Real-Time Communications in Token Ring Networks

by

Li-Jun Yao

A thesis submitted for the degree of

Doctor of Philosophy

in

Department of Computer Science

The University of Adelaide

January 1994

Awarded 1994

Abstract

Real-time communications differ from traditional data communications as it imposes a time constraint, such as *deadline*, on each message transmission. It is well known that the Earliest Deadline First (EDF) scheduling policy is optimal for task scheduling in a centralized real-time system. However, implementing the EDF policy in a distributed real-time system is different from its counterpart in a centralized real-time environment due to its non-negligible scheduling overhead involved. This work deals with the implementation of the EDF policy in the context of a specific distributed system — a token ring local area network. Its main objectives are (i) to propose a token ring protocol which implements the exact network-wide EDF policy for real-time message transmission, and (ii) to address the fundamental issue of the appropriate level of implementing an optimal scheduling policy in a distributed real-time environment.

In brief, the work is concerned with the design and performance evaluation of three token ring protocols for real-time communications, which implement variations of the EDF transmission policy with different overheads. The first is an existing *token passing* protocol which does not adhere to the EDF policy but has a minimal overhead. The second is a modified *priority-driven* protocol which approximates the EDF policy with a moderate overhead. The third is the *window* protocol which is proposed for the token ring networks for the first time. It implements the exact EDF transmission policy, but its contention overhead may be potentially high.

The worst case performance of the three protocols is analyzed and compared. It is found that the performance of a distributed communication protocol is determined not only by the transmission policy employed, but also by the contention overhead incurred when implementing such a policy. It is also shown that no protocol can always outperform the others for the entire parameter ranges considered and that each protocol has its own applicable region where its performance is the best.

Furthermore, the average case performance of the three protocols is evaluated through simulation. It is found that under the current token ring network technology, the proposed window protocol achieves the best performance as a result of implementing the EDF policy. However, it is also shown that when the ring gets faster, the difference in the performance of the three protocols is reduced.

Therefore, it is concluded that in designing a distributed scheduling algorithm, such as a communication protocol, one should seek a balance in achieving an optimal scheduling policy and minimizing the scheduling overhead.

Declaration

This work contains no material which has been accepted for the award of any other degree or diploma in any university or other tertiary institution and, to the best of my knowledge and belief, contains no material previously published or written by another person, except where due reference has been made in the text of the thesis.

I give consent to this copy of my thesis, when deposited in the University Library, being available for loan and photocopying.

SIGNED:....................................   **DATE:** Jan 1994

Acknowledgments

I wish to express my deep gratitude and appreciation to my former supervisor Prof. Wei Zhao for suggesting this project and for his devoted guidance, support and help. I am very grateful to Prof. Chris Barter for having facilitated my coming to Australia to undertake this research and for his immense enthusiasm, support and encouragement as my later supervisor. I thank Dr. Moshe Zukerman for expanding my research horizon tremendously when I was working in the Network Analysis Section, Telecom Research Laboratories in 1991. I also thank Dr. Cheng-Chew Lim from the Department of Electrical and Electronic Engineering for his helpful advice and comments.

I am deeply indebted to my parents and my sister who are remote but near and whose love, understanding and support are invaluable. Special thanks to Piero Ammirato for his love and support and for sharing the good times and bad times over the past four years. I thank Fred Thornett who introduced Australia to me seven years ago for his friendship, help and confidence in me. Finally, I am thankful to all the staff members in the department and all my friends who in one way or another have contributed to the making of this thesis.

This work was supported by the Univerity of Adelaide Postgraduate Research Scholarship.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation and Scope

The need for high speed real-time communications has emerged rapidly over the last decade from many new real-time applications, such as office automation, intelligent manufacturing, advanced air traffic control, space and military projects, distributed processing and robotics systems [6, 12, 18, 31, 45]. Real-time communications differ from traditional data communications as it imposes explicit timing constraints on individual messages. Hence, the correctness of a real-time system depends not only on the logical results of the computation but also on the time when the results appear [20, 33, 35, 36, 59].

Local area networks (LANs) have gained increasing popularity in supporting these new real-time applications due to their distributed nature and the potential for providing high speed reliable resource sharing. The key to the successful use of a LAN to support real-time communications is an adequate distributed scheduling algorithm, which ensures the timely message transmission on geographically distributed nodes.

In relation to Open System Interconnection (OSI) reference model, the Medium Access Control (MAC) protocol is primarily responsible for scheduling message transmission over a LAN. It arbitrates access to the network and determines which message is to be transmitted at any given time. The main design considerations of existing MAC protocols are to maximize the throughput and to minimize the average delay. In contrast, a MAC protocol for transmission of real-time messages must address the timing constraints of individual messages. The most important design objective of a real-time MAC protocol is to ensure that message deadlines are met or that the number of messages that miss their deadlines is minimized [21]. Therefore, any traditional MAC protocols that do not consider individual message timing requirements are inadequate.

Clearly, protocols intended for distributed real-time communications must schedule message transmission based on global message timing information. However, this objective may not be achieved in practice, as scheduling real-time messages over a LAN differs from scheduling tasks in a processor environment. In a LAN, messages are located on physically separated nodes and their timing information is local to the nodes, hence gathering global message timing information not only may incur a non-negligible overhead, but also may be approximate. As a result, it is conceivable that the benefit of employing an optimal centralized scheduling algorithm can be nullified in a distributed environment if the implementation overhead is too high. Consequently, under some network and traffic conditions it may be desirable that scheduling decisions are made with less accurate knowledge about the timing constraints of messages on other nodes in the network. Thus, one of the fundamental issues in the design of real-time MAC protocols is to determine the appropriate level of implementing the optimal scheduling policy in a distributed real-time environment.

Much progress has been made in designing protocols for real-time message transmission in LANs over the last decade, predominantly using CSMA/CD (Carrier Sense Multiple Access/Collision Detection) and ring networks. The approach taken by CSMA/CD related work is to incorporate well-established centralized scheduling algorithms to the basic CSMA/CD networks to support real-time message transmission. As a result, these enhanced CSMA/CD protocols support both real-time synchronous and asynchronous message transmission. On the other hand, the majority of the solutions developed for ring networks only consider timing constraints of real-time synchronous/periodic messages. Furthermore, the issue of achieving scheduling optimality and minimizing scheduling overhead has not been addressed.

The main objectives of this work are

- to propose a token ring protocol which implements the Earliest Deadline First (EDF) optimal scheduling policy for real-time message transmission, and
- to address the fundamental issue of the appropriate level of implementing an optimal scheduling policy in a distributed real-time environment.

Specifically, the work is concerned with the design and performance evaluation of three token ring protocols for real-time communications, which implement variations of the EDF transmission policy.

The first is an existing *token passing* protocol which sends messages in the nearest-neighbor-first order and does not consider individual message timing requirements at all. As a result, this simple token passing protocol does not adhere to the EDF policy but has a minimal overhead.

The second is a modified *priority-driven* protocol. Although the most existing token ring protocols are priority-driven, they are not suitable for transmitting time-constrained messages as they do not explicitly address the individual message deadlines. Hence, we propose to incorporate a priority assignment function which assigns each message an access priority based on its deadline. This modified priority-driven protocol then sends messages in the highest-priority-first order. It can be shown that when the number of priority levels is sufficient, the protocol implements the exact EDF policy; otherwise it only approximates the EDF policy. Thus this protocol considers the individual message deadlines to certain degree with a moderate overhead.

The worst case performance of the token passing and the priority-driven protocols for real-time message transmission is analyzed. The results show that the worst case performance of the two protocols is poor. Specifically, given a message set, the two protocols can send only half of the messages even if the overhead is assumed to be zero. This implies that the dominant factor in deciding the protocol performance in supporting real-time message transmission is whether or not the transmission policy used considers the individual message timing requirements explicitly. Therefore, it is imperative to design a token ring protocol which implements the exact EDF policy.

This leads to the third protocol which is a *window* protocol specifically designed for token ring networks. It is an original contribution of this work. Towsley and Venkatesh [43] proposed the original window protocol for non real-time message transmissions in LANs. A desired property of the window protocol is that it can uniquely locate a message according to some message parameter by partitioning the window recursively. Zhao *et al* and Znati have successfully designed window protocols for transmission of real-time messages in CSMA/CD networks. The CSMA/CD window protocols are contention based, in which stations schedule message transmission without the global knowledge of message deadlines. When a collision occurs, stations back off and the window is partitioned in a binary manner.

Motivated by the CSMA/CD window protocols, our goal is to apply the concept of window protocol to a token ring network, which transmits messages in the earliest-deadline-first order. However, unlike the CSMA/CD networks, token ring networks are control based and collision-free. Thus, it requires a special controller to gather global message deadline information in order to coordinate window operations and to locate the earliest deadline message. As a result our new window protocol for token ring networks differs significantly from the existing window protocols for CSMA/CD networks. It will become clear later that this new window protocol is much more sophisticated in window operations and has much faster convergence in locating the desired message than the existing CSMA/CD window protocols. It can be shown that the proposed window protocol implements the exact EDF transmission policy, but the need for gathering global message deadline information indicates that its overhead may be high.

The worst case performance of the three protocols is analyzed and compared. It is found that no protocol can always outperform (in terms of the fraction of messages sent) the others for the entire parameter ranges considered. It is also shown that each protocol has its own applicable region where its performance is the best. This implies that the performance of a distributed communication protocol is determined not only by the transmission policy employed, but also by the contention overhead incurred when implementing such a policy. As a result, there may not exist a communication protocol for message transmission which would be optimal for all operating environments.

