PRIORITIES AND DYNAMIC RING MEMBERSHIP IN AN 802.4 NETWORK

Mangala Gorur

Computer Science Report No. RM-86-05
July 1986

PRIORITIES AND DYNAMIC RING MEMBERSHIP IN AN 802.4 NETWORK

A Thesis

Presented to

the Faculty of the School of Engineering and Applied Science

University of Virginia

In Partial Fulfillment

of the Requirements for the Degree

Master of Science (Computer Science)

by

Mangala Gorur

January 1987

ABSTRACT

Local area networks using the token bus technology are anticipated to meet the needs for office, factory and process control automation applications. The IEEE 802.4 token bus protocol is emerging as the popular standard for these applications and is the only token bus standard that offers broadband facilities. The enormous interest in this protocol is due to the following features: high reliability, prioritized transmissions, deterministic access of the medium and robust error detection and recovery.

This thesis studies the issues of prioritized traffic, dynamic ring membership and transient loading conditions in a token bus network by analytic and simulation modeling. An analytic model has been developed to predict the timer settings that ensure the implementation of the priority feature as specified by the IEEE 802.4 standard. This is very useful in determining the range of network throughput over which optimum service can be obtained for messages of a certain priority. The issue of dynamic ring membership of a station has been studied and an analytic model has been developed to predict the delivery times of messages. This study gave us better insight into the issues pertaining to the transient ring membership of a station in an 802.4 network, and the benefits of a station being a permanent member of the logical ring as opposed to being a transient member. Simulation studies have been conducted to observe the robustness of the protocol under transient loading conditions. Results of this study illustrate the robustness of the protocol under heavy transient loads.

ACKNOWLEDGEMENTS

Special thanks go to my advisor Dr. Alfred C. Weaver for the guidance and assistance he provided to me to make this research possible. I am sincerely grateful to him for the encouragement he gave and for all the valuable suggestions he made during this research.

I would like to thank Catherine F. Summers for the use of her simulator during this research. Thanks also to Mark J. Smith for his help in maintaining the Apollo network, and my past and present office mates for helping me in the formatting of this thesis.

I am thankful to my parents for the opportunities they afforded me.

This research was funded by NASA-Lewis Research Center and the Virginia Center for Innovative Technology. I would like to express my thanks to them for funding this research.

Table of Contents

List of Figures

CHAPTER 1

INTRODUCTION

1.1. Local area networks

Local area networks are becoming increasingly popular as a communications mechanism to interconnect a large number of computers in a small geographic area. "Local area network" refers to a general purpose network interconnecting a variety of devices like minicomputers, terminals and other peripherals. The advantages of interconnecting computers are increased reliability, availability of more computer power, efficient distribution of computational loads, and resource sharing. Some of the disadvantages of local area networks are that security and privacy are not guaranteed and data integrity is also not assured. One of the major issues in the design of local area networks is the design of its medium access control protocol.

1.2. IEEE 802 Committee

The IEEE 802 committee has proposed a set of six standards for local area networks. The standards are in the form of a 3-layer communications architecture encompassing the functionality of the physical and data link layers as defined in the ISO OSI model [Zimmermann 80]. Figure 1.1 illustrates the mapping of the physical and data link layers as defined in the ISO OSI model to the layers as defined by the IEEE 802 standards.

The document IEEE 802.1 describes the relationship among the six standards and relates them to the ISO OSI reference model. The standard IEEE 802.2 [IEEE 1985a] gives a description of the protocols at the logical link control (LLC) layer. The logical link control layer corresponds to the top of the OSI data link layer.

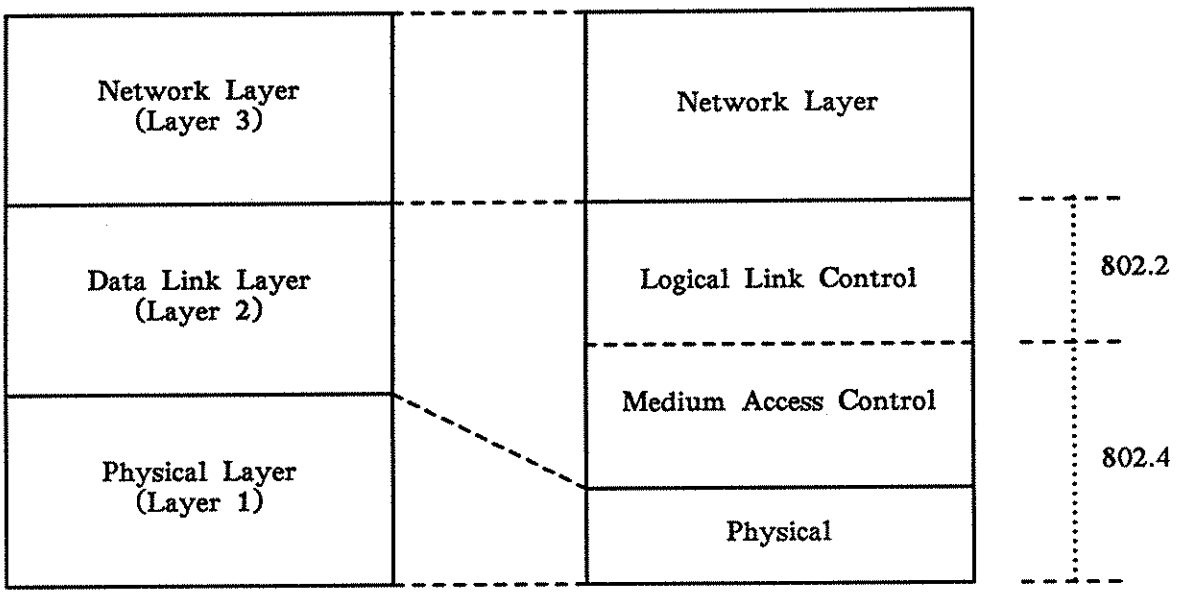


Figure 1.1 Relation of the IEEE standards to the ISO OSI Model

The standards IEEE 802.3 [IEEE 1985b], IEEE 802.4 [IEEE 1985c], and IEEE 802.5 [IEEE 1985d] define 3 types of medium access control and associated physical media.

1) IEEE 802.3: Bus topology utilizing CSMA/CD as the medium access method.

2) IEEE 802.4: Bus topology utilizing explicit token passing as the medium access method.

3) IEEE 802.5: Ring topology utilizing explicit token passing as the medium access method.

The medium access control sub layer (MAC) extends the OSI data link layer to provide additional services at the data link layer. The IEEE 802.6 describes a metropolitan area network. Figure 1.2 illustrates the relationships among the

different standards.

1.3. IEEE 802.4 token bus

The IEEE 802.4 is a standard for local area networks (LANs) with bus topology and an explicit token passing scheme for controlling access to the transmission medium. Every station in the network may receive all signals transmitted on the medium. As the token is passed from station to station, a logical ring is formed. The right to access the medium passes from station to station. Every station which is a member of the logical ring is permitted to transmit messages for a limited amount of time. Each station is permitted to transmit messages when it holds the token. At a bit rate of 1 Mbps, the token in an 802.4 network is an explicit message of at least 96 bits for networks using 16 bit addresses; it is at least 160 bits for stations using 48 bit addresses. The amount of time a station can transmit is

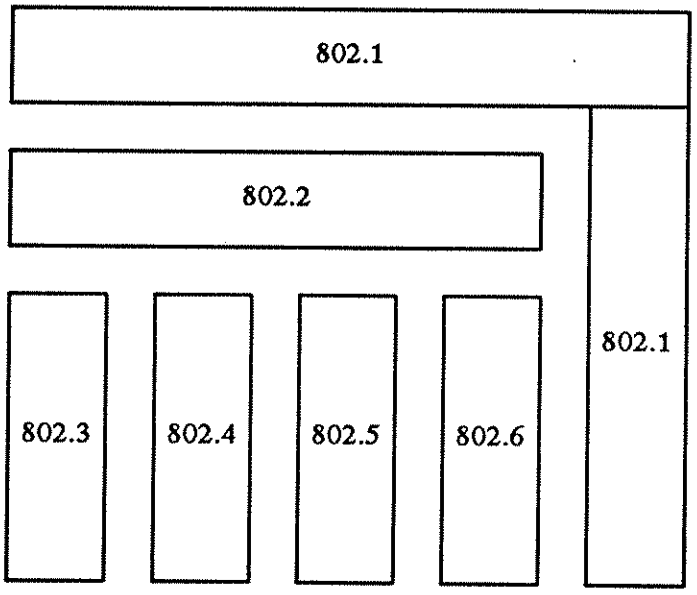


Figure 1.2 802.x Relationships

limited by suitably setting a series of timers. In an 802.4 network, stations can transmit messages at 4 different priorities or access_classes. By suitably setting the timers associated with each access_class, the available bandwidth can be allocated to messages from different access_classes. Stations are connected in the logical ring in descending order of addresses, hence the token is passed from station to station in descending order of addresses. This type of access control method coordinates the stations' transmissions so that collisions are minimized. It also guarantees fairness to a limited extent. It is also intended to be a robust protocol so that it can tolerate and survive multiple concurrent errors. Chapter 2 discusses the features of the protocol at the medium access control (MAC) sublayer.

1.4. Focus of the thesis

This thesis describes the study and performance analysis of specific characteristics of the IEEE 802.4 protocol for local area networks. The performance and sensitivity of the protocol have been studied with regard to the following characteristics of the protocol.

1.4.1. Access_classes and priority of service

The 802.4 token passing access method offers four levels of service called the access_classes, hence messages can be transmitted at any of the four priorities. The four access_classes in descending order of priority are:

1) Synchronous access_class

2) Urgent_Asynchronous access_class

3) Normal_Asynchronous access_class

4) Time_Available access_class

By suitably setting the variables associated with the implementation of the priority

scheme, it can be ensured that this scheme gives preference to frames of higher priority. The implementation of the priority scheme by any station is optional. The priority scheme when implemented by a station allocates network bandwidth to lower priority traffic only when there is time available and there is no higher priority traffic.

The main intent of studying this aspect of the protocol is to be able to predict the timer settings associated with each access_class and the range of network throughput over which optimum service can be obtained at that access_class. The timer setting at each access_class limits the token cycle time at each access_class. An approximate analytic expression for the token cycle time has been developed. The analytic expression can be used to calculate the values of timer settings that will ensure implementation of the priority scheme as specified by the IEEE 802.4 standard. The details of the priority scheme as specified by the IEEE standard are discussed in chapter 3. This is followed by a discussion of the mathematical model and an illustration of the use of the analytic expression to calculate the timer settings. The values of timer settings to obtain optimum service at an access class have been calculated. Network configurations with the timers set to the values calculated from the above expression have been simulated. The results of the simulation have been related to the expected values generated by the analytic model.

1.4.2. Dynamic ring membership

An 802.4 network usually consists of two or more stations that are members of the logical ring. Stations need not be permanent members of the ring. Stations can join and leave the ring dynamically, as dictated by their traffic or station management decisions. Each station can join the logical ring through a controlled contention process. The waiting time for a station to be admitted into the logical

ring varies depending on the number of stations waiting to join the ring and on the settings of several variables. Stations can leave the logical ring at any time by ignoring the token on the next token rotation or by informing the preceding station of its new successor.

The intent of studying this aspect of the protocol is to know the effect of overhead due to ring reconfigurations on the delivery times of messages. The logical ring needs to be patched each time a station leaves the token passing ring. Ideally the stations should experience minimum delay while joining the ring. The overhead due to transient ring membership of stations also affects the total throughput of the network.

The role played by different parameters in deciding the time of entry into the ring or exit from the ring has been studied. Network configurations with suitable parameter settings have been simulated to observe the role played by the relevant parameters in allowing the stations to join the ring. Chapter 4 discusses a mathematical model which predicts the average delivery time of messages from a station which is a transient member. Simulation results have been included to show the effect of transient ring membership on the delivery times of messages.

1.4.3. Transient loading

All the stations in the 802.4 network will not offer uniform loading on the network at any time. Occasionally, a station might have a large amount of data to be transmitted during a token cycle. This may cause the total offered load to exceed the bus capacity (i.e., overloading). The transmission of such a load causes an increase in the token cycle time. Succeeding token cycles will also be longer. If the token cycle time does not recover from the imposition of such a transient load, then queue lengths could grow very large at every imposition of a transient load.

The main intent is to study the robustness of the 802.4 protocol under such loading conditions. The effect of transient loading on an 802.4 network has been studied by imposing large loads at varying intervals on the network. The impact of two forms of transient loading, numerous short messages vs. fewer longer messages, has been studied. The variation of token cycle time and average delivery time of messages (with respect to the offered load) under such transient loading conditions has been studied. Chapter 5 discusses the effect of transient loading on an 802.4 network. Suitable network configurations have been simulated to observe the effect of imposing such loads on an 802.4 network. The last chapter summarizes the results obtained from this study.

CHAPTER 2

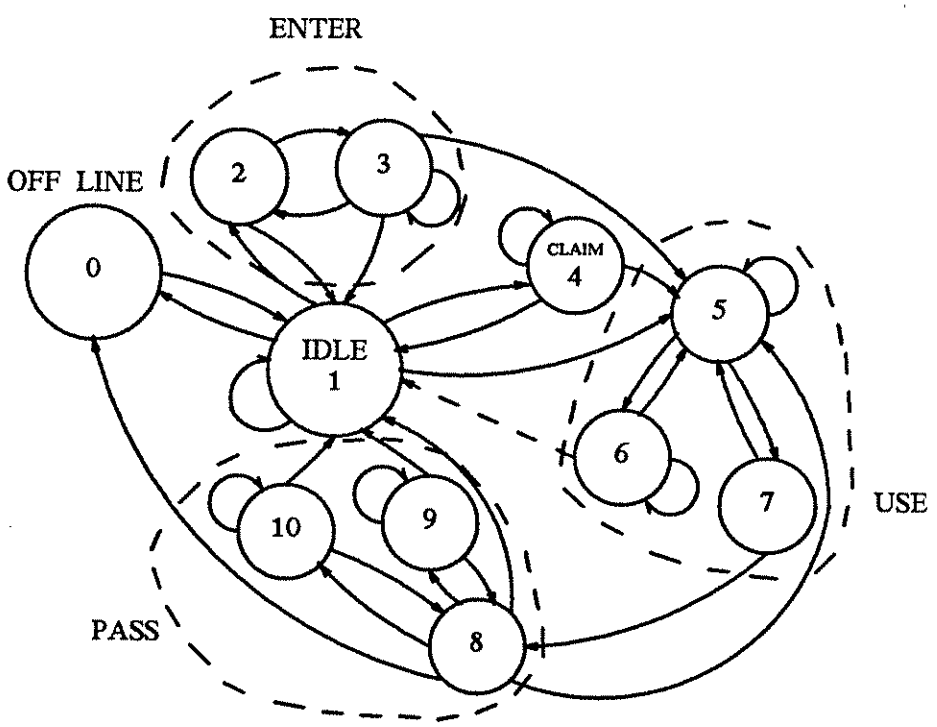
FEATURES OF THE MAC

This chapter discusses the features of the IEEE 802.4 protocol at the medium access control sublayer. The relationship of this sublayer to other layers as specified by the ISO OSI model and as specified in the IEEE 802.4 standard has been discussed in the first chapter.

The MAC sublayer provides sequential access to the shared medium by passing control of the transmission medium from station to station in a logically circular fashion. The protocol at the MAC sublayer at each station manages an ordered access to the medium by recognizing and accepting the *token* frame (refer to section 4 of the IEEE 802.4 standard for frame formats). It also determines the duration each station can hold the token before it is passed to the successor station. It also handles transient ring membership of a station, i.e., a station can join and leave the ring as dictated by its traffic. The MAC protocol can also handle faults caused by communication errors or station failures. This MAC protocol is intended to be a robust protocol so that it can recover from multiple errors.

2.1. States of the MAC

The MAC logic in a station has been described in the IEEE 802.4 standard as a finite state machine with 11 distinct states. The states and the sequence of transitions are illustrated in Figure 2.1. The sequence of transitions and the function of each MAC state are discussed in the forthcoming sections.



0 - OFFLINE
1 - IDLE
2 - DEMAND_IN
3 - DEMAND_DELAY
4 - CLAIM_TOKEN
5 - USE_TOKEN
6 - AWAIT_IFM_RESPONSE
7 - CHECK_ACCESS_CLASS
8 - PASS_TOKEN
9 - CHECK_TOKEN_PASS
10 - AWAIT_RESPONSE

Figure 2.1 MAC Finite State Machine Diagram

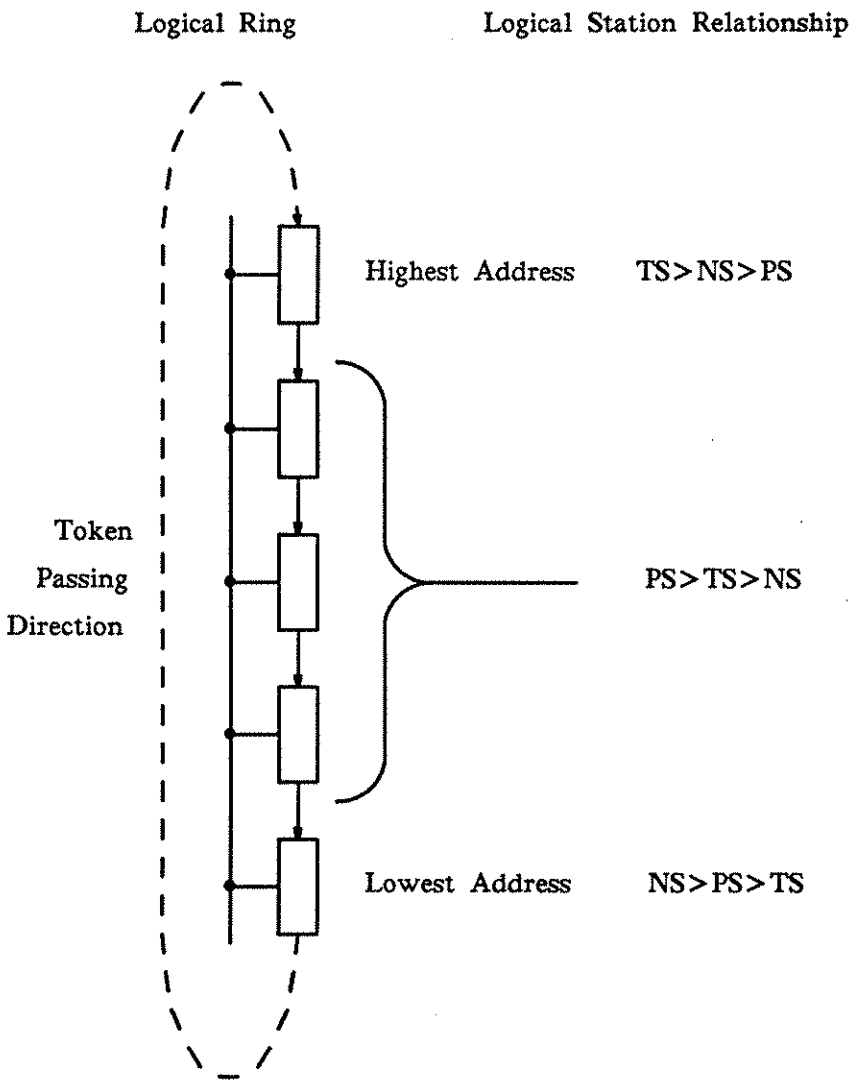
2.2. Logical ring of stations

As the token gets passed from station to station, a logical ring of stations is formed. Each station in the network may receive all signals transmitted on the

medium. But a station has to be a member of the logical ring to be able to access the transmission medium. The right to access the medium, i.e., the token frame, passes from station to station in descending order of addresses. But the station with the lowest address passes the token to the station with the highest address to close the logical ring. This is illustrated in Figure 2.2.

2.3. Logical ring initialization

Initialization of the logical ring is a process similar to adding new stations to the ring. The standard specifies at least two stations in the ring at any time. Associated with each station is a "bus_idle_timer". This timer measures the duration of lack of activity on the bus. If the station detects lack of activity for a duration longer than the expiration of the above timer, then that station enters the **claim_token** state. In this state the station sends a *claim_token* frame. As there can be more than one station trying to initialize the logical ring, the initialization algorithm resolves such multiple claims to initialize the logical ring. This is done by address sorting the initializers. The information field length of this frame varies depending on the address of the initializing station. Hence the length of the *claim_token* frame sent by each station varies depending on its address. The length of the *claim_token* frame is calculated by indexing through the station's address and using two bits at a time (refer to section 6.5.1 of the standard). Depending on the address bits used for sorting the information field, the frame length can be 0, 2, 4 or 6 times the slot_time (refer to section 6.1.9 of the standard for a definition of the slot_time). After transmission of this frame each station waits for one slot_time and then samples the state of the medium. If the station senses non_silence, then some other station with a longer transmission is trying to initialize the ring. Hence the station defers to those stations with numerically higher addresses and remains **idle**. If silence was detected, then another *claim_token* frame is sent using the next



TS - This station's address
NS - Next station's address
PS - Previous station's address

Figure 2.2 Logical Token_Passing Ring

two address bits. If all bits have been used and silence is still detected, then the station has won the initialization process and is the token holder. Hence there is a

unique token in the network. The token holder then enters the **use_token** state. On entering this state the station starts the "token_hold_timer" which limits the amount of time a station can transmit data frames in this state and transmits any enqueued data frames. A station can transmit data frames in this state either until the expiration of the "token_hold_timer" or until the queues are emptied. After transmitting data frames in this state the station adds more stations into the ring by a controlled contention process discussed in section 2.4.

2.4. Resolution process

2.4.1. Resolution of multiple requests for logical ring membership

Stations are added to the logical ring by a controlled contention process using "response_windows". A response_window is an interval of time during which a station which has sent a frame pauses and waits for a response. After transmitting any data frames in the **use_token** state, the station checks the value of "inter_solicit_count". This count at each station determines when response windows are opened at each station. If it is zero then response windows are opened to solicit successors into the ring. If the value is not zero, then the count is decremented and the token is passed to the successor. The MAC control frames used to add new stations into the ring, i.e., to solicit successors into the ring, are *solicit_successor* frames. The *solicit_successor* frame opens response windows and specifies a range of addresses for stations which are interested in contending for membership. Stations whose addresses fall in this range and which desire to become ring members can respond to this call by entering the **demand_in** state and sending a request for membership by transmitting a *set_successor* frame. A station can send repeated requests for ring membership if its address falls within the range specified by the token holder. If there is no response during the interval in which the response window is open, then the token holder resets the "inter_solicit_count" and

passes the token to its successor. If a single station responds, then the token holder makes the responding station its new successor and passes the token to its new successor. If multiple stations respond to a *solicit_successor* frame the desired result is for the token holding station to pass the token to the highest addressed contending station. The contention for the token by multiple stations is resolved by having the stations wait in the **demand_in** state for different time intervals before sending the next *set_successor* frame. Each station waits for an interval of time decided by the one's complement of the station's address. Thus stations with higher numerical addresses wait for a shorter interval of time before requesting membership again. After sending its request, each station waits for a response in the **Demand_delay** state. In this state a station can expect to hear one of the following frames:

a) a token from the token_holder indicating that its request for membership has been granted.

b) a *resolve_contention* frame indicating that the contention for the token has not been resolved and hence the stations have to contend for membership again.

c) requests for ring membership from other stations which the station can ignore.

d) any other frame.

If the station hears (a) then it switches to the **use_token** state. If it hears (b), then it enters another delay interval in the **demand_in** state before sending the next *set_successor* frame. If no other contenders have been heard by the time the delay interval expires then another *set_successor* frame is transmitted. If the station hears any transmissions in this delay interval then it assumes that there is another station with a higher numbered address, and hence drops out of the contention process and returns to the **idle** state. If the station hears any frame other than (a),

(b) or (c) then it assumes that another station with a higher numbered address is contending for the token and hence drops out of the contention process. Thus the station with the highest numbered address wins the token. Each time a multiple response is heard for a *solicit_successor* frame the call for stations to contend for membership is repeated. The contention resolution process should resolve so that the contending station with the highest address wins the token. However there is one special case which results when two or more stations have been erroneously assigned the same address; they will sequence through the same states and remain unresolved. In this case, each station makes one more attempt by generating a two-bit random number (i.e., an extension of the 16-bit or 48-bit address space). If there is a clear winner, it becomes the token holder. Otherwise, all stations involved abandon this process and await the next *solicit_successor* frame.

2.4.2. Resolution of multiple token claims

If the bus_idle_timer expires and there was no activity on the medium for the entire duration then the station enters the **claim_token** state. In this state the station attempts to initialize (which has been explained earlier) or reinitialize the logical ring by claiming the token. Multiple stations may enter this state and claim the token. Multiple claims are resolved by a contention process. The length of the *claim_token* frame depends on the address of the station and can be 0, 2, 4 or 6 times the slot_time. Thus stations with numerically higher addresses send longer *claim_token* frames. After sending the *claim_token* frame each station waits for one slot_time listening for other transmissions. If the station hears any transmissions at or before the expiration of this time interval then it assumes that a station with a numerically higher address is contending for the token and hence drops out of the contention process and remains idle. If there are no transmissions at the end of the slot_time the station sends an additional *claim_token* frame and repeats the delay

and transmission check. If the station has sent "max_pass_count" number (an integer equal to half the station's address length in bits plus one) of *claim_token* frames without hearing other transmissions then the station has successfully claimed the token. The station then goes to the **use_token** state. Thus the resolution process results in the numerically highest addressed station winning the token.

2.5. Protocol management overhead

The logical ring maintenance functions discussed in the previous sections: the initialization or reinitialization of the logical ring, the claim token process, resolution process in the event of multiple token claims, the opening of response windows to solicit successors into the ring, and the resolution process in the case of multiple contenders, all of the above mentioned functions contribute to the network overhead. As described earlier each process involves the transmission of many MAC control frames which consume a certain fraction of the total network bandwidth, hence resulting in a protocol overhead.

2.6. Priority service

As mentioned in the first chapter, the MAC layer offers four priorities of service or access_classes. The access_classes have been identified as class six (the Synchronous access_class), class four (the Urgent_Asynchronus access_class), class two (the Normal_Asynchronous access_class) and class zero (the Time_Available access_class) in descending order of priority. The implementation of the priority scheme by any station is optional. If the priority scheme is not implemented then all of that station's frames are transmitted at highest priority. The states associated with the priority mechanism, and the variables and timers associated with the implementation of the priority scheme are discussed in detail in chapter 3.

