A Slotted Ring Protocol for High-Speed Local and Metropolitan Area Networks

S. K. Vuong

Technical Report CSRI-262
February 1992

Computer Systems Research Institute
University of Toronto
Toronto, Canada
M5S 1A1

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

A Slotted Ring Protocol for High-Speed Local and Metropolitan Area Networks

by

Sang Kim Vuong

A thesis submitted in conformity with the requirements
for the degree of Master of Science in the
University of Toronto

Department of Computer Science
University Of Toronto
Toronto, Canada
April 1991

Acknowledgements

First I would like to thank my supervisor, Prof. Mart Molle, for his suggestion of the topic and his excellent and supportive guidance.

Next I would like to thank my second reader, Prof. Ken Sevcik, for taking time reading this thesis and providing useful suggestions and comments.

I would also like to thank Prof. Songnian Zhou for providing me with access to the "White" Distributed Systems, and all the "White" Systems members for their understanding, tolerance and patience when I ran my simulations.

I would like to thank all of my U of T friends whose friendship made my graduate life more enjoyable. I am grateful to Speros Armyros, Harjinder Sandhu, Hui Li, Jin Liu, Ali Mahmoodi, and Nandini Srikantiah for their help and friendship. I also want to thank my "other" friends, Alan Chan and Ho Luu, for their support and encouragement.

I must thank my parents, sisters and relatives for their love, support, understanding and patience throughout the years of my study.

Last, but not least, I would like to express my gratitude to the Natural Sciences and Engineering Research Council of Canada for their generous financial support.

Abstract

In this thesis, we propose a new access scheme for high-speed local and metropolitan area networks (LANs and MANs), the Slotted Ring with window service. We show that this access scheme exhibits many desirable performance properties including high capacity, low packet delays, and fairness among the stations that is insensitive to the network speed, distance, and number of stations. This is because window service equalizes the waiting time among the stations and imposes a global order of service that is approximately First-Come-First-Server (FCFS). We also consider a modified scheme in which busy slots are released by the destination stations. We find that this modification can double the system throughput without significantly affecting the packet delay. Our new scheme compares favorably with existing protocols for large and/or high speed networks. It offers a mean packet delay that is much lower than either a token ring or Cyclic Reservation Multiple Access (CRMA), and comparable to Distributed Queue Dual Bus (DQDB), which is close to the M/G/1 lower bound. Furthermore, unlike DQDB, the fairness among the stations does not degrade even when the system load is high.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Advances in technology have led to significant changes in the constraints for communication networks. Current fiber optics networks already provide data-rates of hundreds of Mbits/s, and future high-speed local and metropolitan area networks (LANs and MANs) are projected to have speeds beyond 1 Gbits/s and span distances in the hundreds of kilometers [HSW87, KGK$^+$87, GKDC88, FHT$^+$88]. Currently, FDDI (Fiber Distributed Data Interface) is the dominant high-speed LAN protocol. However, FDDI was designed for networks of only up to a few hundred Mbits/s and tens of kilometers. Previous studies have shown that the performance of current LAN protocols, such as Ethernet, token bus and token rings (including FDDI), degrades as the speed-distance product of the medium increases.

In *random access schemes,* such as ALOHA [Abr70, KL73], CSMA/CD (Carrier Sense Multiple Access with Collision Detection) [KT75] and Ethernet [MB76, IEE85a], non-zero propagation delays between the stations can result in packet collisions, and hence wasted capacity. The performance of all random access schemes degrades (i.e., throughput decreases and delay increases) dramatically as the speed-distance product increases. Tobagi [Tob80] showed that the capacities of ALOHA and CSMA/CD drop dramatically as a, the normalized propagation delay[1], increases. Moreover, the random retry mechanism, which is used to prevent continual packet collisions, can leave the bus idle while packets are waiting for transmission.

In *controlled access schemes,* such as token passing (token bus [IEE85b], token rings [IEE85c] and FDDI [ANS86]) and DAMA (demand assignment multiple access) [FT84], the control overhead can result in high access delay even under light load. All controlled access schemes are sensitive to network speed and distance. Fine and Tobagi [FT84] showed that the capacities of DAMA access schemes decrease significantly as a increases. Stallings [Sta84] showed that the efficiencies of token passing access schemes drop sharply as a function of the parameter a. And recently, Rodrigues [Rod90] and Müller et al. [MNW$^+$90] showed that the performance of FDDI also degrades dramatically as a increases.[2] Therefore, more efficient protocols are required for future networks with much higher speed-distance products.

---

[1] The normalized propagation delay, a, is defined in the literature as the ratio of the propagation delay to the mean packet transmission time (see section 2.1).

[2] Actually, the drop-off in capacities of these access schemes as the speed-distance product increases that were shown in these studies was based on the assumption of only one packet transmission per token acquisition. Of course, when more packets may be transmitted per token visit (such as by using exhaustive service in token rings and relaxing the target token rotation time (TTRT) in FDDI), higher utilization can be achieved – but at the cost of higher packet delay, i.e., an upper bound on packet delay cannot be guaranteed.

Recently, several new access schemes for high-speed LANs and MANs have been introduced. Among them Distributed Queue Dual Bus (DQDB) [NH86, IEE90] and Cyclic Reservation Multiple Access (CRMA) [Nas90a, Nas90b] are two protocols for which the capacity is independent of the network speed and distance. However, DQDB was found to have dramatic throughput unfairness among the stations when the load on the network is high [Won89, Mar89, KC90, CGL91]. Also, CRMA was found to have high packet delay, even at very low load, when the network distance is long [LM90]. Thus, new protocols are needed for new high-speed LANs and MANs, which are insensitive to network speed and distance. In this thesis, we consider a new slotted ring in the context of high-speed LANs and MANs.

In slotted rings, since the size of each slot is fixed, long packets have to be segmented into a number of slot-size segments and transmitted one-by-one. In the past, this characteristic of slotted rings was viewed as a significant disadvantage (e.g., [Sta85, section 5.4] and [BG87, section 4.5.3]). Many previous performance comparison studies showed that token rings out-performed slotted rings [AAG78, Bux81, LHG82].[3] This is because slot sizes were necessarily small in a typical LAN environment,[4] and thus header overhead is much more significant in a slotted ring than in an Ethernet or a token ring, which support much larger frames. Bertsekas and Gallager [BG87] also pointed out that artificially lengthening the ring to permit the use of larger slot sizes has the unwanted side effect of increasing packet delay

[3]In all of these performance comparison studies, the networks considered were typical low-speed LANs, i.e., the network speeds were only a few Mbit/s and the network lengths were only a few kilometers.

[4]Stallings [Sta85] estimates the storage capacity of a 10 Mbit/s, 100 station slotted ring to be equivalent to about 150 bits, divisible into four 37-bit slots.

under light load. Furthermore, current slotted ring protocols, such as the Cambridge ring, limit each station to a single outstanding segment, so in the case of unbalanced traffic many slots will go unused.

The situation is quite different in the context of very high data rates and/or metropolitan area distances, however. First, we expect the storage capacity of the ring to be large rather than small, so there is no difficulty in creating a number of reasonable-sized slots. Second, external factors (notably the need for compatibility with asynchronous transfer mode (ATM) “cell switching” equipment) mean that we can *expect* most successful protocols for high-speed MANs to be based on small constant-length slots (or “cells”). These differences between earlier systems and the high-speed environment we are interested in are sufficient to neutralize most of the previously-observed disadvantages of slotted rings. Furthermore, with the introduction of the window service access rule described below, the efficiency of the protocol is independent of both the storage capacity of the ring and the distribution of traffic among its stations.

1.2 Thesis Objectives

The goal of this thesis is to study the performance the Slotted Ring with window service (SRWS) in the high-speed LAN/MAN environment, to see how much the changes in assumptions affect the desirability of this type of access scheme. First the medium access scheme of the proposed protocol is described in detail, then the performance and fairness of the protocol is investigated.

1.3 Thesis Outline

The thesis is organized as follows. Chapter 2 provides the necessary background needed for subsequent chapters. Of particular importance are: i) the slotted ring, the network architecture on which the proposed access protocol will be used; and ii) the window-gated service rule of the Token Ring with Window-Gated service (TRWS) protocol, upon which the window access rule of SRWS is based.

Chapter 3 gives the detailed description of the SRWS protocol. This chapter also describes the SRWS/DR protocol, a modification to the protocol in which slots are released by the destination stations. Assuming uniform traffic, this modification doubles the capacity of a slotted ring.

In chapter 4, we first introduce the moving server queueing system, the underlying queueing theoretic model that was used in the analysis of packet delays for the TRWS protocol. We then describe how we use this moving server queueing model to obtain the analytical formulae for the system capacity and bounds on the mean packet delay for the SRWS protocol.

Chapter 5 shows the simulation results on the performance of SRWS. First, the simulation methodology, assumptions and traffic models are described. Then, the simulation results for both SRWS and SRWS/DR are presented. The results from simulation are also used to evaluate the accuracy of the analytical formulae presented in chapter 4.

In chapter 6, the performance of the SRWS protocol is compared to several other alternatives, notably an M/G/1 queueing system, a token ring with exhaustive service, Cyclic Reservation Multiple Access (CRMA), and Distributed Queue Dual Bus (DQDB).

Chapter 7 gives a summary of our conclusions and suggests directions for future research.

Lastly, appendix A describes a recursive batch mean analysis procedure that can be used to control the run length of a simulation experiment. This procedure uses a fixed number of batches and recursively doubles the batch size until the test for batch independence is satisfied and the desired confidence interval is achieved. We believe that this procedure is also applicable to other simulation experiments where batch mean analysis is used.

Chapter 2

Background

This chapter provides the necessary background needed for subsequent chapters. Section 2.1 gives the definitions for high-speed LANs and MANs. Section 2.2 describes slotted rings in general and the access rule used in the Cambridge slotted ring. Section 2.3 first describes token rings in general and the service disciplines used in token rings, then gives a detailed description of the Token Ring with Window-Gated service (TRWS). Section 2.4 and 2.5 describe two typical high-speed MAN protocols: DQDB and CRMA, respectively.

2.1 High-Speed LAN and MAN Definitions

A **local area network (LAN)** is usually owned by a single organization, and covers the area which is also owned by that organization (e.g., a building or a university campus). Conversely, a **wide area network (WAN)** spans a much wider geographic area, perhaps an entire country, and is owned by multiple organizations. In between the LAN and the WAN is the **MAN (metropolitan area network).** A MAN is a network that covers an entire city, and may be owned by one or more organizations [Tan88]. One of the most important applications for MANs is to serve as a backbone network for interconnecting possibly heterogeneous LANs (such as Ethernet, slotted ring, token ring, and FDDI) and large computers (see figure 2.1). The characteristics of the three network types are summarized in table 2.1.

Type	Area Coverage	Size	Owner	Examples
LAN	Organization	10 km	Private	Ethernet, slotted ring, token ring
MAN	City	100 km	Private/Public	DQDB
WAN	Country	unlimited	Public	ARPANET, DATAPAC, NSFNET

Table 2.1: Network Types and Characteristics.

In analyzing the performance of LANs and MANs, the two most important factors are the *bit rate (B)* of the medium (or *network speed*) and the *end-to-end signal propagation delay (D)* between the stations[1] on the network. The propagation delay reflects the length of the medium which is used to connect the stations together. In fact, it is the product of these two terms, $B * D$ (the *speed-distance product*), that is the single most important parameter for determining the performance of an access scheme for LANs or MANs. The product $B * D$ is equal to the length of the transmission medium in bits. The ratio of the length of the medium, expressed in

[1]The terms *station* and *node* are used interchangeably in the literature, as well as in this thesis.

