Prioritized-Virtual-Time CSMA: Head-of-the-Line Priority Classes without Added Overhead

Mart L. Molle, *Member, IEEE*

Abstract— **The prioritized-virtual-time carrier sense multiple access (PVT–CSMA) protocol is a media access algorithm for cable- and radio-based local area networks that incorporates message-based priority classes. PVT–CSMA implements the well-known head-of-the-line (HOL) priority queueing discipline, in which higher priority messages are always transmitted in preference to lower priority messages, and messages are transmitted first-come–first-served within each priority class. This priority discipline works by altering the way in which stations manipulate the "virtual clocks" that scan the time axis for message arrivals, to ensure that higher priority messages are encountered by the virtual clocks before lower priority messages. Thus, unlike other approaches for incorporating message-based priority classes in CSMA protocols, ours works implicitly, without adding any channel overhead compared to the nonpriority case. Expressions are obtained for both throughput as a function of offered load and mean delay as a function of throughput for asynchronous (unslotted) PVT–CSMA, which are easily solvable for any number of priority classes and for class-dependent message lengths, arrival rates, retransmission rates, etc. Specialization of our delay analysis to the single class case yields a substantial improvement over our previous results for asynchronous virtual-time CSMA. Comparisons are made between our analysis and simulations of PVT–CSMA, and with simulations of Tobagi's P–CSMA protocol.**

I. Introduction

CSMA protocols allow a set of nearby *stations* to share a common channel under distributed control. Each station executes a copy of the protocol to decide when it should transmit its messages, using its own observations of the state of the channel (i.e., whether it is idle or busy, and, if collision detection is available, whether the ongoing message transmission period represents a successful transmission or a destructive collision among several messages). CSMA protocols are *random access* protocols, which means that they provide an efficient way to share bandwidth among large numbers of low duty-cycle (or "bursty") stations and that one can expect small access delays in steady-state operation under moderate load. However, random access protocols do not ordinarily discriminate between stations or message classes. While fairness is in general a useful property in networks, it may not be appropriate when the network must carry different classes of traffic. For example, packetized speech (for which both mean and variance of delay must be small, but some loss is acceptable) should be handled differently from file transfers (for which high throughput may be required, but mean and variance of delay are unimportant). In the case of heterogeneous traffic, we can improve the perceived grade of service for each class of traffic by discriminating on the basis of priority class. Below, we follow the usual terminology from priority queueing (e.g., [10], ch. 3) in assuming that there are P message classes in the network, and that messages of class p have higher priority than (i.e., should be transmitted before) those of class $p - 1$.

Paper approved by the Editor for Wide eArea Networks of the IEEE Communications Society. Manuscript received December 15, 1985; revised August 15, 1986, September 15, 1988 and February 15, 1990. This paper was presented in part at the IEEE International Conference on Communications, ICC' 85, Chicago, IL, June 1985.

The author is with the Computer Systems Research Institute, University of Toronto, Toronto, Ont. M5S 1A4, Canada.

IEEE Log Number 9144895.

Previous approaches for incorporating priority classes into CSMA were based on the addition of reservation periods to the protocol. In P–CSMA [26] each transmission is *followed* by a reservation period consisting of at most P reservation slots. Here each priority class has a turn to transmit a *reservation* (actually a short burst of noise) to indicate that they have a message. If the first reservation occurs in the pth slot, then the channel is reserved, until the next transmission, for access by class p messages (or for classes $P, P-1, \cdots, p$ in the semipreemptive version, which more closely resembles our HOL priority queueing discipline) using the standard p-persistent CSMA protocol. In Priority Ethernet [7], each transmission *begins* with a reservation period in which the stations make use of collision detection and variable length preambles to resolve collisions in favor of the highest priority message, if it is unique. Here stations of class p transmit their preamble for at most p reservation slots. If after j reservation slots, a station does not detect any colliding transmissions $1 \leq j \leq p$ it transmits the message. Otherwise, if the collision lasts through the pth reservation slot, the station must reschedule its transmission because the collision included at least one other message with the same or higher priority.

Extended Hyperchannel [1] and Twentenet [23] are similar to P–CSMA in that a reservation period follows each transmission period. However, the priority mechanism is further extended to encode both the message class and the identity of the transmitting station. For large numbers of stations and/or message classes, this expansion of the reservation period can add considerable channel overhead. Twentenet tries to overcome this overhead by dividing the reservation process into rounds. A first-round reservation period is used to reserve the channel for a given priority class. If a collision occurs following this first round, then a second-round reservation period takes place based on the high-order address bits of the stations that collided in the first round. If a collision occurs following this second round, then a unique station is selected in the third-round reservation period, based on the low-order address bits of the stations that collided in the second round.

To varying degrees, these reservation-period based approaches suffer from the following set of drawbacks. First, they require synchronous operation, with all stations advancing the algorithm in lockstep. Second, they impose channel overhead, since there is a reservation period for each transmission period. In addition to consuming bandwidth that could have been used for transmitting messages, this makes them incompatible at the physical level with existing networks. Third, the overhead is an increasing function for the number of priority classes, and, for both Extended Hyperchannel and Twentenet, also of the number of stations. Fourth, particularly for the Twentenet protocol, each station must keep track of more state information about the network, such as which class (if any) has reserved the network, when the last transmission ended, and so on. And finally, for both Twentenet and Priority Ethernet the reservation overhead is an increasing function of throughput. In our proposal for PVT-CSMA, we have been able to avoid these problems by avoiding the use of explicit reservation periods.

II. Description of PVT–CSMA

PVT–CSMA is an extension of the virtual-time CSMA protocol described in [17], [18], [20]. A slightly different priority implementation called R–VT–CSMA, which is intended for the integration of voice and data, has recently been introduced by Lea and Meditch. In [14], they have analyzed the stability region for R–VT–CSMA, and in [13] they have presented its throughput-delay characteristics obtained via simulation.

The operation of virtual-time CSMA is quite different from the well-known CSMA protocols described in [8], [25], in that messages are assigned transmission times during the idle time on the channel by "recording" the arrival time for each message with the aid of a real-time clock, and then "playing back" the recording of the arrival process with the aid of a virtual-time clock that runs *only* during the idle periods on the channel (since new transmissions must defer to a busy channel), and at an accelerated rate of η so that the (intermittent) virtual-time clock can keep up with the (continuous) real-time clock over the long run. A message is transmitted when the virtual clock reading reaches its recorded arrival time. Since this virtual clock mechanism is essentially forcing the messages to queue for access to the channel, we assume that η times the proportion of idle time on the channel is greater than one to ensure that the queueing delay remains finite. But in this case, the virtual clocks cannot always run at rate η when the channel is idle without scheduling some messages for transmission before they were generated. Thus, to ensure causality, we limit the speed of the virtual-time clocks to unity once they catch up to real-time. Notice that if the virtual clocks are *not* running in step with real-time, they must be either *gaining* on real-time at rate $\eta - 1$ if the channel is idle, or else *falling further behind* at rate unity, if the channel is busy. Consequently, whenever the channel is busy for X time units, the virtual clocks are "busy" (because they are not simply running in step with real-time) for $X + X/(\eta - 1) = \beta X$, where $\beta \triangleq \frac{\eta}{\eta-1}$.

The extension in PVT–CSMA involves changing the "play back" scheme. Thus, messages belonging to class P are scheduled first, by letting only the class P virtual clocks run at rate η_p until the next class P message is encountered or they catch up to real-time. Once the class P virtual clocks have caught up to real-time (possibly after some additional class P transmissions have taken place), we allow messages belonging to both classes $P-1$ and P to be scheduled: the class P clocks continue to run at rate unity (giving newly arriving class P messages free access to the channel), while the class $P - 1$ clocks run at rate $\eta_p - 1$. When the class $P - 1$ virtual clocks eventually catch up to real-time, we start scheduling messages belonging the classes $P - 2, \cdots, P$, and so on. It should be clear that no message from class $c - 1$ or below is transmitted until the class c virtual clock has caught up to real-time, and that within a given class the messages are scheduled in first-come–first-served order. Thus, the scheduling of transmissions in PVT–CSMA is done in HOL-priority order.[1] Notice that whenever the channel is busy for X time units, the class c virtual clocks are "busy" for $X\beta_c$ time units where

$$\beta_c \triangleq \prod_{p=c}^{P} \frac{\eta_p}{\eta_p - 1}, \quad 1 \le c \le P.$$

Figs. 1 and 2 show the effect of imposing two priority classes on a busy period in virtual-time CSMA. Notice that the order in which the high-priority message (H_1) and the preceding low-priority message (L_2) are transmitted has been reversed—as it must be under HOL scheduling, since the arrival of H_1 occurred before the scheduled transmission time for L_2. Also, we emphasize that each station starts its class 1 clock running (at rate η_1) as soon as its class 2 clock slows down to rate unity. Thus, unlike the reservation-period based priority mechanisms described above, each station knows implicitly when to allow traffic belonging to each priority class into the channel *without* having to introduce "gaps" in the channel traffic at priority changeover events. Obviously, this means that class 2 transmissions may collide with class 1 transmissions. However, since at all times the class 1 virtual clock reading is no greater than the class 2 virtual clock reading, it is clear that no class 1 message can ever be delivered "out of turn" in the presence of a class 2 message.