2.7. Error recovery

2.7.1. Lost tokens

If the token holding station fails before passing the token then another station will recover the token. The bus_idle_timer at each station is used for this purpose. Lack of activity on the bus is allowed for a duration up to the expiration of this timer. Beyond this time interval, if the bus is still idle, then the stations whose timers have expired start the claim token process. The lowest addressed station on the ring sets its bus_idle_timer to a smaller value than other stations. Hence this station starts the claim_token process earlier. This is to minimize interference during the start of the recovery process of the lost token.

2.7.2. Token pass failure

After passing the token the station enters the **check_token_pass** state and waits for a response that will indicate that its successor has received the token and is active. If a valid frame is heard on the medium, then the station which transmitted the token assumes that the token pass has been successful. Otherwise it assumes that the token pass has failed and transmits the token again. If the token pass fails again, then the token holding station assumes that its successor station has failed. The sender then sends an MAC control frame, the *who_follows* frame, with its successor's address in the data field of the frame. This is to find the station next to its successor so that the ring can be patched across its successor. If there is no response for the first *who_follows* frame then it is transmitted for the second time. If there is still no response then the token holding station transmits a *solicit_successor* frame addressed to itself so that any station can respond to this frame. Any operational station can respond to this request if it needs to be part of the logical ring. Then the logical ring is reestablished by the process described in

section 2.3. If all these attempts fail, then the station assumes that either all stations have failed or all stations have quit the ring or the medium is broken or its own receiver has failed. In such situations the station stops attempting to maintain the ring and remains silent.

2.7.3. Deaf station

A station with an inoperative receiver can disrupt proper operation of the network because it cannot hear other transmissions on the medium. If such a station is in the **idle** state then, when its "bus_idle_timer" expires, it starts the claim_token process as it senses an idle bus. After claiming the token, it switches to **use_token** state and transmits its messages and will not be able to hear its successor claiming the token. It will solicit successors and will not hear any response. If there are any messages enqueued at this time it will switch back to **use_token** state and transmit the messages. If there are no messages, then it sets "sole_active_station" true. This is a Boolean variable used to mute stations that have faulty receivers. If this variable is true then a station is prevented from claiming the token unless it has data to send. This ensures that a station with a faulty receiver and with no data to transmit remains passive.

2.7.4. Duplicate addresses

A Boolean variable "just_had_token" is used to detect this error. This is set true when the station passes the token and reset when the station hears a valid frame from another station. If a station hears a valid frame with a source address equal to its own and "just_had_token" is false, then the station cannot have sent the frame, in which case there is another station with the same address. At this point the MAC sublayer informs the station management of such an error and enters the **off_line** state.

CHAPTER 3

ACCESS_CLASSES

This chapter discusses the study of the priority feature of the IEEE 802.4 protocol. The first section of this chapter gives the details of implementation of the priority mechanism, as specified by the standard. The role played by several variables and timers in the implementation of the priority feature is discussed in detail. An analytic expression for the token cycle time is developed for a network with traffic at different priorities. This is discussed in section 3.2 of this chapter. The analytic model has been used to calculate the timer settings which ensure the implementation of the priority feature. With known timer settings, the above expression has also been used to calculate the range of network throughput to obtain optimum service at each access_class. This is described in section 3.3 of this chapter. Network configurations have been simulated with timers set to values calculated from the analytic model. In the last section, the results of these simulations are discussed, followed by a comparison of the simulated results to the expected values from the analytic model.

3.1. Priority service

The IEEE 802.2 logical link layer protocol is intended to operate with any of the three MAC protocols: 802.3 CSMA/CD, 802.4 TOKEN BUS and 802.5 TOKEN RING. The MAC sublayer as specified by the IEEE 802 committee offers 8 levels of service to the LLC sublayer and higher level protocols. The LLC entity at each station can request transmission of a frame at any of the 8 levels by means of the service request primitive *MA - DATA.request*. This primitive is a request from the LLC entity to the MAC entity to transmit a data frame at the desired priority on

the local area network. The MAC sublayer transmits the LLC frame at the priority requested by the LLC entity.

The IEEE 802.5 protocol offers 8 classes of service for the LLC sublayer and the higher level protocols. The 802.4 token bus protocol offers only 4 classes of service for the LLC sublayer and the higher level protocols. These levels of service are called access_classes. The access_classes in descending order of priority are

1) Synchronous access_class - class 6

2) Urgent_Asynchronous access_class - class 4

3) Normal_Asynchronous access_class - class 2

4) Time_Available access_class - class 0

The LLC requests at priority levels 7 and 6 are mapped onto class 6; priorities 5 and 4 are mapped onto class 4; priorities 3 and 2 are mapped onto class 2; priorities 1 and 0 are mapped onto class 0.

3.1.1. Variables and timers

3.1.1.1. Station management variables

The following parameters are specified by station management software at each station. These parameters are used to implement the access_class structure and to limit the amount of time for which a station can transmit frames from any access_class.

1) High_Priority_Token_Hold_Time (HPTHT) :

This is a network wide parameter which decides the amount of time each station can transmit frames from the Synchronous access_class. If the priority option is not implemented, then this is the maximum amount of time a station can transmit messages before passing the token to its successor. This is an integer in

the range 0 to 2^{16} - 1 octet_times.

2) **Target_Rotation_Times (TRTs) :**

The station management software sets an upper bound on the token cycle time for each of the lower three access_classes. This is different for each access_class, and is the *Target_Rotation_Time* for each access_class. For each of the lower three access_classes, the station measures the amount of time it takes the token to circulate around the logical ring. If the token returns to a station in less than the *Target_Rotation_Time*, then the station can send messages of that access_class until the *Target_Rotation_Time* is reached. If the token returns after the *Target_Rotation_Time* is reached, then the station cannot send frames of that access_class on this pass of the token. The *Target_Rotation_Time* for each access_class can take on values in the range 0 to 2^{21} - 1 octet_times.

3) **Max_Ring_Maintenance_Rotation_Time :**

This variable serves two purposes. It determines the maximum token cycle time when used in conjunction with the priority scheme. The second function is related to the solicit successor procedure to admit new stations into the ring. If the token cycle time exceeds the *max_ring_maintenance_rotation time*, the station does not open response windows to solicit successors into the ring. The solicit successor procedure is deferred until the token cycle time is less than the *Max_Ring_Maintenance_Rotation_Time*.

3.1.1.2. Timers

The following timers are used to implement the priority scheme and limit the amount of time a station can transmit messages at any access_class. Each of these timers has a granularity of one octet transmission time.

1) **token_rotation_timers (trt) :**

Each station in the ring which implements the priority scheme provides a token_rotation_timer for each of the three lower access_classes. Each is a countdown timer. Each timer counts down to zero from a value equal to the TRT of the access_class with which it is associated. The timers stop counting when they reach zero, at which point their status is considered "expired". When a station considers transmitting messages at an access_class, the token_hold_timer is loaded with the residual value in the token_rotation_timer associated with that access_class and the token_rotation_timer is reloaded with the *Target_Rotation_Time* for that access_class. At network initialization, these priority timers are set to a value of zero.

If a timer associated with an access_class has expired when a station receives the token, then that station cannot send messages of that priority on this pass of the token.

2) **ring_maintenance timer** :

The ring_maintenance timer at each station measures the time it takes for the token to circulate around the logical ring. This timer, in conjunction with the max_inter_solicit_count, determines how often response windows are opened to solicit successors. Initially it is set to the value, *ring_maintenance_timer_initial_value*.

3) **token_hold_timer** :

This timer limits the amount of time a station can transmit messages at any access_class. This is a countdown timer. It stops counting when it reaches a value of zero, at which point, its status is considered to be "expired".

When a station receives the token, it enters the **use_token** state, loads this timer with the *High_Priority_Token_Hold_Time* and transmits frames at the Synchronous access_class. After transmission of frames at Synchronous access_class, if the priority feature is implemented, then the residual time from the current token_rotation_timer is loaded into the token_hold_timer. The station can send mes-

sages at an access_class until the expiration of the token_hold_timer.

3.1.2. Implementation of the priority feature

The implementation of the priority feature by any station is optional. Any station not implementing the priority feature will transmit all its frames at the highest priority. A station which does not use the priority feature can transmit frames at the highest priority, up to a maximum time set by station management software: the *High_Priority_Token_Hold_Time*. Every station is guaranteed this amount of time to transmit messages at this priority. When a station receives the token, it enters the **use_token** state and loads the token_hold_timer with the *High_Priority_Token_Hold_Time*. The station transmits frames from the Synchronous class until the queue is emptied or until the expiration of the token_hold_timer. When either of these two events occur, the station enters the **check_access_class** state. This MAC state controls the transmission of frames for different access_classes. In this state the station checks for enqueued messages at any of the lower three access_classes. It transmits the messages at each access_class if the timer associated with that access_class has not expired. After entering the **check_access_class** state, the station takes the following actions, if there are messages enqueued in the lower access_classes :

1) It decrements the current access_class.

2) It loads the token_hold_timer with the residual time in the token_rotation_timer. If the token_rotation_timer has expired, then frames of this access_class cannot be transmitted on this pass of the token. In such a case, the station enters the **check_access_class** state and repeats the above actions.

3) It reloads the token_rotation_timer with the value of the *Target_Rotation_Time* for that access_class.

4) It then returns to the **use_token** state. The station is allowed to transmit

frames of this access_class until the expiration of the token_hold_timer. When this timer expires, the station again enters the **check_access_class** state. Thus the station alternates between **use_token** and **check_access_class** states until the lowest access_class is served.

When implemented by a station, the priority scheme allocates network bandwidth to lower priority traffic only when there is no higher priority traffic and the token_hold_timer has not expired. By suitably setting the *Target_Rotation_Time* of each access_class, it can be ensured that this scheme gives preference to frames of higher priority. This implies that the transmission of Time_Available messages should not delay the transmission of Normal_Asynchronous messages and in turn the transmission of Normal_Asynchronous messages should be secondary to that of Urgent_Asynchronous messages. To achieve this, the *Target_Rotation_Times* should be set to values in the following order :

$$TRT_{ua} > TRT_{na} > TRT_{ta}.$$

Due to this setting, the token_rotation_timer of the Time_Available access_class expires sooner than the token_rotation_timer of the Normal_Asynchronous access_class; similarly the token_rotation_timer of the Normal_Asynchronous access_class expires sooner than the token_rotation_timer of the Urgent_Asynchronous access_class. Hence there is more time to transmit messages from the Normal_Asynchronous class as compared to messages from the Time_Available class and more time available to transmit messages from the Urgent_Asynchronous class as compared to messages from the Normal_Asynchronous class. Thus this scheme clearly gives preference to messages of higher priority.

3.2. Analytic model for the token cycle time

In this section an analytic expression is developed for the token cycle time of a network carrying traffic at different priorities. The token cycle time increases as the offered load is increased. The amount of service at an access_class depends upon the TRT setting of that access_class and the token cycle time seen by that access_class. As the token cycle time exceeds the TRT of an access_class, messages from that access_class will not get transmitted on the same cycle as they arrive and will therefore suffer longer queueing delays.

The range of network throughput over which optimum service can be obtained at an asynchronous access_class can be calculated from the expression for the token cycle time. This expression can also be used to calculate the network throughput at which the service time at an asynchronous access_class reaches a maximum, if the individual TRTs are known. If the message arrival rate at each access_class is known, then TRTs for each access_class can be calculated which will yield the maximum possible service until the specified throughput is reached.

3.2.1. Definitions

An active access_class at a station is termed a server. The following notation is used:

X_T ≡ duration of a token transmission (seconds/token transmission).

s ≡ the Synchronous access_class.

ua ≡ the Urgent_Asynchronous access_class.

na ≡ the Normal_Asynchronous access_class.

ta ≡ the Time_Available access_class.

λ_{ac} ≡ the mean message arrival rate at each server of an access_class, where ac is s, ua, na or ta.

S ≡ the set of all *Synchronous access_class* servers.

UA ≡ the set of all *Urgent_Asynchronous access_class* servers.

NA ≡ the set of all *Normal_Asynchronous access_class* servers.

TA ≡ the set of all *Time_Available access_class* servers.

R ≡ set of all distinct servers on the logical ring

≡ (S, UA, NA, TA).

λ_x ≡ the mean message arrival rate at server $x \in R$ (messages/second).

$HPTHT$ ≡ the *High_Priority_Token_Hold_Time*.

TRT_x ≡ the *Target_Rotation_Time* at server $x \in (UA, NA, TA)$.

TRT_{asy} ≡ the *Target_Rotation_Time* at each server of an access_class, where asy is ua, na, or ta.

$TCT_{x,i}$ ≡ the time from the end of the $(i-1)^{st}$ service period until the beginning of the i^{th} service period (seconds)

≡ token circulation time as seen by server x on the i^{th} token cycle, i.e., the time interval for which the token is away from x.

$TC_{x,i}$ ≡ time from the end of the $(i-1)^{st}$ service period until the end of the i^{th} service period (seconds).

≡ token cycle time as seen by server x on the i^{th} token cycle.

$A_{x,i}$ ≡ the number of message arrivals at x during the interval from the end of the $(i-1)^{st}$ service period until the end of the i^{th} service period.

$Q_{x,i}$ ≡ the number of messages enqueued at server x after the $(i-1)^{st}$ service period.

$f(n)$ ≡ time required to transmit n messages (octet_times).

t ≡ residual time in the token_hold_timer (octet_times).

$eff(t)$ ≡ the effective amount of time which a server can transmit messages.

= $max(0, t + f(1) -$ one *octet_time*) octet_times.

$TS_{x,i}$ ≡ duration of the i^{th} service period.

$\overline{TS}_{ac}$ ≡ average service time available to every server of an access_class, where ac is s, ua, na or ta.

3.2.2. Assumptions

In the following section, an analytic expression for the token cycle time of an 802.4 network is derived. The derivation is based on the following assumptions about the characteristics of the network:

1) The protocol management overhead is assumed to be negligible (refer to section 2.5)

2) Stations are permanent members of the logical ring.

3) Each station has traffic at all four access_classes.

4) The message arrival at each station follows a Poisson process.

5) Messages from all the servers are of constant length l_M bits and take X_M seconds for transmission.

6) All Synchronous access_class servers have the same HPTHT setting.

7) Each station has HPTHT set to the maximum allowed value (52.43 msec for a 10 Mbps bus). Hence messages from the Synchronous access_class normally are transmitted on the same token cycle as they arrive.

8) All servers of the same priority have identical traffic.

9) All servers of the same priority have the same TRT settings.

10) The TRTs are set in the order $TRT_{ua} > TRT_{na} > TRT_{ta}$.

11) Each active access_class has N servers.

3.2.3. Token cycle time

The token cycle time seen by server x on the i^{th} token cycle is

$$TC_{x,i} = TCT_{x,i} + TS_{x,i}. \quad (1)$$

The time interval during which the token is away from server x ($TCT_{x,i}$) can be expressed as the sum of

1) $N \cdot X_T$ and

2) the sum of the i^{th} service period at all servers other than server x.

This can be expressed as

$$TCT_{x,i} = N \cdot X_T + \sum_{s \in S} TS_{s,i} + \sum_{s \in UA} TS_{s,i} + \sum_{s \in NA} TS_{s,i} + \sum_{s \in TA} TS_{s,i} \quad (2)$$

for all s $\neq$ x.

The token cycle time seen by server x on the i^{th} token cycle from equations (1) and (2) can be expressed as

$$TC_{x,i} = N \cdot X_T + \sum_{s \in S} TS_{s,i} + \sum_{s \in UA} TS_{s,i} + \sum_{s \in NA} TS_{s,i} + \sum_{s \in TA} TS_{s,i} + TS_{x,i} \quad (3)$$

for all s $\neq$ x.

Assuming an average service time at each server, the average token circulation time can be expressed as

$$\overline{TCT}_x = N \cdot X_T + \sum_{s \in S} \overline{TS}_s + \sum_{s \in UA} \overline{TS}_s + \sum_{s \in NA} \overline{TS}_s + \sum_{s \in TA} \overline{TS}_s \quad (4)$$

for all s $\neq$ x.

The average token cycle time is then

$$\overline{TC}_x = N \cdot X_T + \sum_{s \in S} \overline{TS}_s + \sum_{s \in UA} \overline{TS}_s + \sum_{s \in NA} \overline{TS}_s + \sum_{s \in TA} \overline{TS}_s + \overline{TS}_x \quad (5)$$

for all s $\neq$ x.

Since x can be either a synchronous server ($x \in S$) or an asynchronous server ($x \in (UA, NA, TA)$), equation (5) can be rewritten as

$$\overline{TC}_x = N \cdot X_T + \sum_{s \in S} \overline{TS}_s + \sum_{s \in UA} \overline{TS}_s + \sum_{s \in NA} \overline{TS}_s + \sum_{s \in TA} \overline{TS}_s . \tag{6}$$

It follows from assumptions 4, 5, 6, 8 and 9 that the average service time is identical for all servers of the same priority. Hence the average token cycle time seen by all servers of the same priority is the same. It also follows from assumptions 4, 5, 6, 8 and 9 that the average token cycle time seen by every access_class is the same. Hence the average token cycle time for the network can be expressed as

$$\overline{TC} = N \cdot X_T + N \cdot \overline{TS}_s + N \cdot \overline{TS}_{ua} + N \cdot \overline{TS}_{na} + N \cdot \overline{TS}_{ta} . \tag{7}$$

The above equation can be expressed as

$$\overline{TC} = N \cdot X_T + N \cdot (\overline{TS}_s + \overline{TS}_{ua} + \overline{TS}_{na} + \overline{TS}_{ta}). \tag{8}$$

From the above equation it can be seen that the token cycle time can be expressed as the sum of

a) the token pass times, and

b) the time taken to transmit the messages from all stations.

Hence the above equation can be expressed as

$$\overline{TC} = N \cdot X_T + N \cdot X_M \cdot \Lambda \tag{9}$$

where Λ denotes the mean number of messages transmitted per station during an average token cycle.

3.2.4. Service time

The amount of time available to any server on any token cycle is limited. If x is a *Synchronous* server then it is guaranteed a certain amount of time on each token cycle. This can be expressed as

$$TS_{x,i} = \min(eff\,(HPTHT),\ f\,(Q_{x,i} + A_{x,i})). \tag{10}$$

A station starts transmitting messages when there is residual time in the token_hold_timer (refer to section 3.1.1.2). The transmission of a message started just before the expiration of the token_hold_timer will be completed even if it runs past the expiration of the timer. Hence the amount of time a station can transmit messages can exceed the actual residual time by up to the time taken to transmit a single message, i.e., $f\,(1)$. Hence the actual time available to serve messages is eff(t), where t is the residual time in the token_hold_timer. If x is an asynchronous server then the amount of time available on any token cycle depends upon the residual time in the token hold timer (refer to sections 3.1.1 and 3.1.2). This can be expressed as

$$TS_{x,i} = \begin{cases} \min(eff\,(TRT_x - TCT_{x,i}),\ f\,(Q_{x,i} + A_{x,i})) & TCT_{x,i} < TRT_x \\ 0 & TCT_{x,i} \geqslant TRT_x \end{cases} \tag{11}$$

If x is a synchronous server then the average service time of x can be expressed as

$$\overline{TS}_x = \min(eff\,(HPTHT),\ f\,(\overline{Q}_x + \overline{A}_x)). \tag{12}$$

If x is an asynchronous server then the average service time of server x can be expressed as

$$\overline{TS}_x = \begin{cases} \min(eff\,(TRT_x - \overline{TCT}_x),\ f\,(\overline{Q}_x + \overline{A}_x)) & \overline{TCT}_x < TRT_x \\ 0 & \overline{TCT}_x \geqslant TRT_x \end{cases} \tag{13}$$

Assuming an average arrival rate of messages λ_x at server x, the number of messages that have arrived while

a) the token is circulating around the ring and

b) x is being served on the i^{th} token cycle is

$$A_{x,i} = \lambda_x \cdot TCT_{x,i} + \lambda_x \cdot TS_{x,i} = \lambda_x \cdot TC_{x,i}. \quad (14)$$

Hence the average number of messages that have arrived at server x during an average token cycle can be expressed as

$$\overline{A_x} = \lambda_x \cdot \overline{TC}. \quad (15)$$

Let X_M be the time taken to transmit a message. Then the time taken to transmit the average number of messages that have arrived at a server x can be expressed as $X_M \cdot \lambda_x \cdot \overline{TC}$, from equation (15). As long as the time to transmit all the messages that have arrived is less than or equal to HPTHT (if x is a synchronous server), or less than or equal to $(TRT_{asy} - \overline{TCT})$ if x is an asynchronous server, the average service time at server x can be expressed as $\overline{TS_x} = f(\overline{Q_x} + \overline{A_x})$. This is true for $\overline{TC} \leqslant HPTHT$ and $\overline{TC} \leqslant TRT_{asy}$. In the region $\overline{TC} \leqslant HPTHT$ it can be assumed that all the messages from the Synchronous access_class get transmitted during the same token cycle. Also in the region $\overline{TC} \leqslant TRT_{asy}$ it can be assumed that all the messages from that access_class and from access_classes of higher priority that have arrived during a particular token cycle get transmitted during the same token cycle. Hence $\overline{Q_x}$ can be considered to be negligible. Eliminating $\overline{Q_x}$ from equation (13) and substituting the value of $\overline{A_x}$ from equation (15), the average service time can be expressed as

$$\overline{TS_x} = \overline{TS_{ac}} = X_m \cdot \lambda_{ac} \cdot \overline{TC}. \quad (16)$$

Traffic from each access_class gets served in the order of priority, starting from the Synchronous access_class, and if sufficient time is available, lower access_classes also get service. The average service time at each of the lower three access_classes can increase with an increase in offered load from that access_class until the average token cycle time equals the TRT of that access_class. Thus the throughput which results in a token cycle time of TRT for an access_class,

corresponds to the throughput until which maximum possible service can be obtained at that access_class. A definition of throughput is given in [Weaver 85].

3.3. Determining TRTs

A problem faced by token bus network designers and operators is the selection of Target_Rotation_Times which will implement a desired priority scheme. It is possible to determine the individual TRT setting of an access_class so as to obtain maximum possible service (most of the messages get transmitted within one token cycle) at an access_class until a specified throughput is achieved.

If the designer decides that the Time_Available access_class should receive maximum possible service until the network throughput is α, where $0 \leqslant \alpha \leqslant 1$, then

$$\alpha = \frac{N \cdot \Lambda_\alpha \cdot l_M}{C \cdot \overline{TC}_\alpha} \tag{17}$$

where $\Lambda_\alpha =$ mean number of messages transmitted per station per token cycle at a throughput of α. Solving for Λ_α,

$$\Lambda_\alpha = \frac{\alpha \cdot C \cdot \overline{TC}_\alpha}{N \cdot l_M}. \tag{18}$$

From the discussion in the previous section, it follows that maximum possible service at an access_class can be obtained in the region where $\overline{TC} \leqslant TRT_{asy}$. Hence the average number of messages that arrive during the time interval $\overline{TC}_\alpha$ (region where $\overline{TC}_\alpha \leqslant TRT_{ta}$) should equal the mean number of messages transmitted in this time interval, i.e., Λ_α. Hence

$$\Lambda_\alpha = (\lambda_s + \lambda_{ua} + \lambda_{na} + \lambda_{ta}) \cdot \overline{TC}_\alpha. \tag{19}$$

Substituting for Λ_α in equation (18) we obtain the sum of λs to be

$$\lambda_s + \lambda_{ua} + \lambda_{na} + \lambda_{ta} = \frac{\alpha \cdot C}{N \cdot l_M}. \tag{20}$$

Many combinations of individual message arrival rates will yield the unique sum satisfying the above equation. From equation (9) the token cycle time can be expressed as

$$\overline{TC}_{\alpha} = N \cdot X_T + N \cdot X_M \cdot \Lambda_{\alpha}. \tag{21}$$

Substituting for Λ_{α} from equation (19) yields

$$\overline{TC}_{\alpha} = N \cdot X_T + N \cdot X_M \cdot (\lambda_s + \lambda_{ua} + \lambda_{na} + \lambda_{ta}) \cdot \overline{TC}_{\alpha}. \tag{22}$$

Since the Time_Available access_class should receive maximum possible service until a throughput of α, it follows from the discussion of service time in the previous section that the token cycle time at this throughput decides the TRT of the Time_Available access_class. Hence solving for $eff\,(TRT_{ta}) = \overline{TC}_{\alpha}$,

$$eff\,(TRT_{ta}) = \frac{N \cdot X_T}{1 - N \cdot X_M \cdot (\lambda_s + \lambda_{ua} + \lambda_{na} + \lambda_{ta})} \tag{23}$$

which is the effective value of the Time_Available *Target_Rotation_Time*.

As the offered load increases, increasing the network carried load beyond α, the token cycle time increases beyond the TRT of the Time_Available access_class. Hence service to the Time_Available access_class class gets reduced and eventually drops to zero (when TCT_{ta} equals TRT_{ta} as given by equation (13)). Hence the traffic from the Time_Available access_class does not contribute to the network throughput anymore. Hence the load carried by the network is contributed by the upper three classes alone.

If the designer determines that the service at Normal_Asynchronous access_class should reach a maximum when the network throughput is β where $0 \leqslant \alpha < \beta \leqslant 1$, then

$$\beta = \frac{N \cdot \Lambda_{\beta} \cdot l_M}{C \cdot \overline{TC}_{\beta}} \tag{24}$$

where Λ_{β} = mean total number of messages transmitted per station per token cycle at a throughput of β. Solving for Λ_{β},

$$\Lambda_\beta = \frac{\beta \cdot C \cdot \overline{TC}_\beta}{N \cdot l_M}. \tag{25}$$

From the discussion in the previous section, it follows that the maximum possible service at an access_class can be obtained in the region where $\overline{TC} \leqslant TRT_{asy}$. Hence the mean number of message arrivals during the time interval $\overline{TC}_\beta$ should equal the mean number of messages transmitted during that time interval, i.e., Λ_β. Hence

$$\Lambda_\beta = (\lambda_s + \lambda_{ua} + \lambda_{na}) \cdot \overline{TC}_\beta. \tag{26}$$

Substituting for Λ_β in equation (25) we obtain the sum of message arrival rates of the upper three access_classes to be

$$\lambda_s + \lambda_{ua} + \lambda_{na} = \frac{\beta \cdot C}{N \cdot l_M}. \tag{27}$$

Many combinations of individual message arrival rates will yield the unique sum in equation (27). From equation (9) the token cycle time can be expressed as

$$\overline{TC}_\beta = N \cdot X_T + N \cdot X_M \cdot \Lambda_\beta. \tag{28}$$

Substituting for Λ_β from equation (26) yields

$$\overline{TC}_\beta = N \cdot X_T + N \cdot X_M \cdot (\lambda_s + \lambda_{ua} + \lambda_{na}) \cdot \overline{TC}_\beta. \tag{29}$$

Since service at the Normal_Asynchronous access_class should be maximum at a throughput of β, it follows from the discussion of service time in the previous section that the token cycle time at this carried load decides the TRT of the Normal_Asynchronous access_class. Hence solving for $eff\,(TRT_{na}) = \overline{TC}_\beta$.

$$eff\,(TRT_{na}) = \frac{N \cdot X_T}{1 - N \cdot X_M \cdot (\lambda_s + \lambda_{ua} + \lambda_{na})} \tag{30}$$

which is the effective value of the Normal_Asynchronous *Target_Rotation_Time*.

