PERFORMANCE OF SINGLE ACCESS CLASSES ON THE IEEE 802.4 TOKEN BUS

M. Alex Colvin, M.S.
Alfred C. Weaver, Ph.D.

Computer Science Report No. TR-85-21
September 18, 1985

"This paper has been submitted for publication to the IEEE *Journal on Selected Areas in Communications*."

PERFORMANCE OF SINGLE ACCESS CLASSES ON THE IEEE 802.4 TOKEN BUS

M. Alex Colvin, M.S.
Alfred C. Weaver, Ph.D.
Department of Computer Science
University of Virginia

ABSTRACT

The IEEE 802.4 token bus defines both *synchronous* and *asynchronous* message access classes. Their performance is compared for networks implementing a single message class. Throughput bounds are derived from the access class timer. It is shown that for some configurations the asynchronous class yields lower *observable* message delays. Mean observable delay is minimized by allowing each *MAC* unrestricted service.

1. The 802.4 Token Bus

In June 1984 the IEEE adopted Standard 802.4 [1] for local area networks using a token-passing bus access method. This standard defines all of the physical layer and part of the data link layer of the ISO OSI model. [3] The standard specifies: (1) the electrical and physical characteristics of the transmission medium; (2) the electrical signaling method used; (3) the frame formats transmitted; (4) the actions of a station upon receipt of a data frame; and (5) the services provided by the Medium Access Control sublayer of the data link layer.

In general, a *token* controls access to the physical medium; in a sense, the token holder is momentarily the master station of the network. Possession of the token allows a station to transmit messages from one or more priority classes, called *access_classes*, using a protocol which we explain in section 2. When a station has transmitted all its messages, or when certain time limits have expired, the token is passed to a known successor. The orderly progression of the token from station to station thus forms a logical ring on a physical bus.

A station's interface to the network is achieved through a Medium Access Controller (MAC). As shown in Figure 1, the MAC acts as an interface between the logical link control sublayer above it and the physical layer beneath it in the OSI model.

The MAC implements the token-passing protocol, including (1) token recognition, passing, and regeneration after loss; (2) message encapsulation and framing; (3) service of the four priorities (*access_classes*) of messages; and (4) error control and recovery.

Although the standard defines a number of techniques for error recovery and network reconfiguration, we have ignored them. The protocol treated here assumes error-free transmission of data and tokens.

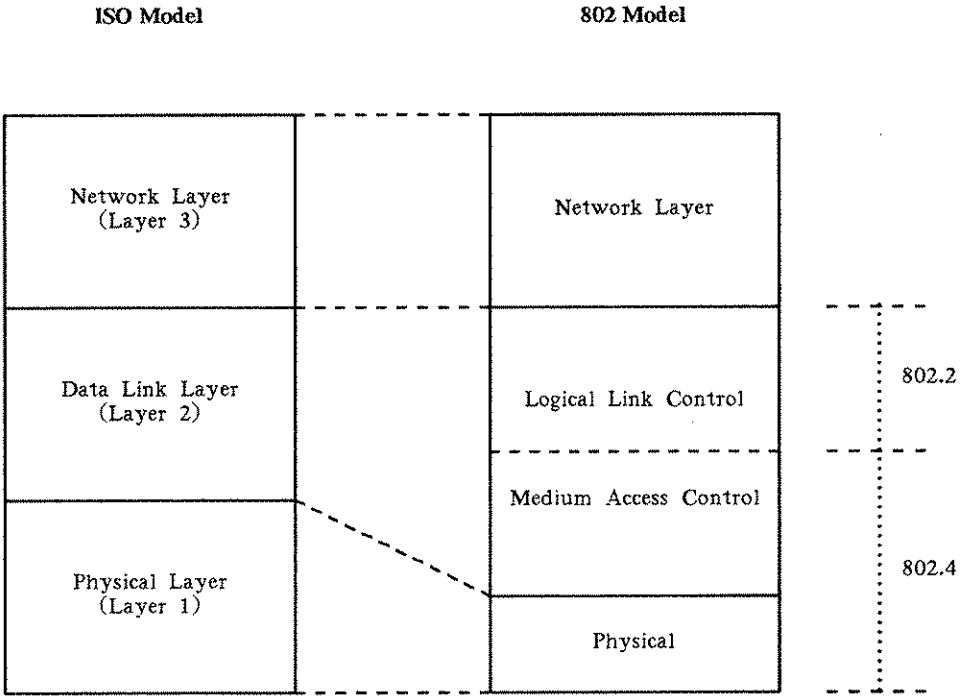

Figure 1. Relation of the IEEE Standards to the ISO OSI Model

2. *MAC* Service

A message arriving for transmission at a *MAC* is queued until the token arrives. When a *MAC* receives the token it may transmit all the messages in its queues (subject to time bounds discussed below), after which the token is transmitted to the next station. The token rotates around the logical ring before returning.