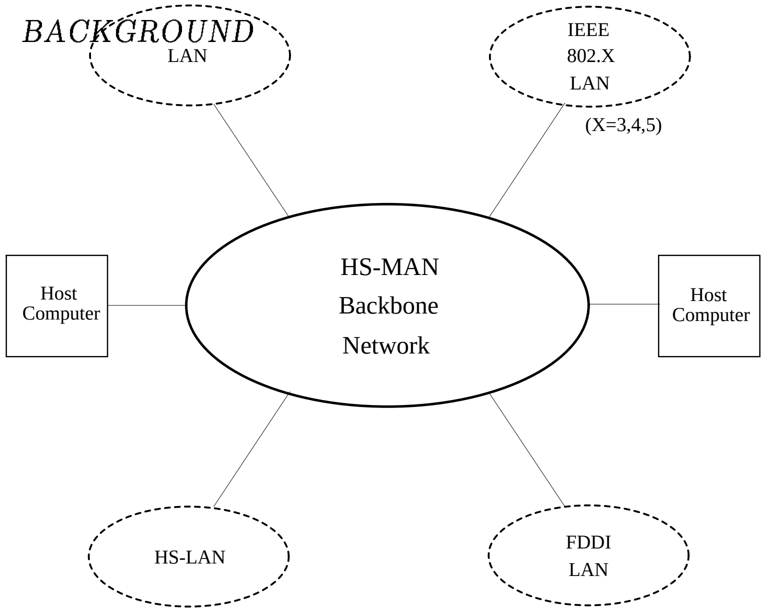


Figure 2.1: A High-Speed MAN Backbone Network.

bits, to the mean packet length is often referred to as the *parameter* a in the literature [Sta84]. If P is the mean packet length, then

$$a = \frac{Network\ Length\ in\ Bits}{Mean\ Packet\ Length} = \frac{B * D}{P} = \frac{End\_to\_End\ Propagation\ Delay}{Mean\ Packet\ Transmission\ Time}. \tag{2.1}$$

The parameter a is also referred to as the normalized propagation delay, because it is also the ratio of the propagation delay to the mean packet transmission time. It has been shown that many widely-used metrics of network protocol performance, such as channel utilization and normalized packet delay, remain the same if the parameter a is held constant [Sta84]. In this thesis, we call a network (LAN or MAN) *high-speed* if it has the normalized propagation delay of greater than 1 (i.e., $a > 1$).

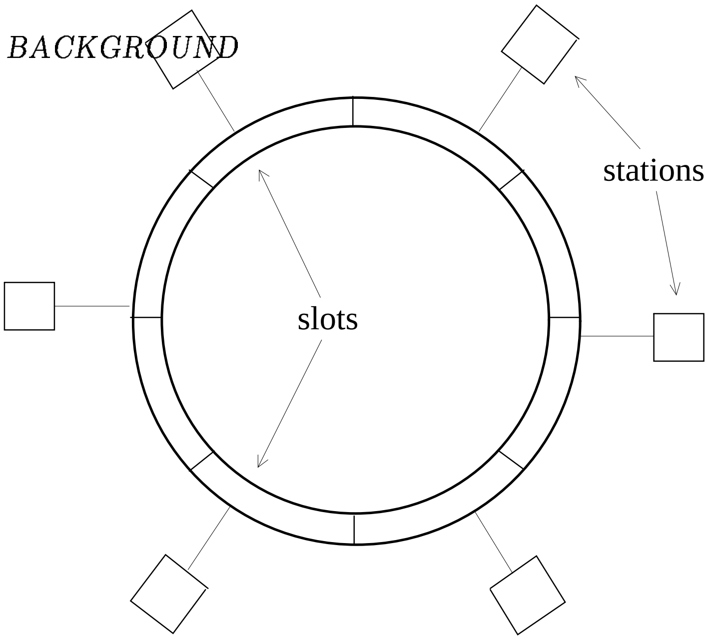


Figure 2.2: A Slotted Ring Network.

2.2 Slotted Rings

A **slotted ring,** e.g., **Cambridge ring** [WW79, Nee79], consists of a constant number of fixed-length slots, which circulate continuously around the ring (see figure 2.2). There may be a gap between the slots, because in a real system the size of the ring may not be an integral multiple of the slot size. Each slot consists of an *access control field* (ACF), which is used for bus access control; and a *data segment field* (DSF), which is used for transmitting a segment of a packet (see figure 2.3). Slotted rings use only one control bit in the ACF of each slot, the *busy/empty* (BUSY) bit, to control access to the ring. The function of the BUSY bit is to indicate whether the slot is currently being used by a station to transmit a data segment (BUSY = 1) or available for access (BUSY = 0).

When a station has data to transmit, it must wait for an *empty* slot to come by,

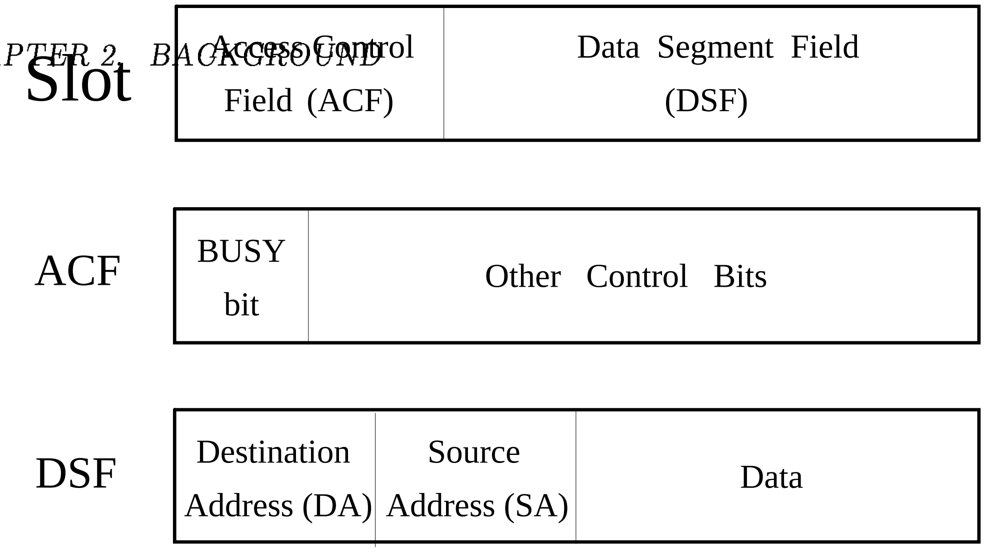


Figure 2.3: General Slot Format used in Slotted Rings, DQDB and CRMA.

mark the slot as *busy* (by setting the BUSY bit to 1), transmit it data segment into the DSF of the slot, and wait for the slot to come back to release it.[2] Each slot makes a complete tour around the ring, from the source to the destination and back to the source, with each station along the way forwarding to the next one. When the busy slot passes by its destination, the destination station makes a copy of the data in the DSF of the slot as it is forwarded to the next station. When the busy slot returns to its source, the source station removes its data segment from the ring (by simply resetting the BUSY bit to 0), so that other stations can use it. To prevent some problem from leaving a slot busy indefinitely, one of the stations on the ring is designated as a monitor that is responsible for releasing slots that have traveled around ring more than once without being reset properly.

[2]In slotted rings, the responsibility of "freeing" a slot is usually assigned to the sending station, although (unlike token rings) there are no fundamental reasons preventing the receiver from freeing the slot – which would effectively double the throughput [BG87]. In the next chapters, we show that this potential for doubling the capacity of slotted rings is, in fact, achievable.

The access rule of current slotted ring protocols, such as the Cambridge ring, as described above, limits each station to a single outstanding outstanding segment. This prevents a station from either monopolizing the channel or being starved of service and ensures fair sharing of the channel bandwidth among the stations. However, it also hinders the efficiency of large rings.

2.3 Token Rings

2.3.1 Token Ring Protocol

A **token ring** consists of a set of stations that are serially connected point-to-point by a unidirectional transmission medium to form a closed loop [FN69, PB79, DSM83, IEE85c, Str87] (see figure 2.4). In a token ring, there is a special bit pattern, called the *token,* which is used to control access to the ring, and is passed around from station to station. A station has the right to transmit data onto the ring when it observes an *idle token* passing by on the medium. Stations that have no packet to send simply pass the token on to the next.

When a station has data to send, it must wait for the idle token to come by, capture the token by modifying it from an idle token to a *busy token,* and transmit its data. The data is transferred sequentially around the ring, bit-by-bit, from one active station to the next, and can be read by all stations on the ring. When the data passes by the destination, the destination station copies the data as it is forwarded to the next station. Finally, the bits that have propagated around the ring come back are

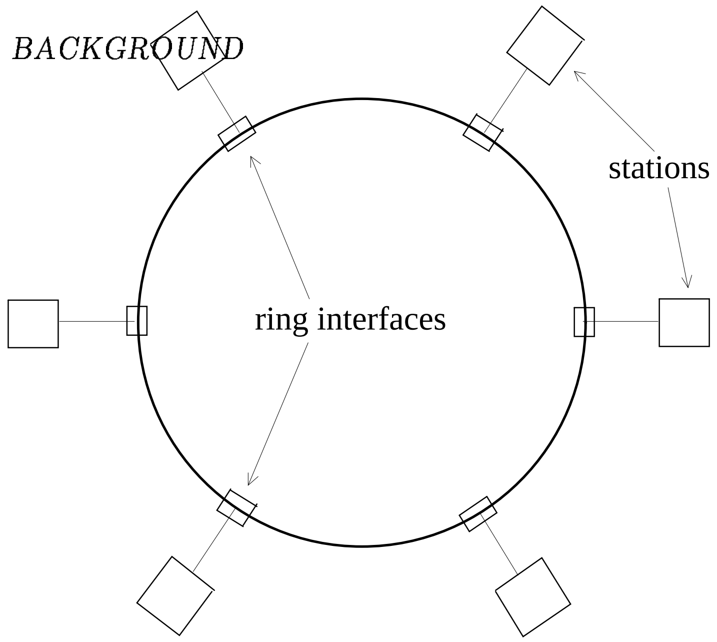


Figure 2.4: A Token Ring Network.

removed from the ring by the source station. At the completion of its transmission, the sending station regenerates a new idle token and passes it to the next station, which gives other stations the opportunity to access the ring. Similar to slotted rings, one of the stations on the ring is designated as a monitor that is responsible for token loss detection and recovery.

2.3.2 Service Disciplines

There are many variations in the detailed operation of a token ring network. Each station could be restricted to sending a single packet each time it acquires the token, or a station could empty its queue of waiting packets before passing the token to the next station. This section lists several usual, as well as unusual, service disciplines which have been used in token ring networks.

- **Exhaustive service:** The server (idle token) continues to serve a station until its queue is empty. Packets which arrive during the service time are also served in the same service period [FA85, Fuh85, Tak86].

- **Gated service:** The server will serve only those packets which are waiting at the station when the token arrives. Those packets which arrive during the service time will be served in the next visit [FA85, Fuh85, Tak86].

- **Limited service:** A station is served until either i) its queue is empty or ii) a specified maximum number of packets are served, whichever occurs first [Fuh85, Tak86].