As the offered load increases, increasing the network throughput beyond β, the token cycle time increases beyond the TRT of the Normal_Asynchronous

access_class. Hence service to the Normal_Asynchronous class gets reduced and eventually drops to zero (when TCT_{na} equals TRT_{na} as given by equation (13)). Hence the traffic from the Normal_Asynchronous access_class does not contribute to the network throughput anymore. Hence the load carried by the network is contributed only by the upper two classes.

If the designer determines that the service at Urgent_Asynchronous access_class should reach a maximum when the network throughput is γ where $0 \leqslant \alpha < \beta < \gamma \leqslant 1$, then

$$\gamma = \frac{N \cdot \Lambda_\gamma \cdot l_M}{C \cdot \overline{TC}_\gamma}. \tag{31}$$

where $\Lambda_\gamma =$ mean total number of messages transmitted per station per token cycle at a throughput of γ. Solving for Λ_γ,

$$\Lambda_\gamma = \frac{\gamma \cdot C \cdot \overline{TC}_\gamma}{N \cdot l_M}. \tag{32}$$

From the discussion in the previous section, it follows that the maximum possible service at an access_class can be obtained in the region where $\overline{TC} \leqslant TRT_{asy}$ and in this region messages get transmitted on the same token cycle as they arrive. Hence the number of messages that arrive during the time interval $\overline{TC}_\gamma$ should equal Λ_γ. Hence,

$$\Lambda_\gamma = (\lambda_s + \lambda_{ua}) \cdot \overline{TC}_\gamma. \tag{33}$$

Substituting for Λ_γ in equation (32) we obtain the sum of message arrival rates at the upper two access_classes to be

$$\lambda_s + \lambda_{ua} = \frac{\gamma \cdot C}{N \cdot l_M}. \tag{34}$$

It can be seen that many values can be chosen for the individual message arrival rates provided that their sum yields the value obtained from the above equation. From equation (9) the token cycle time can be expressed as

$$\overline{TC}_\gamma = N \cdot X_T + N \cdot X_M \cdot \Lambda_\gamma. \tag{35}$$

Substituting for Λ_γ from equation (33) yields

$$\overline{TC}_\gamma = N \cdot X_T + N \cdot X_M \cdot (\lambda_s + \lambda_{ua}) \cdot \overline{TC}_\gamma. \tag{36}$$

Since service at Urgent_Asynchronous access_class should be maximum at a throughput of γ, it follows from the discussion of service time in the previous section that the token cycle time at this throughput decides the TRT of the Urgent_Asynchronous access_class. Hence solving for $eff\,(TRT_{ua}) = \overline{TC}_\gamma$,

$$eff\,(TRT_{ua}) = \frac{N \cdot X_T}{1 - N \cdot X_M \cdot (\lambda_s + \lambda_{ua})} \tag{37}$$

which is the effective value of the Urgent_Asynchronous *Target_Rotation_Time*.

As the offered load increases, increasing the network carried load beyond $\gamma \cdot C$, the token cycle time increases beyond the TRT of the Urgent_Asynchronous access_class. Hence service to the Urgent_Asynchronous class gets reduced and finally drops to zero when TCT_{ua} equals TRT_{ua} as given by equation (13).

3.3.1. An example

This section illustrates the calculation of TRTs for a network configuration using the equations derived in the previous section. Consider a 32 station network with each station offering service at all four access_classes, with identical traffic at all servers of an access_class. The bus capacity is 10 Mbps. The size of data frames is 272 bits including framing. Tokens are 112 bits in length and require 16.2 microseconds (including 50 bit times of propagation delay) for transmission.

N = 32

X_M = .0000272 seconds = 27.2 microseconds.

X_T = .0000162 seconds = 16.2 microseconds.

Suppose that the user decides that the Time_Available access_class should receive maximum possible service until a throughput of 0.18 where $\alpha = 0.18$. Then the sum of λs as given by equation (20) is

$$\lambda_s + \lambda_{ua} + \lambda_{na} + \lambda_{ta} = \frac{\alpha \cdot C}{N \cdot l_M} = \frac{(0.18) \cdot (10^7)}{32 \cdot 272} = 207 \text{ messages per second.}$$

Individual λs are chosen such that their sum is equal to the above value. Substituting the above value in equation (23),

$$eff\,(TRT_{ta}) = \frac{N \cdot X_T}{1 - N \cdot X_M \cdot (\lambda_s + \lambda_{ua} + \lambda_{na} + \lambda_{ta})}$$

$$= \frac{32 \cdot 16.2 \cdot 10^{-6}}{1 - 32 \cdot 27.2 \cdot 10^{-6} \cdot (207)} = 0.632 \text{ msecs.}$$

Suppose that the user decides that the Normal_Asynchronous access_class should receive maximum possible service until a throughput of 0.24 where $\beta = 0.24$. Then the sum of λs as given by equation (27) is

$$\lambda_s + \lambda_{ua} + \lambda_{na} = \frac{\beta \cdot C}{N \cdot l_M} = \frac{(0.24) \cdot (10^7)}{32 \cdot 272} = 275 \text{ messages per second.}$$

Individual λs are chosen such that their sum is equal to the above value. Substituting the above value in equation (30),

$$eff\,(TRT_{na}) = \frac{N \cdot X_T}{1 - N \cdot X_M \cdot (\lambda_s + \lambda_{ua} + \lambda_{na})} = \frac{32 \cdot 16.2 \cdot 10^{-6}}{1 - 32 \cdot 27.2 \cdot 10^{-6} \cdot (275)} = 0.6821 \text{ msecs.}$$

Suppose that the user decides that the Urgent_Asynchronous access_class should receive maximum possible service until a throughput of 0.44 where $\gamma = 0.44$. Then the sum of λs as given by equation (34) is

$$\lambda_s + \lambda_{ua} = \frac{\gamma \cdot C}{N \cdot l_M} = \frac{(0.44) \cdot (10^7)}{32 \cdot 272} = 505 \text{ messages per second.}$$

Individual λs are chosen such that their sum is equal to the above value. Substituting the above value in equation (37),

$$eff\,(TRT_{ua}) = \frac{N \cdot X_T}{1 - N \cdot X_M \cdot (\lambda_s + \lambda_{ua})} = \frac{32 \cdot 16.2 \cdot 10^{-6}}{1 - 32 \cdot 27.2 \cdot 10^{-6} \cdot (505)} = 0.9256 \text{ msecs.}$$

From the effective values of TRTs calculated in the above equations, the actual values of TRTs are given by $(eff\,(TRT_{asy}) - f\,(1) + 1\ octet\_time)$, where f(1) is equal to 34 octet_times. $eff\,(TRT_{ta})$ is 790 octet_times, $eff\,(TRT_{na})$ is 852 octet_times, $eff\,(TRT_{ua})$ is 1157 octet_times, and the values are

TRT_{ta} = 0.6056 msecs.

TRT_{na} = 0.655 msecs.

TRT_{ua} = 0.8992 msecs.

3.4. Predicting peak throughput points for known TRTs

If the user knows the TRT of each access_class, then it is possible to determine the throughput until which an access_class receives maximum possible service (messages get transmitted on the same token cycle as they arrive). Thus the range of carried load (throughput) over which optimum service can be obtained at an asynchronous access_class can be predicted.

If TRT_{ta} is known, it is possible to calculate the throughput until which messages from the Time_Available access_class get transmitted on the same token cycle as they arrive. From the discussion on service time in section 3.2.4 it follows that the Time_Available access_class receives maximum possible service until the token cycle time reaches TRT_{ta}. This value can be calculated from equation (9) with $\overline{TC} = eff\,(TRT_{ta})$. Hence rewriting equation (9) with $\overline{TC} = eff\,(TRT_{ta})$,

$$eff\,(TRT_{ta}) = N \cdot X_T + N \cdot X_M \cdot \Lambda_{ta\_max} \tag{38}$$

where Λ_{ta_max} is the maximum value of the mean number of messages that can be transmitted per station per token cycle in the region where $\overline{TC} \leqslant TRT_{ta}$. Solving for Λ_{ta_max} in the above equation,

$$\Lambda_{ta\_max} = \frac{eff\,(TRT_{ta}) - N \cdot X_T}{N \cdot X_M}. \tag{39}$$

Hence the throughput until which maximum possible service can be obtained at the Time_Available access_class (at which point $\overline{TC} = TRT_{ta}$) is calculated as

$$\frac{N \cdot \Lambda_{ta\_max} \cdot l_M}{C \cdot eff\,(TRT_{ta})}. \tag{40}$$

Hence to obtain maximum possible service at Time_Available class until this throughput, the average number of arrivals within a time interval of TRT_{ta} should equal the number of messages that were transmitted in this interval. Hence

$$(\lambda_s + \lambda_{ua} + \lambda_{na} + \lambda_{ta}) \cdot eff\,(TRT_{ta}) = \Lambda_{ta\_max}.$$

Solving for the sum of λs the above equation can be written as

$$(\lambda_s + \lambda_{ua} + \lambda_{na} + \lambda_{ta}) = \frac{\Lambda_{ta\_max}}{eff\,(TRT_{ta})}. \quad (41)$$

Note that many combinations of individual message arrival rates can yield a unique sum satisfying the above equation.

Knowing the value of TRT_{na} it is possible to determine the throughput until which Normal_Asynchronous access_class receives maximum possible service. From the discussion on service time in section 3.2.4 it follows that the Normal_Asynchronous access_class will receive maximum possible service until the carried load results in a token cycle time of TRT_{na}. This value can be calculated from equation (9) with $\overline{TC} = eff\,(TRT_{na})$. Hence rewriting equation (9) with $\overline{TC} = eff\,(TRT_{na})$,

$$eff\,(TRT_{na}) = N \cdot X_T + N \cdot X_M \cdot \Lambda_{na\_max} \quad (42)$$

where Λ_{na_max} is the maximum number of messages that can be transmitted in the region $\overline{TC} \leqslant TRT_{na}$. Solving for Λ_{na_max} in the above equation,

$$\Lambda_{na\_max} = \frac{eff\,(TRT_{na}) - N \cdot X_T}{N \cdot X_M}. \quad (43)$$

Hence the throughput until which maximum possible service can be obtained (at which point $\overline{TC} = TRT_{na}$) at the Normal_Asynchronous access_class is calculated as

$$\frac{N \cdot \Lambda_{na\_max} \cdot l_M}{C \cdot eff\,(TRT_{na})}. \quad (44)$$

Hence to obtain maximum possible service at the Normal_Asynchronous class until this throughput the number of arrivals within a time interval of TRT_{na} should equal the number of messages that got transmitted. Hence

$$(\lambda_s + \lambda_{ua} + \lambda_{na}) \cdot eff\,(TRT_{na}) = \Lambda_{na\_max}.$$

Solving for the sum of λs the above equation can be written as

$$(\lambda_s + \lambda_{ua} + \lambda_{na}) = \frac{\Lambda_{na\_max}}{eff\,(TRT_{na})}. \quad (45)$$

Note that many combinations of individual message arrival rates can yield a unique sum satisfying the above equation.

From the discussion on service time in section 3.4 it follows that service at Urgent_Asynchronous access_class reaches a maximum when the network throughput results in a token cycle time of TRT_{ua} . This value of the carried load can be calculated from equation (9) with $\overline{TC} = eff\,(TRT_{ua})$. Hence rewriting equation (9) with $\overline{TC} = eff\,(TRT_{ua})$.

$$eff\,(TRT_{ua}) = N \cdot X_T + N \cdot X_M \cdot \Lambda_{ua\_max} \tag{46}$$

where Λ_{ua_max} is the maximum number of messages per station per token cycle that can be transmitted in the region $\overline{TC} \leq TRT_{ua}$. Solving for Λ_{ua_max} in the above equation

$$\Lambda_{ua\_max} = \frac{eff\,(TRT_{ua}) - N \cdot X_T}{N \cdot X_M}. \tag{47}$$

Hence the throughput until which maximum possible service can be obtained at the Urgent_Asynchronous access_class is calculated as

$$\frac{N \cdot \Lambda_{ua\_max} \cdot l_M}{C \cdot eff\,(TRT_{ua})}. \tag{48}$$

Hence to obtain maximum possible service at Urgent_Asynchronous access_class until this throughput, the number of arrivals within a time interval of TRT_{ua} should equal the number of messages that got transmitted. Hence

$$(\lambda_s + \lambda_{ua}) \cdot eff\,(TRT_{ua}) = \Lambda_{ua\_max}.$$

Solving for the sum of λs the above equation can be written as

$$(\lambda_s + \lambda_{ua}) = \frac{\Lambda_{ua\_max}}{eff\,(TRT_{ua})}. \tag{49}$$

Note that many combinations of individual message arrival rates can yield a unique sum satisfying the above equation.

3.4.1. An example

This section illustrates the calculation of the peak throughput point for a given TRT setting, until which each access_class receives uncurtailed service. The illustration is for a network configuration with 32 stations. Each station offers service at all four access_classes. Traffic in an access_class is identical for all stations. The bus capacity is 10 Mbps.

N = 32

Size of data frame = 272 bits including framing.

Size of token = 112 bits

X_M = .0000272 seconds = 27.2 microseconds

X_T = .0000162 = 16.2 microseconds

Supppose the user decides the TRTs to be

TRT_{ua} = 0.9 msecs = .0009 seconds

TRT_{na} = 0.8 msecs = .0008 seconds

TRT_{ta} = 0.7 msecs = .0007 seconds.

The effective values are 0.9264, 0.8264, and 0.7264 msecs respectively. From equation (39),

$$\Lambda_{ta\_max} = \frac{eff\,(TRT_{ta}) - N \cdot X_T}{N \cdot X_M} = \frac{0.7272 \cdot 10^{-3} - 32 \cdot 16.2 \cdot 10^{-6}}{32 \cdot 27.2 \cdot 10^{-6}} = 0.240 \text{ messages per token cycle.}$$

From equation (40) the throughput can be calculated as

$$\frac{N \cdot \Lambda_{ta\_max} \cdot l_M}{C \cdot eff\,(TRT_{ta})} = \frac{32 \cdot 0.240 \cdot 272}{10^7 \cdot 0.7272 \cdot 10^{-3}} = 0.29.$$

Hence messages from Time_Available access_class can get transmitted without suffering large queueing delays until a throughput of 0.29. From equation (41) the sum of λs is

$$(\lambda_s + \lambda_{ua} + \lambda_{na} + \lambda_{ta}) = \frac{\Lambda_{ta\_max}}{eff\,(TRT_{ta})} = \frac{0.240}{0.7272 \cdot 10^{-3}} = 330 \text{ messages per second.}$$

Note that many combinations of individual message arrival rates can yield a unique

sum satisfying the above value.

From equation (43),

$$\Lambda_{na\_max} = \frac{eff\,(TRT_{na}) - N \cdot X_T}{N \cdot X_M} = \frac{0.8272 \cdot 10^{-3} - 32 \cdot 16.2 \cdot 10^{-6}}{32 \cdot 27.2 \cdot 10^{-6}} = 0.354 \text{ messages per token cycle.}$$

From equation (44) the throughput can be calculated as

$$\frac{N \cdot \Lambda_{na\_max} \cdot l_M}{C \cdot eff\,(TRT_{na})} = \frac{32 \cdot 0.354 \cdot 272}{10^7 \cdot 0.8272 \cdot 10^{-3}} = 0.37.$$

Hence messages from Normal_Asynchronous access_class can get transmitted as they arrive until a throughput of 0.37. From equation (45) the sum of λs is

$$(\lambda_s + \lambda_{ua} + \lambda_{na}) = \frac{\Lambda_{na\_max}}{eff\,(TRT_{na})} = \frac{0.354}{0.8272 \cdot 10^{-3}} = 429 \text{ messages per second.}$$

Many combinations of individual message arrival rates can yield a unique sum satisfying the above value.

From equation (47),

$$\Lambda_{ua\_max} = \frac{eff\,(TRT_{ua}) - N \cdot X_T}{N \cdot X_M} = \frac{0.9272 \cdot 10^{-3} - 32 \cdot 16.2 \cdot 10^{-6}}{32 \cdot 27.2 \cdot 10^{-6}} = 0.470 \text{ messages per token cycle.}$$

From equation (48) the throughput can be calculated as

$$\frac{N \cdot \Lambda_{ua\_max} \cdot l_M}{C \cdot eff\,(TRT_{ua})} = \frac{32 \cdot 0.470 \cdot 272}{10^7 \cdot 0.9272 \cdot 10^{-3}} = 0.44.$$

From equation (49) the sum of λs is

$$(\lambda_s + \lambda_{ua}) = \frac{\Lambda_{ua\_max}}{eff\,(TRT_{ua})} = \frac{0.470}{0.9272 \cdot 10^{-3}} = 507 \text{ messages per second.}$$

3.5. Comparison and conclusions

In order to verify the values derived in sections 3.3 and 3.4, network configurations have been simulated with the various parameters set to values derived in the above mentioned sections.

The simulator used for this purpose was designed and developed by Catherine F. Summers [Summers 85]. The simulator can generate reports about important network characteristics such as throughput, bus utilization, delivery time of a message, queue lengths in the stations in the network, statistics on network usage and statistics on service at each station.

The TRTs have been set to values derived in the previous sections. A series of network configurations have been simulated to show the variation of different parameters with throughput. The individual message arrival rates at each access_class have been varied from one simulation to another, to increase the total offered load. The results obtained from the reports have been displayed in graphical form and discussed in the next section.

3.5.1. Comparison of simulation results with derived values

This section discusses the results obtained from the simulations of network configurations discussed in the previous section. It can be observed from Figure 3.1 that average service time at the Time_Available access_class keeps increasing until a throughput of about 0.18. As the throughput increases further, the average service time starts falling and eventually drops to zero. From Figure 3.2 it can be observed that the average delivery times are reasonably low until the network throughput is 0.18. This shows that most messages are getting transmitted on the same token cycle as they arrive. Beyond a throughput of 0.18 the token cycle time has exceeded TRT_{ta}. This can be observed from Figure 3.7. Hence the Time_Available class is getting reduced service. Hence messages are suffering higher

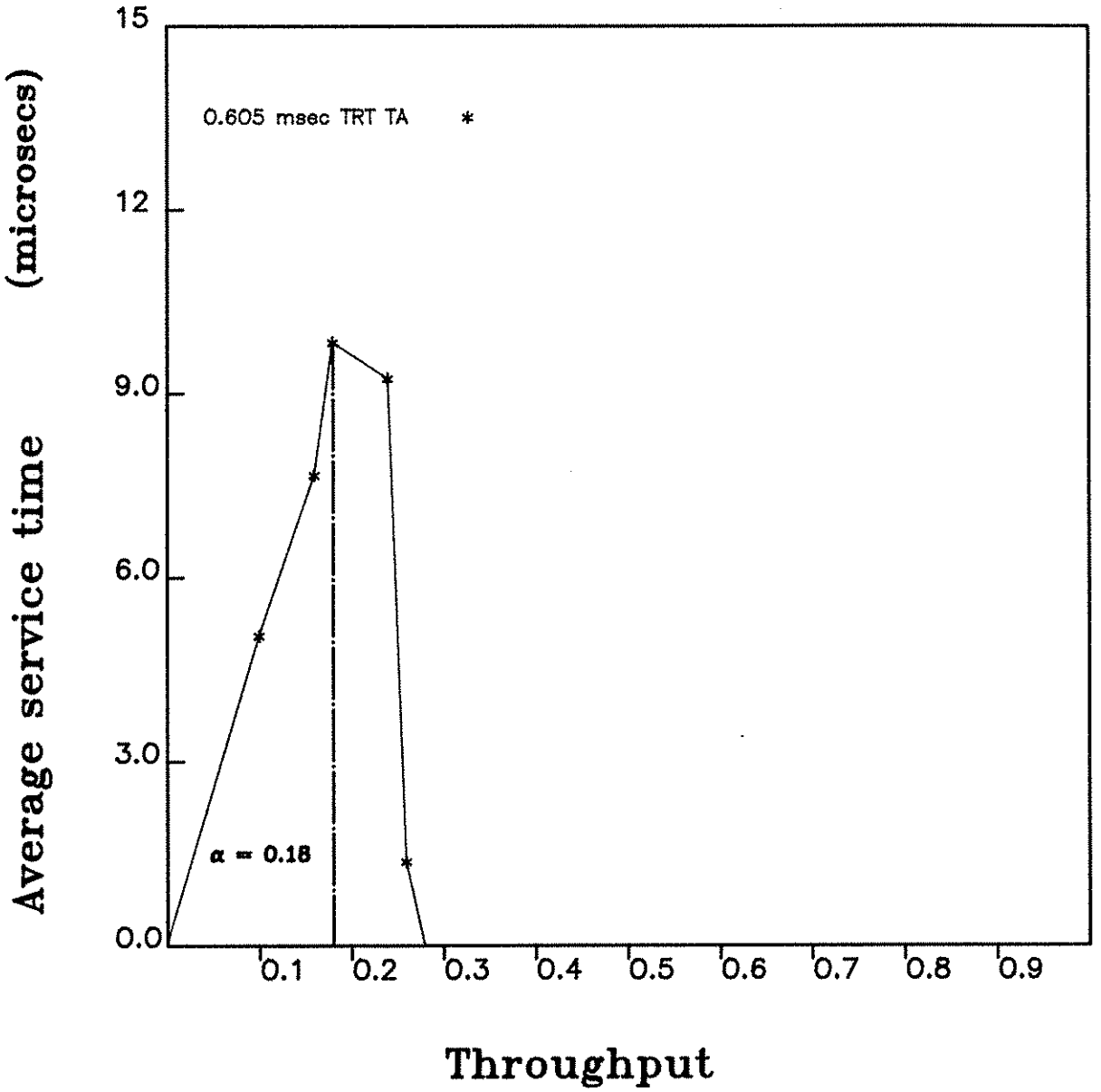


Figure 3.1
Average service time for Time_Available class

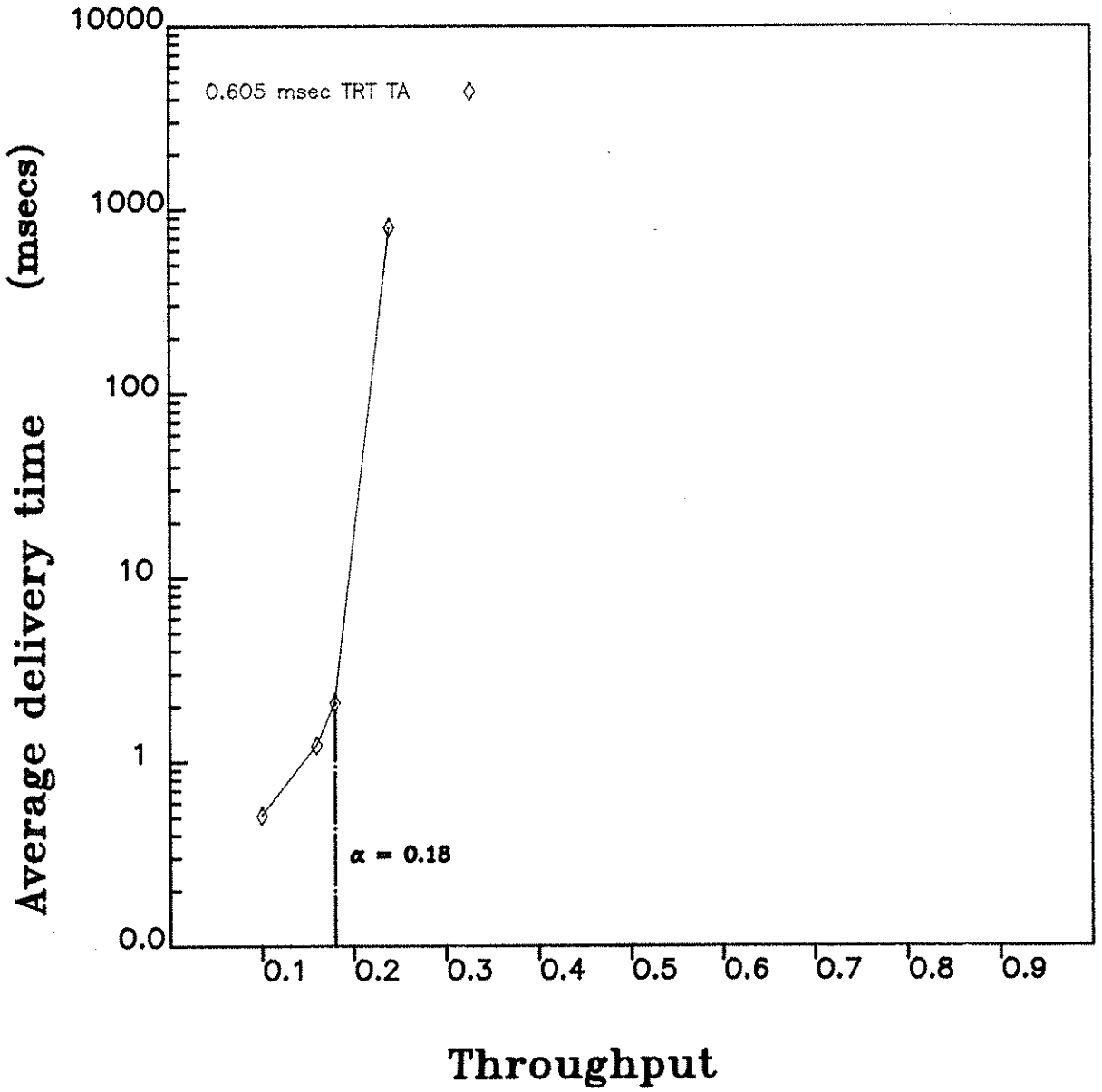


Figure 3.2
Average delivery time for Time_Available class

queueing delays, so the delivery times are increasing exponentially in this region. This can be observed from Figure 3.2.