Furthermore, the average case performance of the three protocols is evaluated through simulations. Simulation results show that the average case performance of a protocol, although significantly better than the worst case case performance, is still determined by the transmission policy used and the contention overhead incurred. More specifically, it is found that under the current technology, the proposed window protocol achieves the best performance as a result of implementing the EDF policy. However, it is also noted that when the ring gets faster, the difference in the performance of the three protocols is reduced. This implies that under very high speeds the benefit of implementing the EDF policy by the window protocol may be nullified by its relatively high contention overhead.

Therefore, it is concluded that in designing a distributed scheduling algorithm, such as a communication protocol, one should seek a balance in achieving an optimal scheduling policy and minimizing the scheduling overhead.

1.2 Real-Time Communications

1.2.1 Definitions and Objectives

A real-time environment is distinguished from a non real-time system by the introduction of *time* as a key factor. The correctness of a real-time system is determined not only by the logical results of the computation, but also by the time at which the results are produced. A *distributed real-time* system, such as a token ring network, is a system where communications are conducted among processes located on geographically distributed nodes. A message in a real-time system is associated with certain timing constraint such as a *deadline* or a *laxity*. The deadline of a message is the time at which the message must be received by its destination. The laxity of a message is defined as the maximum time the message can wait before its transmission has to start in order to meet its deadline.

The most important aspect of real-time communications is that a message must be received by the destination before its deadline expires; otherwise, it is considered *lost.*

Typically, real-time applications generate mixed traffic, including packetized voice and video, network error, alarm and sensor messages and data transactions. They can be grouped into two categories: *synchronous* (periodic) and *asynchronous* (aperiodic). Synchronous messages, such as voice and other polling messages, arrive at the system periodically. Their lengths and periods are known *a priori.* In many cases, they must be transmitted before the next arrival from the same message stream. The transmission of such real-time synchronous messages can be scheduled by a static scheduling algorithm. Real-time asynchronous messages, on the other hand, arrive randomly during run-time and are associated with different deadlines. It is generally perceived that synchronous messages are urgent and require bounded response times while asynchronous messages can tolerate longer delays. However, real-time asynchronous messages, such as network error, alarm and management message, are equally urgent and require fast delivery. Furthermore, real-time asynchronous message transmission requires dynamic scheduling in order to share the transmission resource efficiently. Thus, the ability to handle real-time asynchronous messages well indicates the good responsiveness of a system, which will be essential in the next generation of highly distributed and dynamic real-time systems.

Hence, a protocol intended for real-time communications, should incorporate explicit timing constraints of both synchronous and asynchronous messages. As a result, the principle performance considerations of real-time communications are fundamentally different from those of traditional data communications.

- The primary performance metrics for conventional data communications is *throughput* and *average delay*, which no longer adequately characterize the performance of a real-time communication protocol. The most important performance metric of a real-time communication protocol is the *loss ratio*, which is the percentage of messages missing their deadlines[1]. The chief design objective of real-time communication protocols is to minimize message loss.
- The performance trade-offs of traditional data communications is *offered load* versus *average delay*, while that of real-time communications is *offered load* versus *message loss* for given message timing constraints.

1.2.2 Selection of the Optimal Algorithm

A MAC protocol is essentially a set of rules which schedule message transmissions. The basic design requirements of a MAC protocol and a centralized task scheduling algorithm are similar: both are constrained by time to allocate a serially-used resource. The most commonly used centralized scheduling algorithms are: *First Come First Serve* (FCFS), *Shortest Task First* (STF), *Fixed Priority Scheme* (FPS), *Minimum Laxity First* (MLF) and *Earliest Deadline First*[2] (EDF). The FCFS policy schedules the tasks according to the arrival order of the requests. The FPS algorithm gives the resource to the task that belongs to the highest priority class. The STF policy always selects the task with the shortest length first. Only MLF and EDF policies schedule tasks according to task timing constraints. That is, they choose the task with the minimum laxity and earliest deadline respectively.

[1] Alternatively, the fraction of messages successfully meeting their deadlines is called the *sent ratio*.
[2] If message lengths are constant, then the MLF and EDF algorithms are essentially the same.

Hence, we have a spectrum of centralized scheduling algorithms that schedule message transmission using different amount (from none to perfect) of message timing information. It has been established from centralized real-time scheduling theory that the MLF and EDF algorithms are optimal[3] in both static and dynamic cases [16, 29]. Furthermore, it is known that even a policy that only approximates the MLF or EDF policies can still result in a significant reduction in task loss [13, 53].

Therefore, it is natural to believe that a protocol that implements the MLF or EDF transmission policy is the most desirable for real-time communications. However, the optimality of MLF or EDF policy is achieved in a centralized scheduling environment where the scheduling overhead is negligible.

Implementing the MLF or EDF policy requires the global knowledge of message timing constraints. In a centralized scheduling environment, all message timing information is known to the scheduler at the time of decision making, while in a distributed environment, such as a token ring network, message timing information is only local to individual nodes. As a result, implementing the MLF or EDF policy in token ring networks is considerably more difficult and requires the protocol to include a special mechanism for collecting the deadline information either explicitly or implicitly.

Furthermore, in a distributed communication network, a MAC protocol implementing the MLF or EDF policy will take some time to gather message deadline information in the network before a message can be scheduled for transmission. The *scheduling overhead*, i.e. the time incurred in collecting the message timing information, is no longer negligible and may be high. Thus, we would expect the performance of distributed scheduling algorithms to depend on not only the scheduling policies they use, but also the scheduling overhead they invoke. That is,

[3]In the sense that they minimize the task loss ratio for a given task set

the performance of a real-time MAC protocol is a trade-off between the scheduling policy it employs and the overhead it incurs. Therefore, it is conceivable that a protocol which incurs a large overhead to implement an 'optimal' (MLF or EDF) scheduling algorithm may not necessarily produce better performance than another protocol that employs a simple, but 'non-optimal' policy. Hence, in designing a distributed scheduling algorithm such as a communication protocol, issues in both achieving 'optimal' (MLF or EDF) scheduling policy and minimizing the scheduling overhead must be addressed.

1.3 Related Work

In this section, we examine the recent developments in real-time communications and discuss their contributions and limitations. The majority of existing protocols for real-time communications can be broadly divided into two areas: CSMA/CD based and token ring based.

1.3.1 Related CSMA/CD Based Work

Over the last few years, various enhanced CSMA/CD protocols have been designed and studied in the context of real-time communications. These protocols intend to implement the optimal transmission policy, such as MLF or EDF, and are useful contribution to the state of the art of real-time communications.

One class of protocols implementing MLF or EDF transmission policy are called *window protocols*. These protocols are the enchancements to the traditional window protocols [43], as they use message laxity or deadline as the window axis. The window protocol proposed by Kurose *et al* [22] implements the MLF policy, but with the assumption that laxities of all messages are constant. Under this assumption, the MLF policy is equivalent to the FCFS policy. Zhao, Stankovic and Ramamritham extended this window protocol by relaxing the above assumption to allow message laxities to have arbitrary distribution. Also, In their model, a newly arrived message is allowed to compete for transmission. However, the MLF policy is not always preserved in the presence of a laxity tie, as the protocol uses a probablistic approach to solve a tie. Furthermore, the protocol uses a constant window in the initial phase to locate the contention window, which may result in slow convergence to the desired minimum laxity message.

Znati [60] designed a window protocol which is intended to improve the performance of the window protocol proposed by Zhao *et al* in three ways. First, a binary search based approach is used during the initial phase to determine a contention window, rather than a constant window. Hence, it results in faster convergence in locating the minimum laxity message. Second the protocol has an additional reservation mechanism that prohibits messages with large laxity from competing with messages that have smaller laxities. This mechanism is useful in reducing the number of deceitful contentions among messages with large disprepancy in their timing constraints. Third, the protocol implements the MLF policy even in the presence of a laxity tie. This protocol has been shown to have a considerable improvement in its performance over other proposed window protocols, but the protocol may still be inefficient under situations where message laxities are very close. This is due to the nature of CSMA/CD networks where nodes are not able to obtain sufficient timing information about messages residing on other nodes. As a result, the binary search may need to partition the contention window many times in order to locate the minimum laxity message.

Therefore, many collisions may occur before a successful message transmission can take place.

Virtual time protocols are another family of protocols that implement the MLF or EDF transmission policy. Zhao and Ramamritham [55] studied the modified virtual time CSMA/CD protocols for real-time communications. In their model, each node maintains two clocks, a real clock and a virtual clock. Whenever a node finds the channel idle, it resets the virtual clock which then runs at a higher rate than the real clock. A node is allowed to transmit a message when the time on the virtual clock is equal to some parameter of the message. By selecting different parameters, such as arrival time, transmission time, laxity or deadline, the protocol implements transmission polices FCFS, STF, MLF and EDF respectively. Their simulation results have shown that virtual time CSMA/CD protocols yields improved performance over pure CSMA/CD and that protocols implementing MLF or EDF policy perform better than those implementing FCFS or STF policy in terms of message loss and channel utilization.

Another class of enhanced CSMA/CD protocols for real-time communications uses a *multi-version message transmission* scheme [19, 27, 57]. The basic idea of the enhancement is that the version of a message to be transmitted depends on the overall network load. When the traffic is heavy or/and the message deadline is tight, only a fraction of the message is sent instead of the full message in order to meet the message deadline. The key element is the *version selection scheme* implemented at each node, which determines which version of the message to be sent. The criteria used by the version selection scheme can be the length of the message queue on a node, the estimation of the time average network load or the message deadline density. It has been shown that the method is effective when used with the window protocol [27].

Finally, *guaranteed protocols* [4, 26, 37] aim to guarantee deadlines of real-time asynchronous messages in a dynamic fashion. It differs from static scheduling in the sense that whether a message deadline will be met depends on the network load and deadlines of existing messages, and is known only after the arrival of the message in the network. This enhancement is useful when some asynchronous messages require guaranteed delivery, but the cost is the reduction in channel utilization for real-time synchronous messages and real-time non-guaranteed asynchronous messages.