- **Single service:** At most one packet is served each time the server arrives at a station [KS86]. This is a special case of the limited service in which the specified maximum number of packets can be served is one. This service rule is sometimes also referred to as **ordinary service** in the literature [Kue79].

- **Helical-Window service:** On the k-th visit of the idle token, a station is allowed to transmit only those packets that arrived during the k-th window. The window is a partition of the time axis into a sequence of equal-length intervals [Mol86, KM89].

- **Window-Gated service:** The window-gated service rule is exactly the same as that of helical-window service, except that window-gated service uses variable-size gated windows instead of constant-size helical windows [Mol90]. This service rule is described in more details in the next section.

- **Timed-Token service:** The server serves each station according to a "timer". Each station has a timer, called *observed token rotation time* (OTRT), and chooses a *target token rotation time* (TTRT). OTRT is used to time the token from arrival to arrival. The difference between TTRT and OTRT, called *token holding time* (THT), at a station, is the amount of time that station allowed to transmit data. Each time the idle token arrives, if OTRT $<$ TTRT then the station has the right to transmit until either the timer THT expires or its queue becomes empty, whichever occurs first. If OTRT $\geq$ TTRT then no packets can be transmitted [Gro82]. This service rule is used in **Fiber Distributed Data Interface (FDDI) Token Ring** [ANS86], with multiple timers, to support different classes of service (synchronous and asynchronous data traffic).

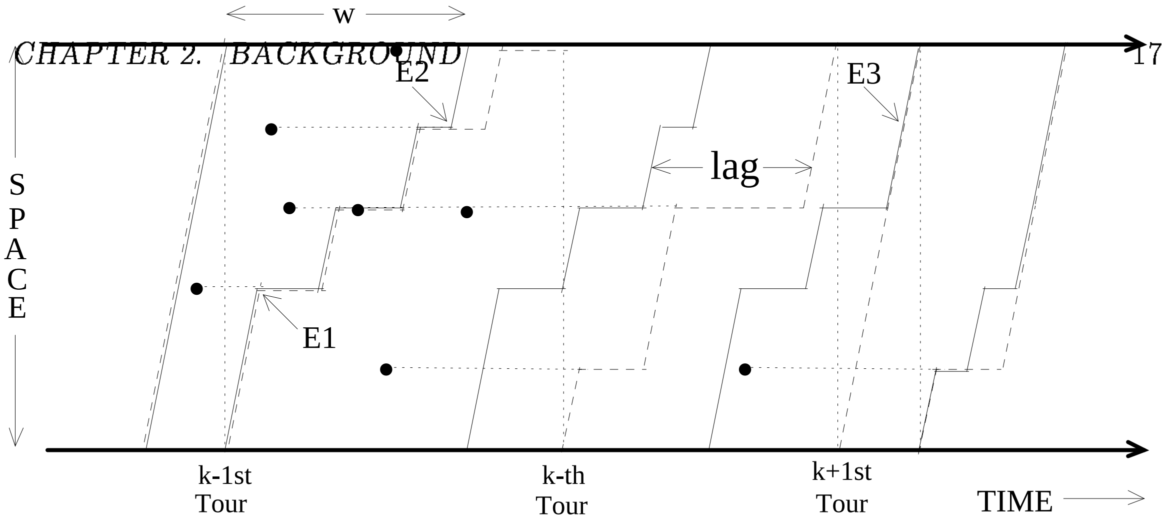
The argument against the exhaustive and gated service rules is *fairness.* Under these two service rules, a heavily loaded station may monopolize the channel, locking out other stations for an extended period of time (or forever). Single service is more fair, but it is less efficient. This is because of the amount of useful bandwidth that is wasted in circulating the token, since no stations can transmit any packets during the time in which the token is traveling from one station to the next. The degradation in performance becomes significant in networks with unbalanced load and/of high-speed.

Similar to limited service, helical-window and window-gated service rules try to compromise between fairness and efficiency. But limited service imposes a fixed maximum on the number of packets that can be transmitted at each station, whereas helical-window and window-gated service select variable numbers of packets for transmission depending on how many have arrival times that fall within a given time interval. Since these window service rules select packets independent of the stations that generated them, they automatically allow heavily loaded stations to send more packets per token acquisition than lightly loaded stations. Helical-window service, however, sometimes requires the idle token to be artificially delayed at a station, and thus is less efficient than window-gated service.

2.3.3 Token Ring with Window-Gated Service

The undesirable wasteful *rest periods* of helical-window service in **Helical-Window Token Ring (HWTR)** [KM89] led Molle [Mol90] to consider a more practical window-based access rule, the window-gated service. Instead of *constant-size helical-windows,* **Token Ring with Window-Gated service (TRWS)** uses *variable-size gated windows,* defined as the minimum of the design parameter, w, and the *lag* of the algorithm at the station when the idle token arrives. The lag of the algorithm is defined as the difference between the completion time of the k-th window at a particular station and the corresponding arrival of the idle token on its k-th tour; whenever the lag is positive, we say that a *busy period* is in progress. The window size is always w when the lag is non-zero; but it takes on values between the *ring walk-time,*[3] and w when the lag is zero, depending on the number of packets each station has generated since its last tour. A time-space diagram of window-gated service access rule is shown in figure 2.5. In this figure, the busy period begins when the idle token reaches

[3]The walk-time (also called the round-trip propagation delay) is the time it takes for the signal to travel around the entire network once.



In this figure, the "solid lines" represent the *window boundaries;* the "dashed lines" represent the *token trajectory;* the "dots (•)" represent *packet arrivals;* and the "horizontal dotted lines" represent the *packet waiting times.*

Figure 2.5: Window-Gated Service Access Rule in TRWS.

event E1, but the window does not reach maximum size until it reaches event E2. Thereafter, the window remains at maximum size until the end of the busy period where the idle token reaches event E3.

Window-gated service has better performance than helical-window service, but it also introduces some dependence among the queues at different stations, which makes its exact analysis much more difficult than that of the helical-window service. Nevertheless, with the assumption of the *extremal-window property,* which ensures that clearing the backlog induced by each packet transmission requires the token to make an integral number of tours of the ring, exact closed-form expressions for the mean, variance, and moment generating function for packet system time were derived. Simulation results showed that the analytical formulae also remain good approximations even when the extremal-window property is not satisfied. The analysis of packet system times for both HWTR and TRWS was based on the **moving server queueing system** [Mol89], which is described in more details in section 4.2.

2.4 Distributed Queue Dual Bus

Distributed Queue Dual Bus (DQDB) [BHN+86, NH86, NBH88, IEE90] consists of a pair of contra-flowing slotted unidirectional buses, called *bus-A* and *bus-B* (see figure 2.6). Both buses are operated synchronously using fixed-size slots. The slots are generated continuously by the slot generator at the head of each bus; they flow down-stream, pass each station, and then are discarded at the end of the bus. The slot format in DQDB is similar to that in slotted rings, i.e., each slot consists of an ACF and an DSF (see figure 2.3). DQDB uses two control bits in the ACF of each slot, the *busy/empty* (BUSY) and *request* (REQ) bits, to control access to the bus. The function of the BUSY bit is the same as in slotted rings, i.e., it is used to indicate whether or not the slot is currently being used by a station to transmit a data segment. The additional REQ bit is used by a *down-stream* station to request for an empty slot from its *up-stream* stations. The operation of the access protocol is identical and independent for each bus. In the following, we only describe the access to bus-A, the access to bus-B is completely symmetric.

2.4.1 Basic Access Protocol

In DQDB, every station continuously monitors the two control (BUSY and REQ) bits and keep a record of its position in the distributed queue. When it is idle (has

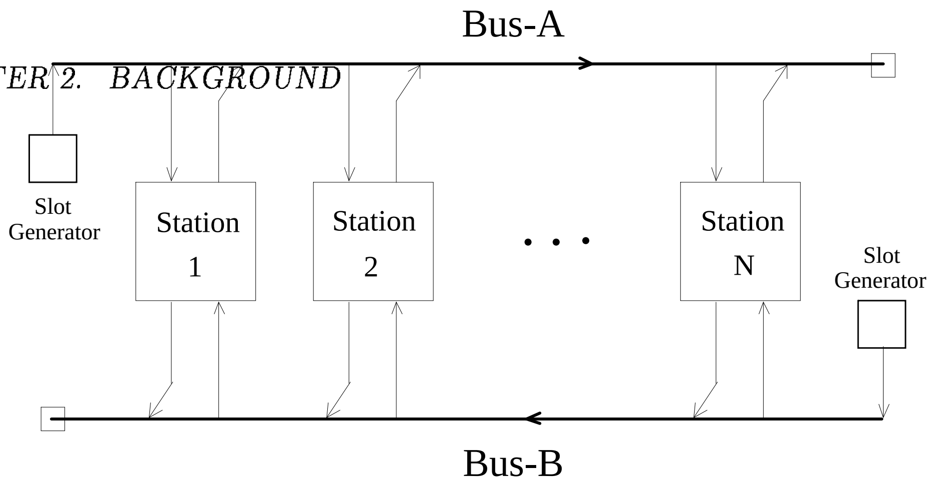


Figure 2.6: Network Configuration in DQDB.

no data to send), the station keeps track of the number of outstanding requests for access to bus-A from downstream stations using the *request counter* (REQ_CTR). The REQ_CTR is incremented by one for each request received on bus-B, and decremented by one, provided it is not zero, for each empty slot passing by on bus-A. When the station becomes active (has data to transmit), it transfers the contents of the REQ_CTR to a second counter called the *count-down counter* (CD_CTR) and resets the REQ_CTR to zero. The station then sends a request on bus-B by setting the REQ bit of the first "request free" slot on bus-B to 1. In the meantime, the REQ_CTR continues to count the number of new requests received on bus-B. The CD_CTR indicates the number of requests for access to bus-A that have to be satisfied before the station can transmit its data segment; it is decremented by one for each empty slot passing by on bus-A until it reaches zero. When the CD_CTR reaches zero, the station transmits its data segment into the DSF of the first empty slot passing by on bus-A. If the station has more data to transmit, it has to transfer the contents of the REQ_CTR to the CD_CTR, reset the REQ_CTR to zero, and repeat the procedure again.

With the use of two counters (REQ_CTR and CD_CTR) at each station, a distributed queue of packets awaiting transmission on a bus is formed. However, it should be noted that because requests for transmissions are not immediately registered (seen) at all stations, the distributed queue is not identically perceived by all stations on the network. Under DQDB, each station is only allowed to have one outstanding request at any given time. Furthermore, each request is only for a single "tiny" slot. These limitations can result in dramatic throughput unfairness when the load on the network is high [Won89].

2.4.2 Bandwidth Balancing Algorithm

In order to reduce the throughput unfairness among the stations in DQDB, an optional *bandwidth balancing* (BWB) algorithm has been included in [IEE90]. With the bandwidth balancing algorithm, DQDB has a system variable, called *bandwidth balancing modulus* (BWB_MOD), and each station has an additional counter for each bus called the *bandwidth balancing counter* (BWB_CTR). If BWB_MOD equals zero, then the BWB algorithm is disabled. If BWB_MOD is greater than zero, it indicates when the value of the BWB_CTR should be rolled over to zero. A non-zero value of BWB_MOD, n say, means that the maximum fraction of empty slots a station can use on either bus is $n/(n+1)$. Each time a station transmits a data segment, the BWB_CTR is incremented by 1. When the BWB_CTR for a bus reaches the value of BWB_MOD, it is reset to zero, and the station has to bypass the use of one empty slot on that bus. Note that the station is allowed to use the next empty slot, without having to make another request, so the BWB algorithm merely tends to let more-down-stream stations get better access to the empty slots without having a significant impact on the capacity.