The TRT_{ta} value was calculated in section 3.3.1 so as to achieve maximum possible service until a throughput of 0.18. The results of the simulation agree with the expected values very closely.

It can be observed from Figure 3.3 that the average service time at the Normal_Asynchronous access_class keeps increasing until a throughput of about 0.24. As the throughput increases further, the average service time starts decreasing and drops to zero. From Figure 3.4 it can be observed that the average delivery times are reasonably low until the network throughput is 0.24. This indicates that most messages are getting transmitted on the same token cycle as they arrive. Beyond a throughput of 0.24 the token cycle time has exceeded TRT_{na}. This can be seen from Figure 3.7. Hence the Normal_Asynchronous class is getting reduced service leading to higher queueing delays. Hence the delivery times rise exponentially in this region. This can be observed from Figure 3.4.

The TRT_{na} value was calculated in section 3.3.1 so as to achieve maximum possible service until a throughput of 0.24. From the above discussion it follows that the results of the simulation agree with the expected values very closely.

It can be observed from Figure 3.5 that the average service time at the Urgent_Asynchronous access_class keeps increasing until a throughput of about 0.44. As the throughput increases further, the average service time starts decreasing. From Figure 3.6 it can be observed that the average delivery times are reasonably low until the network throughput is 0.42. This indicates that most messages are getting transmitted on the same token cycle as they arrive. Beyond a throughput of 0.42 the token cycle time has exceeded TRT_{ua} and hence the Urgent_Asynchronous class is getting reduced service. This can be observed from Figure 3.7. Hence mes-

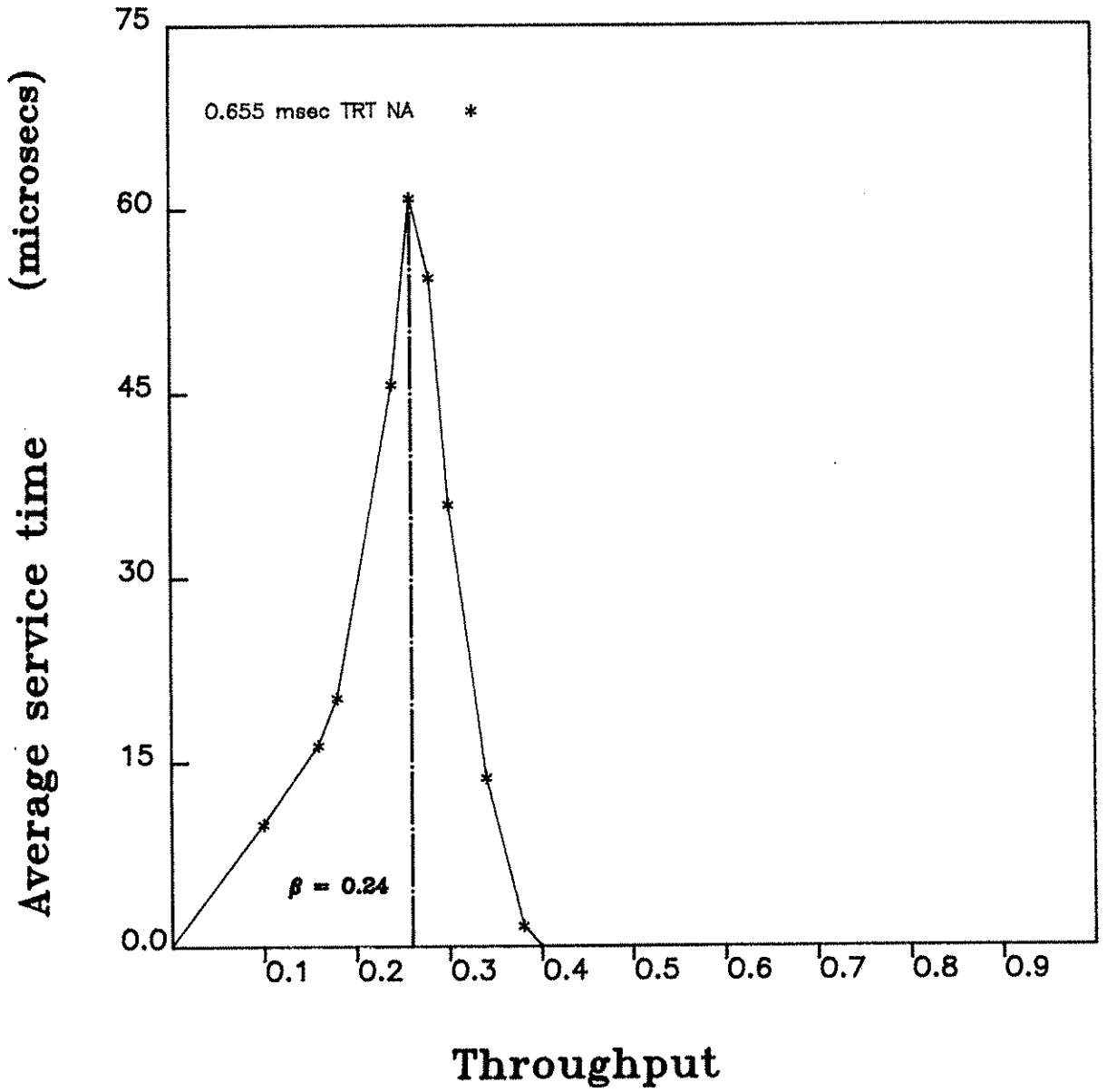


Figure 3.3
Average service time for Normal_Asynchronous class

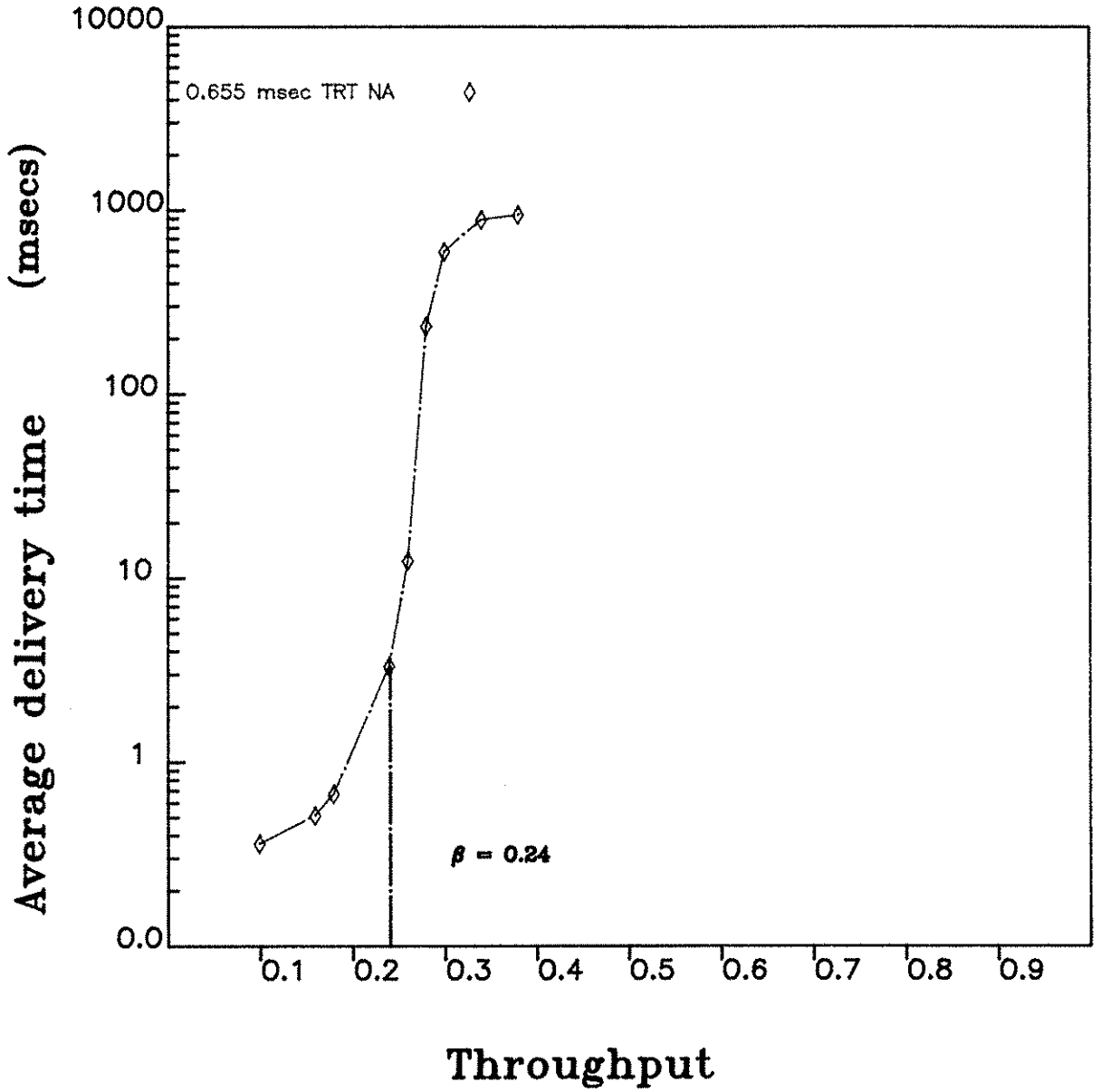


Figure 3.4
Average delivery time for Normal_Asynchronous class

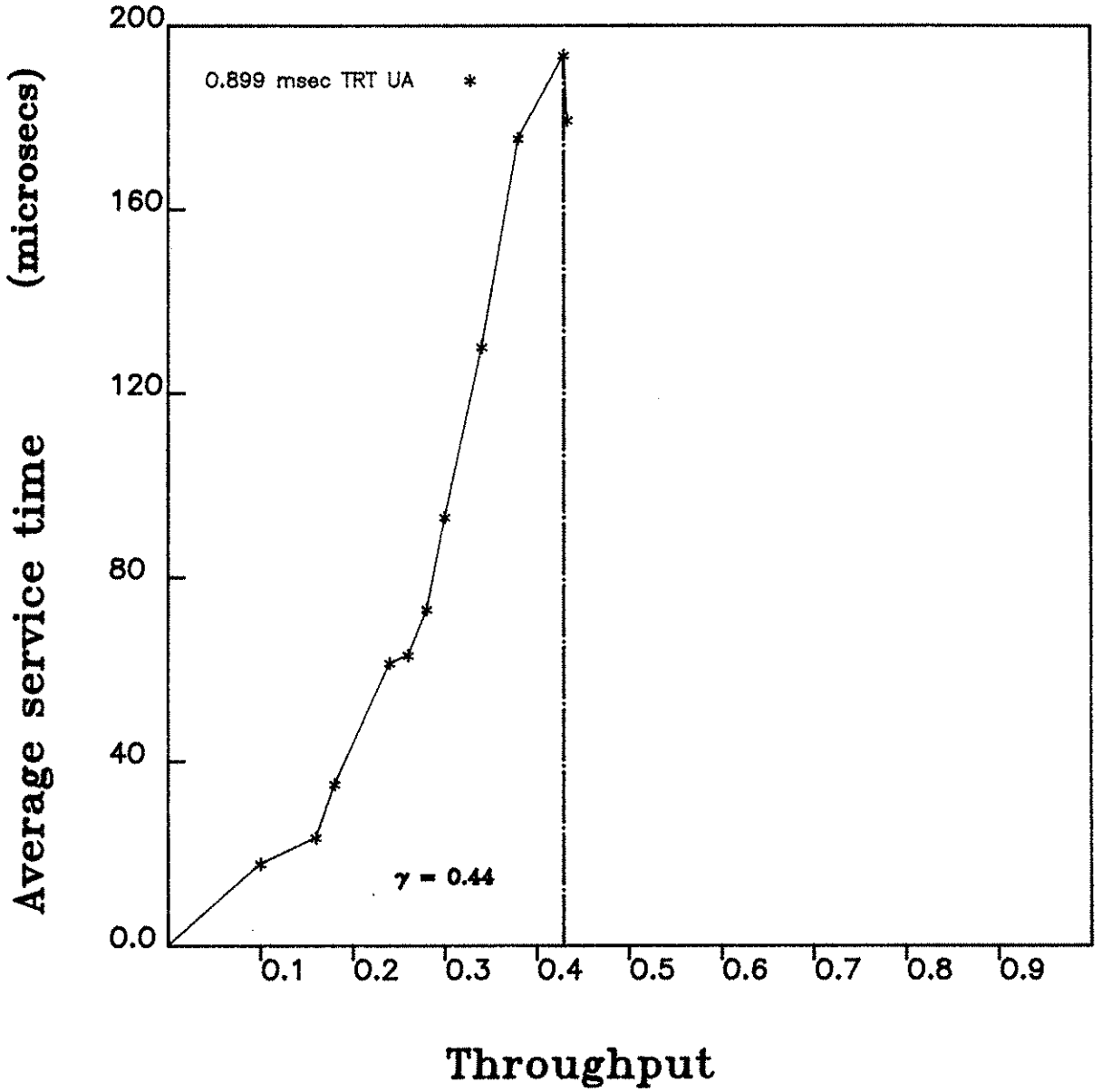


Figure 3.5
Average service time for Urgent_Asynchronous class

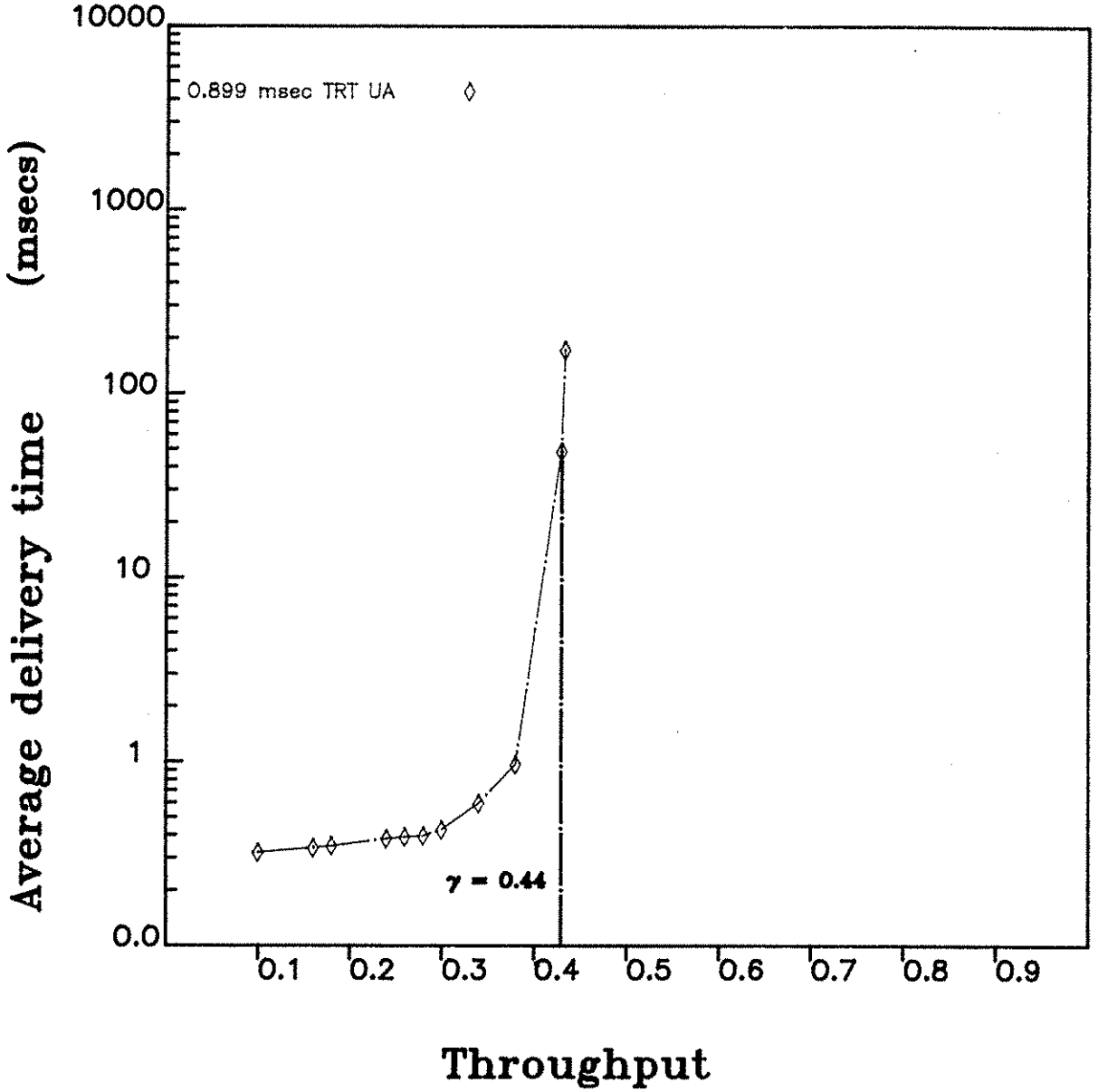


Figure 3.6
Average delivery time for Urgent_Asynchronous class

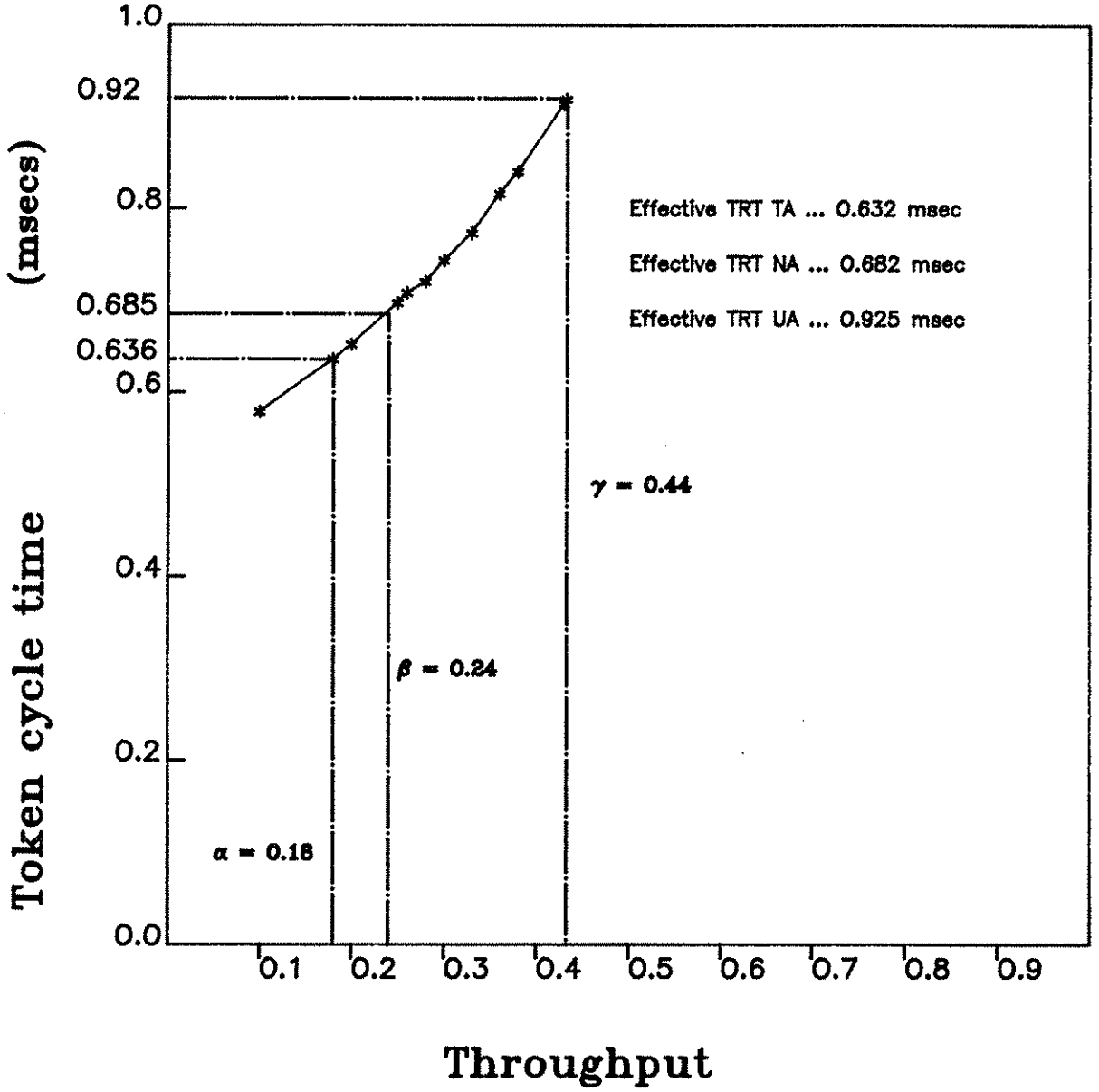


Figure 3.7
Token cycle time

sages are suffering higher queueing delays. Hence the delivery times rise exponentially in this region. This can be observed from Figure 3.6.

The calculations in section 3.3.1 showed that maximum possible service can be obtained at Urgent_Asynchronous access_class until a throughput of 0.44. The same is illustrated by the simulation results.

3.5.2. Comparison of simulation results with derived values

It was predicted in section 3.4.1 that uncurtailed service at the Time_Available access_class can be obtained (for the configuration discussed in section 3.4.1) until a throughput of 0.29. From Figure 3.8 it can be observed that average service time is increasing until a throughput of about 0.34. The delivery time of messages from the Time_Available access_class is reasonably low up to a throughput of about 0.3. Beyond this value the delivery time increases exponentially due to reduced service and hence leading to exponentially growing queue lengths. This can be seen in Figure 3.9. The service is reduced because the token cycle time has exceeded the TRT_{ta} in this region. This can be observed from Figure 3.14 which shows the variation of token cycle time vs. throughput. This shows that simulation results agree very closely with the result derived in section 3.4.1 for the Time_Available access_class. In section 3.4.1 it was demonstrated that maximum possible service can be obtained at the Normal_Asynchronous access_class, for the configuration discussed in section 3.4.1 until a throughput of 0.37. From Figure 3.10 it can be observed that average service time is increasing until a throughput of about 0.38. The delivery time of messages from Normal_Asynchronous access_class is reasonably low upto a throughput of about 0.38. Beyond this value the delivery time increases exponentially due to reduced service and hence exponentially growing queue lengths. This can be observed from Figure 3.11. The service is reduced because the token cycle time has exceeded the TRT_{na} in this region. The variation of token cycle time as a function of throughput can be observed from Figure 3.14. Hence it can be seen that simulation results agree very closely with the expected result in section 3.4.1. For the network configuration discussed in section 3.4.1 the expected value of throughput until which uncurtailed service can be obtained at the Urgent_Asynchronous access_class was calculated as 0.44. From Figure 3.12 it can be observed that average service time is increasing until a throughput of about