1.3.2 Related Token Bus/Ring Based Work

In token bus/ring network[4], a control frame called *token* circulates around the ring. A node wishing to transmit a message must capture the token and then send the message. After the completion of a message transmission, the sending node is responsible for releasing the token. Various enhancements, such as reservation scheme, prioritized access, cyclic control, may be incorporated into the basic token passing protocol.

Over the last decade, token bus and token ring networks have become the standard for real-time communication systems. They include Manufacturing Automation Protocol (MAP), IEEE 802.4 token bus, IEEE 802.5 token ring, Fiber Distributed Data Interface (FDDI) and FDDI-II. The increasing popularity of token bus/ring networks is mainly due to their performance characteristics, such as the bounded time delay, ability to provide guaranteed bandwidth, inherent fairness, and high utilization under overload.

MAP [45] has been largely accepted as the OSI solution to real-time manufacturing communications. Its two bottom layers are based on IEEE 802.4 timed token protocol

[4]Physically, a token bus is a linear or tree-shaped cable onto which stations are attached. Logically, stations are organized into a ring.

[2] which regulates message transmission using a set of timers. Synchronous messages, assigned the highest priority, are provided with guaranteed bandwidth. Asynchronous messages are transmitted only if the current token rotation time does not exceed the pre-defined target token rotation time.

The IEEE 802.5 token ring [1] uses a reservation scheme together with a prioritized access mechanism. It provides eight priorities to accommodate different message classes, with synchronous messages assigned higher priorities. In this protocol, the token has a *priority* field and a *reservation priority* field. Upon the arrival of the token, a node is allowed to transmit a message if the priority of the message is equal to or higher than the token priority; otherwise the node can reserve a future token by writing its message priority into the token reservation priority if the former is greater than the latter. This way, the message with the highest priority is always sent first.

A variation of the 802.5 standard is the IBM token ring, which uses a different prioritized access mechanism. In supporting synchronous traffic, one of the nodes is assigned as the *synchronous bandwidth manager* and raises the token priority periodically so that synchronous messages receive guaranteed bandwidth.

The Fiber Distributed Data Interface (FDDI) is the ANSI standard for an 100 Mbit/s fiber token ring. It uses a timed token rotation protocol which is an extension to the 802.4 standard. In a FDDI network, synchronous messages receive guaranteed bandwidth, while asynchronous messages are transmitted only when the current token rotation rate is faster than the pre-defined target token rotation time.

Although the timed token rotation protocol guarantees that the maximum token rotation time is at most twice the target token rotation time [32], it is found that the timed token rotation scheme alone is not sufficient to meet deadlines of urgent messages. To compensate this, ANSI defines the FDDI-II to be upward compatible with FDDI. It adds circuit switching capability to the basic FDDI to better support urgent synchronous messages. However, the performance in transmitting real-time asynchronous messages is adversely affected.

Clearly, for the standard token bus/ring protocols, it is a common practice that synchronous messages are given higher priorities than asynchronous messages. As a result, these protocols support real-time synchronous messages well. However, they may produce poor performance in meeting timing requirements of real-time asynchronous messages, as they do not explicitly incorporate timing constraints of real-time asynchronous messages.

While the standard token ring protocols were being developed, many non-standard token ring protocols have also been proposed over the last decade. They intend to support integrated real-time applications such as transmission of packetized voice, video and data. We now examine some of these protocols.

Goyal and Dias [11] proposed a reservation based prioritized access protocol, which is an variation of the 802.5 token ring protocol. It aims to increase the ring utilization of the 802.5 token ring by having early token release. Furthermore, it intends to improve and delay performance of time-critical packets by allowing high priority nodes (having time-critical packets) to send special packets to the token holder to reserve a future token or to pre-empt the token holder if it is transmitting a low priority packet. Performance evaluation shows that this enhanced protocol has a good responsiveness and great ability to support time-critical packets.

Huang and Chen [15] designed an integrated token ring protocol accommodating voice/data services. It uses a cyclic control mechanism with cycle length being the period of voice packetization time so that voice packets are delivered within the bounded delay. In their model, there are three access types indicated in the token, namely voice only, voice/expedited data and expedited/normal data. Real-time asynchronous messages are transmitted if the offered load of voice traffic is not heavy.

Tsai and Rubin [44] suggested a token scheme supporting delay-constrained priority messages with a synchronous cycle and a prioritized access arbitriation mechanism. In their model, at any time each node is allowed to have at most one real-time connection for synchronous messages, while transmissions of asynchronous messages are determined by a global contention resolution scheme through the use of token, which is similar to the one used in the 802.5 standard.

Kang *et al* [17] proposed a dual-channel hybrid token ring, which is also a modified 802.5 token ring. It consists of a ring channel and a direct channel. The ring channel is used for non real-time message transmission, while direct channels are inserted dynamically to support real-time and interactive communications between node pairs. Simulation results show that under light traffic load, the delay performance of this hybrid token ring is almost the same as that of the standard token ring, while under heavy load, the hybrid token ring has a much improved delay performance over the standard token ring.

Cherkassky *et al* [8] proposed a new LAN architecture INSTNET to support real-time traffic, which is also based on the 802.5 standard. The INSTNET incorporates a logically separated control channel in addition to the ring channel. Messages are transmitted over the ring channel according to a a reservation based priority contention resolution mechanism. Real-time traffic is allowed to pre-empt non real-time traffic by sending interrupts on the control channel. Simulation results show that the delay performance of real-time traffic is not affected by the load of non real-time traffic.

Although varying in details, these non-standard token ring protocols are similar to the standard token ring protocols in two ways. First, they do not consider individual message timing constraints explicitly. Furthermore, they give higher priorities to real-time synchronous messages at the cost of real-time asynchronous messages. As a result, the support for real-time asynchronous messages is poor.

However, noticeable progress has been made in designing token ring protocols that address the explicit timing constraints of synchronous messages. Strosnider, Lehoczky and Sha [23, 37] proposed a deferrable server algorithm to meet the deadlines of synchronous messages and also, to improve the response time for asynchronous messages. Strosnider and Marchok [38] applied this algorithm to token ring networks, which results in a highly responsive and deterministic service, and is an useful enhancement to the IEEE 802.5 token ring protocol. Shin and Hou [34] investigated the average performance of various token ring protocols in a real-time environment. Clearly, this pioneering work has contributed greatly to the state of the art of real-time communications in token ring networks. However, the problem of meeting timing requirements of real-time asynchronous messages in token ring networks, to our best knowledge, has not been fully addressed.

1.4 Desirable Properties of Protocols for Real-Time Communications

From the preceding discussions, we can conclude that a MAC protocol suitable for real-time communications must address the explicit timing constraints of individual messages rather than differentiate messages in terms of their classes. In specific, a desired real-time MAC protocol should

- incorporate message timing requirements, such as laxity or deadline,
- support real-time synchronous messages as well as real-time asynchronous messages, i.e. to address message timing requirements whether or not they are known *a priori*.
- schedule message transmission based on the optimal transmission policy and minimize overhead incurred.

These desired properties provide a guideline for design and performance evaluation of real-time MAC protocols.

1.5 Thesis Outline

We have defined the scope of real-time communications, identified its objectives and discussed advances and limitations of recent work in real-time communications.

The rest of the thesis is organized as follows. In Chapter 2, we first introduce the network and message models together with their parameters and attributes. We then define the protocol notations and performance metrics. Finally, we describe the methodology for analizing the worst case performance of a real-time MAC protocol.

Chapters 3, 4 and 5 study and analyze three different token ring protocols respectively. In each of the three chapters, we start with a detailed description of the protocol operation. We then derive the worst case performance ratio of the protocol. Finally, based on the worst case performance ratio, we discuss the protocol behavior under different values of network, protocol and message parameters.

Using the results obtained from Chapters 3, 4 and 5, Chapter 6 compares the worst case performance of the three protocols and discusses the underlying implication, which leads to one of the important conclusions of this work.

Chapter 7 evaluates and compares the average case performance of the three protocols through simulation. First, we introduce the simulation program and simulation language used. We then describe the traffic model on which the simulation experiments are conducted. Finally, we present, compare and discuss various simulation results.

In the final chapter, we summarize the results, significance and contribution of this work. We then propose the direction and agenda for future research in the field of real-time communications.

Chapter 2

System Models

In this chapter, we first introduce the network and message models together with their parameters and attributes. We then define the protocol notations and performance metrics. Finally, we describe the methodology for the worst case performance analysis of a real-time MAC protocol, which lays the foundation for chapters 3, 4 and 5.

2.1 Network Model

In the OSI reference model, the data link layer is subdivided into a Medium Access Control (MAC) sublayer and a Logic Link Control (LLC) sublayer [41]. A generic real-time communication layer model is given in Figure 2.1. The primary function of the MAC protocol, called the *global access contention resolution mechanism*, is to provide access control to the shared medium and to schedule the network-wide message transmission according to the transmission policy used.