2.5 Cyclic Reservation Multiple Access

Cyclic Reservation Multiple Access (CRMA) [Nas90a, Nas90b] is an access scheme for networks based on slotted *unidirectional bus structures (UBS),* both folded and dual-bus configurations (shown in figures 2.7 and 2.8, respectively). The slot format in CRMA is similar to those in DQDB and slotted rings, i.e., each slot consists of an ACF and an DSF (see figure 2.3). CRMA includes of two basic access control mechanisms called *cyclic-reservation access* and *reservation-cancellation back-pressure.* In this section, we briefly describe the cyclic-reservation access and reservation-cancellation back-pressure mechanisms for the folded-bus configuration. We then discuss how these two mechanisms function in the dual-bus configuration. Lastly, we describe the *unallocated-slot access* algorithm, which has recently been proposed as a way to reduce its high access delay under light load.

2.5.1 Basic Cyclic-Reservation Access

In CRMA, the stations access the bus according to cycles of slots. Each cycle is explicitly numbered by an integer, called the *cycle-number.* The lengths of the cycles are not fixed and are a function of station demands. The stations reserve slots in each

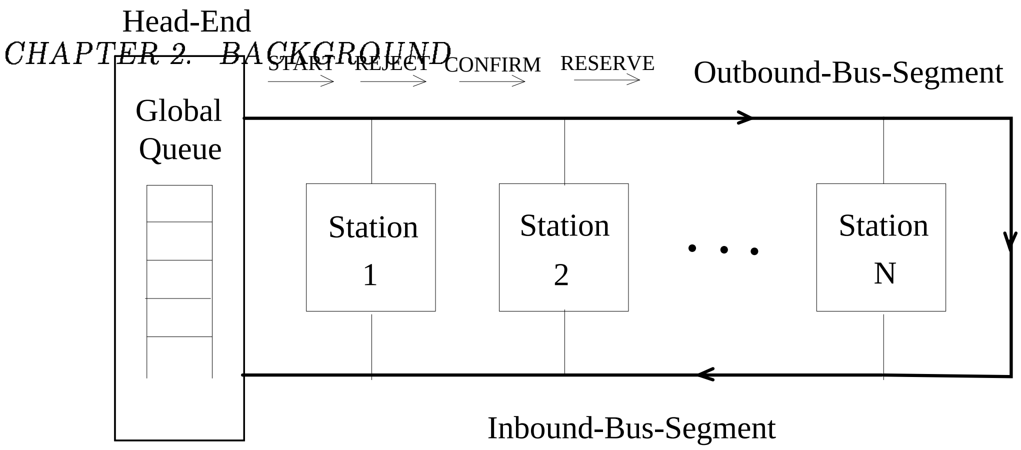


Figure 2.7: Folded-Bus Configuration in CRMA.

cycle and the *head-end* generates the cycle sufficiently long to satisfy these reservations. The cycles for which there were no reservations are skipped. The reservation and generation of the cycles are based on two access commands called *RESERVE* and *START*, respectively. These commands are issued by the head-end and have *cycle-number* as an argument. Each RESERVE command also has the *cycle-length* as an argument.

In a folded-bus CRMA network, the portions of the bus preceding and following the fold are referred to as *out-bound* and *in-bound* segments, respectively (see figure 2.7). Packets are transmitted on the out-bound segment and received on the in-bound segment. The head-end periodically issues RESERVE commands, with cycle-length of zero. As the RESERVE command of each cycle passes by a station on the out-bound bus, a station can make its reservation for that cycle by adding the number of slots it wishes to reserve to the cycle-length. The station then stores the number of slots reserved, as *reserve-length,* together with the cycle-number in a *local reservation queue.* The stations do not modify the cycle-length on the in-bound bus segment.

When the RESERVE command returns to the head-end, a reservation, containing the cycle-number and cycle-length, is entered into a *global reservation queue.* This queue is served according to a First-In-First-Out (FIFO) order. The head-end serves each reservation by issuing a START command containing its cycle-number, followed by as many empty slots as indicated by its cycle-length. Upon observing a START command, each station checks its local reservation queue. If there is an entry with the cycle-number matching that of the START command, the station waits for the next empty slot to arrive, then transmits its data in as many slots as indicated by the reserve-length.

2.5.2 Reservation-Cancellation Back-Pressure

In CRMA, the generation rate of RESERVE commands should be high enough to satisfy the throughput requirement of the stations. However, a high generation rate of RESERVE commands can result in congestion at the global reservation queue, and thus increases the access delay observed by all stations. In order to minimize the worst-case access delay under CRMA, a back-pressure mechanism was developed. The back-pressure mechanism monitors the total number of reserved slots in the cycles queued at the head-end. When the return of a RESERVE command increases this number beyond a certain threshold, the generation of the RESERVE commands is inhibited, and all RESERVE commands that have been issued but not yet returned are canceled. The RESERVE command generation is resumed when the number of reserved slots drops below the *back-pressure threshold.*

CRMA access mechanism under reservation-cancellation back-pressure requires two additional access commands: *CONFIRM* and *REJECT.* The CONFIRM command is used by the head-end to confirm the RESERVE command for each cycle. The REJECT command is used by the head-end, when the total number of reserved slots exceeds the back-pressure threshold, to cancel all RESERVE commands that have been issued but not yet confirmed. When a station observes a REJECT command, it dequeues all unconfirmed reservations from it local queue and initiates reservation retry.

2.5.3 Dual-Bus Configuration

The advantage of dual-bus configuration is the potential for doubling the throughput, when the load is equally distributed among the two buses. In a dual-bus CRMA, there are two head-ends: *head-end-A* and *head-end-B* (see figure 2.8). Every station is able to transmit and receive on both buses. The selection of the bus to be used for transmission is based on the relative location of the destination. Similar to DQDB, the operation of the access protocol in a dual-bus CRMA is identical and independent for each bus. In the following, we only describe the access to bus-A, the access to bus-B is completely symmetric.

To access bus-A, reservations are made to the RESERVE commands on bus-B issued by head-end-B. The RESERVE command accumulates the station reservations and enters the global queue reservation at head-end-A. The CONFIRM, REJECT and

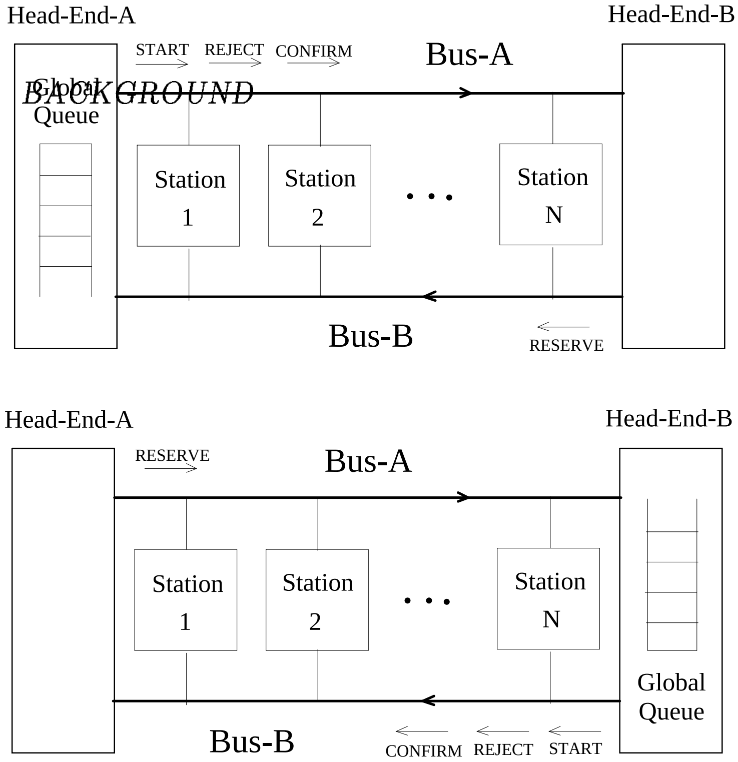


Figure 2.8: Dual-Bus Configuration in CRMA.

START commands are issued on bus-A by head-end-A. As in the case of the folded-bus configuration, a CONFIRM command is used to confirm each RESERVE command, and a REJECT command is issued when the total number of reserved slots exceeds the back-pressure threshold. However, upon receiving a REJECT command, head-end-B immediately begins reissuing RESERVE commands associated with unconfirmed cycles. This variation from the CRMA on the folded-bus configuration came from the fact that each head-end has no detailed information about the global reservation queue at the other end.

2.5.4 Unallocated-Slot Access

In a dual-bus CRMA, a station which accesses bus-A through the reservation mechanism incurs a minimum access delay equal to two times the propagation delay between that station and head-end-A. (It is equal to the round-trip propagation delay in a folded-bus CRMA.) This minimum access delay increases linearly with the speed-distance product, and can be significantly large in high speed and/or long distance networks. This high access delay of CRMA has lead Nassehi [Nas90b] to develop the unallocated-slot access algorithm.

The unallocated-slot access algorithm requires an additional control bit, *allocation* (ALLC) bit, in the ACF of each slot. When the head-end initiates cycles on a bus, by issuing a START command, it sets the ALLC bit of the slots to 1. Otherwise, i.e., when the head-end issues no cycle, it sets the ALLC bit of the slots to 0. By observing the ALLC bit in each slot, a station can determine whether or not it is allocated through the reservation mechanism. If the slot is unallocated and empty (both ALLC and BUSY bits are 0), the station can transmit a data segment in that slot after setting the ALLC and BUSY bits to 1.

In order to preserve the slot-continuity property of CRMA, a packet transmission is assumed to be completed only when all its segments are transmitted in a set of consecutive slots. If the station detects a slot which is allocated and/or busy before the end of a packet transmission, the transmission is aborted. The station then tries to retransmit the entire packet. Since a station does not know in advance whether or not it will be able to transmit a packet using unallocated slots, it assumes that it will not, and reserves sufficient slots for each packet. When the reserved slots for a packet arrive, if the station has not been able to transmit that packet in a set of (consecutive) unallocated slots, it uses the reserved slots; otherwise, it deallocates (i.e., sets the ALLC bit to 0) those reserved slots, so that downstream stations can utilize them.

Chapter 3

Slotted Ring with Window Service Protocol

This chapter describes a new access scheme for high-speed LANs and MANs, the Slotted Ring with window service (SRWS) protocol. Section 3.1 describes how the window service access rule is used in a slotted ring. Section 3.2 proposes the SRWS with destination removal (SRWS/DR) protocol, which can double the capacity of a slotted ring by having the busy slots released by the destination stations.

3.1 Basic Medium Access Control Protocol

The **Slotted Ring with window service (SRWS)** protocol uses a *window-based* access rule, similar to that of the Token Ring with Window-Gated service (TRWS) protocol described in section 2.3.3. Empty slots in SRWS play a similar role to the idle token in TRWS, except that, in general, several empty slots (as opposed to one idle token) may need to be acquired to transmit all the packets in a given window.