As the network traffic increases, the queues at individual *MAC*s grow longer, and more messages are transmitted between token passes. The token transmissions become a smaller fraction of network bandwidth, and the network becomes more efficient.

2.1. Access Classes

The Standard defines eight message service classes to be used to enforce message service priorities. A *MAC* maps these onto as many as four message access classes. The access class is an attribute of the message, not of the *MAC*. The *MAC* has a separate message queue for each supported access class. When the token is received, service begins for the highest priority class and proceeds from there to the lowest priority class. The token is passed when all classes have been served.

Service is not preemptive. Once a *MAC* has begun service for a class, arriving higher priority messages must wait for the token to pass around the logical ring. The access classes take effect by restricting the time available for message transmissions when a station receives the token. A station loads a "token hold timer" when beginning service for a class. The timer is checked before each message transmission. If it has expired, service for the class is terminated. Because the timer is not checked during message transmission, it is possible to overrun this limit by one message time.

The highest priority class is called *synchronous*. The token hold timer for synchronous service is loaded with the constant *hi_pri_token_hold_time*.

There are three *asynchronous* classes. In descending priority they are *urgent_asynchronous*, *normal_asynchronous*, and *time_available*. There is a constant *target_rotation_time* for each asynchronous class. For asynchronous service the token hold timer is loaded with the residue from another timer, the *token_rotation_timer* for that class. The *token_rotation_timer* is then reset to the class *target_rotation_time*. The *token_rotation_timer* times one complete token rotation before its residue is copied into the token hold timer.

Synchronous servers are guaranteed service each time the token is received, but there is a fixed limit on the amount of such service. If the traffic is light at some stations, heavily loaded stations may not increase their service. Asynchronous servers have no guarantee of service when the token is received. However, the amount of service can adapt to the network load. Time not used by lightly loaded stations becomes available to those more heavily loaded.

The synchronous class timer is applied to each *MAC* individually. The asynchronous class timer is applied to the network as a whole, as a single large distributed queue. This makes the asynchronous timer less sensitive to variations in load at an individual *MAC*.

2.2. Delay

One of the principal metrics of network performance is delay. The delay encountered by a message is the time elapsed from message creation at the source until its reception at the destination. *Queueing delay* is the delay as the message waits to reach the head of the access class queue. *Access delay* is the wait for token arrival. *Transmission delay* is the time spent transmitting the message, including any overhead, propagation delay, and switching delay.

Use of token hold timers does not place a bound on total delay. On the contrary, mean delay is increased, as some messages may be required to wait for several token rotations before transmission.

3. Bounds

The following discussion applies to an error-free network of N identical *MAC*s, each with a mean utilization of u, for a total utilization of $U = Nu$. The token transmission time is X_{token}, including address, framing, propagation, and

switching delays. Constant length messages arrive at each *MAC* from identical Poisson processes. Each requires time X for transmission, including address, framing, propagation, and switching delays.

All messages in the network belong to a single message class. Each *MAC*, therefore, serves a single queue.

3.1. Token Rotation

The token rotation time is the time from reception of the token, at some arbitrarily selected server, to its subsequent reception at the same server, after passing around the logical ring.

At any time, either a message or a token is being transmitted. Let U be the network utilization, the fraction of the bandwidth consumed by messages; then $1-U$ is the fraction consumed by tokens. Let $\overline{C}$ be the mean token rotation time. The mean time transmitting messages during one token rotation is $U\overline{C}$. The time passing the token during a token rotation is NX_{token}. Thus,

$$\overline{C} = U\overline{C}+NX_{\text{token}} = \frac{NX_{\text{token}}}{1-U} \quad . \tag{1}$$

This mean rotation time requires that the network be stable ($U < 1$) but does not depend on the token hold timers.