If half the traffic belongs to each priority class, then the values $\eta_2 = 2\eta$ and $\eta_1 = 2\eta - 1$ that we used in Fig. 2 are "optimal" in the following sense. First, these values preserve the total channel overhead imposed by the virtual clock mechanism that was present in the single class case because

$$\beta_1 \equiv \frac{2\eta}{2\eta - 1} \cdot \frac{2\eta - 1}{2\eta - 2} = \frac{\eta}{\eta - 1} \equiv \beta$$

(This result is also evident from Figs. 1 and 2 where the length of the "busy period" for the lowest/only class is exactly the same.) And second, they preserve the channel traffic rate, ηG,

[1] Note that some of these scheduled transmissions may result in collisions, in which case we assume that the backoff algorithm randomly assigns a "new" arrival time to each colliding message. Although our scheme does ensure that these colliding messages are transmitted in HOL-order based on their "new" arrival times, we obviously cannot guarantee anything about the order of delivery in terms of their original generation times.

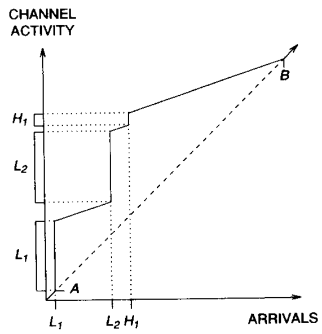


Fig. 1. Operation of virtual-time CSMA without priorities. (A busy period begins at the point marked A and ends at the point marked B.)

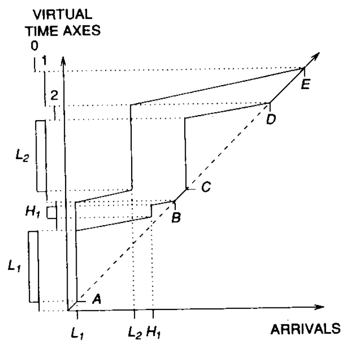


Fig. 2. Operation of PVT-CSMA showing how the channel activity is partitioned into 3 virtual-time axes. (Busy periods for class 2 begin at the points marked A and C, and end at the points marked B and D, respectively. A single busy period for class 1 begins at the point marked A and ends at the point marked E.)

throughout the busy period: if only the class 2 clocks are running, then the channel traffic rate is $(2\eta)G/2 = \eta G$; if the clocks for both priority classes are running, then the channel traffic rate is $G/2 + (2\eta - 1)G/2 = \eta G$.

The same "optimal" imposition of priority classes is easily extended to the general case of P priority classes with traffic intensity G_p for class p, $\sum_{p=1}^{P} G_p = G$. In this case, we let

$$\eta_p = \frac{\eta G - \sum\limits_{i=p+1}^{P} G_i}{G_p}, \quad 1 \le p \le P,$$

so that

$$\beta_1 \equiv \frac{\eta G/G_P}{\eta G/G_P - 1} \cdot \frac{(\eta G - G_P)/G_{P-1}}{(\eta G - G_P)/G_{P-1} - 1} \cdots \frac{\left(\eta G - \sum\limits_{i=2}^{P} G_i\right)\Big/ G_1}{\left(\eta G - \sum\limits_{i=2}^{P} G_i\right)\Big/ G_1 - 1} = \frac{\eta}{\eta - 1} \equiv \beta,$$

as before.

It should be clear from the description above, that adding HOL priority classes to virtual-time CSMA need not introduce any additional overhead to the protocol. All that happens is that the order in which the waiting messages are transmitted is permuted such that higher priority messages are transmitted before lower priority messages. In addition, a station that does not generate messages from all priority classes need not keep track of the evolution of the virtual clocks for every priority class. For example, if the station never generates messages from classes $i, i-1, \cdots, 1$, then it can ignore those classes completely, to the point of being unaware of the ***number*** of lower priority classes. Similarly, if the station ***does*** generate class $j-1$ messages, but never any messages from classes j, $j+1, \cdots, k$, then it can replace the $k - j + 1$ virtual clocks by a single "equivalent" clock with rate $\eta_{j,k} = \beta_j / (\beta_j - \beta_{k+1})$. Thus, one could configure a PVT-CSMA system with a large number of priority classes without imposing an excessive processing burden on any station.

III. Throughput Analyses for PVT-CSMA

In this paper, we assume that the network consists of a large number of low duty cycle stations that collectively generate traffic (consisting of both new messages and retransmissions) according to a Poisson distribution at the rate of G_p for priority class p, $p = 1, \cdots, P$. For simplicity of notation, we define $G_0 \equiv 0$, and $\eta_0 \equiv 1$. Messages of *variable* length within each class are permitted. In particular, whenever a class p message is transmitted, we assume that its transmission time, t_p, is drawn independently from a (class dependent) general distribution with probability density function $f_{t_p}(t_p)$, mean $\overline{t_p}$, and second moment $\overline{t_p^2}$. We will consider only *asynchronous* (i.e., "unslotted") operation on the worst case "star" topology, in which the propagation time between every pair of stations is some fixed constant a. For the synchronous (or "slotted") version of the protocol, see [20] for an analysis of the nonpriority case using similar techniques, and [14], [15] for a more traditional embedded Markov chain approach.

Because our analysis in this paper is for asynchronous version of the protocol, our results depend on the timing interactions of events originating at different points in the network. Thus, it is worth summarizing the sequence of events that takes place whenever there is a successful transmission or a collision. Fortunately, this task is made much simpler because of our previous assumptions. Because of the "star" topology, all other stations will sense the actions of a given station (i.e., starting or terminating a message transmission) simultaneously. And because of the large population assumption, almost all the stations (i.e., those that have not transmitted during the current busy period) will see exactly the same channel history, which is the same as an observer sitting at the "hub" of the star sees but delayed by $a/2$ time units.[2]

[2]Had we considered a topology other than the worst case "star," then a precise analysis of the events in a transmission cycle would have been complicated by the fact that the times at which stations in an asynchronous protocol sense those channel state changes depend on their positions relative to the transmitting station(s) [21]. It is important to note, however, that even though the present *analysis* assumes a worst case "star" topology, the *operation of the protocol* does not depend on that assumption. Other topologies would merely allow some drift to occur between the local copies of the class p clocks at different stations, giving those stations with faster clocks a slight priority advantage over the rest until they all catch up to real-time and (implicitly) resynchronize. The magnitude of the clock drift between stations was studied in [11] where it was found to be quite small.

Define a *transmission cycle* to be an interval of time between two successive channel state changes from "busy" to "idle" as observed at the hub of the star. Suppose a cycle begins at $t = 0$. Then at time $a/2$, all stations (except the last one to have stopped transmitting) sense the state change to "idle" and resume scheduling their messages. Once the first transmission begins, a further time of $a/2$ elapses before the state change reaches the hub. Thus, assuming that the same station does not transmit twice in succession, the length of the idle period will be the constant a plus the minimum of the "think times" of all the stations. Note that a time units elapse before the other stations sense the start of the first transmission, so it will be successful if no other stations begin their own transmissions in this time. In either case, the channel will be busy at the hub from $a/2$ units after *first* station begins transmitting until $a/2$ units after the *last* station stops transmitting.

A. Partitioning the Channel Time Line into $P + 1$ Virtual-Time Axes

The direct analysis of the sequence of transmission cycles in real-time is not easy because abrupt changes in the channel traffic may occur whenever the virtual clocks for some class catch up to real-time. Thus, even under the above "strong" Poisson traffic assumption, the lengths of the channel idle times are *not* exponentially distributed. However, we can partition the channel time axis into $P + 1$ *virtual-time axes,* each of which sees a time-homogeneous Poisson arrival process [20]. As shown in Fig. 2, virtual-time axis i, $1 \le i \le P$, consists of the union of all intervals of channel idle time where the virtual clocks for priority class i are running at rate η_i, together with all message transmission periods that directly follow some channel idle time belonging to that virtual-time axis. The leftover parts of the real-time axis where all virtual clocks have caught up to real-time, are assigned to virtual-time axis 0. Thus, during the idle periods on virtual-time axis i, $0 \le i \le P$, we will say that the class p virtual clocks run at rate

$$r_{p,i} = \begin{cases} 1 & p > i \\ \eta_p & p = i \\ 0 & p < i \end{cases} \tag{1}$$

and that the channel traffic rate is

$$G_i^* = \sum_{p=i}^{P} r_{p,i}\, G_p. \tag{2}$$

Once we have divided the channel time line into $P + 1$ virtual-time axes, we can then partition each virtual-time axis into a sequence of transmission cycles. Note that in reality, a transmission cycle contained in virtual-time axis i, or *i-cycle*, may consist of the union of several disjoint intervals of channel time, namely parts of one or more channel idle periods in real-time, terminating with a message transmission period. Nevertheless, it should be clear that we can partition the entire channel time line into i-cycles, $0 \le i \le P$, each of which is contained in a single virtual-time axis. Let $\{\pi_i\}$ be the limiting distribution for the fraction of such transmission cycles that are i-cycles.

If we restrict our attention to the i-cycles, their analysis becomes quite straightforward. In this case, we appear to have a time-homogeneous Poisson arrival process (say at rate G_i^* on virtual-time axis i), and the durations of the message transmission times appear to be i.i.d. random variables (say $\boldsymbol{X}_i$ on virtual-time axis i). Note that because of the memoryless property of Poisson arrivals, it does not matter that the channel idle time attributed to an i-cycle may not be contiguous on the channel time line.