SRWS uses a *timed-window* to select the set of packets that a station may transmit. Each station has a *local clock,* t_{clock}, and maintains a local variable, t_{window}, that indicates the upper limit of the most recently completed window, and exactly those packets (if any) with arrival times, t_{arr}, satisfying

$$t_{window} \leq t_{arr} \leq t_{window} + w \tag{3.1}$$

can be transmitted, where w is a design parameter specifying the maximum size of the window. Whenever a station does not have any more packets in the current window, it lets one empty slot go to the next station, and then advances to the next window via

$$t_{window} \leftarrow min\,\{t_{window} + w,\ t_{clock}\}\,. \tag{3.2}$$

Each station wishing to join the system initializes itself via $t_{window} \leftarrow t_{clock}$. Thereafter, each time a *usable* slot (i.e., an empty slot, or a busy slot that is being freed by this station) arrives, the station checks to see if there are any more data segments with arrival times within the current window, $[t_{window}, t_{window} + w]$. If there are, it sets the BUSY bit of the slot to 1 and transmits the next data segment in the DSF of the slot. Otherwise, it sets the BUSY bit to 0 and lets the slot go to the next station, then advances its window boundary according to equation (3.2).

It is interesting to note that the system is *self-synchronizing,* in the sense that new stations can join the system at any time, simply by initializing themselves via

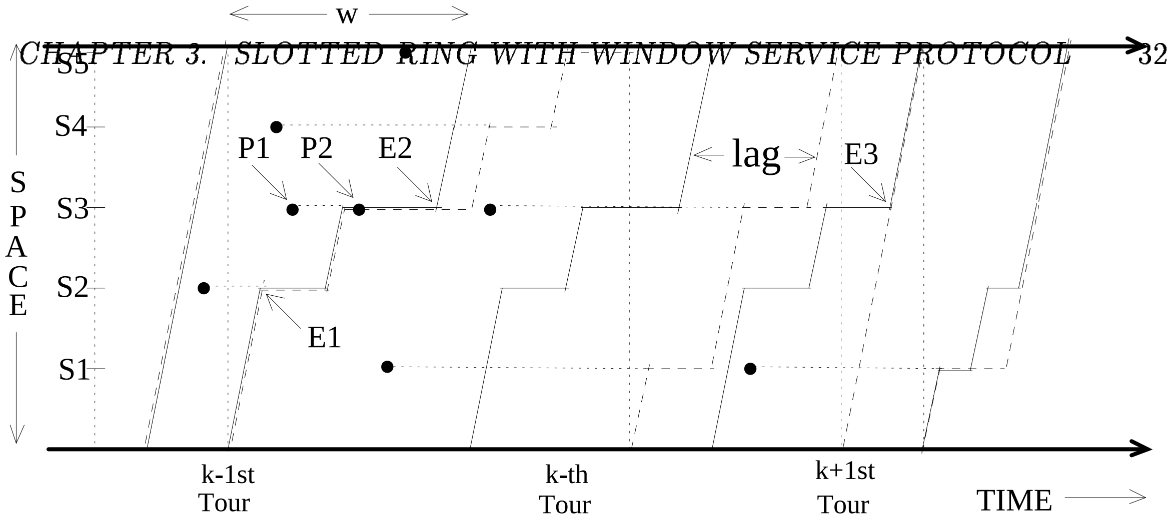
$t_{window} \leftarrow t_{clock}$. At first the new stations will experience lower access delays than the rest, but they will automatically re-synchronize with the remaining stations at the end of the first (and every subsequent) *busy period*, at the point where all stations carry out the assignment $t_{window} \leftarrow t_{clock}$. The operation of the SRWS protocol is summarized in table 3.1.

```
When a slot arrives:

If BUSY = 1 then
      If DA = the station's address or recognized multicast-address then
            Copy DSF into receiving buffer
      Else-if SA = the station's address then
            Classify the slot as USABLE
      End-if
Else
      Classify the slot as USABLE since BUSY = 0
End-if

If the slot is USABLE then
      If the station has a data segment with t_arr ∈ [t_window, t_window + w] then
            Set BUSY ← 1
            Set DSF ← next data segment
      Else
            Set BUSY ← 0
            Set t_window ← min{t_window + w, t_clock}
      End-if
End-if

Forward the slot to the next station
```

Table 3.1: Slotted Ring with Window Service (SRWS) Protocol.

Observe that as long as a station has packets belonging to the current window, it is allowed to capture *every usable slot* for delivering those packets, but as soon as those packets are delivered, the station must let one empty slot pass by before it can begin serving the next window. This makes the system more responsive than typical slotted ring protocols, such as the Cambridge ring, which limit the number of outstanding packets per station. Furthermore, when the system is idle, every slot is a usable slot and hence packets can be transmitted without any queueing or access delays.

Similar to the window-gated service rule in TRWS, our window service rule uses variable-size windows. However, the window service rule described in this section is *not gated,* i.e., packets that arrive after the service to the current window has begun but before the upper limit, $t_{window} + w$, are also transmitted in the same service period. When the system is "backlogged" (e.g., under heavy load), these two service rules are equivalent, since they both use the maximum size window, w. However, when the system is not "backlogged" (e.g., under light load), the effective window sizes in window service can be larger than those in window-gated service. The difference between window service and window-gated service can be seen in figure 3.1. In this figure, similar to figure 2.5, the busy period begins at event E1, the window reaches maximum size at event E2, and remains at maximum size until the end of the busy period at event E3. However, because of the difference between the window service and window-gated service, packet P2 is now transmitted in the same service period as packet P1, so that events E2 and E3 occur at station S3 instead of S4.



In this figure, the “solid lines” represent the *window boundaries;* the “dashed lines” represent the *token trajectory;* the “dots (•)” represent *packet arrivals;* and the “horizontal dotted lines” represent the *packet waiting times.*

Figure 3.1: Window Service Access Rule in SRWS.

3.2 SRWS with Destination Removal Protocol

In slotted rings, the responsibility of “freeing” a slot is usually assigned to the sending station. This has an advantage that the receiving station can send its *acknowledgement* (ACK) back to the sending station by simply setting the ACK bit of the given slot to 1. However, this mechanism also has a major disadvantage, namely that it wastes about half the capacity of the ring, because busy slots carry only one bit of useful information on the return trip from the destinations back to the senders.

Using piggy backed **medium access control (MAC)** level ACKs also complicates the access protocol (and slows down the processing at both sending and receiving sites) while providing only limited benefits, since these ACKs are issued immediately by the network interface hardware and do not indicate acceptance of the frame by higher layer protocols running in the host. As discussed in [SRC84], correct operation requires higher acknowledgements whether or not MAC level ACKs are used.

In this section, we describe the **SRWS with destination removal (SRWS/DR)** protocol, in which busy slots are released by the destination stations. Since each busy slot only travels half of the ring on average, the throughput of the system can be twice as high because each slot can be used twice per ring walk-time instead of once. With the SRWS/DR protocol, a destination station, upon observing a busy slot passing by, checks the DA field in the slot. If the DA field is a multicast (broadcast) address, then the destination station makes a copy of the data segment in the DSF of the slot and forwards it to the next station as before. However, if the DA field is equal to the station's (unique) address, it makes a copy of the data segment from the DSF of the slot and considers the slot to be usable. If the station has both a usable slot and at least one more data segment with an arrival time within the current window, $[t_{window}, t_{window} + w]$, it transmits the next data segment in the DSF of the slot (note that the BUSY bit of the slot is already set to 1). If the station has a usable slot but no more data segments in the current window, it resets the BUSY bit to 0 and lets the slot go to the next station, then advances its window boundary according to equation (3.2).

The source station, after completing the transmission of its data segment, removes that data segment from its local buffer (i.e., letting higher level protocols do the error checking and packet retransmission). If the data segment is sent to two or more destinations (multicast), then the source station expects the busy slot to return a ring walk-time later as before, but higher-level ACKs are still needed since the source station does not know which station sets the ACK bit. If the data segment is sent to a single destination, then the source station expects the destination to free the slot, but examines its status a ring walk-time later. If the slot is empty or contains another station's data segment then the source station gets an implicit ACK; otherwise the slot still contains the old segment, the source station gets an implicit NACK (negative acknowledgement) and the source station must release (or re-use) the slot. The operation of the SRWS/DR protocol is summarized in table 3.2.

Note that the SRWS/DR protocol described above requires the destination station to see the DA field in a slot in order to reset it. Thus, to implement the proposed destination removal algorithm, the storage at each ring interface has to be large enough to contain both the BUSY bit and the DA field. However, with the SRWS/DR protocol, there is a saving of 1 bit per slot, because the ACK bit in the slot is not needed, and thus can be used for transmitting data or other access control functions.

When a slot arrives:

```
If the slot is the one that was sent by this station a ring walk-time ago then
    If DA = the station's address then
        Get an implicit NACK
        Set BUSY ← 0
    Else
        Get an implicit ACK
    End-if
End-if

If BUSY = 1 then
    If DA = recognized multicast-address then
        Copy DSF into receiving buffer
    Else-if DA = the station's address then
        Copy DSF into receiving buffer
        Classify the slot as USABLE
    End-if
Else
    Classify the slot as USABLE since BUSY = 0
End-if

If the slot is USABLE then
    If the station has a data segment with t_arr ∈ [t_window, t_window + w] then
        Set BUSY ← 1
        Set DSF ← next data segment
    Else
        Set BUSY ← 0
        Set t_window ← min{t_window + w, t_clock}
    End-if
End-if

Forward the slot to the next station
```

Table 3.2: SRWS with Destination Removal (SRWS/DR) Protocol.

Chapter 4

Analytical Results

This chapter presents the analytical formulae for the system capacity and bounds on the mean packet delay of Slotted Ring with window service (SRWS). Section 4.1 first describes the system parameters and assumptions used. Section 4.2 introduces the moving server queueing system, the underlying queueing theoretic model that was used in the analysis of packet delays for the Token Ring with Window-Gated (TRWS) service protocol, and is used to obtain bounds on the packet delay for the SRWS protocol. The analytical formula for the system capacity is then presented in section 4.3. And finally, section 4.4 gives lower and upper bounds on the mean packet system time for SRWS.

4.1 System Parameters

In our analytical model, we make the following assumptions:

- The ring operates at B bit/s.

- The size of the slots are fixed at F bits.
- The length (L) of the ring is always chosen so that there are exactly R slots on the ring, i.e., there is no wasted gap.
- The size of the window is w slots.
- The system load is normalized at G $(0 < G < 1)$ packets per mean packet transmission time.
- There are N stations *arbitrarily* spaced around the ring.
- All N stations have equal load, with identical and independent Poisson packet arrivals at normalized rate of G/N packets per mean packet transmission time.
- The packet length has a general transmission time distribution with a mean of $\overline{x}$ units of time and a second moment of $\overline{x^2}$ squared units of time, independent and identically distributed (IID) for all stations.