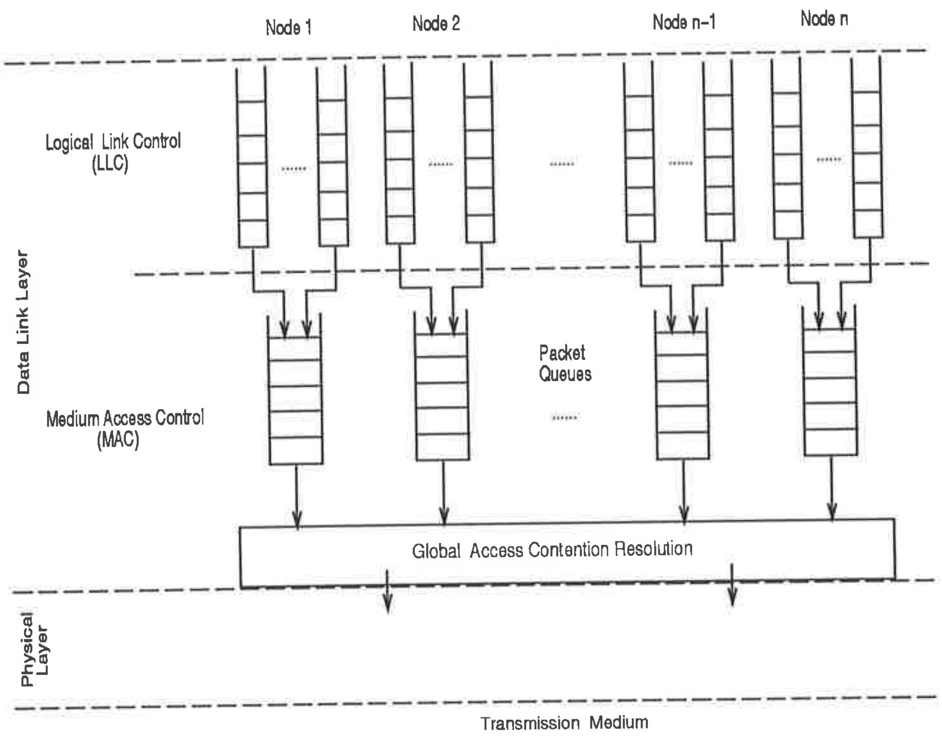

Node 1
Node 2
Node n-1
Node n
Logical Link Control (LLC)
Data Link Layer
Packet Queues
Medium Access Control (MAC)
Global Access Contention Resolution
Physical Layer
Transmission Medium

Figure 2.1. Generic Real-Time Communication Layer Model

Each node maintains a *packet queue* at the MAC layer. Messages from the LLC layer are segmented into fixed-length packets[1] before they are admitted to the corresponding MAC packet queues[2]. A packet queue can be an unsorted queue such as a FCFS queue where packets are kept in their arrival order or a prioritized queue where messages are kept according to a parameter such as access priority, laxity or deadline.

The focus of this work is primarily on the MAC protocol used for real-time communications in token ring networks, which is a dominant factor in determining the

[1]The deadlines of all packets from a message are the same as that of the message. Hence, a successful message transmission requires all component packets of the message be received before their deadlines expire; otherwise, the message is considered lost.

[2]Messages from different LLC connections at a node are buffered in the common packet queue at the MAC layer.

performance of any LANs supporting real-time applications. That is, we are mainly concerned with transmission of time-constrained messages in token ring networks[3].

In a token ring network, nodes are serially connected by a unidirectional point-to point transmission links which forms a ring. Nodes communicate in the network via the ring. Figure 2.2 shows a token ring network of n nodes.

Each station[4] is connected to the ring via a *ring interface* which has two operating modes: *listen* and *transmit.* In the listen mode, the ring interface repeats each input bit to the output. In the transmit mode, the interface breaks the connection between the input and output and transmit its own data onto the ring. Devices (e.g. terminals, workstations, printers) are attached to the ring through nodes to communicate with other devices in the network.

A special control frame called the *token* circulates around the ring to provide access opportunities for contending nodes to gain access. Once a node detects a token passing on the ring, it may capture the token by modifying certain fields in the token. It then transmits its message by appending the data and adding appropriate control, address and information fields. At any time, only the node possessing the token has the right to transmit.

A packet transmitted by a node onto the ring is passed unidirectionally from one node to the next. One or more nodes may be identified as destinations in the destination address field of the packet, and such destination nodes copy the packet as it passes by. The source node (the one that has transmitted the packet onto the ring) removes the transmitted packet from the ring when it returns. The node then issues a new token and resumes its listen mode.

[3]Issues pertaining to higher layer protocols such as routing, flow control and congestion control, retransmission, reliability and so forth, are the themes for our future research.

[4]In the following, station and node will be used interchangeablely.

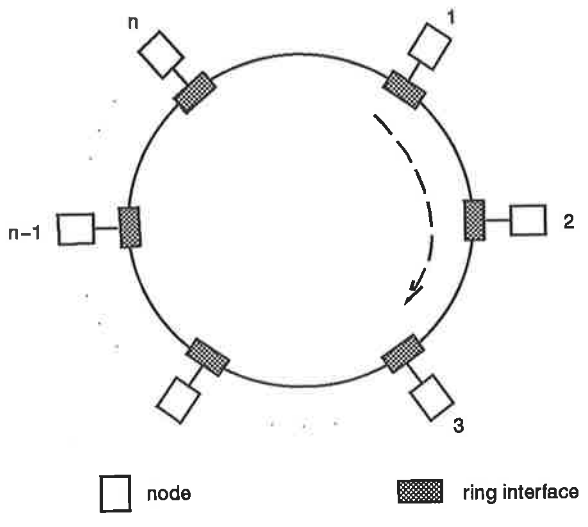


Figure 2.2. A Token Ring Network

2.2 Network Parameters

We now define some parameters to model the simple token ring network described in the last section. Let $N(n, w)$ represent a network of n nodes with a token node-to-node delay of w. We assume that the ring is of length l with a medium propagation speed of ρ. Hence, the medium propagation delay is $l * \rho$. Suppose that the network has a population of n nodes, evenly spaced on the ring, thus the node-to-node propagation delay is $(l * \rho)/n$. The *token node-to-node delay* w is the time for the token to travel between two adjacent nodes. It includes the medium propagation delay τ/n and the station bit delay θ, which is the delay introduced at a station to monitor, repeat or modify the token fields. Hence,

$$w = \frac{l * \rho}{n} + \frac{\theta}{c} \tag{2.1}$$

It follows that the *normalized token node-to-node delay* w' is

Notation	Parameter
n	number of nodes
c	ring speed
l	ring length
ρ	media propagation speed
θ	station bit delay
w	token node-to-node delay
W_t	token walk time
L_p	packet length
L_t	token length

Table 2.1. Network Parameters

$$w' = \frac{(l * \rho)/n + \theta/c}{(Lp + Lt)/c}, \tag{2.2}$$

where L_p and L_t is the packet and token length respectively. Finally, the *token walk time* W_t is the total time needed for the token to complete a full circulation around the ring when the ring idle. That is, W_t is the summation of the token node-to-node delay w over n links on the ring,

$$W_t = n * w. \tag{2.3}$$

Table 2.1 is a summary of the notations and interpretations of these parameters.

2.3 Message Model and Parameters

In this section, we introduce parameters and models that characterize a time-constrained message and message set respectively.

A time constrained message M is associated with the following attributes upon its arrival.

- *arrival time* (γ): the time when the message arrives.

- *length* (l_M): the time needed to complete the message transmission.
- *position* (p): the identification of the node at which the message arrives.
- *deadline* (d): the latest time when the message transmission must be completed.
- *destination* (ϱ): the identification of the node to which the message is destined.

Thus, a message can be characterized by these five parameters γ, l_M, p, d and ϱ. Without loss of generality, we make the following assumptions which are intended to reduce the complexity of the analytical analysis [34, 37, 39].

- We assume that all messages are waiting in the network when the protocol is invoked.
- We assume that each message consists of a single packet whose length including the overhead is normalized to one unit of time[5]. Hence, the normalized token node-to-node delay is the same as the token node-to-node delay, i.e. $w' = w$. We also assume that the token walk time is less than the message transmission time, i.e.

$$nw < 1. \tag{2.4}$$

- We assume that the network operates reliably such that there is no loss of tokens and messages. Messages are always received correctly, hence no retransmission is needed.

[5]For a time constrained message $M = (d, p)$, we require that the transmission of message M should start no later than time $d - 1$. This is because a message that cannot be received at its destination by its deadline is regarded as lost. If the transmission of the message cannot start by $d - 1$ (given the message length is one), it will not be transmitted but will be dropped.

- We assume that a sending node releases the token immediately after it completes a message transmission in order to improve the ring utilization. Hence, the time incurred for the transmitted message to reach the destination is not part of the protocol overhead. Therefore, the destination node is of no significance in determining the protocol performance.

With the above assumptions, parameters γ, l_M and ϱ can be omitted. As a result, we can now denote a message M by a simplified notation (d, p), where d is its deadline and p $(1 \leq p \leq n)$ is the identification of the node where the message is waiting to be sent. This implies that the number of messages sent depends on the distribution of deadlines and positions of all messages in the network, which is the essential information needed to implement a network-wide EDF policy.

A message set $A(\eta)$ is defined as a collection of individual messages denoted by parameters d and p,

$$A(\eta) = \{(d_1, p_1), (d_2, p_2), \cdots, (d_\eta, p_\eta)\}, \tag{2.5}$$

where the number of messages η is defined as the *size of message set* $A(\eta)$. Without loss of generality, we assume that messages in a set have been sorted according to their deadlines, i.e. $d_i \leq d_j$ if $i < j$.

As the network offered load increases, it will reach a point where a protocol can no longer control the message transmission in some predefined manner and exhibits poor performance and instability. Thus, it is important to study the protocol behavior under these situations. Hence, in this work we assume that the network is fully loaded with message(s) to be sent at every node, i.e. $\eta \geq n$.

On the other hand, given η ($\eta \geq n$) messages residing on different nodes, we define the *eligible message set at time t* to be the one containing those messages being considered for transmission at time t. As at any time only one message from each node (the head of the message queue) is qualified to compete for transmission, thus only a maximum of n messages are considered for transmission at any time[6]. This implies that the maximum size of an eligible set is $\eta = n$. How to schedule the transmission of n messages from an eligible set is the responsibility of the MAC protocol, which will be the focus of this study.