In the remainder of this section, we will compute the throughput for each priority class S_i for the given values of G_i and η_i, $i = 1, 2, \cdots, P$. This will be done in three stages. First, for each class $p = 1, 2, \cdots, P$, we find its contributions $\boldsymbol{H}_{p,i}$ and $\boldsymbol{X}_{p,i}$ to the throughput and channel busy time, respectively, in an i-cycle, $i = 0, 1, \cdots, P$. Second, we make use of a set of "global balance equations" on the average advance of real and virtual-time per transmission cycle (i.e., *without* conditioning on i) to find $\{\pi_i : i = 0, \cdots, P\}$. And, finally, we find the total throughput per class as the ratio of the expected contribution to the class p throughput per transmission cycle to the expected duration of a transmission cycle.

B. Derivation of $E[\boldsymbol{H}_{p,i}], E[\boldsymbol{X}_{p,i}]$ and $E\left[(\boldsymbol{X}_{p,i})^2\right]$

Because of the Poisson assumption, the conditional probability that a message belongs to class p given that it was transmitted on virtual-time axis i is

$$\zeta_{p,i} \triangleq \frac{r_{p,i} G_p}{G_i^*}, \quad i \le p \le P. \tag{3}$$

Consequently, an i-cycle may contain the transmission of messages from any class $p \ge i$, and, indeed, a collision may contain messages from several classes. However, without loss of generality, we can attribute the channel busy time to class p if it begins with a class p transmission. Because of carrier sensing, the probability of success for this first message, namely, $e^{-aG_i^*}$, is independent of its class. Thus, we have

$$E[\boldsymbol{H}_{p,i}] = \zeta_{p,i} E[\boldsymbol{t}_p] e^{-aG_i^*}. \tag{4}$$

The moments of $\boldsymbol{X}_{p,i}$ can now be found using the results for $f_x(x)$ in the appendix. Using the definition of $\boldsymbol{X}_{p,i}$, it is clear that the pdf, $f_t(x)$, for the length of the first message in the channel busy time must be that of class p, i.e.,

$$f_t(x) \equiv f_{t_p}(x). \tag{5}$$

Similarly, using the definition of an i-cycle, it is clear that the pdf, $f_{t'}(x)$, for the lengths of the colliding messages (if any) in the channel busy time will be a weighted (by $\{\zeta_{p,i}\}$) average of the pdf's for classes i, $i + 1, \cdots, P$, namely,

$$f_{t'}(x) \equiv \sum_{c=i}^{P} \zeta_{c,i} f_{t_c}(x). \tag{6}$$

Thus,

$$E[\boldsymbol{X}_{p,i}] = \int_{x=0}^{\infty} x f_x(x)\, dx \tag{7}$$

and

$$E\left[(\boldsymbol{X}_{p,i})^2\right] = \int_{x=0}^{\infty} x^2 f\boldsymbol{x}(x)\,dx \tag{8}$$

where $f\boldsymbol{x}(x)$ is given by either (A.7) or (A.4), if collision detection is or is not present, respectively. When collision detection is present, (7) and (8) may be simplified to obtain

$$\begin{aligned}E[\boldsymbol{X}_{p,i}|\text{collision detection}] &= e_i^{-aG^*}(\overline{t_p} - a)\\ &\quad + \left(1 - e^{-aG_i^*}\right)\cdot\ (1/G_i^* + a + c),\end{aligned} \tag{9}$$

and

$$\begin{aligned}E\Big[&(\boldsymbol{X}_{p,i})^2|\text{collision detection}\\ &= e_i^{-aG^*}\left(\overline{\mathbf{t}_p^2} - a(3a + 2c + 2/G_i^*)\right)\\ &\quad + \ \left(1 - e^{-aG_i^*}\right)\left((1/G_i^* + a + c)^2 + 1/G_i^*\right).\end{aligned} \tag{10}$$

C. Derivation of $\{\pi_i\}$ and the Throughput for Each Class

Let $E[\boldsymbol{L}_i]$ be the average duration of an i-cycle. Since the mean idle time in an i-cycle is $1/G_i^* + a$, and since the mean channel busy time is found from $\{E[\boldsymbol{X}_{p,i}]\}$, with respective weights $\zeta_{p,i}$, we have

$$E[\boldsymbol{L}_i] = \frac{1}{G_i^*} + a + \sum_{p=i}^{p} \zeta_{p,i} E[\boldsymbol{X}_{p,i}]. \tag{11}$$

Recall that π_i is the probability that a transmission cycle chosen at random is an i-cycle. To find $\{\pi_i\}$, we make use of the following "global balance" equations on the average advance of the class p virtual clocks over a transmission cycle in equilibrium. More precisely, if the mean delay for class p messages is finite, then in equilibrium the average advance of the class p virtual clocks over a cycle, i.e.,

$$\delta_p \triangleq \sum_{i=0}^{p} r_{p,i}\left(\frac{1}{G_i^*} + a\right)\pi_i, \tag{12}$$

and the average duration of a cycle, i.e.,

$$E[\boldsymbol{L}] = \sum_{i=0}^{p} E[\boldsymbol{L}_i]\pi_i, \tag{13}$$

must be equal. Now suppose the delays for all classes c, $c+1, \cdots, P$ are finite (and $c = 1$ if all classes experience finite delays). Then, since $\delta_j = E[\boldsymbol{L}]$ for all $j \geq c$, we can equate δ_c and δ_{c+1} after dropping the common terms to obtain

$$\eta_{c+1}\left(\frac{1}{G_{c+1}^*} + a\right)\pi_{c+1} + \left(\frac{1}{G_c^*} + a\right)\pi_c = \eta_c\left(\frac{1}{G_c^*} + a\right)\pi_c,$$

from which we find after some algebraic manipulations that

$$\pi_i = \pi_c \prod_{j=c}^{i-1} \frac{(\eta_j - 1)(1/G_j^* + a)}{(\eta_{j+1})(1/G_{j+1}^* + a)} \quad i \geq c. \tag{14}$$

Since $\pi_i = 0$ for all $i < c-1$, we can substitute (14) into (13) and equate it to (12) evaluated at $p = c$ to obtain

$$\pi_{c-1} = \pi_c \frac{-\eta_c(1/G_c^* + a) + \sum\limits_{i=c}^{p} E[\boldsymbol{L}_i] \sum\limits_{j=c}^{i-1} \frac{(\eta_j - 1)(1/G_j^* + a)}{\eta_{j+1}(1/G_{j+1}^* + a)}}{1/G_{c-1}^* + a - E[\boldsymbol{L}_{c-1}]}. \tag{15}$$

To find the throughput for each priority class S_p we simply take the ratio of the expected class p throughput per transmission cycle to $E[\boldsymbol{L}]$, i.e.,

$$S_p = \frac{1}{E[\boldsymbol{L}]} \sum_{i=0}^{p} E[\boldsymbol{H}_{p,i}]\pi_i. \tag{16}$$

IV. Delay Analysis

Our estimate for the mean message delay for each priority class in PVT–CSMA is based on the standard "equilibrium" model for random access protocols [10]. That is, we partition T_p into several disjoint components, i.e.,

$$T_p = T_{0,p} + \left(\frac{G_p \overline{t_p}}{S_p} - 1\right) T_{R,p} + \overline{t_p} + a \tag{17}$$

where $T_{0,p}$ includes the initial waiting time up to the first transmission attempt for class p messages, $T_{R,p}$ includes the delay at each unsuccessful attempt,[3] and $\overline{t_p} + a$ includes the transmission and propagation time during the final, successful attempt. It can be shown [20] that under stable operation (i.e., $T_{0,p} < \infty$), the average time for the virtual clocks to advance by $\boldsymbol{R}_p$, the retransmission delay, is simply $\boldsymbol{R}_p$. Thus, since $T_{R,p}$ is the sum of the durations for an unsuccessful transmission attempt and the subsequent retransmission delay, we have immediately that

$$\begin{aligned}T_{R,p} &= E[\boldsymbol{R}_p]\\ &\quad + \sum_{i=0}^{p} \Pr[\text{transmission was on axis } i \,|\, p, \text{ collision}]\\ &\quad \cdot E[\boldsymbol{X}_{p,i} \,|\, \text{collision}]\\ &= E[\boldsymbol{R}_p] + \sum_{i=0}^{p} \frac{\Pr[\text{collision} \,|\, p, i]\ \Pr[i \,|\, p]}{\Pr[\text{collision} \,|\, p]}\\ &\quad \cdot E[\boldsymbol{X}_{p,i} \,|\, \text{collision}].\end{aligned} \tag{18}$$

Of course, this expression for $T_{R,p}$ is an approximation because we have chosen to evaluate $T_{0,p}$ and $T_{R,p}$ according to the Poisson total traffic assumption. However, it will become clear below that retransmissions are rare in virtual-time CSMA, so the dominant term in (17) is the initial delay rather than the retransmission delay. Thus, we shall devote the remainder of this section to estimating $T_{0,p}$; we will return to the Poisson assumption and the associated stability issue below in Section V.

[3]Note that in the absence of collision detection, we are making an "independence assumption" in which a message does not retain its length on successive transmission attempts. However, unlike ALOHA (where long messages are much more likely to suffer a collision), this approximation is not very important in our case because the collision probability is low and because of carrier sensing, independent of the message length.