From the parameters B, F, and R, the length (L) of the ring can be determined using the following equation:

$$R * \frac{F}{B} = T_{walk} = \frac{L}{v}\,, \tag{4.1}$$

where T_{walk} is the ring walk-time (or round-trip propagation delay) and v is the signal propagation speed. From equation (4.1), we see that R is the ratio of the round-trip propagation delay (T_{walk}) to the segment (frame) transmission time (F/B), hence R

is also the normalized propagation delay measured in units of slots, and $a = R/P$, where P is the mean packet length (in slots).[1]

4.2 Moving Server Queueing Systems

In a **moving server queueing system** [Mol89], the server traverses an infinite tour along which it encounters customers requiring service. The server, when it is not busy serving a customer, moves along the tour at one of two different rates depending on its position with respect to some constantly advancing *reference point.* When the server is caught up with the reference point, it advances in step with it; when it is behind (because it had to stop to serve a customer), it speeds up by a factor of η ($\eta > 1$) trying to catch up. Moving server queueing systems differ from conventional queueing systems, such as M/M/1, M/G/1, etc.. Instead of asking the customers to come to the server (and join a central queue) to obtained their service, moving server queueing systems require the server to come to the customers. This requirement incorporates a particular type of changeover time known as *moving server overhead.*

Molle [Mol89] analyzed the delay of moving server queueing systems for First-Come-First-served (FCFS) and Head-of-the-Line (HOL) priority orders of service with no specific assumptions on the arrival process and service time distribution. In his analysis, he showed that, by a suitable transformation, a moving server queueing system can be transformed into a conventional single server system, which he called

[1] The normalized propagation delay, a, is defined in the literature as the ratio of the propagation delay to the mean packet transmission time (see section 2.1).

a **synthetic queueing system,** with proportional waiting time. Thus, one is free to use the entire body of queueing theoretic results for conventional queueing systems to solve for the performance of the synthetic queueing system, and hence that of a moving server queueing system. Applications of moving server queueing systems include the Virtual-Time CSMA/CD [MK85], prioritized Virtual-Time CSMA/CD with HOL priorities classes [MW85, Mol91], window access tree conflict resolution (TCR) algorithm [PM87], Helical-Window and Window-Gated service Token Rings [Mol86, KM89, Mol90], and Slotted Ring with window service, which is the subject of this thesis.

4.3 System Capacity

The window service access rule in the SRWS protocol is an example of a moving server queueing system described in section 4.2. The window in SRWS can be viewed as the server in a queueing system that is moving along the time axis at a finite speed in search of customers (i.e., data segments to send). Thus, whenever the window encounters some data segments at its current position, it stops (allowing them to be transmitted at the rate of one segment per usable slot), and thereafter it resumes its motion.

There is a "cost" associated with advancing the window, namely an empty slot must be left unused. Thus, the maximum speed, η, at which the window advances is limited to w slots per empty slot (if the system is "backlogged," i.e., the leading window edge has fallen behind real time) or to one slot per empty slot (otherwise).

Note that since real time is advancing at the rate of one slot per slot, the rate at which the backlog decreases is only $(\eta - 1)$ slots per empty slot. Hence, SRWS includes an overhead factor of:

$$\beta = \frac{\eta}{\eta - 1} = \frac{w}{w - T_{slot}} \tag{4.2}$$

usable slots per transmitted frame, where T_{slot} is the slot transmission time. Thus at most $1/\beta$ of the slots can be utilized for carrying data segments. Therefore, from equation (4.2), the capacity of the SRWS protocol is limited to:

$$\overline{x}G \leq \frac{1}{\beta} = \frac{w - T_{slot}}{w}\,. \tag{4.3}$$

Notice that the capacity of SRWS depends only on the window size, w (measured in slots); it does not depend on the network speed, distance or the number of stations. Also, the capacity of SRWS is directly proportional to the window size, hence the mean packet delay decreases monotonically as the window size increases.

The operation of SRWS is closely related to Token Ring with Window-Gated service (TRWS) described in section 2.3.3, with the major difference being that the window size in TRWS must be taken as η times the total walk-time (T_{walk}) of the ring instead of η times the size of a single slot (T_{slot}), as in SRWS. For the same window size, w (but measured in two different units), SRWS and TRWS have the same speed-up factor (η), overhead factor (β), and capacity ($\overline{x}G$).

4.4 Bounds on Mean Packet Delay

Similar to window-gated service in token rings, our window service rule for slotted ring introduces some dependence among the queues at different stations, which makes its exact analysis difficult. Indeed, SRWS has a major complication due to the presence of multiple slots – which leads to a *non-work conserving multi-server queueing model.* Finding an exact analytical solution for the mean packet delay of SRWS appears to be a hard problem, and is beyond the scope of this thesis. Nevertheless, we were able to find lower and upper bounds for the mean packet system time, which are presented in sections 4.4.1 and 4.4.2, respectively.

4.4.1 Lower Bound

When there is only one slot on the ring, the operation of SRWS is equivalent to TRWS, except that long packets are now segmented into fixed-size segments and sent in consecutive slots. Thus, since we do not include any overhead for the tokens or slot headers, the mean packet system time of the SRWS protocol is the same as that of the TRWS protocol when $R = 1$ slot. Therefore, the analytical formula for the mean packet system time of TRWS in [Mol90] is valid for a 1-slot SRWS system when T_{walk} is relaced by T_{slot}, which gives

$$\overline{T}_{SRWS|R=1} = \overline{x} + \frac{T_{slot}}{2}\left[1 + \frac{G\overline{x}}{N}\right] + \frac{\beta G\overline{x^2}}{2(1-\beta G\overline{x})}\,. \tag{4.4}$$

Using arguments similar to those used by Kleinrock in [Kle76, section 5.1], where he shows that a queueing system with a single "fast" server and a single queue has better performance than those with separate "slow" servers and/or multiple queues, we can see that the mean packet system time of an 1-slot SRWS is a lower bound for that of an R-slot ($R > 1$) SRWS system. In Kleinrock's case, the advantage of a single fast server was that the entire service capacity of the system could be put to work as soon as one customer entered the system, whereas in a multi-server system some capacity would be wasted until more customers arrived. In our case, the argument is based on large windows being more efficient than small ones. Thus, in an 1-slot system, all stations begin a busy period within 1 slot-time instead of R slot-times. Since larger (and more efficient) windows are used within busy periods, a 1-slot system switches to its most efficient mode of operation more quickly than an R-slot system. Thus, for the same system load G, an 1-slot SRWS has lower mean packet delay than an R-slot SRWS. Therefore, the formula in equation 4.4 for the mean packet system time of a 1-slot SRWS system is also a lower bound for a SRWS system with $R > 1$ slot, i.e.,

$$\overline{T}_{SRWS} \geq \overline{x} + \frac{T_{slot}}{2}\left[1 + \frac{G\overline{x}}{N}\right] + \frac{\beta G\overline{x^2}}{2(1-\beta G\overline{x})} = \overline{T}_{SRWS|R=1} \,. \tag{4.5}$$

4.4.2 Upper Bound

To find an upper bound for the mean packet system time, we consider the following modifications that will increase the packet delay at the tagged station:

i) Increase the length of the ring and the number of stations. If the ring is "short", then the tagged station can "capture" a particular slot until it has emptied its current window. If the ring length is large compared to the length of a busy period (as seen by the tagged station), then the return of a captured slot will be too late to be useful. The worst case is the length of the ring $R = \infty$, i.e., no busy slots will return. Furthermore, under the symmetric workload assumption, increasing the number of stations means that the return of captured slots are less frequent. In the limit, they can be ignored.

ii) Increase the window size at other stations. The *tagged station* follows the usual algorithm, but all other stations can send as many segments as possible before releasing an empty slot to the tagged station. Thus, the other stations can, on average, send more packets ahead of the tagged station, and the tagged station has to wait longer for empty slots to be released from them. The worst case is the other stations use window size $w = \infty$, i.e., exhaustive service.

iii) Aggregate the $(N-1)$ remaining stations into a single "big" station with aggregated workload demands $((N-1)G/N)$. This single aggregated station has the same average load as the $(N-1)$ separate stations, but it is more effective at blocking out the tagged station. Consider two "other" stations S_i and S_j, where S_i is upstream of S_j, who is in turn upstream of the tagged station. Assume that S_i has more segments to send after S_j has run out of segments. If a busy slot previously used by S_j comes by, then S_i cannot use it even though S_i has more data because it is still busy until it returns to S_j, and S_j does not use it because S_j has no more data. Thus, the tagged station gets to use it even though S_i still has waiting segments. However, if S_i and S_j were aggregated into one, then S_i could re-use the slot S_j would have freed, and the tagged station is delayed even more. Notice that the relative position of the aggregated station with respect to the tagged station is irrelevant, since we would get the same behavior if we shift the arrival process by the inter-station propagation time.

The resulting system from the above three modifications is a 2-class Head-of-the-Line (HOL) moving server queueing system with *preemptive priority* for *other stations,* and lower priority for the tagged station. There is no moving server overhead associated with the higher priority class, however, since the window size for other stations $w_o \to \infty$ implies that $\beta_o \to 1$. Therefore, the mean system time of the lower class in a 2-class HOL moving server queueing system is an upper bound for the mean system time of SRWS.

If we let P_o be the probability that a slot is used by a station other than the tagged station, that is

$$P_o = P[\mathit{slot\ is\ used\ by\ other\ stations}] = (N-1)\frac{G\overline{x}}{N}\,, \tag{4.6}$$

then using the results from moving server with HOL priority service in [Mol89] and preemptive HOL priority queue in [Kle76] with *preemptive priority* for *other stations,* the formula for an upper bound on the mean packet system time of SRWS is:

$$\overline{T}_{SRWS} \leq \frac{\overline{x}}{(1-P_o)} + \frac{T_{slot}}{2}\left[1 + \frac{G\overline{x}}{N(1-P_o)}\right] + \frac{\beta G\overline{x^2}}{2(1-P_o)(1-\beta G\overline{x})}\,. \tag{4.7}$$

Chapter 5

Simulation Results

Since our efforts at finding an exact analytical solution for the packet delay were unsuccessful, we chose to develop a simulation model to investigate the performance of Slotted Ring with window service (SRWS) in more detail. The simulation results are also used to evaluate the accuracy of the analytical results presented in chapter 4. In this chapter, section 5.1 first describes the methodology and assumptions used in our simulation. The simulation results for SRWS with source removal are then presented in section 5.2. And finally, section 5.3 discusses on the performance of SRWS with destination removal (SRWS/DR).

5.1 Simulation Methodology

5.1.1 Simulation Parameters

Our simulation parameters are similar to those for the analytical model described in section 4.1 except that, to reduce the complexity of the simulation, we assume that the N stations are equally (instead of arbitrarily) spaced around the ring. However, this assumption will have no effect on the performance of SRWS, because in slotted rings all locations are equivalent, i.e., there is no position advantage or disadvantage. The assumptions used in our simulation model are:

- The ring operates at B bit/s.
- The size of the slots are fixed at F bits.
- The length (L) of the ring is always chosen so that there are exactly R slots on the ring, i.e., there is no wasted gap.
- The size of the window is w slots.
- The system load is normalized at G ($0 < G < 1$) packets per mean packet transmission time.
- There are N stations *equally* spaced around the ring.
- All N stations have equal load, with identical and independent Poisson packet arrivals at normalized rate of G/N packets per mean packet transmission time.

5.1.2 Traffic Models

To study the performance of SRWS under different workload characteristics, we consider three different traffic models:

- **Basic Poisson Traffic Model**

Under basic Poisson traffic model, we assume that packet arrivals follow a Poisson process and that the packets fit perfectly into a single slot. That is, the packet length distribution is a *constant* with a mean of 1 slot and a squared coefficient of variation of zero. Although Poisson traffic model is probably quite unrealistic, it is used in many analytical and simulation studies because of its tractability property.

- **Hypergeometric Packet Length Traffic Model**

Under this traffic model, we still assume that packet arrivals follow a Poisson process. However, this time the packet length distribution is a truncated *hypergeometric* with a mean of 10 slots, a squared coefficient of variation of 1.5, and a maximum of 150 slots. The parameters used in this traffic model were chosen for compatibility with [MNW+90], where the performance of CRMA protocol was studied. In this thesis, the hypergeometric packet length traffic model is used to characterize bursty computer traffic with a moderate squared coefficient of variation.