3.2. Access Classes

Even in a network with a single message class, the access class mechanism may be used to bound the token rotation time and, with it, the bus access delay.

For a network with synchronous traffic, the *hi_pri_token_hold_time* (T_{sync}) bounds the token rotation time C by

$$0 \leqslant C < C_{\max} = N(T_{\text{sync}}+X_{\text{token}}) \tag{2}$$

This restricts the maximum network utilization U.

$$0 \leqslant U < U_{max} = \frac{T_{sync}}{T_{sync}+X_{token}} \tag{3}$$

For an asynchronous network the *target_rotation_time* (T_{async}) limits the duration of one token rotation $\overline{C}$ followed by one service period $u\overline{C}$. The utilization and cycle time are bounded by $\overline{C}+u\overline{C} < T_{async}$. This limits the network utilization U to be

$$0 \leqslant U < U_{max} = N\left(\frac{T_{async}}{\overline{C}} - 1\right) = \frac{T_{async}-NX_{token}}{T_{async}+X_{token}} \tag{4}$$

and

$$T_{async} = X_{token}\frac{N+U_{max}}{1-U_{max}} \tag{5}$$

3.3. Delay

A recent paper by Fuhrmann[2] expresses the mean delay as the sum of half the cycle time, half the queueing delay, and the transmission delay. This applies when the token holding time is unlimited. The sum of the three terms is

$$\overline{D} = \frac{\overline{C}}{2} + \frac{XU}{2(1-U)} + X = \frac{NX_{token}+XU}{2(1-U)} + X \tag{6}$$

The mean token rotation time is unaffected by the token hold timers. Mean delay is inceased by reducing the timers. When finite token hold times are used, messages may have to wait several token rotations to gain access to the bus, increasing the delay.

4. Simulations

The figures show the results of computer simulations. In the simulated configuration messages arrive at 64 *MAC*s from identical Poisson processes. Each message transmission requires a constant X = 306 bit times. This time includes

data, address, framing, propagation, and switching delays. Token passing requires X_{token} = 146 bit times (.477 X). This corresponds to a 96 bit token with 50 bit times of propagation and switching delay.

4.1. Mean Delay

Figure 2 shows the mean delay ($\overline{D}$) for messages. This includes access delay, queuing delay, and transmission time. Delay is normalized to message transmission time (X), and plotted on a logarithmic scale.

The simulated load was varied from 0 to U_{max}. In these simulations there was no limit on token hold times, hence U_{max} = 1.

The curve shows the delay described in (6) above. Circles indicate mean delay measured from simulations. The data points show good agreement with the behavior predicted by equation (6).

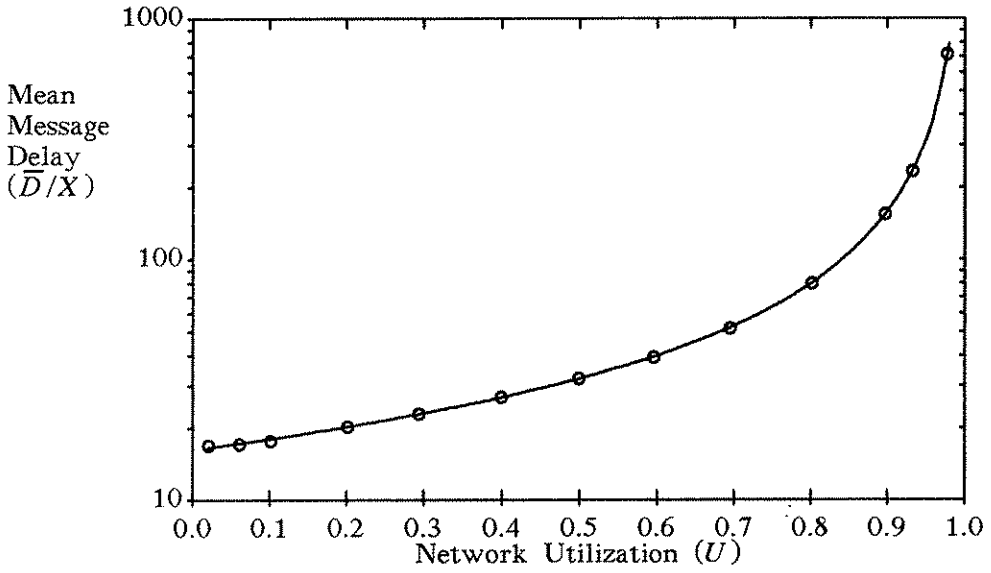

Figure 2. Mean Delay vs. Utilization

4.2. Access Classes

To examine the difference between synchronous and asynchronous access classes, simulations were done with both types of traffic using three different values of U_{max}. For the synchronous-only networks the *hi_pri_token_hold—time*, T_{sync} = 1, 2, and 4 packet times (X).