A. The "Moving Server" Head-of-the-Line $M/G/1$ Queueing System

In a moving server queueing system with first-come–first served (FCFS) order of service, the server comes to the customers by walking along the arrival time line, instead of the usual case where the customers come to the server and form a queue. Whenever the server encounters a customer, he pauses to offer service, and then continues his walk at an accelerated rate (of η, say) until he either encounters the next customer or catches up to real-time (in which case, by slowing his walk to unit speed, he is certain to encounter the next customer at the moment of his arrival to the system). Obviously, message scheduling in (single class) virtual-time CSMA is an example of such a moving server system.

In [12], we showed that for FCFS moving server systems with arbitrary customer interarrival and service time distributions, the waiting time for each customer is proportional to the waiting time for the corresponding customer in a "synthetic" queueing system. The key step is to observe that if $A^{(k)}$ and $X^{(k)}$ represent the sequence of interarrival and service times, respectively, in the moving server system, then the waiting time sequence satisfies the recurrence relation

$$\begin{aligned}\tilde{W}^{(k)} &= \max\left\{\tilde{W}^{(k-1)} + X^{(k-1)} - A^{(k)} + A^{(k)}/\eta, 0\right\}\\ &= \max\left\{\tilde{W}^{(k-1)} + X^{(k-1)} - A^{(k)}/\beta, 0\right\}. \qquad (19)\end{aligned}$$

Since the recurrence relation reduces to

$$W^{(k)} = \max\left\{W^{(k-1)} + X^{(k-1)} - A^{(k)}, 0\right\} \qquad (20)$$

for an ordinary FCFS queuing system without moving server overhead, we see that (19) can be put into the same form as (20), by increasing each service time by a factor of β and ignoring the moving server overhead—thus forming the synthetic system—and then introducing a change of variable from $\beta\tilde{W}^{(k)}$ to $W^{(k)}$. Hence, under FCFS order of service, the waiting time statistics for a moving server system may be obtained from the corresponding synthetic system using the relation

$$\tilde{W} = W/\beta. \qquad (21)$$

Now consider the generalization of the model to a moving server system with P HOL-priority classes. In this case, there are P separate arrival time lines—one for each class—with a moving server walking along each one. Whenever *any* of the servers encounters a customer, *all* of them stop until the service is complete. Thereafter, only the class P server begins walking at rate η_p, and when he has caught up with real-time (and reduces his speed to unity) the class $P-1$ server begins walking at rate η_{p-1}, and so on. We say that the class p server is *busy* whenever he is stopped (either because some customer is actually being served, or because the class $p+1$ server is still busy) or walking at rate η_p, and that he is *idle* otherwise. Obviously, message scheduling in PVT–CSMA is an example of such a moving server system.

In [22] we showed that the waiting times for customers of each class in an $M/G/1$ HOL priority queue with moving server overhead can be obtained from the corresponding results for a conventional $M/G/1$ HOL priority queue with class j arrival rate λ_j and service time x_j [2], [10]. In particular, the mean class p waiting time for an $M/G/1$ moving server HOL priority queueing system is given by [22]

$$\tilde{W}_p = \frac{\beta_{p+1}\cdot\beta_p W_0}{\left(1-\beta_p \sum_{j=p}^{P} \overline{x_j}\lambda_j\right)\left(1-\beta_{p+1} \sum_{j=p+1}^{P} \overline{x_j}\lambda_j\right)} \qquad (22)$$

where

$$W_0 = \frac{1}{2}\sum_{k=1}^{P} \lambda_k \overline{x_k^2} \qquad (23)$$

is the mean residual life of the service time for the customer (if any) in service at his arrival, the ratio of the numerator to the left-hand factor in the denominator represents the component of his mean delay due to customers in the queue on his arrival, and the ratio of that component to the right-hand factor in the denominator represents the mean duration of the delay busy period that ends with his entry into service.

B. Application to the Initial Delay in Virtual-Time CSMA without Priorities

Some care is required in using the results above, for "ideal" moving server queueing systems, to obtain the initial delay in virtual-time CSMA and PVT-CSMA. This is because the nonzero propagation time inherent in CSMA systems manifests itself as *state dependence* (according to the current virtual-time axis) in both the customer arrival process and the service time distribution in the corresponding moving server system. In particular, we found in Section III-B. that the service time $X_{c,i}$ for a class c customer encountered in an i-cycle depends on G_i^*, which governs the probability that the customer represents a successfully transmitted class c message [and, in the absence of collision detection, also on the message lengths for each class and their respective channel traffic rates on virtual-time axis i via (6) and (A.4)]. Furthermore, the arrival rate for class c customers in the moving server system is also a function of the virtual-time axis $i \in \{c, c-1, \cdots, 0\}$ where those customers were encountered.[4]

Surprisingly, the results from the previous section *can* be used to obtain an exact expression for T_0 in asynchronous virtual-time CSMA under the Poisson total traffic model. This expression is a significant improvement over our previous results, [20] where we obtained [weak] bounds on T_0 by assuming that the best/worst case state dependent values,

[4]Note the distinction between the arrival of a customer in the moving server system, which happens exactly once per transmission cycle, and the arrival of a message in the PVT-CSMA system, which is governed by a Poisson process, at intensity G_c for class c, and is independent of the state of the protocol. Since several message arrivals are reduced to a *single* customer arrival in the event of a collision (by "erasing" all but the first message arrival, say), and since *none* of these colliding messages contribute towards the throughput in the PVT–CSMA system, the state dependent customer arrival rate for each class, $\lambda_{c,i}$, must satisfy $S_c/\overline{t_c} < \lambda_{c,i} < G_c$. It is also important to note the distinction between the above state dependence in the customer generation process, and the state dependence of the mapping from a customer's generation time to the time of his entry into service, as shown on the horizontal "Arrival Time" and vertical "Channel Time" axes of Fig. 2, respectively.

respectively, for the customer arrival rate and service times were to hold over all time. The improvement in this paper is obtained by applying the transformation introduced in [24] to eliminate the state-dependent "gaps" in the (Poisson) arrival process, and then solving the resulting synthetic system as an M/G/1 queue in which the first customer of each busy period has a different service time distribution [6, Section 4.3] to account for the lower collision probability (and hence atypical service times) on virtual-time axis 0.

Let $X^{(k)}$ and $A^{(k)}$, respectively, be the service time and interarrival time leading up to the kth customer in the synthetic system. Recall that $A^{(k)}$ consists of a "gap" in the arrival process (of duration $r^{(k-1)}a$ where $r^{(k-1)} \in \{1, \eta\}$ is the virtual clock rate when the kth customer was encountered) followed by an exponential interarrival time with parameter G. Thus, if we subtract each "gap" from the corresponding interarrival time, the remainder, namely, $A^{(k)} - r^{(k-1)}a$, is clearly Poisson distributed with parameter G. Having thus transformed the sequence $A^{(k)}$ in the synthetic system into a Poisson process, we must now apply a complementary transformation to the sequence $X^{(k)}$ so as to preserve the customer waiting times, $W^{(k)}$. Using (20), it is clear that the transformed service times must be $X^{(k)} - r^{(k)}a$. Unfortunately, with collision detection the transformed service time for a collision is upper bounded by $(2 - r^{(k)}) \cdot a + c$, which is likely to be negative for $r^{(k)} = \eta$. Thus, the result below is only useful if there is no collision detection. Fortunately, our approximate result for PVT–CSMA, specialized to $P = 1$, was found to give excellent agreement in our numerical examples.

Thus, T_0 for asynchronous virtual-time CSMA without collision detection follows immediately from [6, (4.41)], as

$$T_0 \approx \frac{\beta G \cdot E\left[(\boldsymbol{X}_{1,1} - \eta_1 a/\beta)^2\right]}{2(1 - G \cdot E[\beta \boldsymbol{X}_{1,1} - \eta_1 a])} + \frac{\beta G \cdot \left(E\left[(\boldsymbol{X}_{1,0} - a/\beta)^2\right] - E\left[(\boldsymbol{X}_{1,1} - \eta_1 a/\beta)^2\right]\right)}{2(1 - G \cdot E[\beta \boldsymbol{X}_{1,1} - \eta_1 a] + E[\beta \boldsymbol{X}_{1,0} - a])} \tag{24}$$

where $\boldsymbol{X}_{1,i}$ is given by (7)–(8).

C. Application to the Initial Delay in PVT–CSMA

Unfortunately, our approach for the nonpriority case described above does not generalize to PVT–CSMA, even without collision detection. This is because the transformation we used to eliminate the "gap" in the arrival process in an i-cycle imposes contradictory constraints on the transformed service time, since the gap in the class c arrival process that we wish to cancel is either, $a, \eta_i a$, or 0, for $c > i$, $c = i$, and $c < i$, respectively. However, we can still improve on the best/worst case bounds for $T_{0,p}$ in PVT–CSMA described in [19], [20] by a substantial margin, by substituting carefully chosen weighted averages of the various state dependent parameters into (22). These weighted average parameter values are obtained as follows.

First, since the servers encounter an average of $\zeta_{c,i}$ class c customers per i-cycle, during which the class c server moved an average distance of $r_{c,i}(1/G_i^* + a)$ along the arrival time line, we obtain

$$\lambda_{c,i} = \frac{G_c}{1 + aG_i^*}, \quad i \le c. \tag{25}$$

Note that (25) is the exact result for the [state dependent] mean customer arrival rate, but its application to (22) will result in an approximation whenever the assumption of Poisson arrivals is required.