- **Trimodal Packet Length Traffic Model**

Under this traffic model, we again assume that packet arrivals follow a Poisson process. But this time we assume that the packet length distribution is a *trimodal* with three different packet sizes:

  ◦ Short packet: 70% of the packets are 1K bits in length or less
  (half of these packets are 1 slot, other half are 2 slots).

- ◦ Medium packet: 20% of the packets are 16K bits in length (32 slots).
- ◦ Long packet: 10% of the packets are 100K bits in length (200 slots).

This trimodal distribution has a mean of approximately 13.75K bits (27.5 slots of size 500 bits) and a squared coefficient of variation of approximately 4.6. The assumptions used in this traffic model are based on the study in [CSK88]. They also conform with measurement studies done by Gusella, reported in [Gus88, Gus90], who found that most packets are short, but most of the data are carried in long packets. In this thesis, the trimodal packet length traffic model is used to characterize bursty computer traffic with a high squared coefficient of variation.

5.1.3 Performance Metrics

From the simulation, we collect the following performance measures of interest:

- System utilization: The system utilization is the proportion of slots that are used to carry data segments.

- Packet system time (mean and variance): The system time of a packet is the sum of packet waiting time (queueing delay) and packet transmission time, i.e., from the arrival time at the station until the last bit of the packet is transmitted. Since the slot length is fixed at F bits, packets that are longer than F bits have to be segmented into a number of F-bit segments. When a packet is segmented, the system time of the packet is the system time of the last segment of that packet. To permit easy comparisons with other protocols that do not require segmentation, in our numerical results we normalize the packet system time by the mean packet transmission time, not the slot size.

- Frame access delay (at each station): The frame access delay is defined as the sum of the channel waiting time and the segment (frame) transmission time, i.e., from the time the segment reaches the head of its local queue until the last bit of the segment is transmitted. Unlike the packet system time, which includes the waiting time while a segment is in its local queue, and hence may depend on local traffic characteristics at each station, the frame access delay concentrates on the delay caused by the medium access control (MAC) protocol. The frame access delay is measured in units of slots.

- Fairness ratio: The fairness ratio among the stations on the network is defined as the ratio of the minimum and maximum mean frame access delays among all the stations on the network.

5.1.4 Simulation Design and Implementation

We designed and implemented our simulation model in "*C*", using approximately one thousand lines of code. Each station has a limited buffer of 2,000 segments, and packets will be dropped if the buffer is full. (This buffer size is large enough for all three traffic models considered, since we only found packets dropped when the system was overloaded.) The simulation was run on *Silicon Graphics SGI 4D/240* and *Digital Equipment DECstation3100* workstations. The simulation uses the recursive batch mean analysis procedure described in appendix A with the number of batches $k = 20$. Each simulation run begins with the minimum batch size of 500 packets, and terminated when the batches satisfy a test for approximate independence and the half-width of the 95% confidence interval for the mean packet system time reached a relative width of 5%, or when the batch size reaches the maximum batch size of 16,000 packets. We only found this maximum batch size reached when the system load is close to the system capacity. We also found that, in all of the cases we studied, the first 2 batches (1/10 of the simulation run time) is more than large enough to cover the initial transient period.

5.1.5 Baseline Configuration

The *baseline configuration* is defined by the set of parameter values in table 5.1. Between various configurations, we keep as many parameters constant as possible to study the sensitivity of the system performance to changes in the given parameters. Each experiment can be described by indicating how the configuration differs from the baseline configuration. A complete list of all simulation parameters used in this thesis is tabulated in table 5.2.

In slotted rings, the slot size is a significant design parameter. It has to be chosen according to the packet length distribution to optimize the system performance. If the slot size is too small, then large packets will have to be segmented into many segments, and the per-segment header overhead will reduce the network capacity significantly. On the other hand, if the slot size is too large, then padding small packets to the slot size will also waste the network capacity. In our simulations, we follow DQDB and CRMA in choosing the slot size close to that of an ATM "cell". DQDB uses a slot size of 69 octets (552 bits) [IEE90], and CRMA uses slot size of

Network Parameters	Notation	Value	
Number of stations	N	10	stations
Network speed	B	100	Mbit/s
Network length	L	100	km
Number of slots	R	100	slots
Window size	w	10	slots
Slot size	F	500	bits
Header overhead	–	0	bit
Buffer size	–	2,000	segments
Simulation Control Parameters	Notation	Value	
Number of batches	k	20	batches
Minimum batch size	m_{min}	500	packets
Maximum batch size	m_{max}	16,000	packets
Confidence level	–	95%	
Relative error in Confidence interval	γ	5%	

Table 5.1: Parameters for Baseline Configuration.

71 octets (568 bits) [MNW+90]. For ease of conversion between the number of slots on the ring (R) and the length of the ring (L), we choose the slot size $F = 500$ bits (62.5 octets). In a practical implementation, compatibility would dictate a slightly different slot size (in the range of 65 – 70 octets). However, this change will have no significant effect on the performance of the SRWS protocol.

Case	Figures	N (stations)	R (slots)	B (Mbit/s)	L (km)	w (slots)
1	5.1 – 5.4, 5.7	10	100	100	100	2 to 100
2	5.5, 5.8	10	1 to 2000	1 to 2000	0.1 to 200	10
3	5.6, 5.9	5, 10, 25, 50	100	100	100	10
4	5.10, 5.11	50	100	100	100	10
5	5.12, 5.13	10	100	100	100	2 to 100
6	6.1	50	150	150	100	20
7	6.2	50	1 to 1000	1 to 1000	0.1 to 100	20
8	6.3, 6.4, 6.5	50	15, 150	150	10, 100	20

Table 5.2: Summary of Network Parameters used in Simulations.

5.2 SRWS with Source Removal

In figures 5.1 to 5.9, we used our simulation model to study the effects of system parameters on the performance of the basic SRWS protocol, assuming busy slots are released (or re-used) by the source stations. In all cases, our starting point is the baseline configuration described in section 5.1.5. In figures 5.1 to 5.4 and 5.7, we varied the window size over the range w = 2, 4, 10, 20 and 100 while keeping R and N fixed. In figures 5.5 and 5.8, we varied the normalized ring walk-time over the range R = 1, 10, 100, 1000, and 2000 while keeping w and N fixed (actually, we ran this test twice – by fixing the speed and varying the ring size, and also by fixing the ring size and varying the speed – and got virtually identical results in the two cases). And in figures 5.6 and 5.9, we varied the number of stations over the range N = 5, 10, 25, and 50 while keeping w and R fixed (see table 5.2). For each set of parameters, we run the simulation for all three traffic models described in section 5.1.2.

5.2.1 Throughput-Delay Characteristics

Figure 5.1 shows that, for this set of parameters, the performance statistics for the SRWS protocol improve monotonically (i.e., capacity increases and mean and variance[1] of system time decreases) as the window size increases. When the window size is small ($w \leq 10$), increasing the window size can substantially improve the performance of SRWS; however, when the window size is large ($w \geq 20$), increasing the

[1]Note that because the variance of the system time gets vary large when the system load is close to the capacity, we plot the square root of the variance, i.e., the standard deviation, rather than the variance.

window size led to little improvement on the mean and variance of system time.

These simulation results were also used to evaluate the accuracy of the analytical formulae for the bounds on the mean packet system time presented in chapter 4. All our simulation results agree closely with the analytical formula for the system capacity in equation (4.3). In particular, for window sizes in the range $w = 2$, 4, 10, 20, and 100 slots, the corresponding capacities are 50%, 75%, 90%, 95% and 99%, respectively (see figure 5.1). In figure 5.2, we include the lower and upper bounds for mean packet system time, which were calculated using equations 4.5 and 4.7. For all three traffic models we considered, the simulation results show that these two analytical formulae do indeed provide useful lower and upper bounds for the mean packet system time of SRWS. It is interesting to note that as the coefficient of variation for the packet size distribution increases, our lower bound becomes quite tight.

In all the cases we studied, we found that the Poisson traffic has the lowest system time (both mean and variance), while the hypergeometric packet length traffic has the medium, and the trimodal packet length traffic has the highest. This is because these three traffic models have packet lengths with different means and squared coefficients of variation. The Poisson traffic has the lowest mean and squared coefficient of variation ($\overline{x} = 1$, $C_b^2 = 0$); the hypergeometric packet length traffic has the medium mean and squared coefficient of variation ($\overline{x} = 10$, $C_b^2 = 1.5$); and the trimodal packet length traffic has the highest mean and squared coefficient of variation ($\overline{x} = 27.5$, $C_b^2 = 4.6$). Although we have normalized the system time by the mean packet transmission time, the differences in the squared coefficients of variation make the system time for these three traffic models different. This can be easily seen by studying the analytical formulae for the bounds on mean system time in equations 4.5 and 4.7. Both these two formulae have the second moment of the packet transmission time, $\overline{x^2}$, as a factor in their last terms, and the second moment of a random variable is directly proportional to its mean and squared coefficient of variation, i.e., $\overline{x^2} = \overline{x}^2(1 + C_b^2)$.

From figure 5.1, we see that increasing the window size can improve the efficiency of the protocol, however, increasing the window size without bound also changes the character of the protocol into an exhaustive service system. With exhaustive service, a heavily loaded station can monopolize the channel, locking out other stations for extended periods of time. This causes the fairness among the stations to be degraded. This effect is illustrated in figure 5.3, where the mean and variance of system time is plotted against "β times" the throughput (i.e., the throughput relative to the capacity for the given moving server overhead factor). When the window size is small ($w < 10$), the mean and variance of the system time of SRWS is close to that of an M/G/1 queueing system with First-Come-First-Served (FCFS) order of service.[2] However, both the mean and variance of system time increase monotonically as the window size increases. Thus, when the window size is small ($w < 10$), SRWS has a global order of service that is close to FCFS, i.e., it is fair but has some loss in capacity due to the moving server overhead. But when the window size is large ($w \gg 10$), it converges to an exhaustive service system, i.e., it has full capacity but exhibits some unfairness among the stations.

[2]The mean and variance of the system time of the slotted M/G/1 queueing system with FCFS scheduling are calculated using equations 6.1 and 6.2, respectively, in chapter 6.

The effect of large window size becomes even more apparent when we examine figure 5.4, where the mean and variance of packet system time is plotted as functions of the window size. From figure 5.4, we see that, under light load ($G = 0.1$), both the mean and variance of system time are very low and independent of the window size. For medium loads ($G = 0.5$ and 0.8), because of high moving server overhead, small window sizes ($w \leq 4$) have high packet system time (both mean and variance). However, when the system is heavily loaded ($G = 1.2$), even though the mean system time slightly decreases when the window size increases, the variance of system time increases rapidly as the window size increases.[3]

Figure 5.5 shows that network size has very little effect on the performance of the SRWS protocol. Varying the ring size (or network speed) over more than three orders of magnitude had no effect on the capacity of the protocol. In addition, this change had only a modest effect on the mean and variance of system time. (Notice that we have normalized the system time by the mean packet transmission time, so that simultaneously changing both speed and length, so as to preserve the normalized ring walk-time, has no effect on the normalized packet system time.) Thus, the capacity and system time of the SRWS protocol is insensitive to the normalized walk-time, and hence to both the speed and length of the network. Once again, we have shown the analytical formulae for upper and lower bounds to the mean system time. As before, the bounds enclose all our simulation data, and the lower bound becomes tighter as the coefficient of variation for the packet size increases.

[3]Note that, because of the finite buffering in our simulation, we show finite values of the mean and variance of the system time under overload conditions.