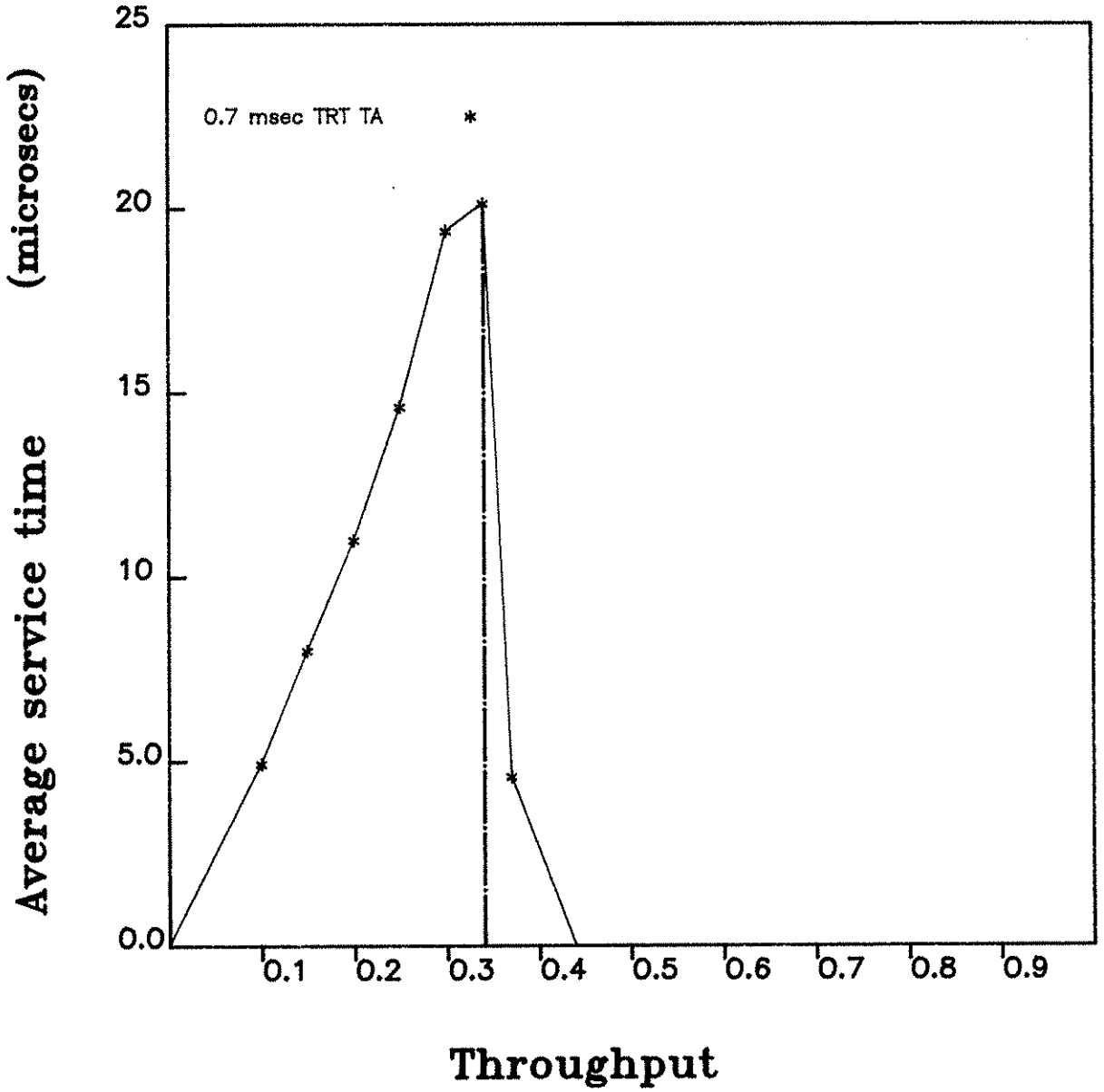


Figure 3.8
Average service time for Time_Available class

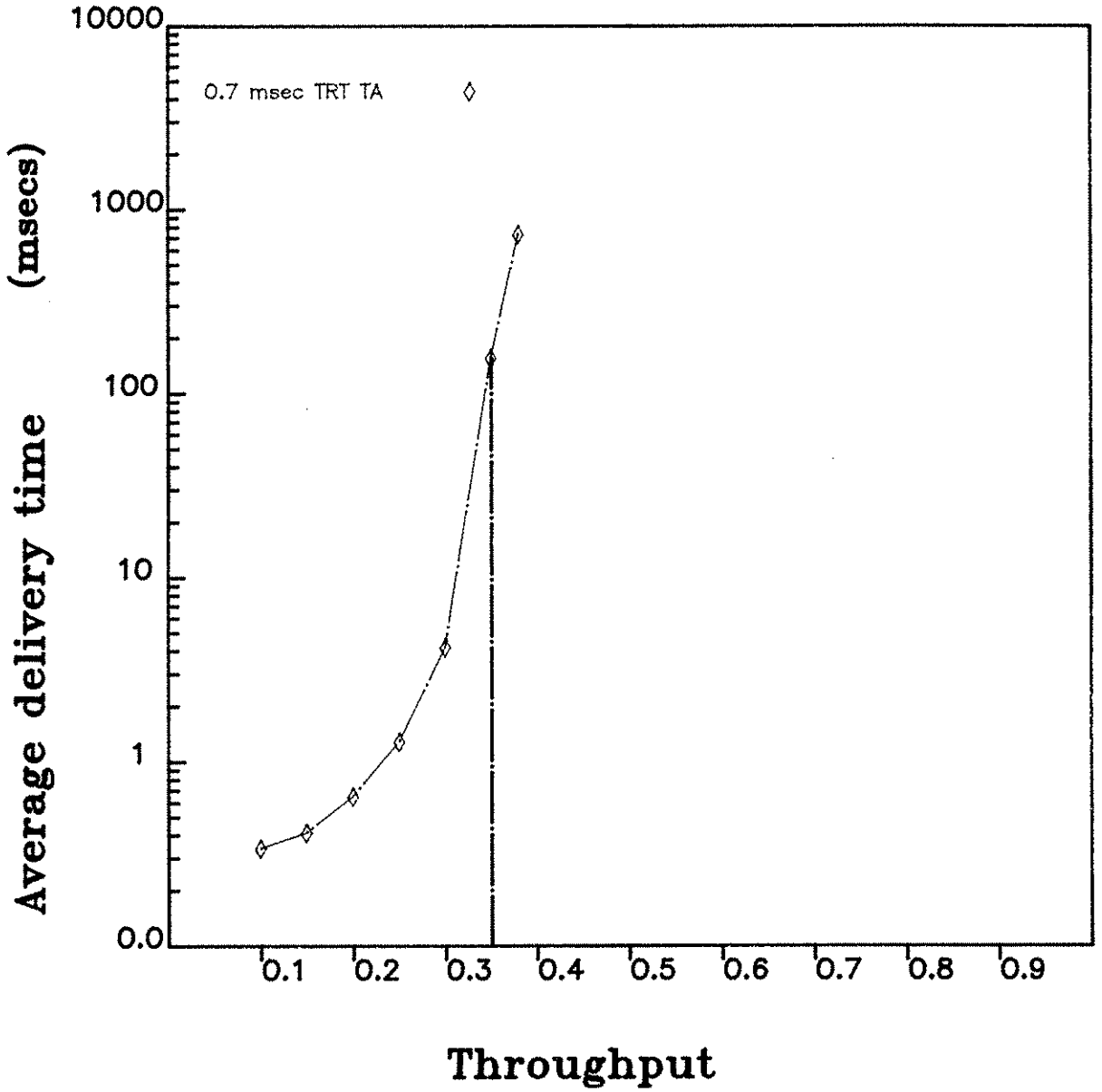


Figure 3.9
Average delivery time for Time_Available class

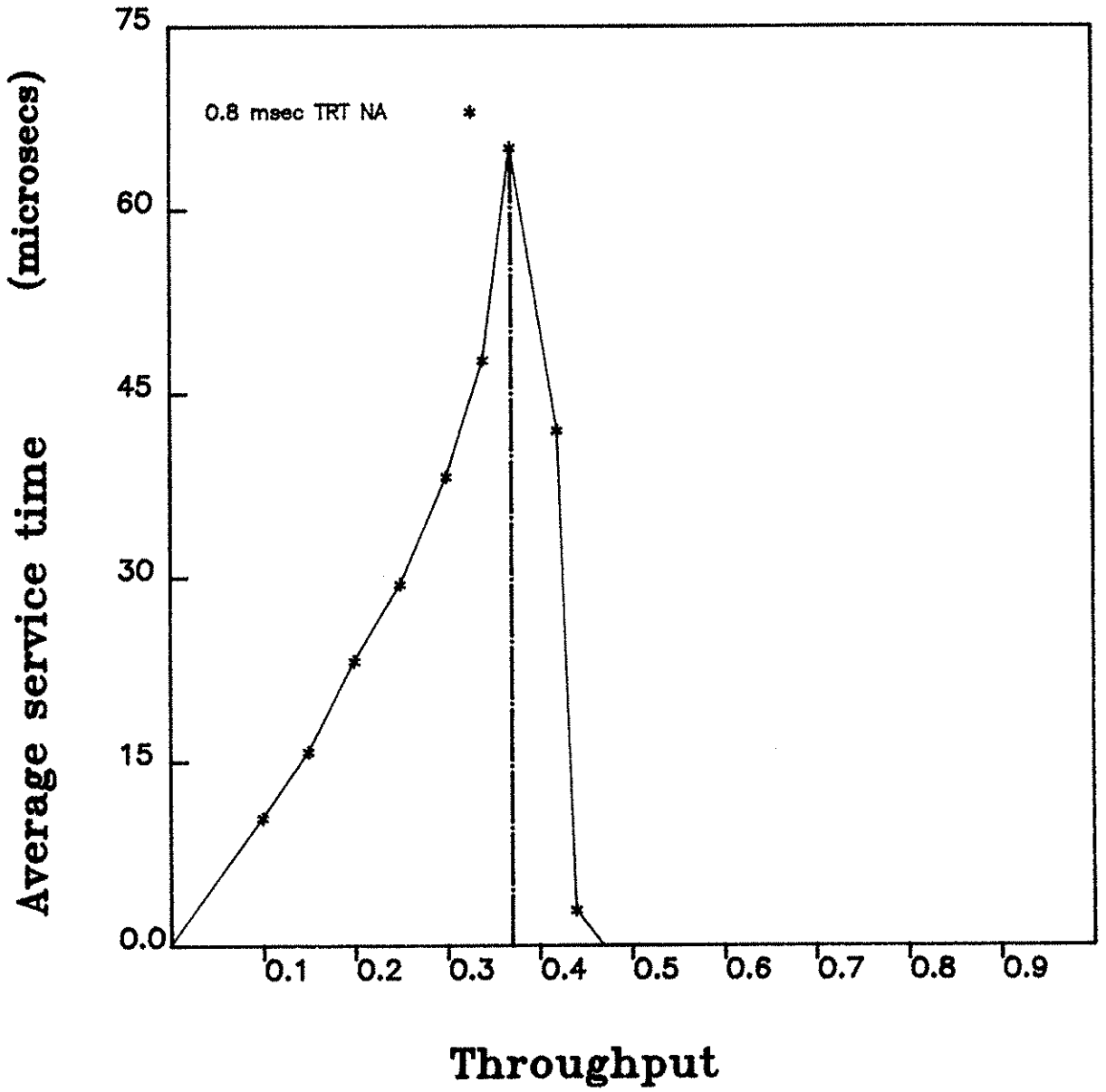


Figure 3.10
Average service time for Normal_Asynchronous class

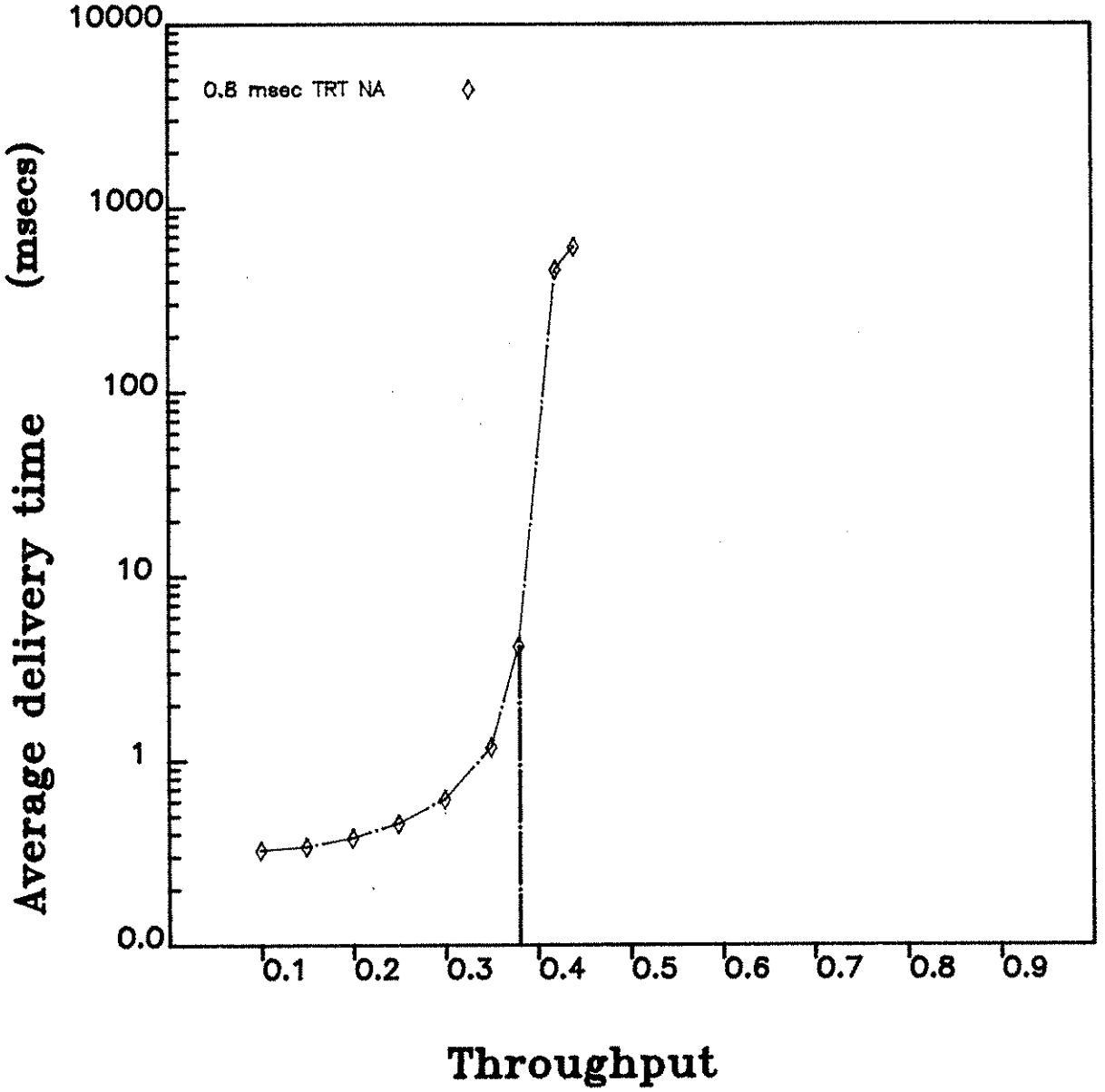


Figure 3.11
Average delivery time for Normal_Asynchronous class

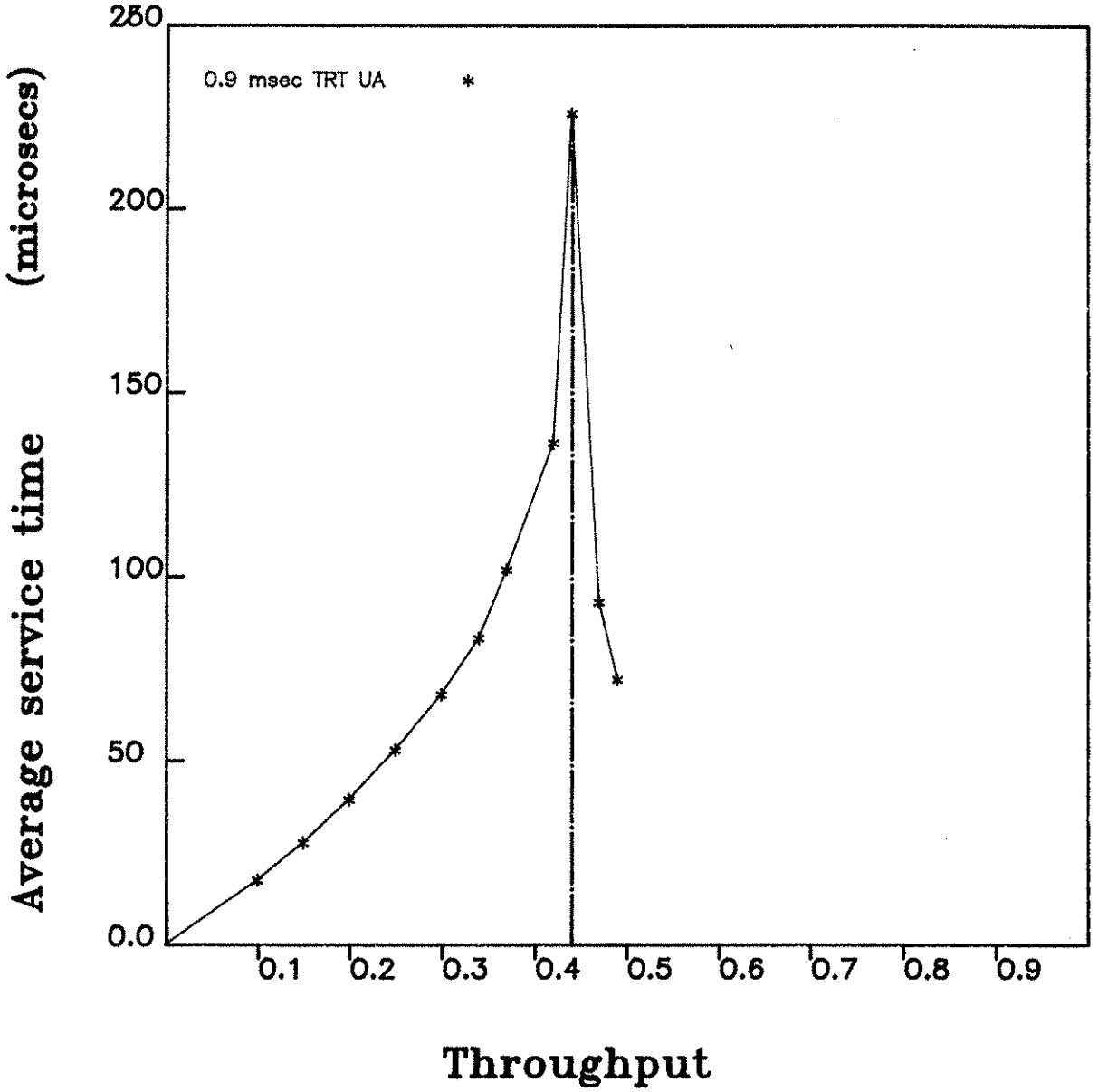


Figure 3.12
Average service time for Urgent_Asynchronous class

0.44. The delivery time of messages from the Urgent_Asynchronous access_class is reasonably low up to a throughput of about 0.44. Beyond this value the delivery time increases exponentially due to reduced service and hence exponentially growing queue lengths. Figure 3.13 illustrates this. The service is reduced because the token cycle time has exceeded the TRT_{ua} in this region. This illustrates that simulation results agree very closely with the expected result in section 3.4.1 for the Urgent_Asynchronous access_class.

Thus this study illustrates that choosing the TRT values in the order $TRT_{ua} > TRT_{na} > TRT_{ta}$ ensures the proper implementation of the priority feature. Also by suitably choosing the TRT value at each access_class, a designer can decide the range of throughput over which an access_class receives optimum service and hence messages do not suffer large queueing delays.

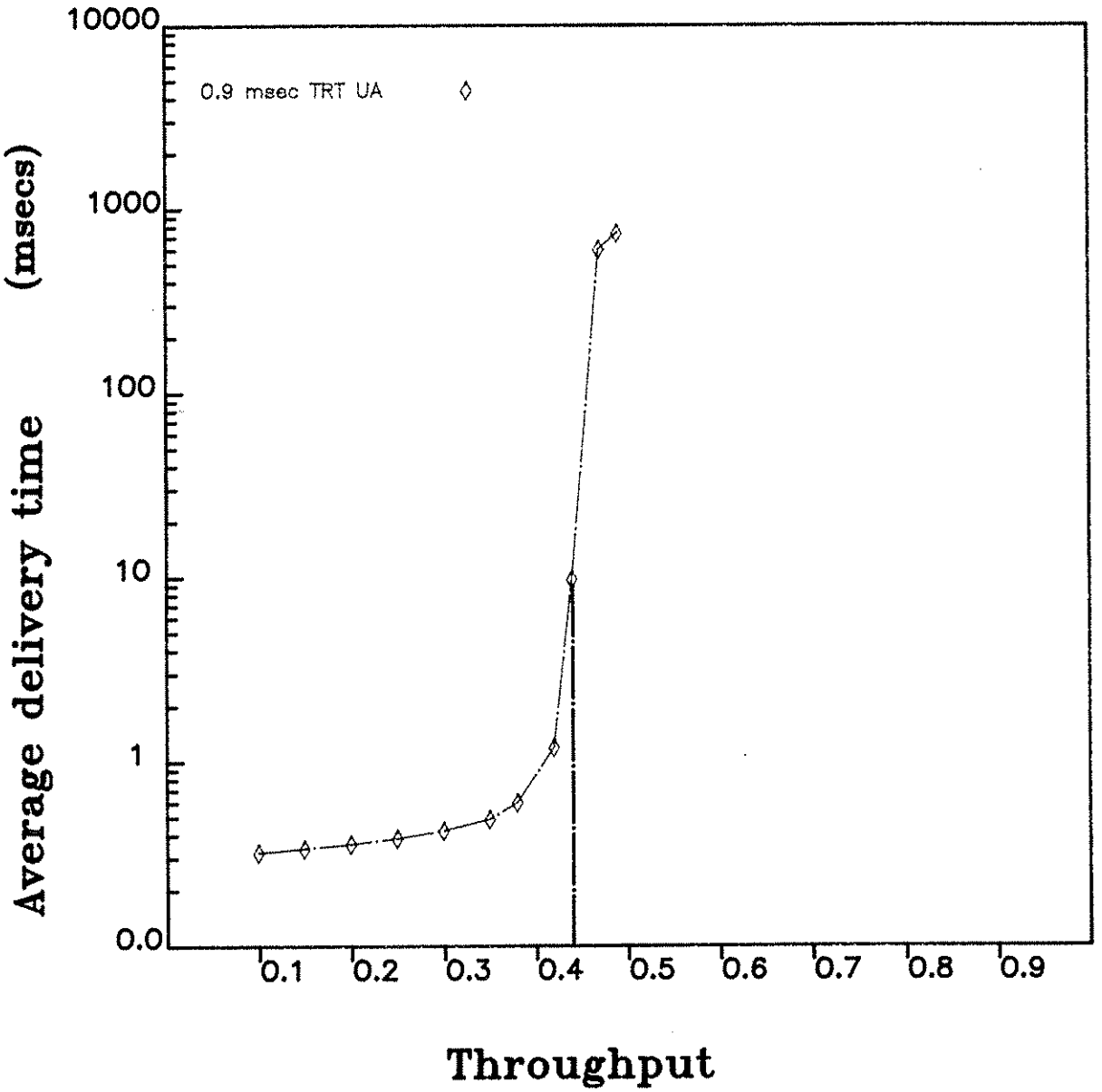


Figure 3.13
Average delivery time for Urgent_Asynchronous class

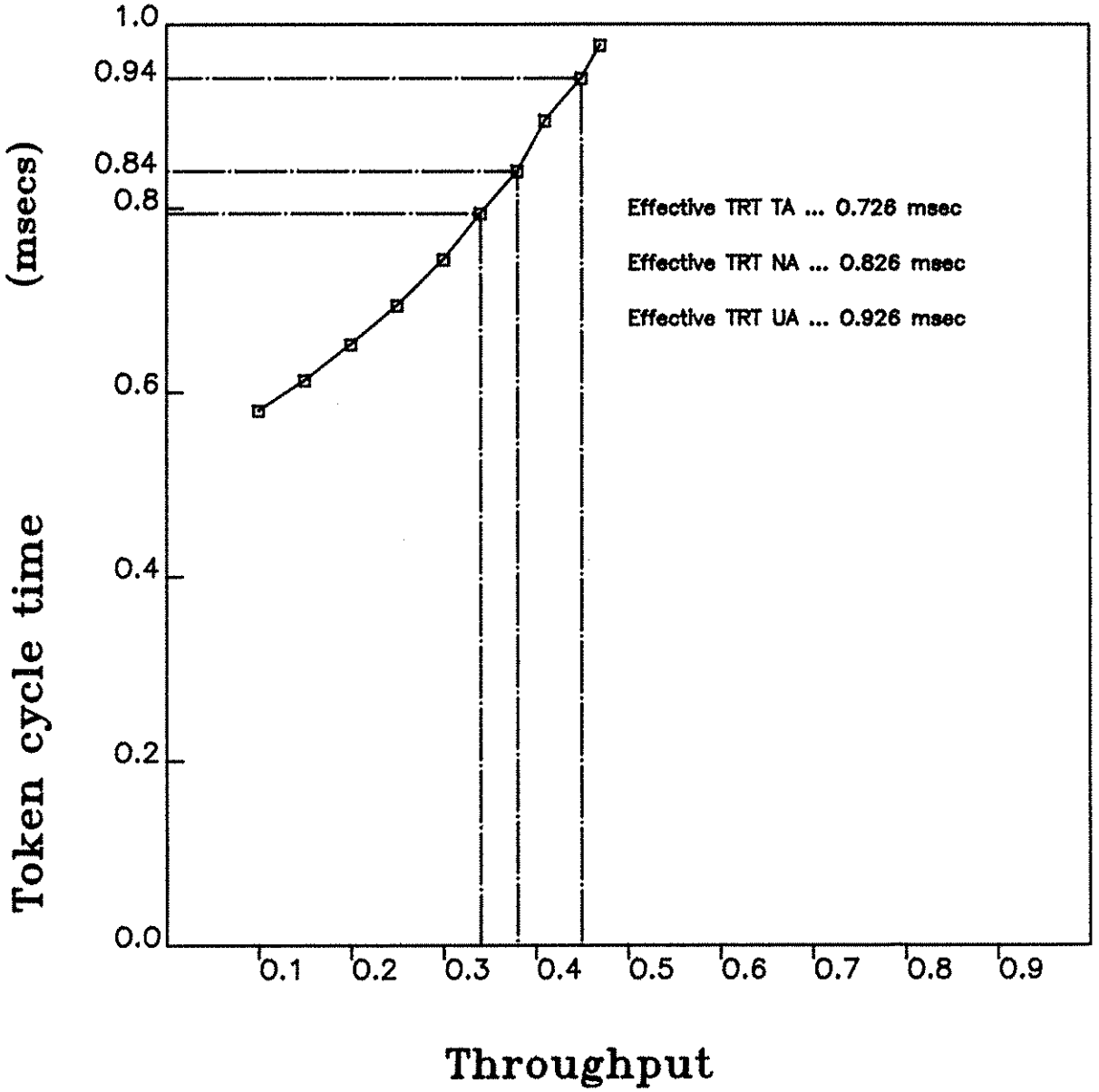


Figure 3.14
Token cycle time

CHAPTER 4

DYNAMIC RING MEMBERSHIP

This chapter discusses the study and analysis of the performance of a dynamic 802.4 network, i.e., a network of stations which can join and leave the ring as dictated by their traffic. The role played by several variables and timers in allowing the stations to be transient ring members has been discussed in the first section of this chapter. An analytic expression for the token cycle time for a network with a single transient ring member has been developed. The delivery times of messages from the station which is a transient member also has been predicted analytically. Network configurations with a single station joining and leaving the ring as dictated by the traffic in its queues have been simulated to observe the effect of transient ring membership on the network. In the last section of this chapter the simulation results have been discussed, followed by a comparison to the expected values from the analytic model.

4.1. Variables and timers

4.1.1. Variables

In an IEEE 802.4 network, stations can join and leave the logical ring as dictated by the traffic in their queues. Specific variables and timers are used to implement this feature. The following variables at each station take part in allowing a station to be a transient member :

1) **max_inter_solicit_count (MIC) :**

This variable can take on values in the range 16 to 255. This variable in addition to the ring_maintenance_timer determines how often a station opens

response_windows to solicit successors into the ring. A station solicits successors into the ring once in every MIC number of token cycles.

To prevent all stations in the ring from opening response_windows during the same token cycle the actual value used for the max_inter_solicit_count is changed by each station by rerandomizing the least significant two bits of the variable at least every 50ms or after every use.

2) **ring_maintenance_timer_initial_value :**

This variable can take on values in the range 0 to 2^{21} - 1 octet_times. This determines the initial value of the ring_maintenance token_rotation_timer. A large value will cause the station to solicit successors immediately on entry into the ring.

3) **any_send_pending :**

This is a Boolean variable which is set "TRUE" if the station has enqueued messages, otherwise it is set "FALSE".

4) **in_ring_desired :**

A Boolean variable which is set "TRUE" if the station desires to join the logical ring and has messages pending transmission.

5) **inter_solicit_count (IC) :**

This takes on values in the range 0 to 255. The value of this variable determines when the station opens response_windows. Before passing the token to its successor, every station decrements the value of the inter_solicit_count. If this value is zero and if the token cycle time is less than *max_ring_maintenance_rotation_time*, then the station opens response_windows to solicit successors. Otherwise it passes the token to its successor. If the value is zero and there is no response during this

time interval, then this variable is reset to max_inter_solicit_count and another count down process is started. However if any response is heard during the response_window then the inter_solicit_count remains at zero until such a token cycle that there is no response for the entire response_window interval.