Equation (23) represents the residual service time of the customer (if any) in service when a "tagged" class p message arrives. Because of the Poisson traffic model, this "tagged" message arrival takes a uniform look at the channel time line.[5] Thus we can solve for W_0 via a renewal type argument that depends on the relative frequency of the different possible service times but not on their arrangement along the (channel) time line. But since $\pi_i \zeta_{p,i}/E[\boldsymbol{L}]$ represents the mean arrival rate (in customers per unit time) of class p customers whose service takes place in an i-cycle, in which case his service time will be given by $\boldsymbol{X}_{p,i}$, (23) reduces immediately to

$$W_0 = \frac{1}{2E[\boldsymbol{L}]} \sum_{i=0}^{P} \sum_{c=i}^{P} \pi_i \zeta_{c,i} E\left[\boldsymbol{X}_{c,i}^2\right]. \tag{26}$$

Now consider the summation in the left-hand term in the denominator of (22), representing the queueing delay for our tagged class-p customer due to customers from classes $c = p, p+1, \cdots, P$ who were waiting in the queue at his arrival. But recall that our priority access rule was constructed so that a class c customer either experiences *no* queueing delay (if his arrival takes place during the idle time in an i-cycle, for some $i < c$), or his service takes place in a c-cycle (and thus *before* any waiting messages from classes $c-1, c-2, \cdots$ could enter service). Thus,

$$\sum_{c=p}^{P} \overline{\boldsymbol{X}_c} \lambda_c \approx \sum_{c=p}^{P} \overline{\boldsymbol{X}_{c,c}} \lambda_{c,c} = \sum_{c=p}^{P} \frac{\zeta_{c,c} \overline{\boldsymbol{X}_{c,c}}}{\eta_c (1/G_c^* + a)}. \tag{27}$$

The situation is more complicated in the right-hand term, representing customers of strictly higher priority who arrive after the tagged class p customer but are served before him. The key observation is the following. A tagged class p customer arriving at time τ will enter service at time $\tau^* \ge \tau$, corresponding to the event that the class p server has reached τ and the servers from classes $p+1, \cdots, P$ have all reached τ^*. But by definition, that part of the tagged customer's waiting included in the numerator and left-hand term in the denominator of (22) represents exactly the time required for the servers from classes $p, p+1, \cdots, P$ to reach τ on the arrival time axis but no farther. (Otherwise, we might have missed giving service to some customer who arrived earlier or inadvertently already given service to some customer who arrived later.) Thus, it should be clear that every customer who contributes to the right-hand summation *must* have been served during a $(p+1)$-cycle, $\cdots$, or a P-cycle. Furthermore, the servers from classes $p+1, \cdots, P$ each move from τ to

[5] Remember that we require $T_{0,p}$ for *messages* arriving to the PVT–CSMA system, before any of them are "erased" from the *customer* arrival process in the corresponding moving server system.

τ^* on the arrival time axis during this part of the tagged customer's delay. But by eliminating common terms from (12), we see that to equal the average advance of the class $p+1$ clocks over a $p+1$-cycle, the class c clocks require π_c/π_{p+1} c-cycles, plus π_{c-1}/π_{p+1} $(c-1)$-cycles, $\cdots$, plus one $p+1$-cycle. Since each i-cycle advances the class c clocks by an average of $r_{c,i}(1/G_i^* + a)$, a fraction

$$\frac{\pi_i r_{c,i}(1/G_i^* + a)}{\pi_{p+1}\eta_{p+1}(1/G_{p+1}^* + a)}$$

of the interval between τ and τ^* will be examined for class c customers in state i. Combining this weight with the expressions above for $\lambda_{p,i}$ and $X_{p,i}$, we obtain after some manipulation

$$\sum_{c=p+1}^{p} \overline{X_c}\lambda_c \approx \frac{\sum_{c=p+1}^{p}\sum_{i=p+1}^{c} \pi_i \zeta_{c,i}\overline{X_{c,i}}}{\pi_{p+1}\eta_{p+1}(1/G_{p+1}^* + a)}. \tag{28}$$

Substituting (26)–(28) into (22) gives us the desired approximation to $T_{0,p}$.

V. Numerical Results and Discussion

A. Comparison to VT–CSMA without Priorities

The first set of performance curves are chosen to match [20, Fig. 13] except for the imposition of two message-based priority classes. That is, we consider asynchronous PVT–CSMA on the worst case "star" topology, assuming that $t_1 = t_2 \equiv 1$, that the propagation time $a = 0.01$, that there is collision detection with collision recovery time $c = 0.001$, and that the mean retransmission delay is 3. The major difference here is that instead of a single class system using $\eta = 10$, we now let two classes share the traffic equally using $\eta_2 = 20$ and $\eta_1 = 19$. (Recall from the discussion of Figs. 1 and 2 that this is the "optimal" way to impose priorities in this example.)

In Fig. 3, we have shown each component of the delay for each class, both from our analysis and from detailed simulations.[6] sAs is evident from the figure, our analysis fits the simulation data remarkably well. The only real discrepancy seems to be that our analysis has overestimated $T_{0,2}$ and underestimated $T_{0,1}$ reaching about 5% relative error in each case under heavy load. We believe that this discrepancy is related to our choice of a simple nonadaptive retransmission algorithm in the simulation. Because of this retransmission algorithm, we expect the transient traffic rate on the channel due to each class to increase during i-cycles $i > 0$ and to decrease during 0-cycles. But by comparing Figs. 1 and 2, it is clear that within each busy period, high-priority messages are

[6]The simulation model faithfully represents the operation of a "star"-shaped LAN in which 21 individual stations (each equipped with a 15-message buffer) are executing local copies of the asynchronous PVT–CSMA protocol. Here only the *newly generated* messages are Poisson—the retransmissions are carried out under the control of a nonadaptive exponential retransmission algorithm where $R_p = 3 \cdot \overline{t_p}$. Note also that each station *independently* (re)schedules the messages contained in its buffer, so the resulting system resembles 315 single-buffered stations arranged in 21 clusters. For more details on the simulation, see [11], [18].

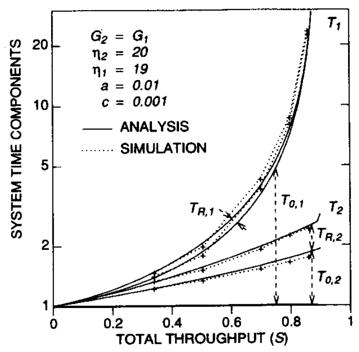


Fig. 3. Delay components for PVT–CSMA (two classes with equal traffic).

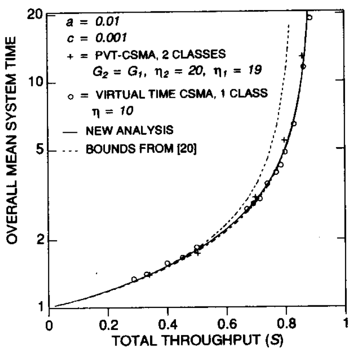


Fig. 4. A demonstration that PVT–CSMA can add priority classes to virtual-time CSMA *without* affecting the overall class-independent mean system time.

transmitted earlier on average than are low-priority messages. (Otherwise, we would not have prioritized access!) Thus, in contrast to the analysis (where we assumed time-homogeneous Poisson traffic), in the simulation we expect low-priority messages to encounter a (slightly) busier system on average.

In Fig. 4, we have shown simulation results for the overall mean delay (ignoring any priority class distinctions) in virtual-time CSMA operating with and without priorities. The nonpriority points (indicated by "o" on the figure) are taken directly from [20, Fig. 13], while the corresponding points for PVT–CSMA with two priority classes (indicated by '+') represent the average of the two sets of class-dependent points from Fig. 3. As expected, there is no significant difference between these two sets of simulation data.

Fig. 4 also contains a single analytical curve, since it can be shown algebraically that for the given set of parameters (i.e., $t_2 = t_1 \equiv 1$, $G_2 = G_1 = G/2$, $\eta_2 = 2\eta$, $\eta_1 = 2\eta - 1$, and $R_2 = R_1 = R = 3$), the average over all classes for each component of T_p from (17) is the same as the corresponding component of T in the nonpriority case. Once again, we see remarkable agreement between our analysis and the simulation data. (Note also the significant refinement of our analysis in comparison to [20], since we have been able to replace our previous [weak] bounds on T_0 by a single estimate that is in good agreement with the simulation results.)

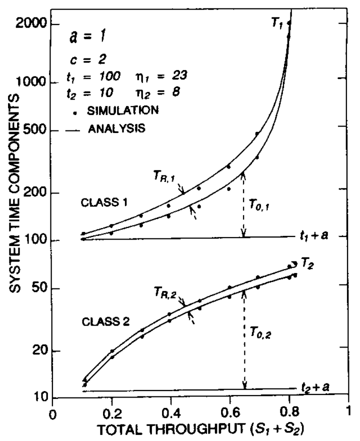


Fig. 5. Delay components for PVT–CSMA with different arrival rates and packet lengths for each class. (Following [5], [26] we fix $S_2 = 0.1$ and increase S_1 from zero to saturation.)