Figure 5.6 shows that varying the number of stations on the network has negligible effect on the capacity, mean and variance of system time of SRWS. This is to be expected, since the windows from different stations are advancing at approximately the same rate, so the time at which a given packet is selected by the access protocol depends only on its arrival time and not on the identity of the station where it arrived. Thus, in summary, we see that varying the window size gives us a tradeoff between efficiency and fairness, whereas varying the network size, speed, or the number of stations has very little effect, if any, on the performance of the protocol.

5.2.2 Fairness Among Stations

Figures 5.7, 5.8 and 5.9 show the fairness ratio among the stations from the three simulation experiments: varying the window size, varying the normalized ring walk-time, and varying the number of stations, respectively.

Figure 5.7 shows that the fairness ratio among the stations remains around 95% for all system loads for the window sizes between 2 and 20 slots, but it drops down below 90% (for some system loads) when the window size $w = 100$. This is because, as discussed before, when the window size is too large the operation of the protocol converges to an exhaustive service system, and the fairness among the stations cannot be guaranteed. The fairness ratio is obviously negatively correlated with the burstiness of the traffic, but it does not seem to be related to either the system load or the window size.

Figure 5.8 shows that network size has no effect on the fairness of the SRWS protocol. For all network sizes, the fairness ratio remains around 90% for all system loads. Again, traffic burstiness seems to have the strongest correlation to the fairness ratio. Thus, the fairness of the SRWS protocol is insensitive to the network speed and length.

Figure 5.9 shows that the fairness ratio slightly decreases when the number of stations is large. But this may be an artifact of the small amount of traffic generated per station over a simulation run when the number of stations is large. Indeed, from the details of mean frame access delay at each station for the case $N = 50$ stations shown in figures 5.10 and 5.11, we see that all stations have approximately the same mean frame access delay for all system loads and all three traffic models. Thus, using Little's law, we can conclude that the difference in the throughput between the stations is primarily due to statistical variations in the packet arrival rates rather than any effects caused by the SRWS protocol. These two figures also show that, in slotted rings, all stations are treated equally, i.e., there is no position advantage or disadvantage. Thus, the fairness of the SRWS protocol is independent of the number of stations and their positions on the network.

Similar to the packet system time, because of the differences in the means and squared coefficients of variation of packet lengths in the three traffic models we considered, the Poisson traffic (which has the lowest mean and squared coefficient of variation) has the highest fairness ratio, and the trimodal packet length traffic (which has the highest mean and squared coefficient of variation) has the lowest fairness ratio. However, for all three traffic models, the fairness ratios are always around 90%

for all system loads and all the cases we studied.

5.3 SRWS with Destination Removal

To study the performance of SRWS/DR, we modified our simulation to allow the destination station to release or re-use the busy slots that were addressed to them. We then ran our modified simulation for the set of parameters with $N = 10$ stations, $R = 100$ slots ($B = 100$ Mbit/s, $L = 100$ km), and $w = 2$, 4, 10, 20 slots, and compared these results to those for SRWS with source removal described in section 5.2.

Figures 5.12 and 5.13 compare the mean and variance of packet system time of SRWS and SRWS/DR, respectively. The results for SRWS are taken directly from figure 5.1, whereas the results for SRWS/DR are obtained from running the modified simulation with the destination removal algorithm included. From these two figures, we see that the throughput (and capacity) of SRWS/DR is indeed twice of that of an SRWS. These figures also show that, for any system load that is feasible for SRWS, SRWS/DR has much lower mean and variance packet system time. Thus, just as DQDB and dual-bus CRMA can use two buses, instead of a folded bus, to double system capacity, the destination removal algorithm can be used to double the capacity of a slotted ring.

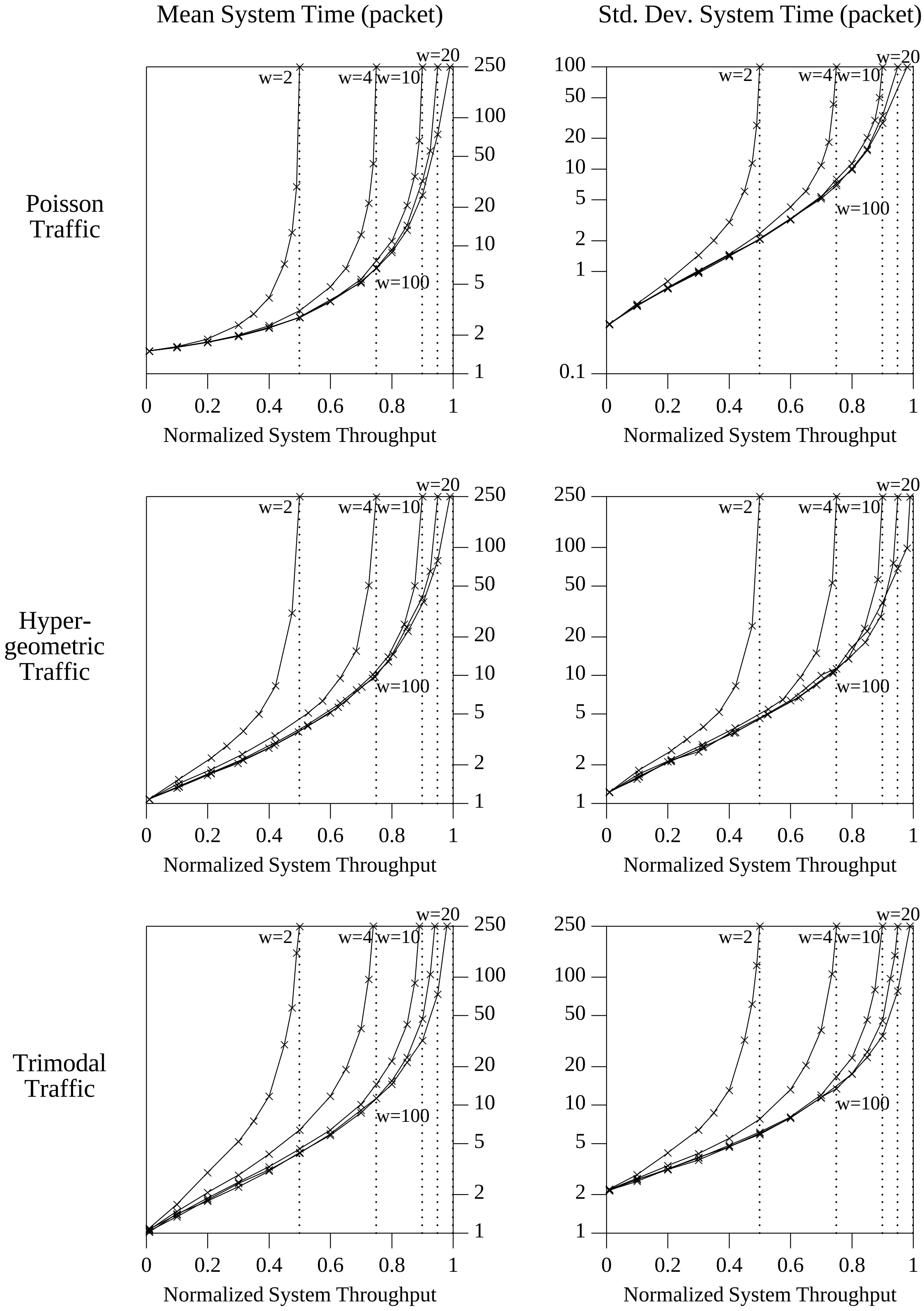


Figure 5.1: Packet Delay vs Throughput for Varying Window Size.

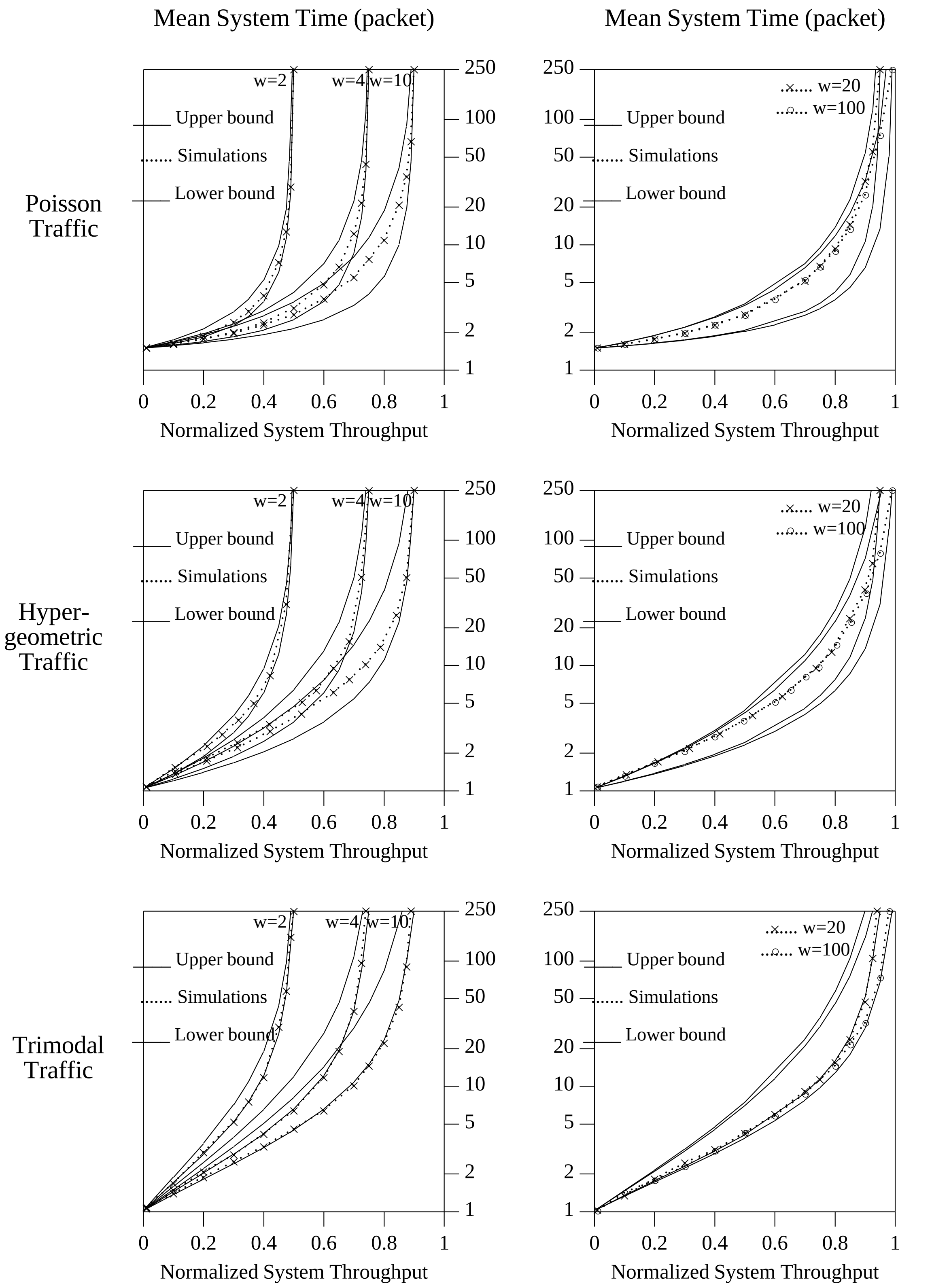


Figure 5.2: Bounds on Mean Packet Delay for Varying Window Size.