6) **heard :**

This is a three state variable used in the **await_response** state. The three states are

a) **nothing :**

The sending station has heard nothing since beginning the resolution process.

b) **collision :**

Multiple stations have responded to the call for logical ring membership.

c) **successor :**

A valid set_successor frame has been heard. Hence the contention process has been resolved.

7) **max_pass_count :**

This has a value equal to half the station's address length in bits plus one (equal to 9 for a 16-bit address length). This is used to limit the number of token contention cycles. After passing through max_pass_count number of cycles, the station abandons the resolution process if a single contender cannot be resolved.

8) **contend_pass_count :**

This variable takes on values in the range 0 to max_pass_count. This is used as an index to select the address bits of a station during the contention process.

9) **resolution_pass_count :**

This is used to count the number of times a station tries to resolve the contention among multiple contenders. This takes on values in the range 0 to max_pass_count. If max_pass_count cycles have passed without resolution then the token holding station gives up the resolution process.

4.1.2. Timers

1) **response_window_timer :**

When a station sends a solicit_successor frame to solicit successors into the logical ring, this timer counts down from a value equal to the number of response windows opened after the transmission of the solicit_successor frame. Hence this timer determines the duration of time for which the station awaits for responses from other stations.

2) **contention_timer :**

While contending for ring membership, each station listens in the **demand_in** state for a response for its request for membership for a duration equal to the duration of this timer. If the station hears a transmission during this time interval, then it assumes that a higher addressed station is contending for membership and drops out of the contention process. The above mentioned timers have a granularity of one slot_time.

4.2. Logical ring membership

Each station in the logical ring solicits successors into the logical ring when it is in the **pass_token** state and when its inter_solicit_count goes to zero. The station solicits successors by transmitting solicit_successor frame. One or two response_windows (refer to section 2.4.1 for a definition of response_window) follow the solicit_successor frame. The response_window timer is set to count this duration. Hence the station which is soliciting successors changes into the

await_response state and waits for a response for this duration. If nothing is heard for the entire duration of the timer, i.e., corresponds to the state "nothing" of the three state variable *heard*, then the station enters the **pass_token** state to pass the token to its successor. If a valid set_successor frame is heard then the station waits for the duration of the response_window to pass and then sends the token to its new successor. If noise is heard, i.e., corresponding to state "collision" of *heard*, then the station initiates a resolution process (refer to section 2.4.1) to resolve the contention for membership by multiple stations.

4.3. Analytic expression for token cycle time and delivery time

In the following sections analytic expressions are developed for the token cycle time and delivery time of a network with all stations being permanent members except one station which joins and leaves the ring as dictated by the traffic in its queues. The effect of a station's insertion and deletion after ring initialization on the token cycle time and the delivery time has been studied in this section.

4.3.1. Assumptions

The derivation is based on the following assumptions about the characteristics of the network:

1) There are N stations in the system with one station being a transient member.

2) Each station has traffic at the Synchronous access_class only.

3) Messages from all the stations are of constant length and take X_M seconds for transmission.

4) The message arrival at each station follows a Poisson process.

5) All stations in the ring have the same HPTHT setting set to the maximum allowed value of 52.43 msecs for a 10 Mbps bus.

6) All stations have identical traffic.

7) *max_ring_maintenance_rotation_time* is set to a very large value.

4.3.2. Analytic expression for token cycle time

The definitions of section 3.1.1 hold in this section also. Let $y \in S$ denote the single transient member. As per the specifications of the 802.4 protocol only one station can open the response_window for allowing y to join the ring. Let server $x \in S$ denote a static member of the ring which is responsible for opening of

response_windows for y to join the ring. Let MIC denote the value of the variable max_inter_solicit count at each station. The token cycle time seen by server x on the i^{th} token cycle is as given by equation (1) section 3.2.3:

$$TC_{x,i} = TCT_{x,i} + TS_{x,i}. \tag{1}$$

The time interval during which the token is away from x can be expressed as the sum of

1) $(N-1)\cdot X_T$;

2) sum of the i^{th} service period at all servers other than x;

3) overhead due to the time spent on soliciting successors during the i^{th} token cycle. This is negligible in this analysis as only one station is assumed to be a transient member.

4) the service period at server y and the token pass from server y. There are two possibilities to consider the service time at y and the token pass from y

a) If y was out of ring during the $(i-1)^{th}$ token cycle, then it can contend for membership on the i^{th} token cycle, if messages have arrived during the $(i-1)^{th}$ token cycle.

The expected number of arrivals that have arrived at y during the $(i-1)^{th}$ token cycle will be served with a probability $\frac{1}{MIC+1}$, i.e., IC_x has gone to zero and y has won the contention for ring membership. The probability that y wins the contention is equal to one because it is a single contender and is sure to win the contention. Hence $TCT_{x,i}$ can be expressed as

$$TCT_{x,i} = (N-1)\cdot X_T + \sum_{s \in S} TS_{s,i} + X_M \cdot \lambda_s \cdot TC_{x,i-1} \cdot \frac{1}{MIC+1} \cdot 1 + X_T \cdot \frac{1}{MIC+1} \tag{2}$$

for all $s \neq x$.

Hence equation (1) can be expressed as

$$TC_{x,i} = (N-1)\cdot X_T + \sum_{s \in S} TS_{s,i} + X_M \cdot \lambda_s \cdot TC_{x,i-1} \cdot \frac{1}{MIC+1} \cdot 1 + X_T \cdot \frac{1}{MIC+1} \cdot 1 + TS_{x,i} \quad (3)$$

for all s $\neq$ x.

Assuming an average service time the average token cycle time can be expressed as

$$\overline{TC}_x = (N-1)\cdot X_T + \sum_{s \in S} \overline{TS}_s + X_M \cdot \lambda_s \cdot \overline{TC} \cdot \frac{1}{MIC+1} + X_T \cdot \frac{1}{MIC+1} + \overline{TS}_x. \quad (4)$$

The traffic has been assumed to be identical at all stations. Hence the average service time is the same for all stations. Substituting for the average service time from equation (16) section 3.2.4 (for Synchronous access_class only) the above equation can be expressed as

$$\overline{TC} = (N-1)\cdot X_T + (N-1)\cdot \lambda_s \cdot X_M \cdot \overline{TC} + \lambda_s \cdot X_M \cdot \frac{1}{MIC+1} \cdot \overline{TC} + X_T \cdot \frac{1}{MIC+1}. \quad (5)$$

The above equation can be rewritten as

$$\overline{TC} = \frac{(N-1)\cdot X_T + \frac{X_T}{MIC+1}}{1 - ((N-1)\cdot \lambda_s \cdot X_M + \lambda_s \cdot X_M \cdot \frac{1}{MIC+1})}. \quad (6)$$

b) If y was a member on the $(i-1)^{th}$ token cycle and has enqueued messages then it is a ring member on the i^{th} token cycle.

Rewriting equation (3) with y in ring

$$TC_{x,i} = N \cdot X_T + \sum_{s \in S} TS_{s,i} + TS_{x,i} + TS_{y,i}. \quad (7)$$

As the average service time is identical for all stations, substituting for this value from equation (16) section 3.2.4, and rearranging, the average token cycle time can be expressed as

$$\overline{TC} = \frac{N \cdot X_T}{1 - N \cdot X_M \cdot \lambda_s}. \quad (8)$$

4.3.3. Analytic expression for delivery time

4.3.3.1. Token cycle time less than message interarrival time

In a dynamic token passing ring stations contend for membership upon the arrival of a message in their queues. In order to calculate the average delivery time of a message from station y it can be expressed as the sum of the wait time involved in becoming a ring member, i.e., ring entrance delay, and the mean waiting time for transmission after receiving the token, i.e., it can be decomposed into the wait time to receive the token and the wait time after receiving the token [Fuhrmann 84]. This can be expressed as

$$D_M = D_{re} + Q_M. \tag{9}$$

Depending on the value of the inter_solicit_count a station has to wait between 0 and MIC number of token cycles to become a member of the logical ring (if a single contender). When the station y is out of the ring it expresses the desire to become a ring member soon after receiving the first message. Depending on the value of the IC of the station which can open a response_window for this station, the station has to wait on the average $1/2 \cdot MIC$ cycles to receive the token, i.e., the sum of integers from zero to MIC averaged over (MIC plus one) multiplied with the average token cycle time is the wait involved to receive the token after the arrival of the first message. This can be expressed as

$$D_{re} = 1/2 \cdot MIC \cdot \overline{TC}. \tag{10}$$

During this time interval the number of messages that have arrived at y can be calculated as

$$n = \lambda_s \cdot 1/2 \cdot MIC \cdot \overline{TC}. \tag{11}$$

The first packet that arrives in this time interval has to wait on the average $1/2 \cdot MIC \cdot \overline{TC}$ secs. The second packet waits $1/2 \cdot MIC \cdot \overline{TC} + X_M$, the third packet waits $1/2 \cdot MIC \cdot \overline{TC} + 2 \cdot X_M$ and so on. Hence the average wait time after the token

arrival is Q_M and is equal to

$$Q_M = (n-1)\cdot\frac{n}{2}\cdot\frac{1}{n}\cdot X_M = \frac{(n-1)}{2}\cdot X_M . \tag{12}$$

Hence the average delivery time can be expressed as

$$D_M = D_{re} + Q_M = \frac{MIC\cdot\overline{TC}}{2} + \frac{(n-1)}{2}\cdot X_M . \tag{13}$$

4.3.3.2. Token cycle time greater than message interarrival time

In this region of offered load messages are arriving at y on the average more frequently than the token. As more messages are enqueued per cycle, the station can transmit more messages per token and hence stays for a longer time interval in the ring. Hence the ring entrance delay does not play a dominant role in this region. Let T denote the total time y is in the system. The number of token cycles completed in T seconds is given by

$$N_{total} = \frac{T}{\overline{TC}}. \tag{14}$$

The probability that y has no message arrivals during an average token cycle is given by

$$P_{no\_arrival} = e^{-\lambda_x\cdot\overline{TC}}. \tag{15}$$

Hence the expected number of token cycles that y is out of the ring is given by

$$N_{out} = N_{total}\cdot P_{no\_arrival}. \tag{16}$$

Hence the number of cycles y has been in ring is given by

$$N_{in} = N_{total} - N_{out}. \tag{17}$$

Station x opens response_windows every MIC number of token cycles. Hence y is allowed to join the ring $\frac{N_{out}}{MIC}$ number of times. But y contends for membership only in the event of a message arrival. Hence the probability that a message has arrived in the time interval $MIC\cdot\overline{TC}$ is given by $(1-e^{-\lambda_x\cdot MIC\cdot\overline{TC}})$. Hence the expected number of times y could have joined the ring is given by

$$y_{n\_join} = \frac{N_{out}}{MIC} \cdot (1 - e^{-\lambda_s \cdot MIC \cdot \overline{TC}}). \quad (18)$$

The expected number of messages y could have transmitted over a time interval of T sec is given by the product of the number of token cycles y has been in ring and the average number of messages transmitted during an average token cycle, i.e.,

$$M_{total} = \lambda_s \cdot \overline{TC} \cdot N_{in}. \quad (19)$$

The delivery of messages that arrive while the station is contending for ring membership is delayed by an amount equal to the ring entrance delay. But this is only for a few messages, as the station stays in ring for a longer time interval. Hence the calculation of average delivery time in this region is the sum of two terms:

1) The delivery time of messages that have arrived while the station is waiting to become a ring member.

Each of the messages that arrive while the station is waiting to become a ring member experiences an average delay as given by equation (13). The number of messages that have arrived while the station is waiting to join multiplied by the number of times the station has joined the ring, can be expressed as

$$M_{wait} = y_{n\_join} \cdot \lambda_s \cdot \frac{MIC \cdot \overline{TC}}{2}. \quad (20)$$

M_{wait} is the number of messages that experience the average delay shown in equation (13).

2) The delivery time of messages that have arrived while the station is in ring.

This delay is the same delay as experienced by a message in a static ring. M. E. Ulug [Ulug 84] shows that delay to be

$$W = \frac{T_r}{2} + \frac{(s-1) \cdot (x+r)}{2}. \quad (21)$$

The term (x+r) denotes the service time which in the above model is X_M. The term T_r denotes the token cycle time. The first term denotes the mean waiting

time to receive the token. The second term is the mean waiting time after the token is received. The number of messages that have arrived during the time interval T_r is given by s which in the above model is $\lambda_s \cdot \overline{TC}$. Each message on the average experiences a delay as given by the above equation. Hence the average delivery time of a message from a transient member in this region of offered load can be expressed as

$$D_M = \frac{M_{wait} \cdot (\frac{MIC \cdot \overline{TC}}{2} + \frac{(n-1)}{2} \cdot X_M) + M_{in} \cdot (\frac{\overline{TC}}{2} + \frac{(\lambda_s \cdot \overline{TC} - 1)}{2} \cdot X_M)}{M_{total}}. \quad (22)$$

4.4. Simulation results

4.4.1. Token cycle time

Network configurations with 32 stations with one station being a transient member and parameters set in accordance with the assumptions in section 4.3.1 were simulated. Static configurations with 32 stations were simulated in order to compare the average delivery times of the 2 configurations. In addition the following is the data pertaining to the simulated configurations:

N = 32; MIC = 16,

Size of data frame = 272 bits including framing,

Size of token = 112 bits,

X_M = 0.0000272 seconds = 27.2 microseconds,

X_T = 0.0000162 seconds = 16.2 microseconds.

Figure 4.1 illustrates the analytic predictions and simulation results of the variation of the token cycle time with offered load. Up to an offered load of sixty percent the token cycle time of the ring (considering only the static ring members) is less than the message interarrival time at y. For example at 30 percent offered load the token cycle time of a static ring of N-1 members from equation (8) is

$$\overline{TC} = \frac{(N-1)\cdot X_T}{1-(N-1)\cdot\lambda_s\cdot X_M} = \frac{31\cdot 16.2\cdot 10^{-6}}{1-31\cdot 344.64\cdot 27.2\cdot 10^{-6}} = 0.7079 \text{ msecs.}$$

The message interarrival time at station y at 30 percent offered load is $\frac{1}{344.64}$ messages/second = 2.9015 msecs. Hence the token is arriving on the average more frequently than the messages. Hence y is out of the logical ring more often than it is in the logical ring. Hence equation (6) has been used for calculation of the token cycle time of the dynamic ring. At 30 percent offered load the token cycle time of the dynamic ring has been calculated from equation (6) as

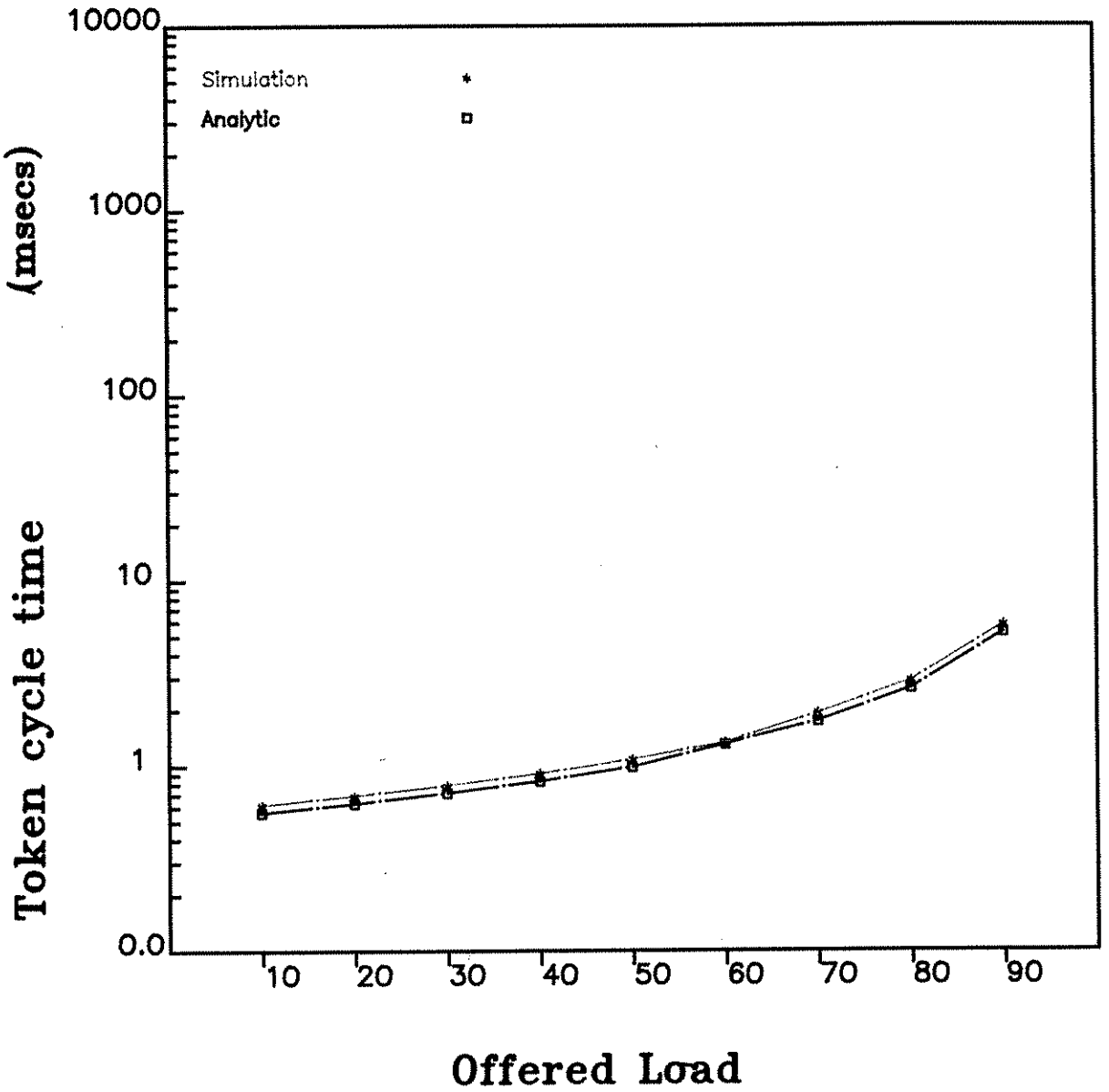


Figure 4.1
Token cycle time

$$\overline{TC} = \frac{X_T \cdot ((N-1) + \frac{1}{MIC})}{1 - \lambda_s \cdot X_M \cdot ((N-1) + \frac{1}{MIC})} = \frac{16.2 \cdot 10^{-6} \cdot (31 + \frac{1}{16})}{1 - 344.64 \cdot 27.2 \cdot 10^{-6} \cdot (31 + \frac{1}{16})} = 0.7099 \text{ msecs.}$$

At offered loads of 60 percent and above the interarrival time of messages at y is on the average higher than the token circulation time. For example at 70 percent offered load the token cycle time of a ring of 31 members from equation (8) is

$$\overline{TC} = \frac{(N-1) \cdot X_T}{1 - (N-1) \cdot \lambda_s \cdot X_M} = \frac{31 \cdot 16.2 \cdot 10^{-6}}{1 - 31 \cdot 804.16 \cdot 27.2 \cdot 10^{-6}} = 1.559 \text{ msecs.}$$

The message interarrival time at y at this offered load is $\frac{1}{804.16} = 1.243$ msecs. Hence the messages are arriving more frequently than the token. Hence equation (8) has been used for the calculation of token cycle time beyond offered loads of 60 percent. For example at 70 percent offered load the token cycle time from equation (8) is

$$\overline{TC} = \frac{N \cdot X_T}{1 - N \cdot X_M \cdot \lambda_s} = \frac{32 \cdot 16.2 \cdot 10^{-6}}{1 - 32 \cdot 27.2 \cdot 10^{-6} \cdot 804.16} = 1.7289 \text{ msecs.}$$

It can be observed from the figure that the analytic predictions agree very closely with the simulation results.

4.4.2. Average delivery time

Figure 4.2 illustrates the variation of delivery time of messages from the transient station in a dynamic configuration of 32 stations and the average delivery time of messages in a static configuration of 32 stations. It can be observed from the graph that the analytic predictions agree very closely with the simulation results at lower offered loads. At offered loads of up to 60 percent the token is arriving more frequently than the messages. For example at 40 percent offered load the token cycle time has been calculated as 0.8225 msecs. The message interarrival time is $\frac{1}{459.52} = 2.176$ msecs. Hence equation (13) has been used for the calculation of the average delivery time of messages in this region. From equation (10)

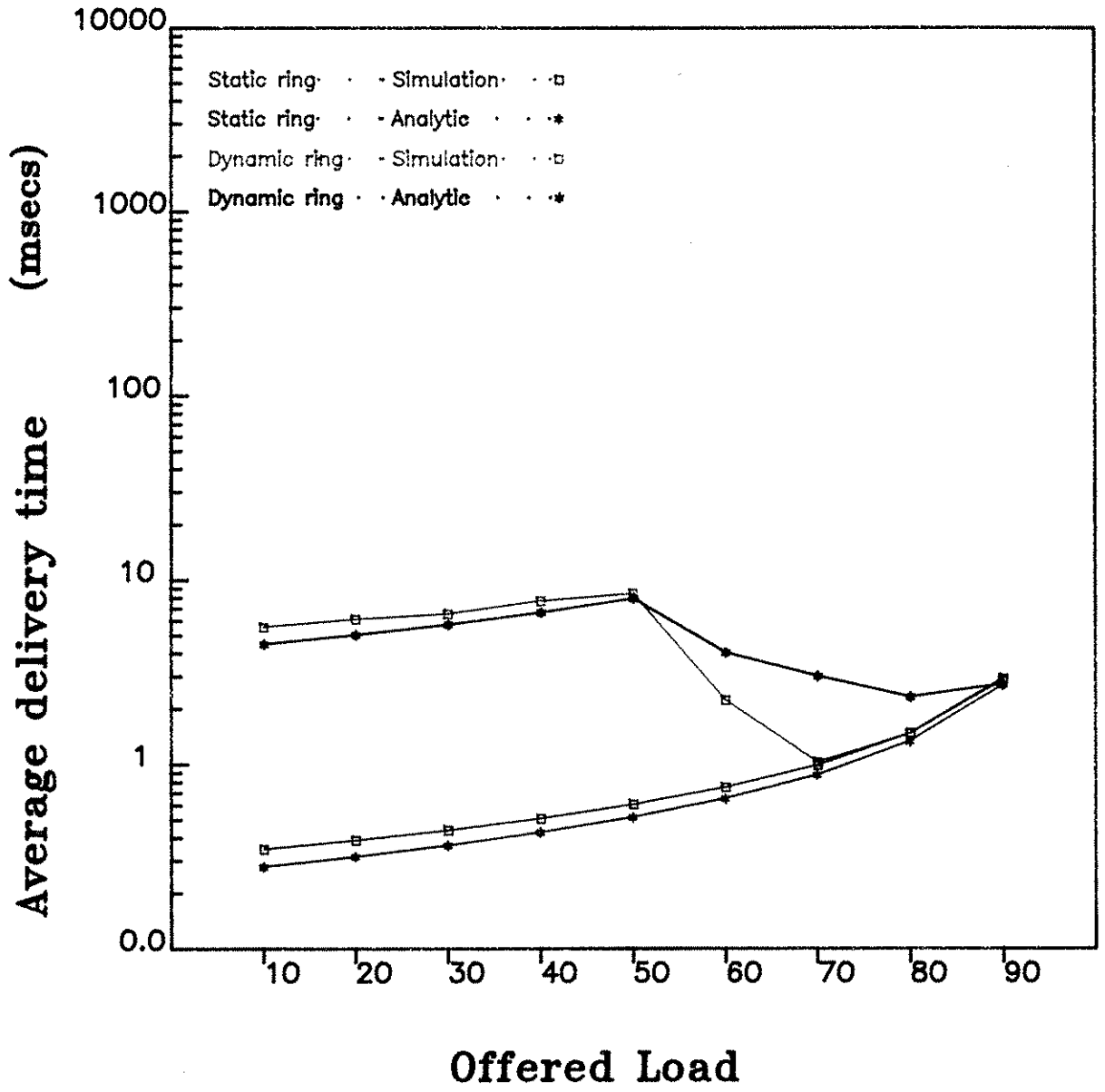


Figure 4.2
Average delivery time

$$D_{re} = \frac{MIC \cdot \overline{TC}}{2} = \frac{16 \cdot 0.8225}{2} = 6.58 \text{ msecs.}$$

From equation (11)

$$n = \lambda_s \frac{MIC \cdot \overline{TC}}{2} = 459.52 \cdot \frac{16 \cdot 0.0008225}{2} = 3.023 \text{ messages.}$$

Hence substituting the above value in equation (12)

$$Q_M = \frac{n-1}{2} \cdot X_M = 0.02752 \text{ msecs.}$$

Then substituting the values of D_{re} and Q_M in equation (13)

$$D_M = D_{re} + Q_M = 6.58 + 0.02752 = 6.607 \text{ msecs.}$$

The delivery time of a message from the same station when it is a static member is given by equation (21)

$$W = \frac{T_r}{2} + \frac{(s-1) \cdot (x+r)}{2} = \frac{0.8639 \cdot 10^{-3}}{2} + \frac{(-0.603) \cdot (0.0000272)}{2} = 0.4237 \text{ msecs.}$$

The token cycle time used in the above equation has been calculated from equation (8).

At offered loads of 60 percent and above the messages are arriving more frequently than the token. Hence the station stays in ring for a longer time interval. Hence on the average a message does not experience the ring entrance delay. For example at 80 percent offered load from equation (19)

$$M_{total} = \lambda_s \cdot \overline{TC} \cdot N_{in} = 919.04 \cdot 0.0025931 \cdot 11202 = 26696 \text{ messages.}$$

From equation (20)

$$M_{wait} = y_{n\_join} \cdot \lambda_s \cdot \frac{MIC \cdot \overline{TC}}{2} = 71 \cdot 919.04 \cdot \frac{16 \cdot 0.0025931}{2} = 1353 \text{ messages.}$$

Hence using equation (22)

$$D_M = \frac{M_{wait} \cdot (\frac{MIC \cdot \overline{TC}}{2} + \frac{(n-1)}{2} \cdot X_M) + M_{in} \cdot (\frac{\overline{TC}}{2} + \frac{(\lambda_s \cdot \overline{TC} - 1)}{2} \cdot X_M)}{M_{total}}$$

$$= \frac{1353 \cdot (20.7 + 0.245) + 25342 \cdot (1.296 + 0.018)}{26696} = 2.309 \text{ msecs.}$$

The simulated value of the average delivery time at this offered load is 1.47 msecs. The departure of the analytic predictions from the simulation results at 70 percent and 80 percent is due to the fact that mean values have been used for the number of times a station could have joined the ring which is higher than the actual number of times a station has joined the ring. Hence the value obtained from equation (20) is contributing a larger value to the calculation of average delivery time. For the same reason equation (22) gives an approximate value of the delivery time for high values of MIC.