Relaxations of these assumptions are possible. For example, in the case where the network is not fully loaded (not all the nodes have pending messages), our analysis would still be valid if w represents the token node-to-node delay between two nearest *active* nodes where there are pending messages. In fact, from time to time, we will extend our analysis results to the case when the network is not fully loaded.

2.4 Protocol Notations and Performance Metrics

A MAC *protocol* is a set of rules used to schedule message transmissions, i.e. to determine when a message should be sent. Given a set of time constrained messages, we wish to determine how many messages can be transmitted before their deadlines expire. That is, we consider the fraction of messages successfully sent by the protocol as the most important performance metric. The goal of designing a protocol for transmitting real-time messages is to maximize the number of messages sent or alternatively to minimize the number of messages lost.

[6]Subsequent eligible sets can be obtained by adding a new message to the eligible set after removing the transmitted message from the set.

Let *Send(P, N(n,w), A(n),* T_s, ζ*)* denote the number of messages sent by a protocol P for a given message set $A(n)$ in a network $N(n,w)$. T_s and ζ denote the initial conditions of the network and protocol. T_s defines the time when the protocol is invoked and ζ denotes the node where the token is at time T_s. Unless otherwise specified, we assume that the protocol starts to operate at time 0 and the token is being released from node n, moving towards node 1, then node 2 and so on. In this case, we can omit T_s and ζ from the argument list of *Send*.

As discussed in Chapter 1, for a given distribution of message deadlines and positions, we expect that the number of messages sent by a protocol is determined not only by the transmission policy employed but also by the overhead incurred. In this study, we will compare the performance of protocols with an ideal protocol implementing the optimal scheduling policy without overhead. We call this ideal protocol the *Centralized Earliest Deadline First Protocol* (CEDF). Under this protocol, transmissions of all messages are assumed to be scheduled by a centralized controller which possesses perfect knowledge about message deadlines without experiencing any time delay. The controller schedules message transmissions in the EDF order. Clearly, this protocol is ideal: it implements an optimal scheduling policy with no overhead. We use it to provide an upper bound on performance of other protocols.

We introduce a metric called *performance ratio* for the protocol performance evaluation.

Definition 2.4.1 *For a token ring of* n *nodes with a token node-to-node delay of* w, *a protocol* P *is used to transmit* n *messages from a message set* $A(n)$, *the performance ratio of protocol* P, *denoted by* $r(P, N(n,w), A(n))$, *is defined by*

$$r(P, N(n,w), A(n)) = \frac{Send(P, N(n,w), A(n))}{Send(CEDF, N(n,w), A(n))}. \quad (2.6)$$

This performance ratio measures the percentage of messages sent by a protocol against that by the ideal CEDF protocol.

Clearly, given n, message sets of sizc n can have different distributions of deadlines and positions. As a result, they may have different performance ratios. We define the lowest value of the performance ratios of all message sets of size n, i.e. the lower bound of the performance ratio, as the *worst case performance ratio.*

Definition 2.4.2 *For a token ring of n nodes with a token node-to-node delay of w, if a protocol P is used to transmit n messages, the worst case performance ratio of protocol P, denoted by $R(P, w, n)$, is defined by*

$$\begin{aligned} R(P, w, n) &= \min_{\forall A(n)} (r(P, N(n,w), A(n))) \\ &= \min_{\forall A(n)} (\frac{Send(P, N(n,w), A(n))}{Send(CEDF, w, A(n))}), \end{aligned} \quad (2.7)$$

where $A(n)$ is any message set of size n.

In general, if message size is h ($h < n$) and the protocol starts at time T_s ($T_s > 0$), then the worst case performance ratio can be written as $R(P, w, n, h, T_s)$.

We define that the *worst case performance* of a protocol occurs when the worst case performance ratio is reached. The message set which results in the worst case performance ratio is defined as the *worst case message set.*

Definition 2.4.3 *For a given protocol P and the number of messages n, the worst case message set is $A_{wc}(n)$, if the following is satisfied.*

$$r(P, N(n,w), A_{wc}(n)) = R(P, w, n). \tag{2.8}$$

We also define that *the average case* performance of a protocol occurs when the performance ratio does not reach the minimum. Although the average case performance indicates the protocol performance under normal situations, the worst case performance demonstrates how poor the performance of a protocol can be under the most unfavorable operating conditions. To provide predictable and reliable performance in a real-time system, it is imperative to investigate the worst case performance of a protocol.

Before moving on to the next section, we introduce a requirement on message deadlines, which is observed by all time-constrained message sets under consideration.

Definition 2.4.4 *A message set $A(n)$ is feasible if the ideal protocol CEDF can send all the messages in $A(n)$. That is,*

$$Send(CEDF, N(n,w), A(n)) = n. \tag{2.9}$$

If a message set is feasible, then the following can be established.

Lemma 2.4.1 *A sufficient and necessary condition for a message set $A(n)$ to be feasible is that the deadline of each message satisfies the following.*

$$d_i \;\geq\; i, \text{ where } 1 \leq i \leq n. \tag{2.10}$$

That is, the deadline of the i-th message will not expire until the completion of its transmission[7].

[7] Although the lemma is intuitively true, we still provide a formal proof for the sake of completeness.

Proof: We prove the sufficient and necessary condition separately.

- To prove $d_i \geq i$ is a sufficient condition.

  That is, if $d_i \geq i$, then message set is feasible. We use the induction method. Let $i = 1$, we wish to prove that $d_1 \geq 1$ implies that the message set is feasible. This is straightforward as the transmission of the message is completed before its deadline expires. Thus, the message is successfully sent. Therefore, the message set is feasible.

  Suppose the lemma holds for $i = k - 1$, That is, $d_{k-1} \geq k - 1$ implies that the message set is feasible. We wish to show that given $d_k \geq k$, the message set is feasible. Clearly, the first $k - 1$ messages are sent and their transmissions complete at time $k - 1$. As messages are sorted in the order of their deadlines and the CEDF protocol always sends the message with the earliest deadline first, the k-th message sent by the CEDF protocol is the one with the deadline of d_k. As $d_k \geq k$, from time $k-1$ there is enough time to transmit the message. Hence, all k messages are successfully sent. That is, the message set is feasible.

- To prove $d_i \geq i$ is a necessary condition.

  That is, if the message set is feasible, then $d_i \geq i$. Similarly, we use the induction method. Let $i = 1$, if the message is sent, then its deadline must satisfy $d_1 \geq 1$.

  Suppose the lemma holds for the case of $i = k - 1$. We wish to prove that it is also true for the case $i = k$. As messages are sorted in the order of their deadlines and EDF policy is used, the k-th message sent by the CEDF protocol is the one with the deadline of d_k. As the message set having k messages is feasible, all the k messages are transmitted. The total time needed to transmit k messages is k, since each message transmission requires one unit time. As the k-th message is transmitted, its deadline must not expire before the end of its transmission, hence, we have $d_k \geq k$.

This concludes the proof. Q.E.D.

Definition 2.4.5 *We define $A_{SFS}(n)$ as the smallest feasible set for message size n, if $A'(n)$ is infeasible, where the deadline of messages in $A'(n)$ are the same as those in $A_{SFS}(n)$ except*

$$d'_i < d_i - 1, \quad 1 \leq i \leq n, \tag{2.11}$$

where d_i and d'_i are deadlines of messages in $A_{SFS}(n)$ and $A'(n)$ respectively.

Obviously, for message sets of size n the smallest feasible set is

$$A_{SFS} = \{1, 2, 3, \cdots, n-1, n\}. \tag{2.12}$$

If a message set is infeasible, then even the CEDF protocol is not able to send all of the messages in the message set, consequently, no protocol can send all the messages. In the extreme case, we may have *Send(CEDF, N(n,w), A(n))* = *Send(P, N(n,w), A(n))* $= 0$, resulting in the performance ratio being an indefinite form, i.e.

$$r(P, w, n) = \frac{0}{0}. \tag{2.13}$$

This makes the calculation of the performance ratio of the protocol meaningless. The *infeasibility* of a message set reflects the correlation of message deadlines. Thus, failure in meeting message deadlines due to this correlation should not be regarded as an index to the protocol performance. Therefore, we are only concerned with the performance ratio of feasible message sets.

Definition 2.4.6 *Given that message sets are feasible, (2.6) and (2.7) become*

$$r(P, N(n,w), A(n)) = \frac{Send(P, N(n,w), A(n))}{n} \tag{2.14}$$

and

$$\begin{aligned} R(P,w,n) &= \min_{\forall A(n)}(r(P, N(n,w), A(n))) \\ &= \min_{\forall A(n)}(\frac{Send(P, N(n,w), A(n))}{n}). \end{aligned} \tag{2.15}$$

In general, an infeasible message set can be transformed into a feasible one before calculating the relevant performance metric of the protocol. The basic idea of the transformation is as follows. Let I be a subset of an infeasible set $A(n)$ such that $A(n) - I$ is feasible, then $A(n) - I$ is the required transform of $A(n)$.

In specific, given an infeasible message set where message deadlines are

$$\{d_1, d_2, \cdots d_n\}, \text{ where } d_i \le d_j,\ i < j, \tag{2.16}$$

we can transform it to a feasible set by removing messages whose deadlines do not satisfy $d_i \ge i$. That is,

$$d'_i = \min(d_k), \quad d_k \ge i \text{ and } 1 \le k \le n, \tag{2.17}$$

where d'_i denotes the deadline of the i-th message in the feasible set after the transformation.