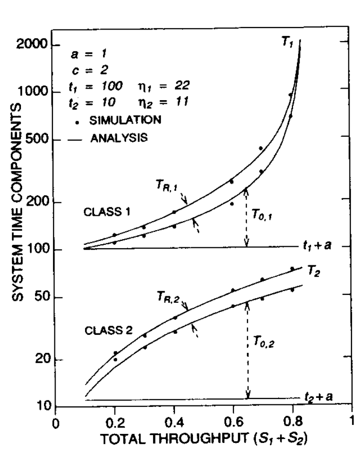


Fig. 6. The effect of perturbing some algorithmic parameters on the system from Fig. 5.

B. Comparison to P–CSMA

The next set of figures is for a different two-class configuration, which has been used previously in numerical results for P–CSMA [5], [26] and in the simulation study of PVT–CSMA by Konstantas [11]. Here we assume that $a = 1$, $c = 2$, $t_2 \equiv 10$, and $t_1 \equiv 100$. In Fig. 5, we have followed Konstantas in assuming that $\eta_2 = 8$, $\eta_1 = 23$ and that $R_p = 3t_p$. This time our analysis shows even better agreement than before with the simulation data for each component of the delay. In Fig. 6, we have perturbed the parameter settings slightly in order to test the predictive power of the analysis. Here we have increased η_2 by almost 40% from 8 to 11, decreased η_1 slightly from 23 to 22, and increased R_2 from 30 to 50. Once again we have excellent agreement between the simulation data and our analysis, and in particular, the analysis correctly predicted that increasing η_2 (and thus reducing β_p) would reduce both $T_{0,2}$ and $T_{0,1}$, but that increasing R_2 would counteract the reduction in $T_{0,2}$ enough to cause a net increase in T_2 in comparison to Fig. 5.

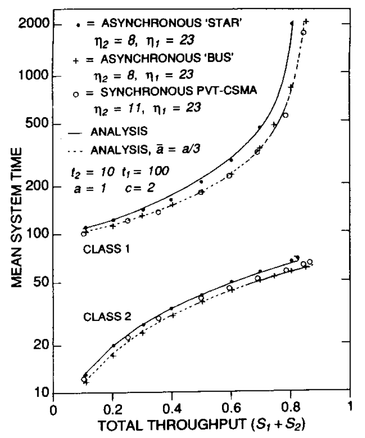


Fig. 7. The effect of changing the system configuration on the mean system time for each class.

In Fig. 7, we investigate the effect of changing the system configuration (as opposed to just changing the parameters of the algorithm) on the performance of PVT–CSMA. In all cases, we retain the previous assumptions of $a = 1$, $t_2 = 10$, $t_1 = 100$, and that there is collision detection such that under asynchronous operation the collision recovery time $c = 2$ and (following Tobagi [26]) under synchronous operation the transmission times for collision fragments are $b = 2$. In addition to the results from Fig. 5 (where 21 stations use the asynchronous protocol on the worst case "star" topology), we have included results for two other configurations.

First, to show the impact of topology under asynchronous operation, we have included results for a system configuration identical to the previous one in all respects except that the stations are now assumed to be equally spaced along a linear "bus" network. As expected, the delay-throughput performance of asynchronous PVT–CSMA improves under this much more favourable topology assumption. What is more surprising, however, is how well our analytical results fit the simulation data for this "bus" network when we just substitute the *average* propagation time, $\bar{a} \equiv a/3$ for the "bus" network, in place of the worst case propagation time a in our expressions for the "star" topology. This heuristic substitution was discussed in [21, Section V], in the estimation of the throughput for nonpersistent CSMA in more general topologies. Unfortunately, the geometry of collisions in space-time is quite complex, so it remains an open problem to determine analytically whether or not this heuristic gives us either an upper or a lower bound. We attribute the remarkable accuracy of the heuristic in this example to the fact that collisions make such a small contribution, on average, to the duration of a transmission cycle. This is because the probability of encountering a collision in a transmission cycle is low, and, should a collision occur, collision detection ensures that its duration will be short.

For the second configuration, we have included results for a synchronous PVT–CSMA system. In this case, however, we follow Tobagi [26] in assuming that there are 50 stations (rather than 21) to facilitate a direct

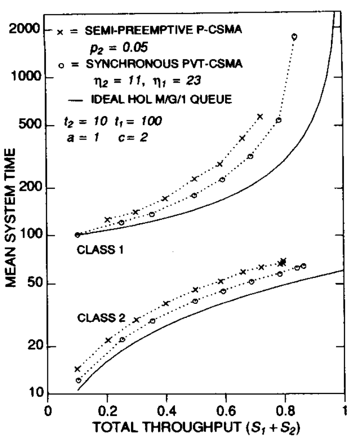


Fig. 8. Comparison of PVT–CSMA and P–CSMA using simulation.

comparison of PVT–CSMA and P–CSMA in Fig. 8. The synchronous PVT–CSMA system also differs slightly from the asynchronous systems in that we have increased η_2 from 8 to 11, thus enabling the class 2 walk times per message to be completed in exactly 10 and 1 steps, respectively, for messages of classes 1 and 2. The previous value of $\eta_1 = 23$ is retained, since the respective class 1 walk times per message are already about as "integral" as we can hope for, namely, 5 and 1/2 steps, respectively. Once again, we see a performance improvement compared to the asynchronous system from Fig. 5.

It is interesting to compare synchronous PVT–CSMA to our previous results for asynchronous PVT–CSMA on the "star" and "bus" topologies. Notice that synchronous PVT–CSMA uniformly outperforms the asynchronous "star" system, while offering worse delay for the high-priority class and comparable delay for the low-priority class in comparison to the asynchronous "bus" system. These results are in agreement with previous studies of the effect of topology on the throughput for nonpersistent CSMA [21, Fig. 11(b)], which show that the throughput for the synchronous system is above the asynchronous "star" but below the asynchronous "bus."

And in Fig. 8, we compare the performance of P–CSMA and PVT–CSMA using simulation. (We use simulation, rather than analysis, because we assumed asynchronous operation in our analysis of PVT–CSMA, and P–CSMA is a synchronous protocol, and because the analytical model for P–CSMA is solvable only for small numbers of stations.) The results shown for P–CSMA are taken from Tobagi [26, Fig. 11]. Although Tobagi's results include a variety of configurations, we have shown only semipreemptive P–CSMA with $p_2 = 0.05$, which, for both classes, attained the lower envelope of the respective sets of delay—throughput curves.[7] The results shown for synchronous PVT–CSMA are taken from Fig. 7. Note that we have carefully chosen the system configuration (i.e., identical values for the message lengths in each class, duration of collision slots, population sizes, etc.) to be as close as possible to Tobagi's results for P–CSMA. The only difference is that each station in the PVT–CSMA system is assumed to have an infinite buffer rather than a single buffer. However, the only effect that this difference could have is to place PVT–CSMA at a disadvantage in the comparison. This is because new messages that are generated while the station is already trying to transmit another message are dropped without penalty in the single-buffer P–CSMA model, and stored in a queue (and thus accumulating delay) in the PVT–CSMA model. Also shown, to illustrate the efficiency of the various protocols in absolute (rather than just relative) terms, are the corresponding ideal results, namely, a centralized, two-class HOL M/G/1 priority queueing system without overhead.

[7]We are excluding fully preemptive P–CSMA from this comparison because it allows a high-priority station to destroy an ongoing low-priority transmission, thereby reducing the delay slightly for the high-priority class under certain [low] throughput conditions while at the same time drastically increasing the delay for the low-priority class. We note, however, that similar response times for the high-priority class are readily obtainable without such preemptions, simply by reducing the maximum low-priority message length.

It is evident from Fig. 8 that both protocols exhibit similar behavior: for fixed S_2, T_2 increases linearly with S_1, whereas T_1 increases without bound as S_1 approaches its maximum. The results also show that synchronous PVT–CSMA offers consistently better performance to both classes than P–CSMA throughout the entire throughput range. Although the margin of improvement is not overwhelming in absolute terms, we must point out that by substituting (synchronous) PVT–CSMA for P–CSMA, we can eliminate more than half of the "excess delay" incurred in comparison to the ideal case. Furthermore, by comparing Figs. 5–7 with Fig. 8, it is evident that the mean delay for each class in P–CSMA is no better than the *worst* of the corresponding results for asynchronous PVT–CSMA in *any* of the configurations shown, even though PVT–CSMA is much simpler to implement, since neither "slotting" nor explicit priority assessment periods on the channel are required.

C. On Stability and the Accuracy of the Poisson Assumption

Those familiar with the importance of the backoff algorithm in determining the performance of other CSMA protocols may be puzzled by the accuracy of our analytical predictions, obtained using an approximate model that excludes the dependence between the channel access and the backoff algorithms. The reason for our success is that even under heavy traffic conditions, PVT–CSMA relies more heavily on *collision avoidance* (via its virtual clock mechanism) rather than on *collision recovery* (using some backoff algorithm). Indeed, it may be readily seen from Figs. 3, 5, and 6 that, for all the system configurations we studied, most of the waiting time in PVT–CSMA occurs *before* the first transmission attempt. Thus, we have chosen to model $T_{0,p}$ in considerable detail, while treating $T_{R,p}$ rather superficially.