Thus it may be observed that at low offered loads a message from a transient member waits for a longer time to be delivered. This is due to the fact that the arrival of messages is infrequent and hence the station is out of the ring more often than it is in ring. Hence D_{re} plays a dominant role in the delivery time of a message. Hence at low offered loads better service can be obtained if a station is a static member as opposed to being a transient member. On the other hand at higher loads messages are arriving more frequently, hence more messages are being transmitted, hence the station stays in ring for a longer period of time. Hence the delivery time is very close to that observed in a static ring. The delivery time for a static ring was calculated to be 1.3143 msecs which is very close to the delivery time of 1.47 msecs for a message from a transient member.

CHAPTER 5

TRANSIENT LOADING

This chapter discusses the performance of the IEEE 802.4 protocol under transient loading conditions. The first section of this chapter discusses the role of the timers and packet sizes in the transmission of a transient load. Section 5.2 of this chapter discusses the various forms of transient loading. Section 5.3 gives the details of the reference configuration. Section 5.4 discusses the details of various network configurations that have been simulated to observe the effect of different forms of transient loading. Section 5.5 discusses the results of the simulated configurations.

5.1. Transmission of the transient load

In an 802.4 network, each station can transmit data when it is the token holder. The duration of a token cycle depends upon the number of messages transmitted at each station during that cycle of the token. The transmission of a load involving large data transfers causes an increase in the token cycle time. Succeeding token cycles will also be longer. If the token cycle time does not recover from the imposition of such a transient load, then queue lengths could grow very large at every imposition of a transient load. There is also a possibility of more than one station offering such huge loads on the network.

Depending on the amount of load to be transmitted, the MAC entity at a station offering the transient load may transmit the load as a single large packet (if the amount of load does not exceed the maximum packet size as specified by the protocol) or split the load into several packets. The load may be transmitted within one or several token cycles depending upon the residual time in the

token_hold_timer. The residual time in the token_hold_timer is dependent on the value of the HPTHT or the residual time in the token_rotation_timers (as decided by the priority of the load).

Let L_t denote the number of octet_times to transmit the load, if it is a single packet, or the transmission time of one packet in a multi-packet load. Let N denote the total number of packets in a multi-packet load. If t octet_times is the residual time in the token_hold_timer then a station can transmit for a time interval given by eff(t) (refer to section 3.2.1). Then the number of packets transmitted on the i^{th} token cycle, i.e., N_i, is given by

$$N_i = \frac{eff(t)}{L_t}. \qquad (1)$$

If N_i is less than N then the remaining packets are transmitted during subsequent token cycles depending on the residual time in the token_hold_timer.

As mentioned earlier, the MAC entity at each station can transmit the load as a few large messages or several smaller messages. The effect of both forms of loading on the network has been studied by simulation.

5.2. Forms of transient loading

The effect of transient loading on the network can be studied by imposing the load on the network in several ways namely

1) The transient load can be imposed on the network as several long messages transmitted from one station to another station. A load of about 0.6 Mbits has been considered for this study. The load has been split into 20 messages, each 32000 bits long, i.e., N has a value of 20.

2) The transient load can be imposed on the network as a large number of smaller messages. Thus a large message could be split up into several smaller messages. For this form of loading, 2508 messages, i.e., N is equal to 2508, each 160

bits in length has been chosen.

5.3. Reference configuration

The reference configuration has 32 stations in the logical ring with all stations in the ring having identical traffic at all four access_classes. The High_Priority_Token_Hold_Time has been set to the maximum value. The Target_Rotation_Time values (TRT) of Urgent_ Asynchronous, Normal_Asynchronous and Time_Available access_classes have been set to 0.9, 0.8 and 0.7 msecs respectively. The different network configurations simulated for observing the effect of transient load are variations of the reference configuration.

5.4. Details of various configurations

5.4.1. Without timer restrictions

For this simulation, the HPTHT was set to the maximum value and the TRTs of the lower three access_classes were set to large values so that each station can transmit all its enqueued messages upon receiving the token. Two network configurations were simulated, one with a larger ring of 32 stations and another with a smaller ring of 8 stations, and one of the stations having a load of 20 messages of 32 kbits each. The messages were preloaded at the Urgent_Asynchronous access_class.

5.4.2. With timer restrictions

The effect of transmission of a transient load on a network consisting of stations with tight TRT settings has been studied. The effect of both forms of transient loading has been observed on a network of 32 stations with TRTs set to the same values as in the reference configuration. The transient load was transmitted from the Urgent_Asynchronous access_class.

5.4.3. Variation of the intensity of loading

In the case of the transient load being imposed in the form of a few large messages, messages from other stations in the network experience large delays due to the transmission of the transient load. The effect of transient load is observed by varying the intensity of the load at a particular offered load. The number of 32 kbit messages transmitted from the Urgent_Asynchronous access_class was varied from 20 down to 4 to observe the variation in delivery times of the messages from other stations in the network. All other parameters were set to values as in the reference configuration.

5.5. Discussion of the results

5.5.1. Variation of the token cycle time

5.5.1.1. Without timer restrictions

a) Eight Stations in the ring :

The variation of token cycle time for the network configuration discussed in section 5.4.1 is as shown in Figure 5.1. The token cycle time for this configuration increases initially to a very large value. This is because two long messages are being transmitted in that cycle. As can be seen from Figure 5.1 it reaches a value of 7 msec and remains around 4 msec for the entire duration of transmission of the transient load. The token cycle time remains at such high values for a duration of 23 msec and returns to its stable value 3 or 4 token cycles after completion of transmission of the transient load. This can be observed from Figure 5.2 which shows token cycle time vs simulation time in milliseconds. As the timers are set to large values, messages get transmitted during every cycle of the token and hence the token cycle time remains close to the time taken to transmit each 32 kbit message.

b) Thirty-Two Stations in the ring :

The variation in token cycle time can be observed from Figures 5.3 and 5.4.

It can be seen from Figure 5.3 that the token cycle time rises to a value of about 64 msec during the cycle when most of the 32 kbit messages get transmitted. Not all messages of the multi-packet load get transmitted during that cycle due to expiration of the token_hold_timer (as explained in section 5.1). It can be observed from the figure that after 2 token cycles there is another increase in the token cycle time, which is due to transmission of the remaining messages of the multi-packet load. It can be observed from Figure 5.3 that a transient load of

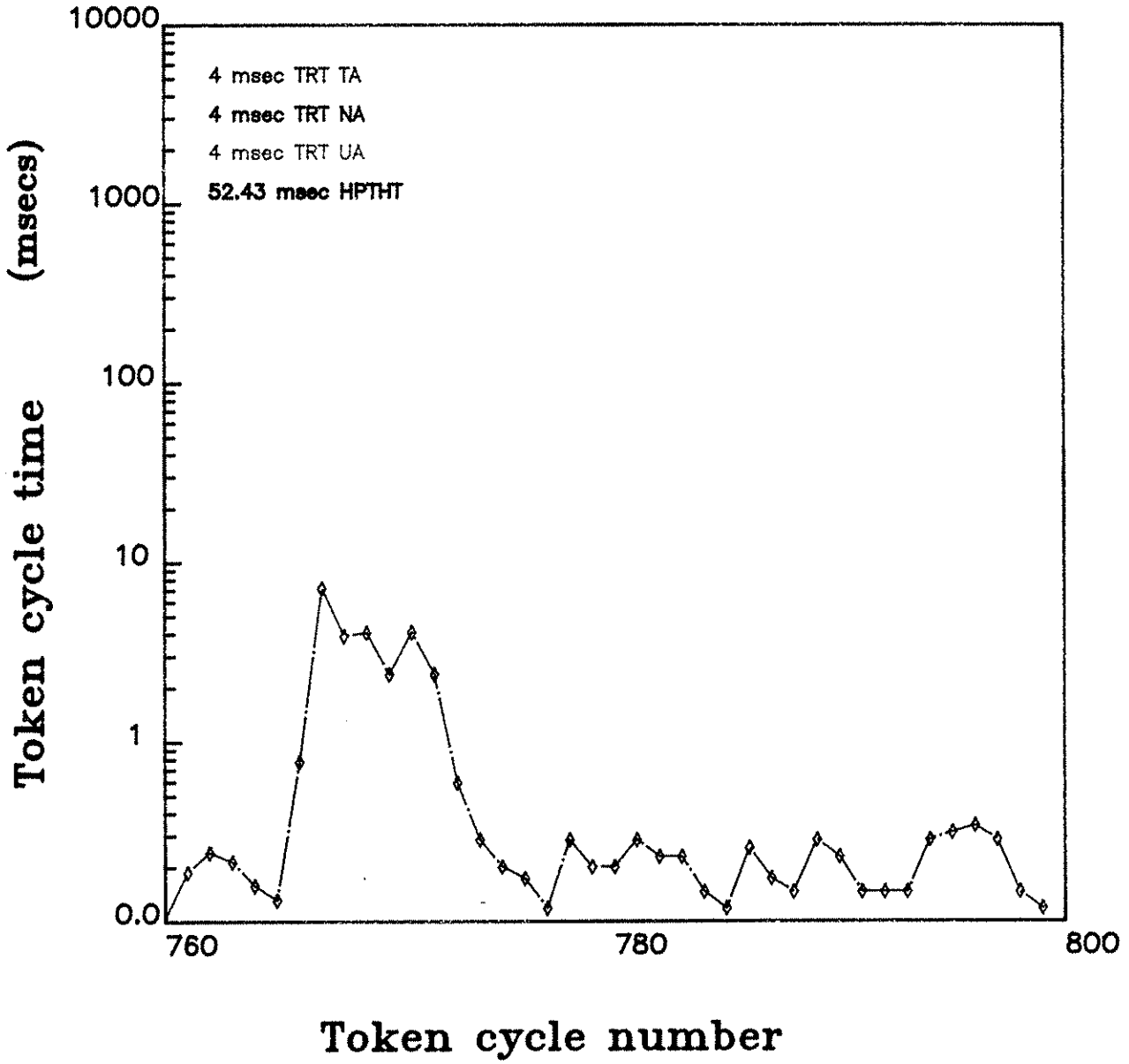


Figure 5.1
Token cycle time vs. token cycle number for 8 stations

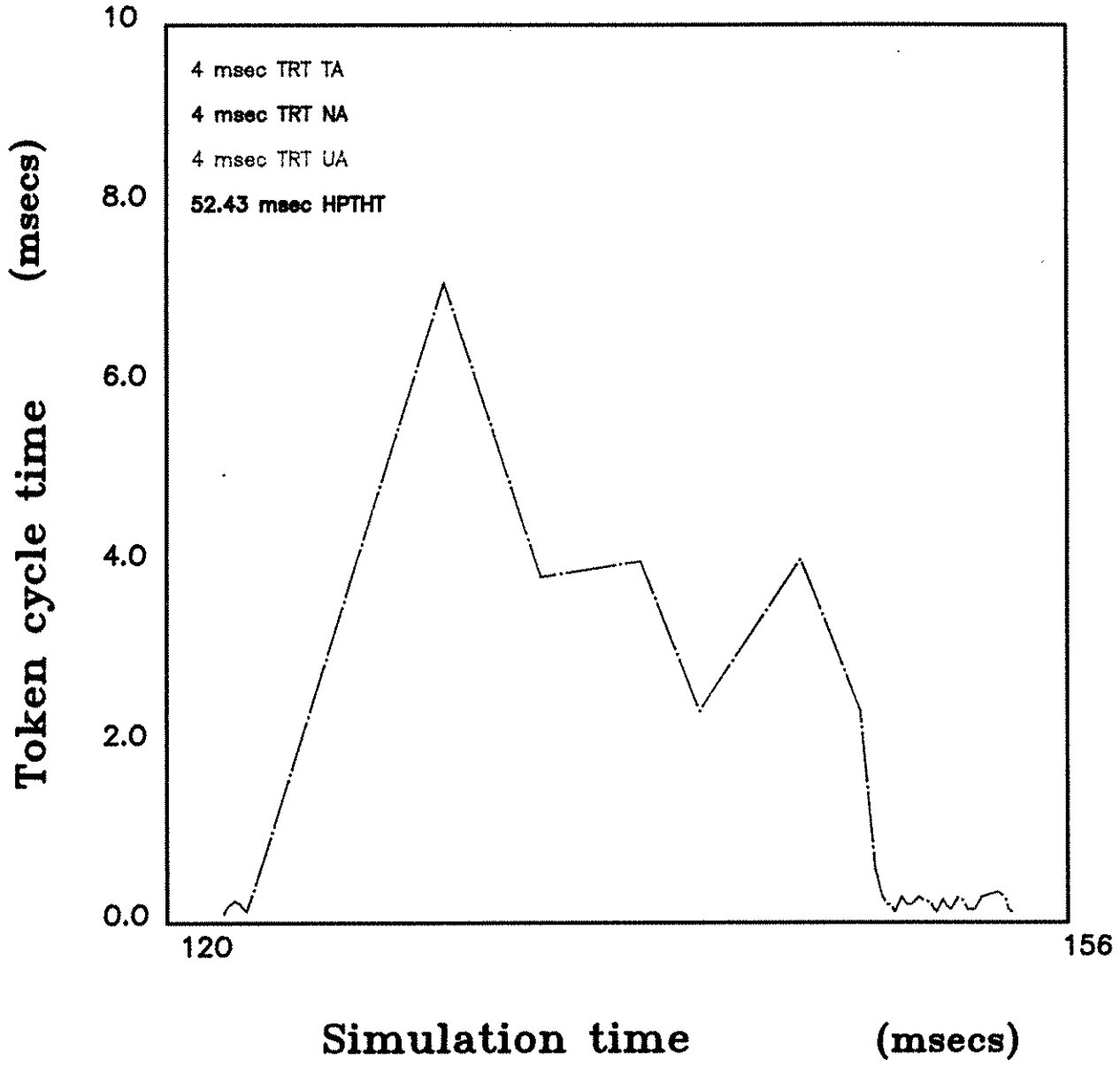


Figure 5.2
Token cycle time vs. simulation time for 8 stations

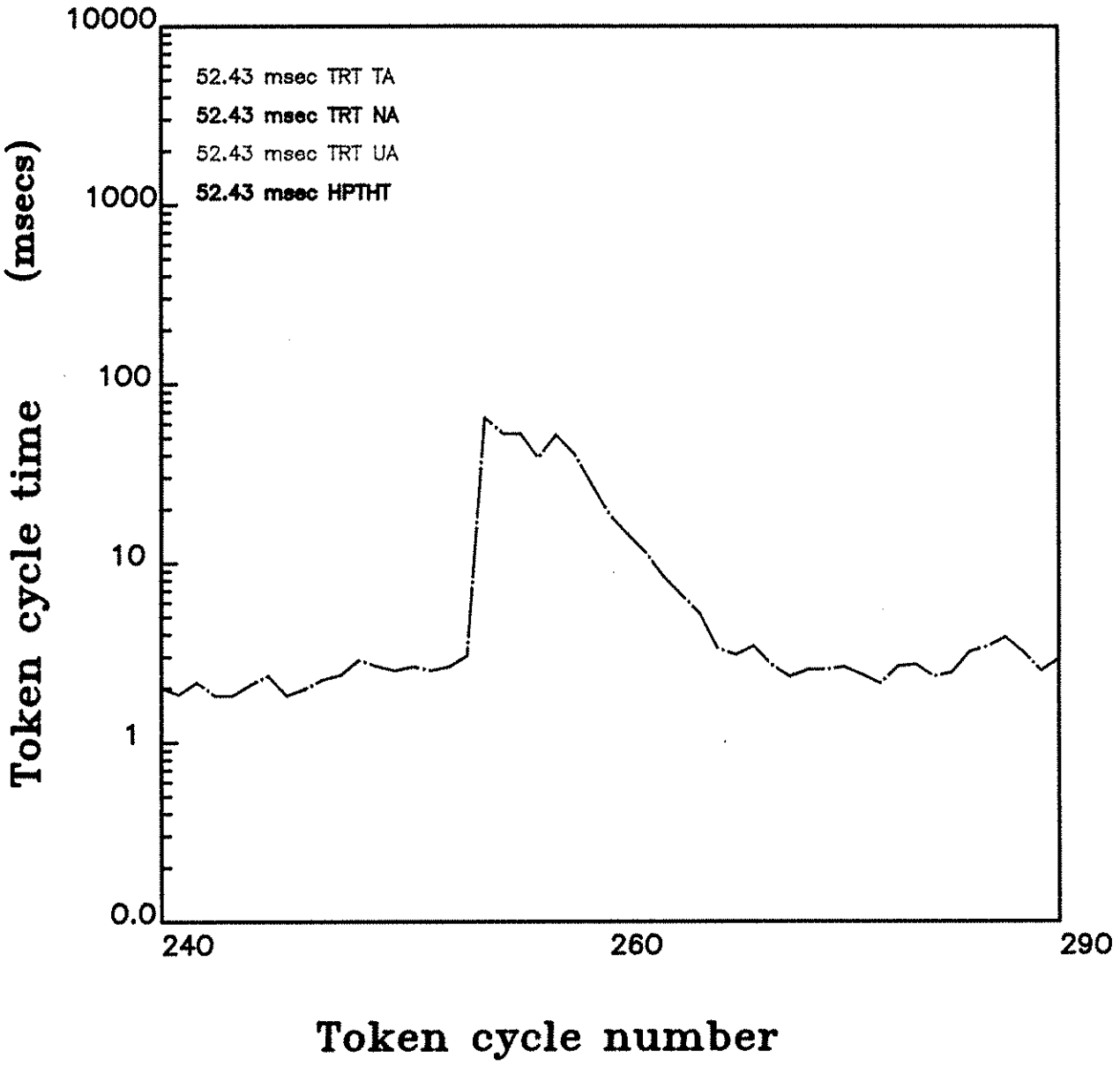


Figure 5.3
Token cycle time vs. token cycle number for 32 stations

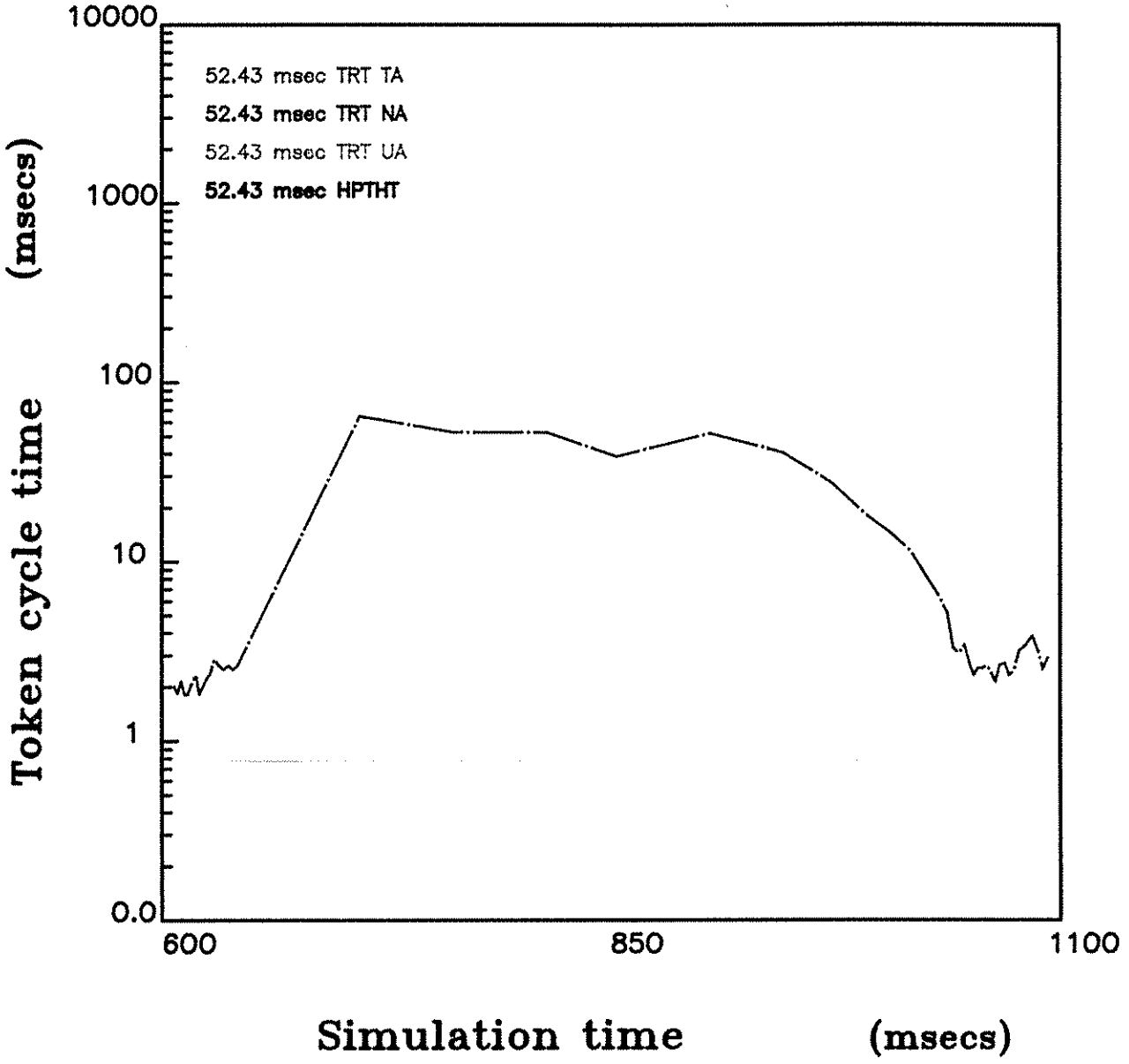


Figure 5.4
Token cycle time vs. simulation time for 32 stations

about 0.6 Mbits is transmitted in 3 token cycles and the token cycle time returns to its stable value after a duration of about 0.5 sec, i.e., after about 10 token cycles.

5.5.1.2. With timer restrictions

With TRTs set to smaller values, the token cycle time rises sharply and recovers within one token cycle after the transmission of the transient load. When the transient load is imposed on the network in the form of few longer messages the token cycle time increases sharply whenever a long message is being transmitted. This is because the transmission of a message once started will go on to completion even if the timers have expired. Hence the transmission of a long message will increase the token cycle time. This can be observed from Figure 5.5. After transmission of each 32 kbit message, the token cycle time returns to its normal value after just one cycle. This is because the time available for each station to transmit messages is limited. Hence an increase in the token cycle time after the transmission of the transient load leads to less residual time in the timers of other stations in the ring and hence less time for other stations to transmit. Hence the token cycle time returns to its normal value very quickly. This can be observed from Figure 5.5.

In the case where the transient load was imposed on the network as a large number of smaller messages, the token cycle time can be seen to be varying very slightly from its normal values. The token cycle time does not increase sharply in this case because the transient load has been split up into large number of smaller messages and hence the load gets distributed over several token cycles. This can be observed from Figure 5.6.