The asynchronous *target_rotation_time* (T_{async}) is not directly comparable to the synchronous timer T_{sync}. For comparison, the asynchronous networks were simulated with T_{async} chosen to yield the same U_{max} as the synchronous networks, according to equation (5) above.

T_{sync}	T_{async}	U_{max}
$1X$	$95.5X$	0.677
$2X$	$160.5X$	0.807
$4X$	$290.5X$	0.893

Figures 3a, 3b, and 3c show the results of simulations of synchronous and asynchronous traffic for the three U_{max}. Each configuration is simulated with loads ranging from 0 to the corresponding U_{max}. The data points representing simulations are marked with a symbol. Vertical lines show the different U_{max}.

The mean delay is everywhere greater for the synchronous-only networks than for the asynchronous-only networks. The asynchronous delay is unaffected by the token hold timers until the load approaches U_{max}.

The least delay (as well as the greatest maximum utilization) is shown by the network with unlimited token hold times.

The mean token rotation time $\overline{C}$ has the same curve for all configurations, namely that described in equation (1) above. In all these configurations the mean delay $\overline{D}$ is increased over that of equation (6).

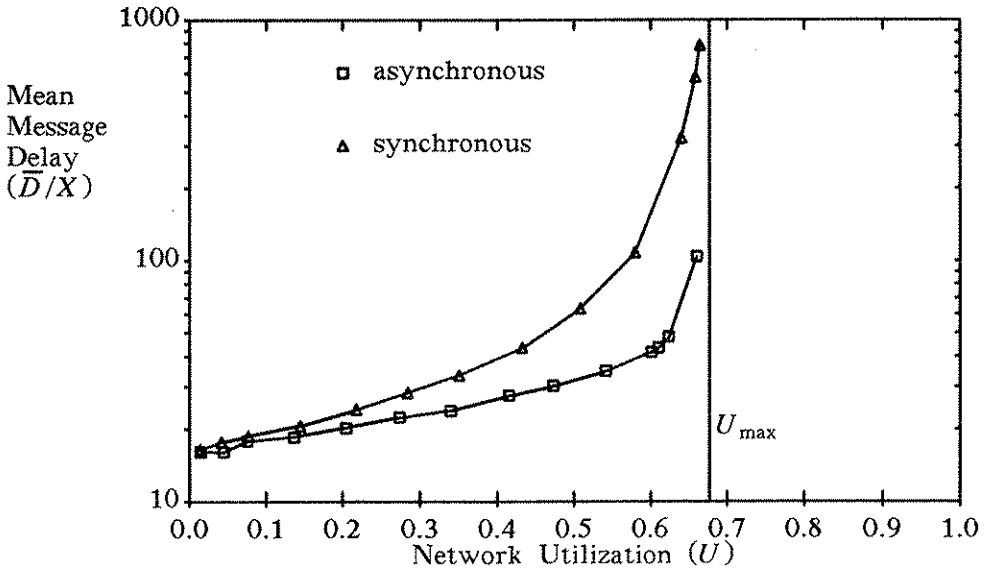

Figure 3a. Mean Delay vs. Utilization with U_{max} = .677

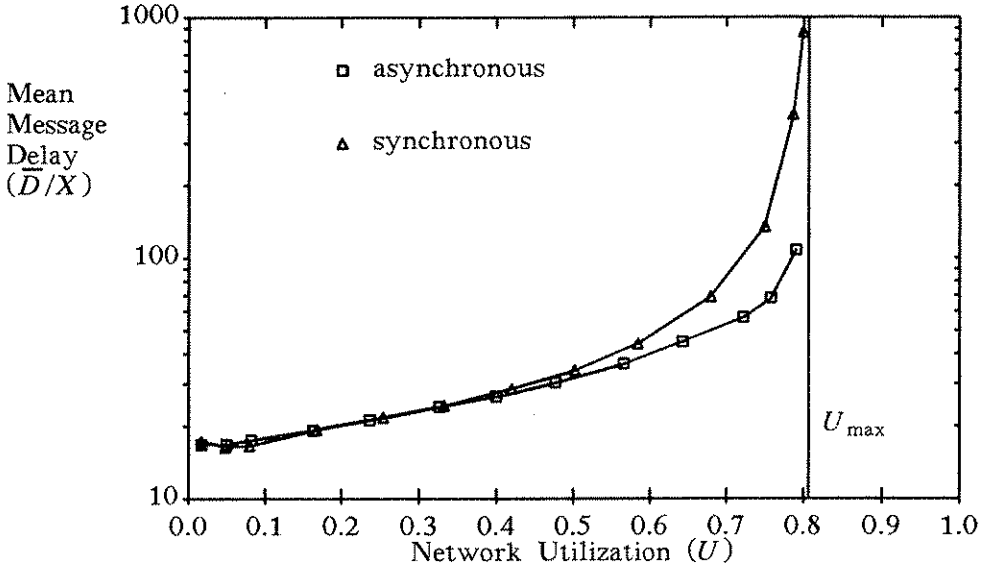