The relative unimportance of the backoff algorithm in PVT–CSMA is best illustrated by means of a simple example. Thus, we consider only virtual-time axis P which, by compressing the scheduled transmission times by η_P, provides a worst case analysis for class P. We assume a true infinite population model, where new messages arrive according to a Poisson process at rate S_P. Furthermore, we assume that each message under the control of the backoff algorithm is rescheduled independently, simply by adding an exponential random variable with mean R_P to its "arrival" time—which is exactly what was done in our simulator.

Define an embedded Markov chain whose state k is the number of class P messages under the control of the backoff algorithm at the end of each P-cycle. Observe that the transition probabilities depend *only* on the number of scheduled transmissions within the first a time units of the channel busy time in the P-cycle (during which time the class P clocks are assumed to run at rate η_P), since the number of blocked messages is invariant during the idle time in a cycle, and the stoppage of the virtual clocks for the busy time in the cycle ensures that all other messages in the system remain queued for the next cycle. (And, in particular, observe that the transition probabilities are independent of the lengths of the messages and/or collision fragments transmitted during the cycle.) Let

$$\alpha_1(k) \triangleq P[\text{initial message is new} \mid k] = \frac{S_P}{S_P + k/R_P},$$

$$\alpha_2(k) \triangleq P[\text{initial message is old} \mid k] = \frac{k/R_p}{S_P + k/R_P} = 1 - \alpha_1(k),$$

$$\nu(j) \triangleq P[j \text{ new messages collide with initial message}] = \frac{(a\eta_P S_P)^j}{j!} e^{-a\eta_P S_P},$$

and

$$q_0(k) \triangleq P[\text{no old messages collide with initial message} \mid k] = e^{-a\eta_P k/R_P} = q_0(k-1) \cdot q_0(1).$$

Now in steady state (if it were to exist!), it is clear that the following set of global balance conditions must hold:

$$\begin{aligned}\pi_k = {} & \pi_{k+1}\alpha_2(k+1)\nu(0)q_0(k) \\ & + \pi_k(\alpha_2(k)(1-q_0(k-1)) + \alpha_1(k)q_0(k))\nu(0) \\ & + \pi_{k-1}(\alpha_1(k-1)\nu(0)[1-q_0(k-1)] \\ & + \alpha_2(k-1)\nu(1)) \\ & + \sum_{j=2}^{k} \pi_{k-j}(\alpha_1(k-j)\nu(j-1) \\ & + \alpha_2(k-j)\nu(j)) \end{aligned} \tag{29}$$

where, for notational simplicity we assume that $\pi_{-1} = 0$.

Using the partial sum approach of Fayolle *et al.* [4], we can show that (29) implies that

$$\frac{\pi_{N+1}}{\pi_N} \geq \frac{1 - \alpha_2(N+1)\nu(0) - \alpha_1(N)\nu(0)q_0(N)}{\alpha_2(N+1)\nu(0)q_0(N)},$$

and thus that π_{N+1}/π_N diverges as $N \to \infty$. Nevertheless, it is instructive to plot the ratios, π_k/π_0, for (relatively) small values of k, as we have done in Fig. 9. The two solid curves show the worst case bounds for class 2 messages using the parameter values from Figs. 3 and 5, when $S_2 = 0.35$ and $S_2 = 0.1$ respectively. (Thus, in both cases we are assuming a reasonably heavy traffic situation, since $S_2 = 0.35$ implies $S = 0.7$ in Fig. 3, while in Fig. 5 our assumptions include all feasible values for S.) These worst case bounds are obtained by assuming that *every* cycle is a P-cycle, in which the class P

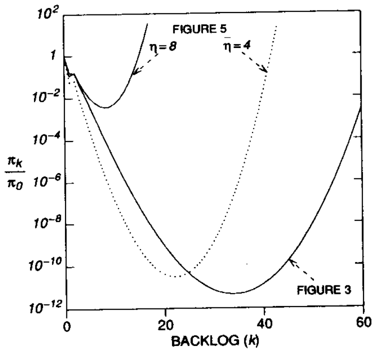


Fig. 9. "Solution" to the embedded chain, showing its initial exponential decrease and ultimate instability, assuming a nonadaptive geometric backoff algorithm.

clocks advance at rate η_P, which maximizes the collision probabilities in every state. Notice that the curve using Fig. 3 data decreases exponentially, to $\pi_k/\pi_0 \approx 10^{-11}$, before asymptotic instability is encountered at $k \approx 30$. However, the corresponding results using Fig. 5 data are not nearly so encouraging, since asymptotic instability is encountered much earlier, at $k \approx 10$ where $\pi_k/\pi_0 \approx 10^{-2}$. However, the situation is not nearly as bad as the worst case bound suggests, since the unusually high value[8] of η_2 means that those worst case transition probabilities are not sustainable. In particular, for $\eta_2 = 8$ we find, using (11)–(12), that $\min_{G_P>0} \frac{1/G_P+\eta_P a}{E[L_P]} \geq 2$ is the *minimum* rate at which the class P clocks can advance over an entire P-cycle, and

$$\min_{G_i^*>0} \frac{1/G_i^* + a}{E[L_i]} \geq \frac{1}{4}$$

is the *minimum* rate at which the class P clocks can advance over an entire i-cycle, $i < P$. Thus, the class P clocks traverse at most 3/7th of the time line at rate η_P, so that the average advance of the class P clocks in a cycle is upper bounded by $\overline{\eta_P} a = (\eta a \cdot 3 + a \cdot 4)/7 = 4a$. Since η_P is based on the most extreme state-dependent values, since $\alpha_1(k)$ and $\alpha_2(k)$are independent of η_P, and since λ^k and $e^{-\lambda}$ are convex $\smile$ functions of λ, substituting $\overline{\eta_P}$ in place of η_P in (29) and solving for π_{k+1} gives us an upper bound on π_{k+1}. This upper bound is shown as the dashed line in Fig. 9.

Although the above stability analysis offers some insight into how our analysis could be in such good agreement with simulations of finite population PVT–CSMA systems (and our primary interest is in the performance of finite population systems), the fact that (29) does not have an equilibrium solution means that there is still an open question about infinite population systems, depending on the form of adaptive backoff algorithm that is used. We note, however, that even our worst case stability analysis shows that, on average, the simple *nonadaptive* infinite population system spends more than 10^{10} transmission cycles in a "quasi-stable" operating mode before

[8] $\eta_2 = 3.2$ would be optimal (in the sense of maximizing capacity) if all traffic were from class 2. But since Fig. 5 involves saturating the channel with class 1 traffic for $S_2 = 0.1$ fixed, Konstantas chose $\eta_2 = 8$ to make [slightly] more capacity available to class 1.

reaching the boundary of the unstable region. Thus, we believe that comparable levels of agreement may be possible if we adopt the dynamic control procedure[9] recently described by Cunningham and Meditch [3].

VI. Conclusion

We have presented the Prioritized-Virtual-Time-CSMA protocol. This protocol is interesting because it adds HOL priority scheduling to an ordinary, nonpriority CSMA protocol implicitly, at no "cost" in increased channel overhead, and without requiring any incompatible timing (e.g., priority assessment periods) or format (e.g., class dependent preambles) changes at the physical level. Since the conversion from the nonpriority to the priority version of the protocol was done entirely by modifying the rules each station uses to update the "window" that selects messages for transmission, this same approach could be used to add priority classes to other sliding-window-based LAN algorithms, including various tree conflict resolution algorithms, e.g., [24], and token rings [12].

We have been able to derive the throughput and delay performance of PVT–CSMA using a generalization and refinement of the analysis for virtual-time CSMA that was introduced in [20]. Unlike models for other prioritized CSMA protocols, our model remains easy to solve for the general case of P priority classes, each with its own message length distribution, virtual clock rate, and retransmission delay. (A side benefit of our analysis has been to find the throughput equation for unslotted nonpersistent CSMA with a general message length distribution, which follows from (13) and (16) when $P = 1$ and $\pi_0 = 1$.) The results we obtain from this analysis were shown to match detailed simulation results remarkably well, and indicate that the performance of PVT–CSMA compares very favorably to other prioritized CSMA protocols.

Appendix

In this Appendix, we derive the density of the channel busy time $f_X(x)$ within a single transmission cycle for asynchronous CSMA protocols *without* collision detection, in which we have 1) the worst case "star" topology (where all stations are assumed to be mutually equidistant); 2) exponential times between successive message transmissions on the idle channel (so our results can be applied to both nonpersistent and virtual-time CSMA, but not to 1-persistent CSMA); 3) a general message length distribution $f_t(x)$ for the first message to begin the channel busy time; and 4) a separate general message length distribution $f_{t'}(x)$ for the colliding messages (if any) such that their lengths are i.i.d. The derivation is complicated by the interactions of messages of different lengths, as well as different starting times, in a collision. In general, neither the last message to *stop* transmitting, nor the *longest* transmitted message will necessarily determine the end of the channel busy time. The corresponding result for the case *with* collision detection is trivial in comparison, since the collision duration depends on the time at which the *second* transmission begins and is independent of the lengths of the messages.

[9]The authors of [3] were primarily concerned with the limiting behavior as $k \rightarrow \infty$, and thus seem to have inadvertently chosen their maximum retransmission rate β so aggressively as to encourage collisions when k is small (and hence $f \approx \beta$). Thus, we suggest that a much smaller maximum rate, for example, the mean retransmission rate $1/\left(3\overline{t_p}\right)$ used in this paper, would be more appropriate, since it gives us the benefit of the exponential initial decrease in p_k/p_0 of the nonadaptive algorithm, while avoiding the asymptotic instability for large k.