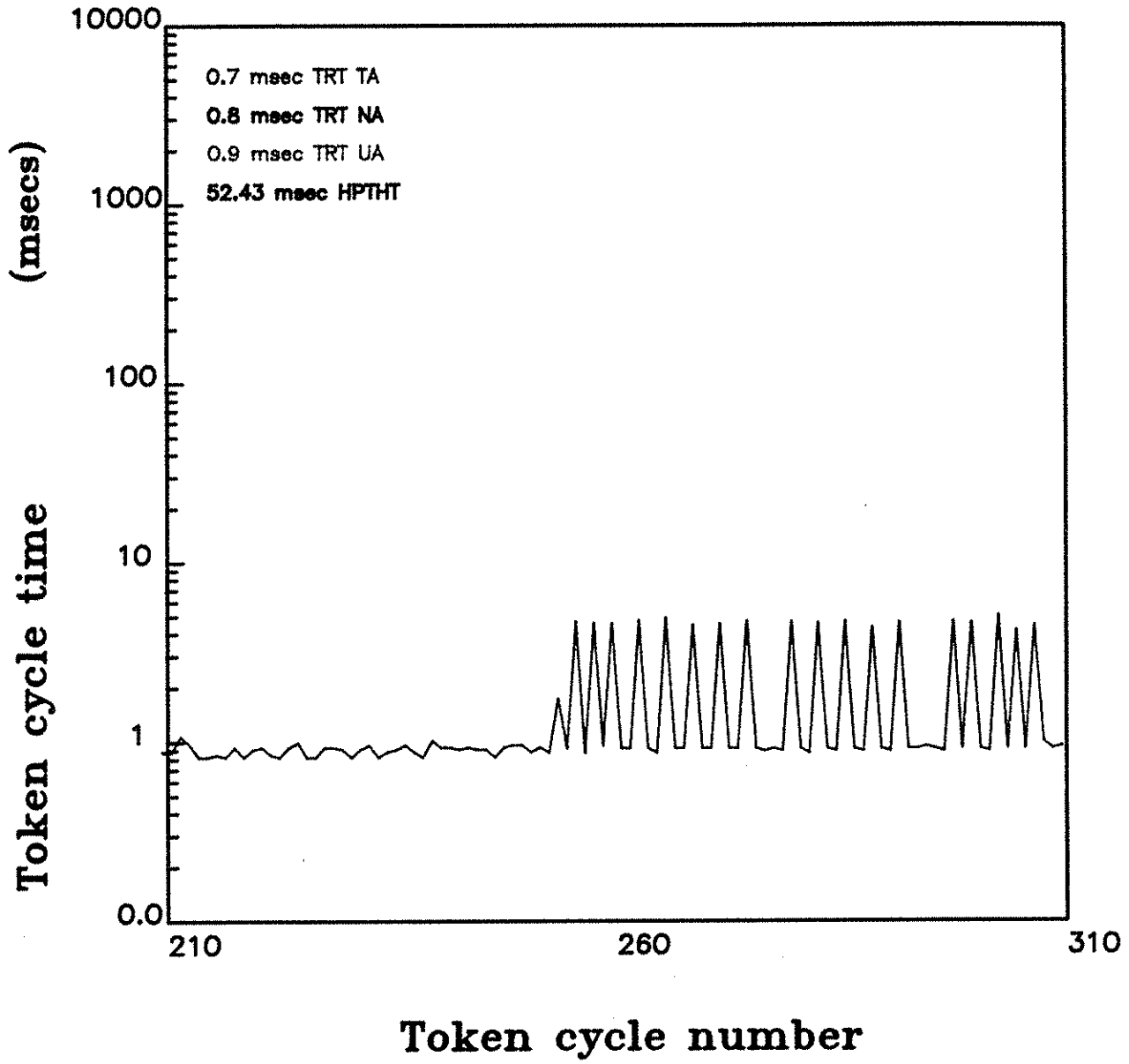


Figure 5.5
Token cycle time vs. token cycle number for 32 stations

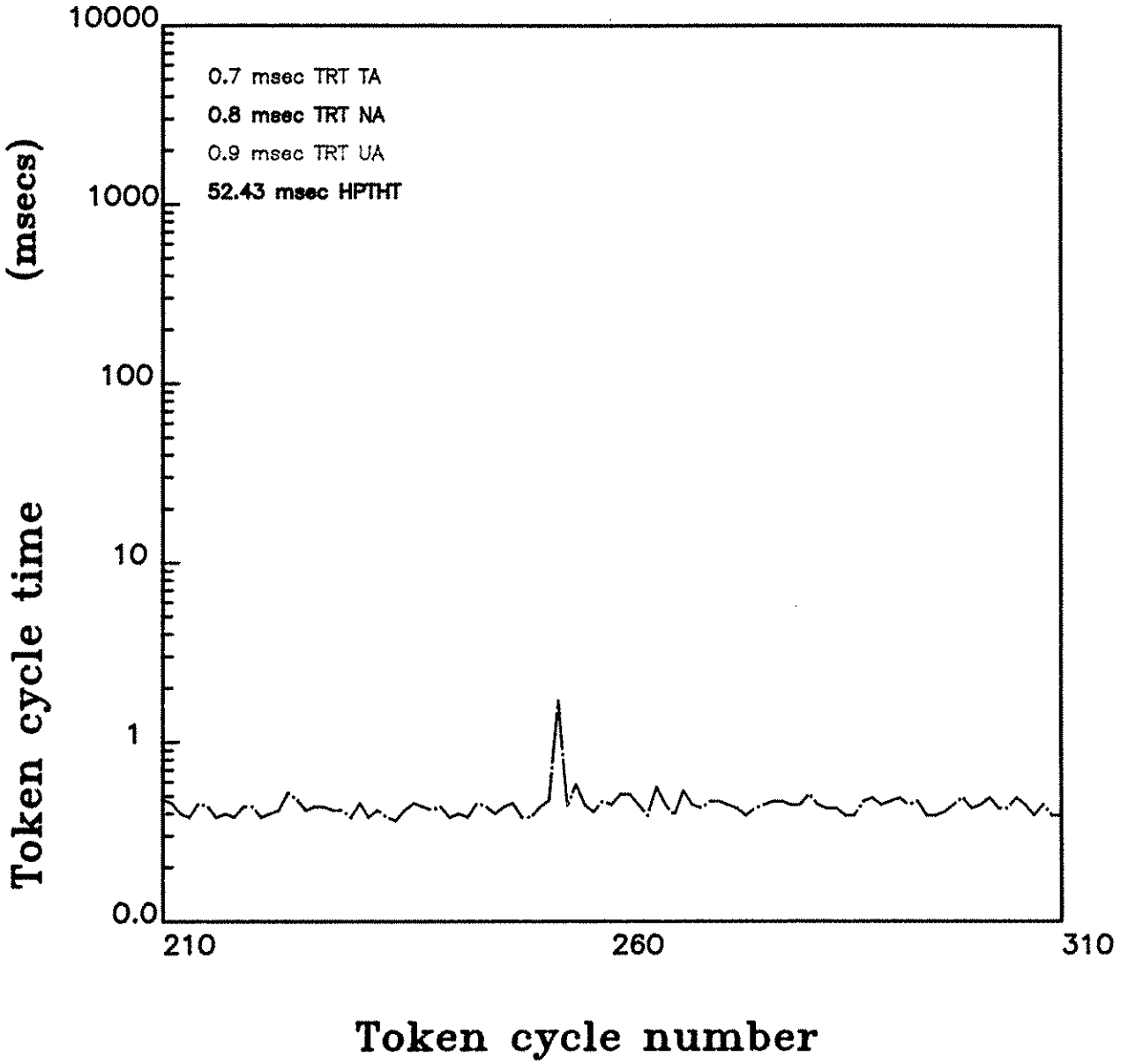


Figure 5.6
Token cycle time vs. token cycle number for 32 stations

5.5.2. Variation of queue length

The variation in the length of the queues at the Synchronous access_class and Urgent_asynchronous access_class can be observed in Figure 5.7. This was observed for 15 token cycles during the transmission of the transient load. The messages from the Synchronous access_class do not suffer large queueing delays because the messages from the Synchronous access_class are transmitted at the highest priority and the HPTHT at each station has been set to the maximum value so that it does not limit the number of messages transmitted at this priority. The queue length at the Urgent_Asynchronous access_class does not show the same variation as the token cycle time. Even though the token cycle time is returning to its stable value one cycle after the transmission of the transient load, the queue lengths are steadily growing at the Urgent_Asynchronous class. This is because the average token cycle time is still above the TRT of the Urgent_Asynchronous class. Hence messages are suffering higher queueing delays before being transmitted.

5.5.3. Variation of the average delivery time

a) The variation of average delivery time of messages at the Synchronous access_class in a network without transient loading (reference configuration) and in a network subjected to transient loading is shown in Figure 5.8. Figure 5.9 shows the variation of delivery time for messages transmitted at the Urgent_Asynchronous access_class. It can be observed from both the figures that messages are suffering higher transmission delays in the case where the transient load was imposed in the form of long messages. This is due to the fact that transmission of long messages by a station results in the station holding the token for a longer interval of time. This causes the timers in other stations to expire leading to larger queueing delays of messages at these stations and hence higher delivery times.

b) Figure 5.10 shows the effect of varying the intensity of the transient load

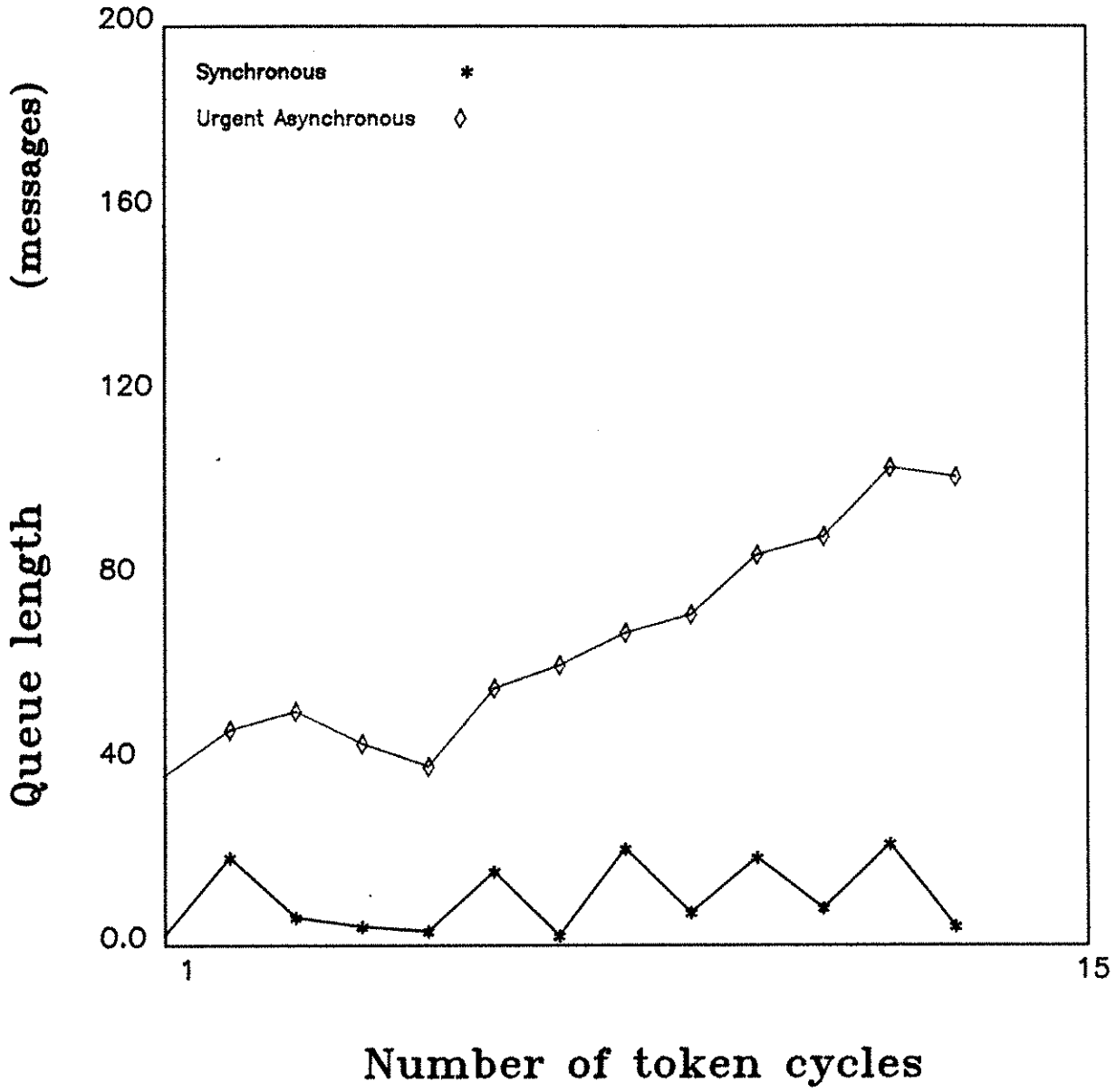


Figure 5.7
Queue length vs. number of token cycles

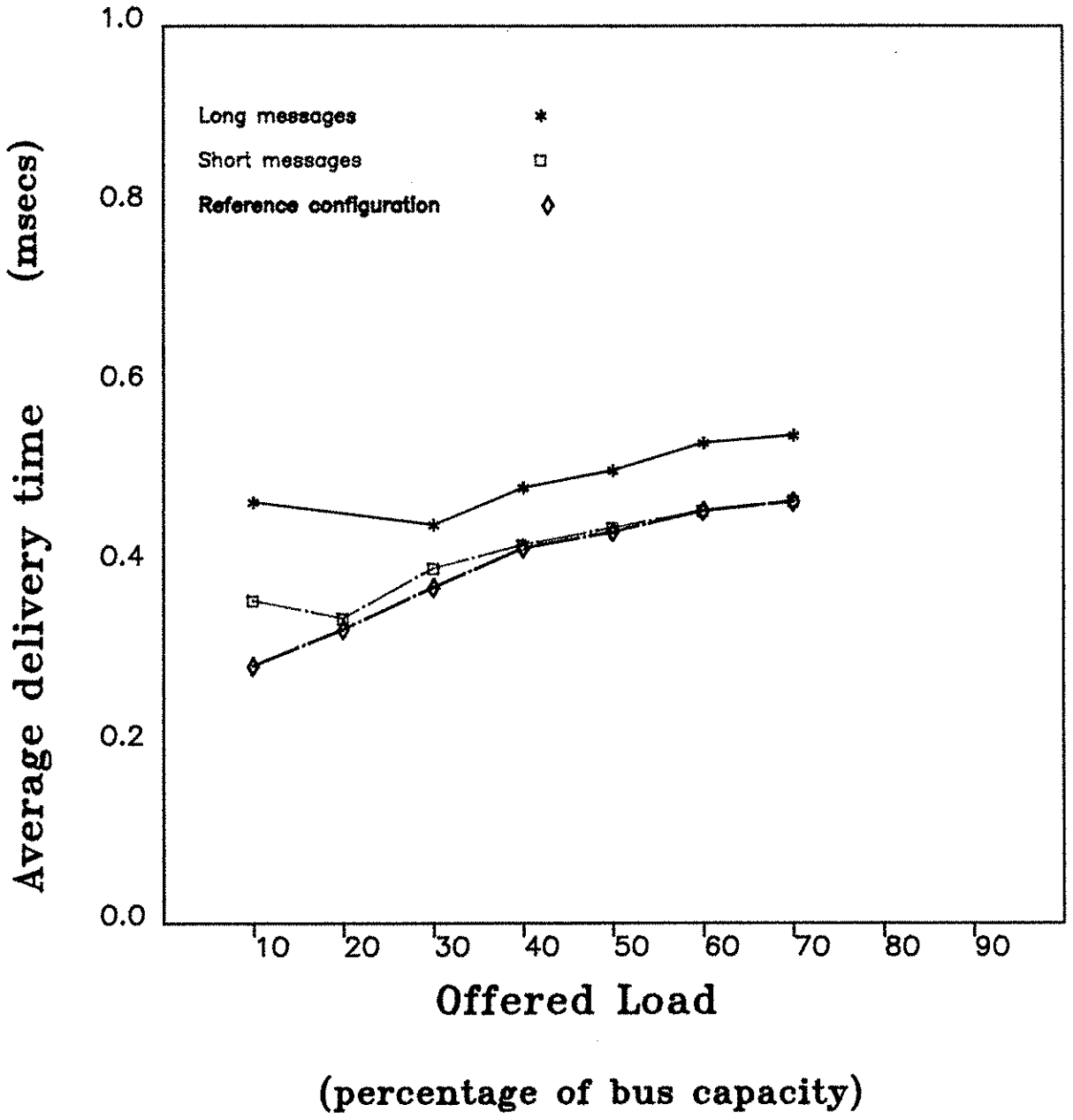


Figure 5.8
Average delivery time for Synchronous class

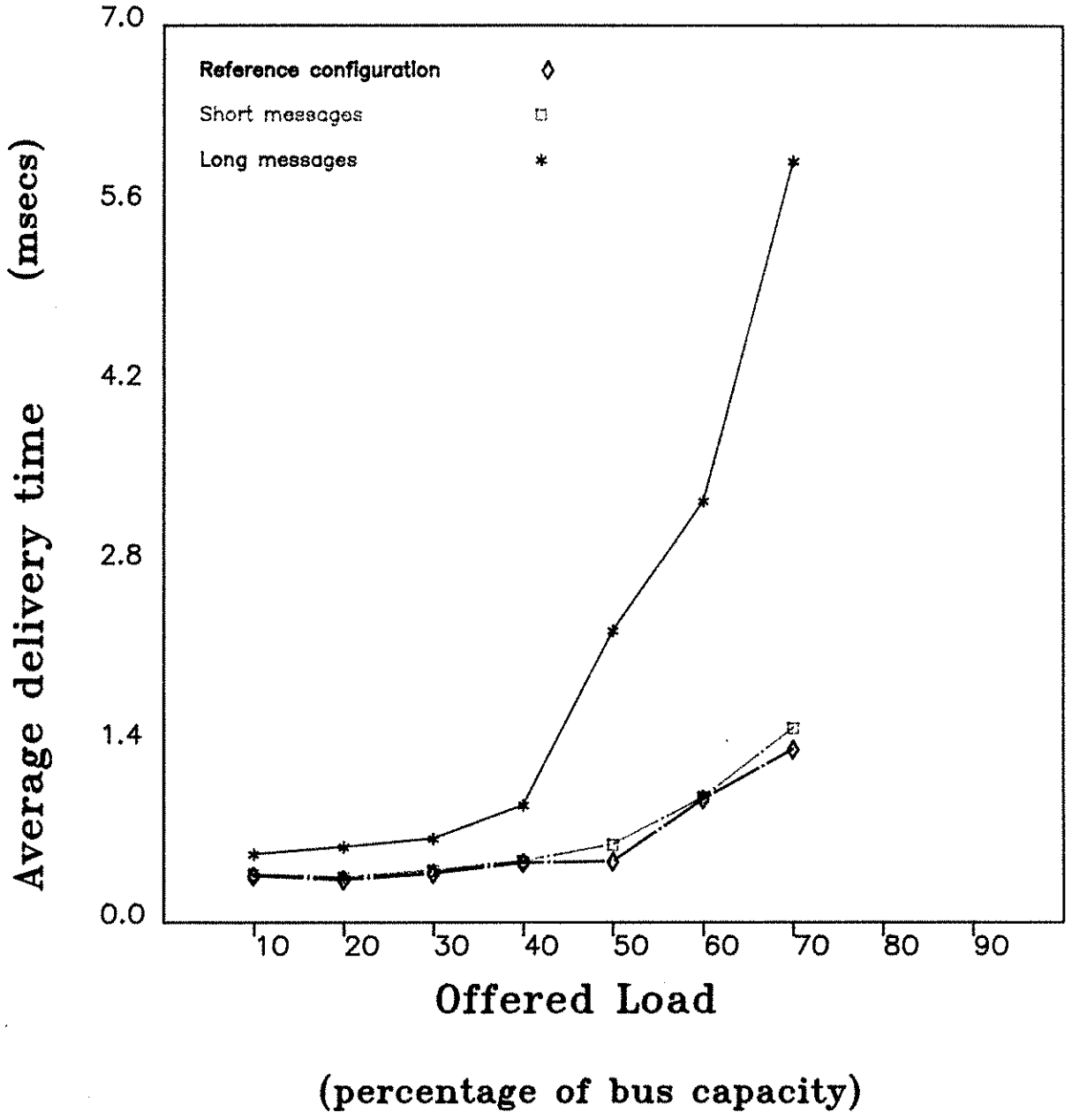


Figure 5.9
Average delivery time for Urgent_Asynchronous class

on the delivery times of messages from other stations transmitted at the same priority as the transient load. A four-fold increase in delivery time was observed as the transient load was varied from 0 to 0.64 Mbits in steps of 0.128 Mbits.

c) It can be observed from Figures 5.8 and 5.9 that the delivery times of messages from other stations in the upper two access_classes in the case where the transient load was imposed in the form of large number of smaller messages is very close to that observed in a network with no transient load. This is because the transmission of the transient load gets distributed over several token cycles and hence does not increase the token cycle time. Hence the transmission of messages at other stations is not delayed as in the previous case. But this increases the delivery time of the transient load itself.

Thus it can be observed that if the stations do not have limited timer settings, then each imposition of the transient load leads to a sharp increase in the token cycle time. The number of token cycles required for the token cycle time to return to its stable value depends on the amount of load imposed on the network. The token cycle time of the network configuration simulated for this study returned to its stable value after 10 token cycles. With limited timer settings also, the token cycle time increases to a very high value but recovers within one token cycle. Also when the transient load is imposed in the form of long messages, it increases the delivery time of other messages being transmitted at higher priorities in the network. With limited timer settings, splitting the transient load into large number of shorter messages, will result in lower delivery times for messages at higher priorities from other stations in the network. In this case the token cycle time remains steady during the entire period of transmission of the transient load.

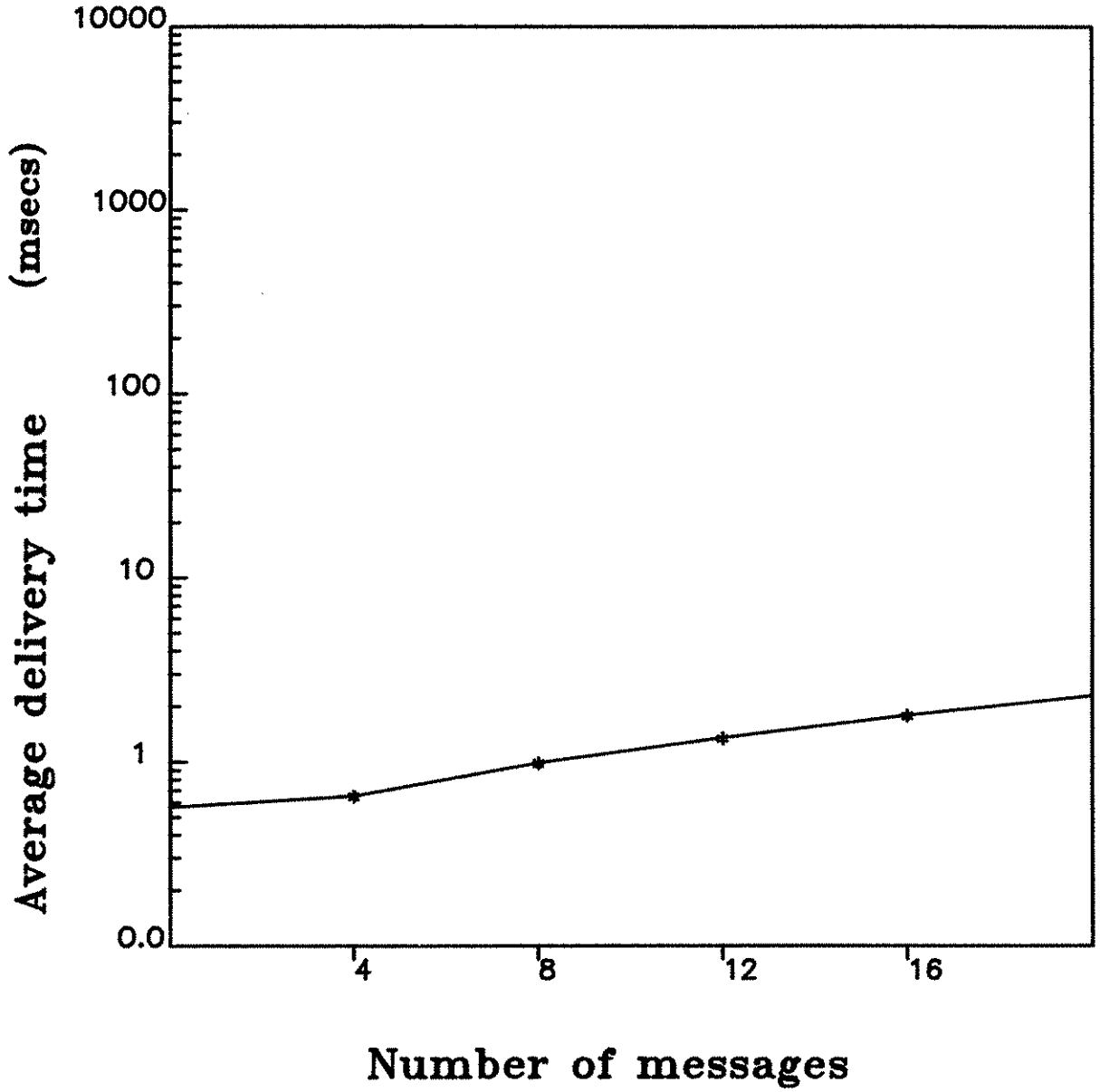


Figure 5.10
Average delivery time at different transient loads

CHAPTER 6

CONCLUSIONS

This thesis has investigated the performance characteristics of the IEEE 802.4 protocol with regard to three aspects of the protocol, namely access_classes, dynamic ring membership and transient loading.

The performance of the token passing bus depends greatly on the value of the timers that can be controlled at the data_link layer. With TRTs set in the order $TRT_{ua} > TRT_{na} > TRT_{ta}$ the protocol clearly gives preference to frames of higher priority. By choosing suitable values for TRTs it is possible to know the range of network throughput over which optimum service can be obtained at an access_class. In this study all participating stations were assumed to be active and with identical message interarrival rates at all stations at an access_class. However the designer of a local area network has to take into consideration the communication traffic at each station at each access_class as it may be specific to each station and each access_class. Also the traffic from each station can be very bursty in nature. Nevertheless this study helped us gain a better understanding of the impact of certain parameters critical to the implementation of the priority feature.

From the study on dynamic ring membership it has been observed that the protocol allocates more bandwidth to stations which have been in the ring for a longer time interval. Hence at low loads it has been observed that the static ring members get a better share of the bandwidth as opposed to the transient member which stays in ring for a shorter time interval due to infrequent message arrivals. But at higher offered loads, messages arrive more frequently and the transient station stays in ring for a longer time interval. Hence the average delivery time of a

message is reduced as the station stays in ring for a longer time interval. This study analyzed the effect of transient ring membership of a single station on the performance of the network. This gave us a better understanding of the role of the variable MIC in determining a station's entry into the logical ring. In actual practice with all stations joining and leaving the ring, the resolution process specified by this protocol can introduce considerable overhead. Hence a designer will have to definitely consider this while estimating the waiting times of messages in a dynamic ring.

The study on the performance of the protocol under transient loading conditions demonstrated that distributing the transient load over several token cycles did not cause any adverse effects on the network. Also the token cycle time remained steady over the entire period of transmission of the transient load. Simulations with loads transmitted as a few maximum-sized packets demonstrated the robustness and the deterministic nature of the protocol. Even though the token cycle time increased to a very large value, it recovered within a few token cycles.

Thus this study has helped us gain a better understanding of certain features of the protocol, role of the various parameters in the implementation of these features and in predicting optimized parameter settings and timer settings for network configurations. In the near future with the expected implementation of this token bus technology it will be possible to compare these results with data from actual measurements.

Bibliography

[Fuhrmann 84]
S. W. Fuhrmann and Robert B. Cooper, *Application of Decomposition Principle in M/G/1 Vacation Model to Two Continuum Cyclic Queueing Models (Especially Token Ring LANs)*, submitted for publication, June 14, 1984.

[IEEE 1985a]
Logical Link Control, IEEE Standard 802.2-1985.

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

[IEEE 1985c]
Token-Passing Bus Access Method and Physical Layer Specifications, IEEE Standard 802.4-1985.

[IEEE 1985d]
Token Ring Access Method and Physical Layer Specifications, IEEE Standard 802.5-1985.

[Summers 85]
Catherine F. Summers and Alfred C. Weaver, *Performance Studies of the IEEE Token Bus*, Master's thesis, DAMACS Report No. 85-06, University of Virginia, Department of Computer Science, May 1985.

[Ulug 84]
M. E. Ulug, Comparison Of Token Holding Time Strategies For a Static Token Passing Bus, *Proc. of Computer Networking Symposium*, December 1984, pp. 37-44.

[Weaver 85]
Catherine F. Summers and Alfred C. Weaver, *The IEEE 802.4 Token Bus - An Introduction and Performance Analysis*, Computer Science Report No. TR-85-19, University of Virginia, August 26, 1985.

[Zimmermann 80]
Hubert Zimmermann, OSI Reference Model - the ISO Model of Architecture for Open Systems Interconnection, IEEE *Trans. on Communications COM-28*, Vol. 4, April 1980, pp. 425-432.