Figure 3b. Mean Delay vs. Utilization with U_{max} = .807

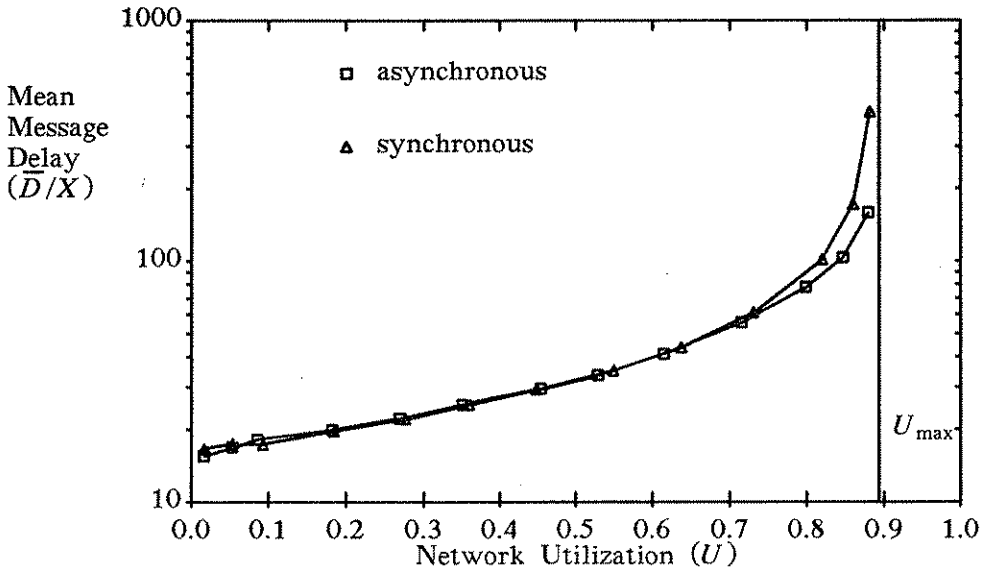

Figure 3c. Mean Delay vs. Utilization with U_{max} = .893

5. Discussion

The standard states that if only one access class is to be implemented it must be synchronous. For some applications the lower mean delays of a single asynchronous class may be preferable.

The token hold times are particularly well suited to a polling network, where a station generates messages only when a token is received. This, however, does not correspond to the separation of logical link and medium access layers described in the Standard.

Restricting token holding times may improve bounds on access delay. However, when the message arrivals are independent of the network servers (as they would be when the *MAC*s are self-contained units), the *observable* measure is the total delay. The minimum mean delay is achieved by avoiding the priority classes altogether.

REFERENCES

1. The Institute of Electrical and Electronics Engineers, Inc., in *IEEE Standard 802.4 — Token-Passing Bus Access Method and Physical Layer Specifications*, IEEE (1985).

2. S. W. Fuhrmann, "A Note on the M/G/1 Queue with Server Vacations," *Operations Research*, **32**(6), p. 1368 (November-December 1984).

3. H. Zimmerman, "OSI Reference Model - the ISO model of architecture for open systems interconnection," *IEEE Trans. on Communications* (April 1980).