Consider the probability $F_X(x \mid t)$ that the channel busy time is at most x, given that the first message had length t. Clearly, for this event to be true no messages with lengths greater than x could have been transmitted. Furthermore, if any messages of length t were transmitted, for $x - a \le t \le x$, they must not have begun after time $x - t$. However, it does not matter whether or not any messages of lengths less than $x - a$ were transmitted. Thus, for $x \ge t$ we have

$$F_X(x \mid t) = e^{-\Gamma(x)} \tag{A.1}$$

where

$$\begin{aligned}\Gamma(x) &= aG \int_{u=x}^{\infty} f_{t'}(u)\,du + G \int_{u=x-a}^{a} (x-u) f_{t'}(u)\,du \\ &= aG[1 - F_{t'}(x) - F_{t'}(x-a)] + G \int_{u=x-a}^{x} F_{t'}(u)\,du.\end{aligned} \tag{A.2}$$

Substituting (A.2) into (A.1) and differentiating, we obtain the conditional density function for $\boldsymbol{X}$, namely,

$$\begin{aligned}f_X(x \mid t) = {}& u_0(x-t) \cdot e^{-\Gamma(t)} + \delta(x-t) \cdot Ge^{-\Gamma(x)} \big[a f_{t'}(x) \\ &+ a f_{t'}(x-a) - F_{t'}(x) + F_{t'}(x-a)\big]\end{aligned} \tag{A.3}$$

where, following [9], we use $u_0(\cdot)$ and $\delta(\cdot)$ to represent the unit impulse and unit step functions, respectively. Unconditioning on t we obtain

$$f_X(x) = \int_{t=0}^{x} f_X(x \mid t) f_t(t)\,dt. \tag{A.4}$$

Notice that if all messages are of constant length, t^* say, then $f_{t'}(u) = u_0(t^* - u)$, and for $t^* < x \le t^* + a$ we have $\Gamma(x) = G(x - t^*)$ and thus

$$f_X(x) = e^{-aG} u_0(x - t^*) + Ge^{-(x-t^*)G}, \quad t^* \le x \le t^* + a,$$

which agrees with the classical result from [8].

Another important special case is where only a discrete set of packet lengths $t_1^* < t_2^* < \cdots < t_N^*$ is permitted, such that $|t_i^* - t_j^*| > a$ for all $i \ne j$. Assume that $t = t_m^*$ for some $m \in \{1, \cdots, N\}$ and also that

$$f_{t'}(u) = \sum_{n=1}^{N} \gamma_n u_0(u - t_n^*); \quad \sum_{n=1}^{N} \gamma_n = 1. \tag{A.5}$$

In this case, the duration of a collision containing at least one message of length t_n^* is independent of the transmissions (if any) of all messages of lengths $t^*_{n-1}, \cdots, t^*_1$. Thus, for x such that $t^*_n \le x \le t^*_n + a$ for some $m \le n \le N$, we have that

$$\Gamma(x \mid t = t^*_m) = (a - x + t^*_n)\gamma_n G + aG \sum_{i=n+1}^{N} \gamma i; \quad t^*_n \le x \le t^*_n + a$$

and hence that

$$f_X(x \mid t = t^*_m) = e^{-\Gamma(t^*_m)} u_0(x - t^*_m) + \sum_{n=m}^{N} \delta(x - t^*_n) \cdot \delta(t^*_n + a - x) \cdot \gamma_n G e^{-\Gamma(x)}. \tag{A.6}$$

The calculation of $f_X(x)$ when there is collision detection is much simpler, since it can be shown [11] that the duration of a collision is the sum of the propagation time a the collision recovery time, c (during which the stations jam the channel after discovering the collision [16]), and the time, y, until the *earliest* colliding transmission begins. Since a and c are constants, and

$$f_y(y, \text{collision}) = Ge^{-yG}, \quad 0 \le y \le a,$$

we have that

$$f_X(x) = f_t(x)e^{-aG} + \delta(x - a - c) \cdot \delta(2a + c - x) \cdot Ge^{-xG}. \tag{A.7}$$

Acknowledgment

The author would like to thank Ms. M.-W. Wu for her help with an early version of this material, and also the anonymous referees whose criticisms of the original manuscript resulted in some substantial improvements in the current version.

References

[1] I. Chlamtac and W. Franta, "Message based priority access to local networks," *Comput. Commun.*, vol. 3, no. 2, Apr. 1980.

[2] R. W. Conway, W. L. Maxwell, and L. W. Miller, *Theory of Scheduling.* New York: Addison-Wesley, 1967.

[3] G. A. Cunningham and J. S. Meditch, "Distributed retransmission controls for slotted, nonpersistent, and virtual-time CSMA," *IEEE Trans. Commun.*, vol. 36, pp. 685–691, June 1988.

[4] G. Fayolle, E. Gelenbe, and J. Labetoulle, "Stability and optimal control of the packet switching broadcast channel," *J. ACM,* vol. 24, no. 3, pp. 375–386, July 1977.

[5] N. Gonzalez-Cawley and F. A. Tobagi, "Simulation of message-based priority functions in carrier sense multiaccess/broadcast systems," Comput. Syst. Lab., Stanford Univ., Tech. Rep. 213, June 1981.

[6] J. F. Hayes, *Modeling and Analysis of Computer Communications Network.* New York: Plenum, 1984.

[7] I. Iida, M. Ishizaka, Y. Yasuda, and M. Onoe, "Random access packet switched local computer network with priority function," in *Conf. Rec., Nat. Telecommun. Conf.,* Dec. 1980.

[8] 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.

[9] L. Kleinrock, *Queueing Systems, Volume 1: Theory.* New York: Wiley–Interscience, 1975.

[10] ——, *Queueing Systems, Volume II: Computer Applications.* New York: Wiley–Interscience, 1976.

[11] D. Konstantas, "Virtual-time CSMA: A study," Comput. Syst. Res. Group, Univ. of Toronto, Tech. Rep. CSRG-149, Jan. 1983; also as a M.Sc. thesis.

[12] F. R. Kschischang and M. L. Molle, "The helical window token ring," *IEEE Trans. Inform. Theory,* vol. 35, Mar. 1989.

[13] C.-T. A. Lea and J. S. Meditch, "A channel access protocol for integrated voice data applications," *IEEE J. Select. Areas Commun.*, vol. SAC-5, pp. 939–947, July 1987.

[14] J. S. Meditch and C.-T. A. Lea, "A virtual-time CSMA protocol for integrated voice data," in *Proc. IEEE INFOCOM '84,* Apr. 9–12, 1984, pp. 185–189.

[15] ——, "Stability and throughput in virtual-time CSMA," *Comput. Networks ISDN Syst.*, vol. 10, no. 1, pp. 19–26, Aug. 1985.

[16] R. M. Metcalfe and D. R. Boggs, "Ethernet: Distributed packet switching for local computer networks," *Commun. ACM,* vol. 19, no. 7, July 1976.

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

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

[19] M. L. Molle and M.-W. Wu, "Prioritized virtual-time CSMA: Head-of-the line priorities without channel overhead," in *IEEE Int. Conf. Commun.*, June 1985, pp. 35.4.1–35.4.6.

[20] M. L. Molle and L. Kleinrock, "virtual-time CSMA: Why two clocks are better than one," *IEEE Trans. Commun.*, vol. COM-33, Sept. 1985.

[21] M. L. Molle, K. Sohraby, and A. N. Venetsanopoulos, "Space-time models of asynchronous CSMA protocols for local area networks," *IEEE J. Select. Areas Commun.*, vol. SAC-5, pp. 956–968, July 1987.

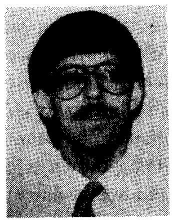
[22] M. L. Molle, "Analysis of a class of distributed queues with applications," *Perform. Eval.*, vol. 9, no. 4, pp. 271–286, Aug. 1989.

[23] I. G. Niemegeers and C. A. Vissers, "Twentenet: A LAN with message priority, design performance considerations," *ACM SIGCOMM'84 Symp. Commun. Architect. Protocols,* June 1984.

[24] G. C. Polyzos and M. L. Molle, "Delay analysis of a window tree conflict resolution algorithm in a local area network environment," in *Proc. ACM SIGMETRICS'87,* May 1987.

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

[26] ——, "Carrier sense multiple access with message-based priority functions," *IEEE Trans. Commun.*, vol. COM-30, pp. 185–200, Jan. 1982.



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 at Kingston, Canada in 1976, and the M.S. and Ph.D. degrees in computer science from the University of California, Los Angeles in 1978 and 1981, respectively.

Since September 1981 he has been on the faculty at the University of Toronto, where he is now a Professor in the Department of Computer Science, with a cross-appointment in the Department of Electrical Engineering, and is a member of the computer systems research institute. In 1987–1988 he was a Visiting Associate Professor at the University of California, Irvine. His research interests include the modelling and analysis of protocols for computer networks and distributed systems. He has published extensively on the performance evaluation of various local area network access protocols and distributed conflict resolution algorithms, and won the best paper award at ICC 1986.

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