Finally, a protocol is *work conserving* if it does not remain idle while there is a message waiting on some node. A non-work conserving protocol may drop all the messages and never send any. Its (worst case) performance ratio would then simply be zero. Thus, only the work conserving protocols are of interest in terms of the (worst case) performance ratio. Therefore, in whatever follows all protocols under study are work conserving.

2.5 Methodology for Worst Case Performance Analysis

In this section, we discuss the methodology and identify some common protocol properties which reveal the quantitative relationship between the performance ratio and the distribution of message deadlines and positions. These properties, presented in the form of lemmas or theorems, form a foundation for the worst case performance analysis of the three protocols, which will be carried out in the next three chapters.

Recall that for a token ring of n nodes, the maximum size of an eligible message set is n, which implies that we only need to be concerned with how the protocol transmits the n messages. That is, to simplify the analysis, we take a snap shot of the system.

Given a message set of size n, the worst case performance ratio of a protocol is the lowest performance ratio achieved by the protocol for all message sets of size n. Hence, the most straightforward way to obtain the worst case performance ratio is to compute the performance ratio for each of those message sets. For instance, suppose the values of message deadlines and positions are in the range of [1, D] and [1, n] respectively, the number of different message sets of size n is $D \times n$. Given a network with a population of n in the order of hundreds and relatively large message deadlines, the number of different message sets can be enormous. Clearly, this approach is formidable, if not impossible.

By definition, the worst case message set yields the worst case performance ratio, hence an alternative is to identify the worst case message set and then to compute its performance ratio. This means that we need to establish an ordering among message sets in terms of the performance ratio. To avoid exhaustive numerical computing as in the first approach, message sets will not be sorted directly by computing and comparing their performance ratios. As the performance ratio is determined by the distribution of message deadlines and positions, it is possible to identify the quantitative relationship (in terms of the performance ratio) between two message sets by comparing their deadline and position distributions, so that message sets can be sorted out by their deadlines and positions respectively. Consequently, the message set having the lowest index of both orderings is the worst case message set. This approach is adopted in this study.

We now formally define the partial ordering among message sets.

Definition 2.5.1 *Let two message sets be*

$$\begin{aligned} A(n) &= \{(d_1, p_1), (d_2, p_2), \cdots, (d_n, p_n)\} \\ A'(n) &= \{(d'_1, p'_1), (d'_2, p'_2), \cdots, (d'_n, p'_n)\}, \end{aligned} \tag{2.18}$$

where $p_i, p_j \in [1, n], 1 \leq i, j \leq n$. *We define deadline partial ordering as*

$$A'(n) \leq_D A(n) \tag{2.19}$$

if for any $(d_i, p_i) \in A(n)$ *and* $(d'_j, p'_j) \in A'(n)$, $p'_j = p_i$ *implies* $d'_j \leq d_i$ $(1 \leq i, j \leq n)$.

The symbol $\leq_D$ is read as *less than or equal to in terms of deadline.* For example, if there are two message sets,

$$\begin{aligned} A(5) &= \{(2,4),(3,2),(6,5),(4,1),(7,3)\}, \\ A'(5) &= \{(1,4),(3,2),(3,5),(4,1),(7,3)\}, \end{aligned} \tag{2.20}$$

we have $A'(5) \leq_D A(5)$.

With the above definition, we can define the sensitivity of a protocol to the deadline distribution of a message set.

Definition 2.5.2 *A protocol P is deadline monotonic, if*

$$Send(P, N(n,w), A'(n)) \leq Send(P, N(n,w), A(n)), \tag{2.21}$$

whenever $A'(n) \leq_D A(n)$.

This implies that if a protocol is deadline monotonic, then the message set having the smallest possible deadlines, i.e. the smallest feasible set, has the minimum number of messages sent.

Lemma 2.5.1 *If a protocol is deadline monotonic and used to transmit n messages, then the smallest feasible set $A_{SFS}(n)$ is the worst case message set.*

Proof: First, for n messages we have

$$A_{SFS}(n) \leq_D A(n). \tag{2.22}$$

As the protocol is deadline monotonic, then with Definition (2.5.2), we have

$$\mathrm{Send}(\mathrm{P}, N(n,w), A_{SFS}(n)) \leq \mathrm{Send}(\mathrm{P}, N(n,w), A(n)), \tag{2.23}$$

where $A(n)$ is any other message set of size n. This implies that in the worst case the smallest feasible set $A_{SFS}(n)$ minimizes the number of messages sent. It follows that it is the worst case set. Q.E.D.

We now define some parameters related to message positions in a network.

Definition 2.5.3 *The distance between nodes i and j $(i \neq j)$ in a network is defined as*

$$g_{i,j} = \begin{cases} j - i & j > i, \\ n + (j - i) & \textit{otherwise.} \end{cases} \tag{2.24}$$

That is, $g_{i,j}$ is the number of links between the two nodes.

Lemma 2.5.2 *The maximum and minimum value of $g_{i,j} (i \neq j)$ is $n - 1$ and 1 respectively.*

Proof: We have two cases to consider.

- $j > i$.

  With (2.24), we have $g_{i,j} = j - i$. It follows that

$$\begin{aligned} \forall_{i,j} \max(j - i) &= n - 1, \\ \forall_{i,j} \min(j - i) &= 1. \end{aligned} \tag{2.25}$$

- $j < i$.

  In this case we have $g_{i,j} = n + (j - i)$, which implies the following.

$$\begin{aligned}
\forall_{i,j} \max(n + (j - i)) &= n - \min(i - j)) \\
&= n - 1, \\
\forall_{i,j} \min(n + (j - i)) &= n - \max(i - j) \\
&= n - (n - 1) \\
&= 1. \qquad (2.26)
\end{aligned}$$

This conclues the proof. Q.E.D.

In a token ring network, the number of links that the token travels before a message transmission starts is translated into a non-zero overhead, which is the time needed for the token to reach the next sending node. We define this overhead as the *contention overhead*.

Definition 2.5.4 *Suppose $M_1, M_2, \cdots, M_{i-1}, M_i, \cdots M_k$ are k messages sent from message set $A(n)$, the contention overhead of transmission of M_i $(1 \leq i \leq n)$, denoted by c_i, is defined as the time elapsed from the instant when the token leaves the node having message M_{i-1} to the instant when the transmission of message M_i starts.*

The contention overhead incurred by each message transmission may vary from one to another depending on the protocol used and message deadline and position distribution in the network. Clearly, the contention overhead of a message transmission can be expressed as the product of the token node-to-node delay w and the total distance the token has traveled from the previous sending node to the current sending node. As we will see, the contention overhead is one of the factors determining the worst case performance ratio.

Chapter 3

The Token Passing Protocol

In this chapter, we study a simple *token passing* (TP) protocol which transmits messages in the nearest-neighbor-first order [24]. The major disadvantage of this protocol is that it does not consider individual message deadlines. As a result, its performance on transmitting time-constrained messages in the worst case may not be desirable.

We first describe the protocol operation and establish some important properties of the protocol. Using these properties, we then identify the message deadlines and positions in the worst case message set to derive the worst case performance ratio. Finally, we use numerical examples to demonstrate the protocol behaviors under different values of network and message parameters. The other two protocols will be presented and analyzed in Chapters 4 and 5 in a similar manner.

3.1 Protocol Description

In the token passing protocol, each node maintains a message queue where messages are kept in their arrival order, irrespective of their deadlines. Messages that have missed deadlines will not be sent but removed from the queue.

A free token is circulating around the ring. If a node wishes to send a message, it captures the token and then sends the first message from the queue. Upon the completion of the message transmission, the node releases the token to the node downstream, and the protocol continues this way.

3.2 Protocol Properties

The operation of the protocol indicates that the token passing protocol schedules the global message transmission in the nearest-neighbor-first order. No mechanism is included to take care of message deadlines. This implies that whether and when a message is sent is determined by the its position and its arrival time regardless of its deadline.

Before proceeding to the worst case analysis, we first outline some properties of this token passing protocol.

Lemma 3.2.1 *The token passing protocol is deadline monotonic.*

Proof: First, we wish to show that for a given message set $A(n)$, if there exists a message set $A'(n)$, such that $A'(n) \leq_D A(n)$, then the number of messages sent from $A'(n)$ is no more than that from $A(n)$.

In specific, let message sets $A(n)$ and $A'(n)$ be

$$\begin{aligned} A(n) &= \{(d_1,p_1),(d_2,p_2),\cdots,(d_i,p_i),\cdots,(d_n,p_n)\}, \\ A'(n) &= \{(d_1,p_1),(d_2,p_2),\cdots,(d'_i,p_i),\cdots,(d_n,p_n)\}, \end{aligned} \tag{3.27}$$

where $d'_i < d_i$ and $d'_j = d_j$, $1 \le i,j \le n$, $j \ne i$. Thus $A'(n) \le_D A(n)$. We wish to show

$$\text{Send}(\text{TP}, N(n,w), A'(n)) \le \text{Send}(\text{TP}, N(n,w), A(n)). \tag{3.28}$$

Let t_i denote the time when the node having message (d_i,p_i) receives the token. We have two cases to consider.

- Case 1: message (d_i,p_i) is lost in $A(n)$

  This implies that from t_i there is not enough time to send message (d_i,i). That is, $d_i < t_i+1$, where 1 is the message transmission time. As $d'_i < d_i$, consequently,

  $$d'_i < d_i < t_i + 1, \tag{3.29}$$

  indicating that d' will expire earlier than d_i. Hence, (d'_i,i) will be lost. The remaining message transmission sequence stays the same. Thus,

  $$\text{Send}(TP, N(n,w), A'(n)) = \text{Send}(TP, N(n,w), A(n)). \tag{3.30}$$

- Case 2: message (d_i,p_i) is sent in $A(n)$

  As the node having (d_i,i) receives the token at t_i, the transmission of the message can start at t_i. Depending on the values of d'_i and t_i, we have the following.

  - $d'_i \ge t_i + 1$.

    This implies that from time t_i onwards, there is a time interval of at least one unit of time before d'_i expires. Hence, it follows that (d'_i, i) will be sent too. The remaining message transmission sequence remains the same. Hence,

$$\text{Send}(TP, N(n,w), A'(n)) = \text{Send}(TP, N(n,w), A(n)). \qquad (3.31)$$

- $d'_i < t_i + 1$.

  This means there is insufficient time to send (d'_i, p_i) from t_i, thus, (d'_i, p_i) will be lost. If there is no message loss in A(n) after (d_i, p_i), then the remaining message transmission sequence remains the same. Thus,

$$\text{Send}(TP, N(n,w), A'(n)) = \text{Send}(TP, N(n,w), A(n)) - 1. \qquad (3.32)$$

  Otherwise, assume (d_j, p_j) is the first message lost in A(n) after (d_i, p_i), whose deadline satisfies the following:

$$t_j \le d_j < t_j + 1. \qquad (3.33)$$

  Let t_j and t'_j be the token arrival times on node p_j when transmitting messages from $A(n)$ and $A'(n)$ respectively. Clearly,

$$t_j = t_i + 1 + (j - i)w. \qquad (3.34)$$

  As (d'_i, i) is not sent, thus $t'_j = t_j - 1$. That is,

$$t_j = t'_j + 1. \qquad (3.35)$$

With (3.33) and (3.35), we arrive at

$$d_j \geq t_j = t'_j + 1. \tag{3.36}$$

Thus, (d_j, p_j) is sent. Those messages on nodes between node p_i and p_j are sent one time unit earlier and the transmission sequence after node p_j remains the same. Therefore,

$$\text{Send}(TP, N(n,w), A'(n)) = \text{Send}(TP, N(n,w), A(n)). \tag{3.37}$$

If there does not exist a lost message whose deadline satisfies (3.33), then those messages lost after t_i will still be lost. Transmission of each remaining message can be started one time unit earlier. In this case,

$$\text{Send}(TP, N(n,w), A'(n)) = \text{Send}(TP, N(n,w), A(n)) - 1. \tag{3.38}$$

With (3.30), (3.31), (3.32), (3.37) and (3.38), we have

$$\text{Send}(TP, N(n,w), A'(n)) \leq \text{Send}(TP, N(n,w), A(n)). \tag{3.39}$$

In general, for any message set $A'(n)$ such that $A'(n) \leq_D A(n)$, $A'(n)$ can always be obtained by decreasing the deadlines of some messages in $A(n)$. Assume that only one message deadline is decreased each time, using a similar argument as above, we can ensure that when a message deadline is decreased, the number of messages sent is either reduced or the same. Hence, using the deduction method, we have

$$\text{Send}(TP, N(n,w), A'(n)) \;\leq\; \text{Send}(TP, N(n,w), A(n)), \tag{3.40}$$

whenever $A'(n) \leq_D A(n)$. This implies that the token passing protocol is deadline monotonic according to Definition 2.5.2. Q.E.D.

Another important property of the protocol is related to message positions in the network.

Definition 3.2.1 *For a given message set $A(n)$, let $A^*(n)$ be another form of $A(n)$ where messages are sorted according to their positions. That is,*

$$A^*(n) = \{(d_1, p_1), (d_2, p_2), \cdots, (d_n, p_n)\}, \tag{3.41}$$

where $p_i < p_j$, if $i < j$. As we assume that each node has exactly one message, thus

$$A^*(n) = \{(d_1, 1), (d_2, 2), \cdots, (d_n, n)\}. \tag{3.42}$$

With this definition, the following lemma can be established.

Lemma 3.2.2 *Let two message sets be*

$$\begin{aligned} A^*(n) &= \{(d_1,1),(d_2,2),\cdots,(d_i,i),(d_{i+1},i+1),\cdots,(d_n,n)\}, \\ A'^*(n) &= \{(d'_1,1),(d'_2,2),\cdots,(d'_i,i),(d'_{i+1},i+1),\cdots,(d'_n,n)\}, \end{aligned} \tag{3.43}$$

where $d'_j = d_j$ for $1 \leq j \leq n$ and $j \neq i$, $d'_i = d_{i+1}$, $d'_{i+1} = d_i$ and $d_i > d_{i+1}$, $1 \leq i \leq n$. If the token passing protocol is used to transmit these two sets of messages, then

$$\textit{Send}(TP, N(n,w), A^*(n)) \leq \textit{Send}(TP, N(n,w), A'^*(n)). \tag{3.44}$$

That is, for a given message set $A^*(n)$, if the deadlines of two messages on the neighboring nodes are interchanged so that the token will visit the node with the smaller deadline message first, then the number of messages sent by the token passing protocol may increase.

Proof: Assume that the token arrives at node i and node $i+1$ at time t_i and t_{i+1} respectively when transmitting messages from $A^*(n)$. Similarly, let their counterparts be t'_i and t'_{i+1} when transmitting messages from $A'^*(n)$. It is clear that

$$t'_i = t_i < t_{i+1}. \tag{3.45}$$

That is, the token arrival times at node i are identical when transmitting messages from $A^*(n)$ and $A'^*(n)$. However, the times when the token arrives at node $i+1$ may differ, depending on whether or not the message on node i is sent after swapping.

As messages (d_i, i) and $(d_{i+1}, i+1)$ from $A^*(n)$ can either be sent or lost, we have four cases to consider.

- Case 1: both messages (d_i, i) and $(d_{i+1}, i+1)$ are sent

  We wish to show that from $A'^*(n)$, (d'_i, i) and $(d'_{i+1}, i+1)$ are also sent. As message transmission sequences for both sets before t_i are the same. We have

$$t'_i = t_i < t_{i+1}. \tag{3.46}$$

  As $(d_{i+1}, i+1)$ in $A^*(n)$ is sent, thus,

$$d_{i+1} \geq t_{i+1} + 1. \tag{3.47}$$

Combining (3.46) and (3.47), we arrive at

$$d_{i+1} \geq t_{i+1} + 1 > t'_i + 1. \tag{3.48}$$

Because $d'_i = d_{i+1} > t'_i + 1$, thus, (d'_i, i) is sent. As node $i + 1$ is the nearest neighbor of node i and (d'_i, i) is sent, thus $t'_{i+1} = t_{i+1}$. Consequently,

$$d_i > d_{i+1} \geq t_{i+1} + 1 = t'_{i+1} + 1. \tag{3.49}$$

It follows that

$$d'_{i+1} = d_i > t'_{i+1} + 1, \tag{3.50}$$

indicating that $(d'_{i+1}, i + 1)$ is also sent. The remaining messages are the same in both sets, thus

$$\mathrm{Send}(TP, N(n,w), A^*(n)) \;=\; \mathrm{Send}(TP, N(n,w), A'^*(n)). \tag{3.51}$$

- Case 2: both (d_i, i) and $(d_{i+1}, i + 1)$ are lost

  We wish to show that (d'_i, i) and $(d'_{i+1}, i + 1)$ from $A'^*(n)$ are also lost. First, as messages positioned before node i are the same in both sets, thus $t'_i = t_i$ holds. Second, as (d_i, i) is lost, we have

$$d_i < t_i + 1. \tag{3.52}$$

As $d_i > d_{i+1}$, from (3.45) and (3.52), we have

$$d_{i+1} < d_i < t_i + 1 = t'_i + 1. \tag{3.53}$$

Thus, it follows that

$$d'_i = d_{i+1} < t'_i + 1, \tag{3.54}$$

which implies that (d'_i, i) is lost. As node $i+1$ is the nearest neighbor of node i and in both sets the messages on node i are lost, thus, the time when the token reaches node $i+1$ is the same, that is $t'_{i+1} = t_{i+1}$. Consequently,

$$d'_{i+1} = d_i < t_i + 1 < t_{i+1} + 1 = t'_{i+1} + 1. \tag{3.55}$$

This implies that $(d'_{i+1}, i+1)$ is also lost. Hence,

$$\text{Send}(TP, N(n,w), A^*(n)) \quad = \quad \text{Send}(TP, N(n,w), A'^*(n)). \tag{3.56}$$

- Case 3: (d_i, i) is sent, but $(d_{i+1}, i+1)$ is lost

  With regard to time t_i and d_{i+1}, we have two subcases to consider.

  - $d_{i+1} \geq t_i + 1$.

    That is,

$$d'_i = d_{i+1} \geq t_i + 1 = t'_i + 1. \tag{3.57}$$

Hence, (d'_i, i) is sent. It follows that $t'_{i+1} = t_{i+1}$. Moreover, if $d_i \geq t_{i+1} + 1$, that is,

$$d'_{i+1} = d_i \geq t'_{i+1} + 1, \tag{3.58}$$

then $(d'_{i+1}, i+1)$ is sent; otherwise, it is lost. Thus, we have

$$\text{Send}(TP, N(n,w), A^*(n)) \leq \text{Send}(TP, N(n,w), A'^*(n)). \tag{3.59}$$

– $d_{i+1} < t_i + 1$.

Because $d'_i = d_{i+1} < t_i + 1 = t'_i + 1$, (d'_i, i) is lost. Let s denote the number of messages sent before the token arrives at node i, we have

$$d_i \geq t_i + 1 = s + iw + 1. \tag{3.60}$$

As d_i is an integer, thus

$$d_i \geq \lceil s + iw + 1 \rceil \tag{3.61}$$

With the assumption $iw \leq nw < 1$, we establish the following:

$$\lceil s + iw + 1 \rceil \quad = \quad \lceil s + (i+1)w + 1 \rceil. \tag{3.62}$$

It follows that

$$\begin{aligned} d'_{i+1} = d_i &\geq \lceil s + iw + 1 \rceil \\ &= \lceil s + (i+1)w + 1 \rceil \\ &\geq t'_{i+1} + 1. \end{aligned} \tag{3.63}$$

Hence, $(d'_{i+1}, i+1)$ is sent.

The remaining messages in both sets are the same. Therefore,

$$\text{Send}(TP, N(n,w), A^*(n)) = \text{Send}(TP, N(n,w), A'^*(n)). \tag{3.64}$$

- Case 4: (d_i, i) is lost, but $(d_{i+1}, i+1)$ is sent

  This means

$$d_i < t_i + 1, \quad \text{and} \tag{3.65}$$

$$d_{i+1} \geq t_{i+1} + 1. \tag{3.66}$$

  Because $d_i > d_{i+1}$ and $t_i < t_{i+1}$, (3.66) becomes

$$d_i > d_{i+1} \geq t_{i+1} + 1 > t_i + 1, \tag{3.67}$$

  which contradicts to (3.65). Hence, this case does not exist.

With (3.44), (3.51), (3.56) and (3.59), we have

$$\text{Send}(TP, N(n,w), A^*(n)) \;\leq\; \text{Send}(TP, N(n,w), A'^*(n)). \tag{3.68}$$

This conclude s the proof. Q.E.D.

3.3 Worst Case Performance Analysis

Lemmas 3.2.1 and 3.2.2 have established the message deadline and position ordering in terms of the worst case performance ratio among the equal-size message sets. Using these lemmas, we can now identify the worst case message set which yields the worst case performance ratio for the token passing protocol.

Lemma 3.3.1 *Given a token ring of n nodes, if the token passing protocol is used to transmit n messages, the worst case message set is*

$$A_{wc}(n) = \{(1,n),(2,n-1),\cdots,(n-1,2),(n,1)\}. \tag{3.69}$$

That is, in the worst case messages are positioned in the network in the decreasing order of their deadlines.

Proof: Lemma 2.4.1 states that message deadlines of any feasible message sets of size n must satisfy $d_i \geq i$ $(1 \leq i \leq n)$. With (2.12), we know that for message sets of size n, the one having message deadlines of $\{1, 2, ..., n-1, n\}$ is the smallest feasible set, thus we have

$$A_{wc}(n) \leq_D A(n), \tag{3.70}$$

where $A(n)$ is any other message sets of size n.

On the other hand, we see that messages in $A_{wc}(n)$ given in (3.69) are positioned in the decreasing order of their deadlines, such that the one with the largest deadline is always sent first by the token passing protocol. From Lemma 3.2.2, we know that for a given message set, if the deadlines of two messages on the neighboring nodes are interchanged so that the token will visit the node with the smaller deadline message first, then the number of messages sent by the token passing protocol may increase. It follows that the number of messages sent from $A_{wc}(n)$ is minimized since messages are positioned such that the largest deadline message is always sent first. Therefore,

$$\text{Send}(TP, N(n,w), A_{wc}(n)) \leq \text{Send}(TP, N(n,w), A(n)). \tag{3.71}$$

That is, $A_{wc}(n)$ is the message set that yields the worst case performance ratio of the token passing protocol Q.E.D.

With the worst case message set given in (3.69), it is straightforward to compute the worst case performance ratio.

Theorem 3.3.1 *The worst case performance ratio of the token passing protocol is*

$$R(TP, w, n) = \begin{cases} 0 & n < w+1, \\ \\ \frac{\lfloor \frac{n+1}{w+2} \rfloor}{n} & \textit{otherwise.} \end{cases} \tag{3.72}$$

Proof: We first calculate the number of messages sent from $A_{wc}(n)$ by the token passing protocol. Let s denote *Send*$(TP, N(n,w), A_{wc}(n))$.

At time 0, the token is released from node n, moving towards node 1 which holds message $(n, 1)$. It takes w units of time for the token to reach node 1. We have two cases to consider.

- No message is sent

  This happens if $n < w + 1$. When the token arrives at node 1, the latest time to send message $(n, 1)$ has passed. As the deadlines of the rest messages are smaller than n, all messages have been lost.

- One or more messages are sent.

  Suppose s messages are sent. As $d_i > d_j$ for $i < j$, and messages are sent in the latest-deadline-first order, those s sent messages must be $(n, 1)$, $(n-1, 2)$, $(n-2, 3)$, $\cdots$, $(n-s+1, s)$. That is, the i-th sent message is on node i. Thus, the overhead for each message transmission is simply the time for the token to travel from one node to its nearest neighbor, i.e. the token node-to-node delay w. As a result, each message transmission takes $1 + w$ units of time. As the message having a deadline of $n - s + 1$ is the last message sent, its deadline must not expire when the transmission of this message is completed. That is,

  $$n - s + 1 \geq s \times (1 + w). \tag{3.73}$$

  Solving (3.73) for s in terms of n and w, we have

  $$s \leq \frac{n+1}{w+2}. \tag{3.74}$$

  As a total of s messages are sent, s must be the largest integer which satisfies (3.74). That is,

  $$s = \lfloor \frac{n+1}{w+2} \rfloor. \tag{3.75}$$

  As the worst case performance ratio is defined as the fraction of messages sent from the message set, this leads to (3.72). Q.E.D.

This theorem implies that the worst case performance ratio is a decreasing function of parameters n and w.

We now extend Theorem 3.3.1 to more general cases.

Corollary 3.3.1 *If the token passing protocol is invoked at time T_s to transmit messages from the worst message set given in (3.69), then its worst case performance ratio is*

$$R(TP, w, n, T_s) = \begin{cases} 0 & n < T_s + w + 1, \\ \frac{\lfloor \frac{n+1-T_s}{w+2} \rfloor}{n} & otherwise. \end{cases} \tag{3.76}$$

This corollary can be easily proved by substituting $s \times (1 + w)$ with $T_s + s \times (1 + w)$ on the right hand side of (3.73), since the protocol is invoked at time T_s.

Corollary 3.3.1 can be further extended to the case where the size of the message set is smaller than the number of nodes n.

Corollary 3.3.2 *If the token passing protocol is invoked at time T_s to transmit h ($h < n$) messages, then its worst case performance ratio is*

$$R(TP, w, n, h, T_s) = \begin{cases} 0 & h < w + 1, \\ \frac{\lfloor \frac{h+1-T_s}{w+2} \rfloor}{h} & otherwise, \end{cases} \tag{3.77}$$

Proof: As the token passing protocol is deadline monotonic, with a similar argument used in the proof of Lemma 3.3.1, the worst case message set for h messages is

$$\{(1,p_1),(2,p_2),\cdots,(h-1,p_{h-1}),(h,p_h)\}, \tag{3.78}$$

where $p_1 > p_2 > \cdots > p_{h-1} > p_h$. As there are n nodes but h messages ($h < n$), these messages can reside on any nodes as long as they are distributed in the decreasing order of deadlines.

Let s ($s < h$) denote the number of messages sent from $A(h)$. As in the worst case the token passing protocol sends messages in the latest-deadline-first order, thus the s sent messages must be

$$(h,p_h),(h-1,p_{h-1}),\cdots,(h-s+2,p_{h-s+2}),(h-s+1,p_{h-s+1}). \tag{3.79}$$

As at time T_s the token is at node n, moving towards node p_h where message (h,p_h) resides, evidently the first message transmission incurs an overhead of $g_{n,p_h}w$ where g_{n,p_h} is the distance from node n and node p_h as defined in Definition 2.24. After the transmission of message (h,p_h) completes, the token moves from node p_h to node p_{h-1}. Thus, the overhead involved in the second message transmission is $g_{h,p_{h-1}}w$. Likewise, the transmission of the s-th message $(h-s+1,p_{h-s+1})$ incurs an overhead of $g_{h-s+2,p_{h-s+1}}w$. Hence, the transmission of s sent messages completes at

$$\begin{aligned} T_s &+ (g_{n,p_h}w+1)+(g_{h,p_{h-1}}w+1)+\cdots+(g_{h-s+2,p_{h-s+1}}w+1) \\ &= Ts+(g_{n,p_h}w+g_{h,p_{h-1}}w+\cdots+g_{h-s+2,p_{h-s+1}}w)+s \\ &= T_s+(g_{n,p_h}+g_{h,p_{h-1}}+\cdots+g_{h-s+2,p_{h-s+1}})w+s. \end{aligned} \tag{3.80}$$

The term $(g_{n,p_h} + g_{h,p_{h-1}} + \cdots + g_{h-s+2,p_{h-s+1}})$ in (3.80) is the total number of links that the token has traveled when s messages are sent. Clearly,

$$s \leq g_{n,p_h} + g_{h,p_{h-1}} + \cdots + g_{h-s+2,p_{h-s+1}} \leq n. \tag{3.81}$$

That is, to send s messages, the token must travel at least s links since these messages reside on s different nodes.

As message $(h-s+1, p_{h-s+1})$ is the last sent message, its deadline $h-s+1$ should not expire when its transmission complete. With (3.80), we arrive at

$$h - s + 1 \geq T_s + (g_{n,p_h} + g_{h,p_{h-1}} + \cdots + g_{h-s+2,p_{h-s+1}})w + s. \tag{3.82}$$

With (3.81), (3.82) becomes

$$h - s + 1 \geq T_s + sw + s. \tag{3.83}$$

Solving s in terms of w, h and T_s, we have

$$s \quad \leq \quad \frac{h+1-T_s}{w+2}. \tag{3.84}$$

As s is the total number of messages sent, s must be the largest integer satisfying (3.84). Hence,

$$s \quad = \quad \lfloor \frac{h+1-T_s}{w+2} \rfloor. \tag{3.85}$$

This concludes the proof. Q.E.D.

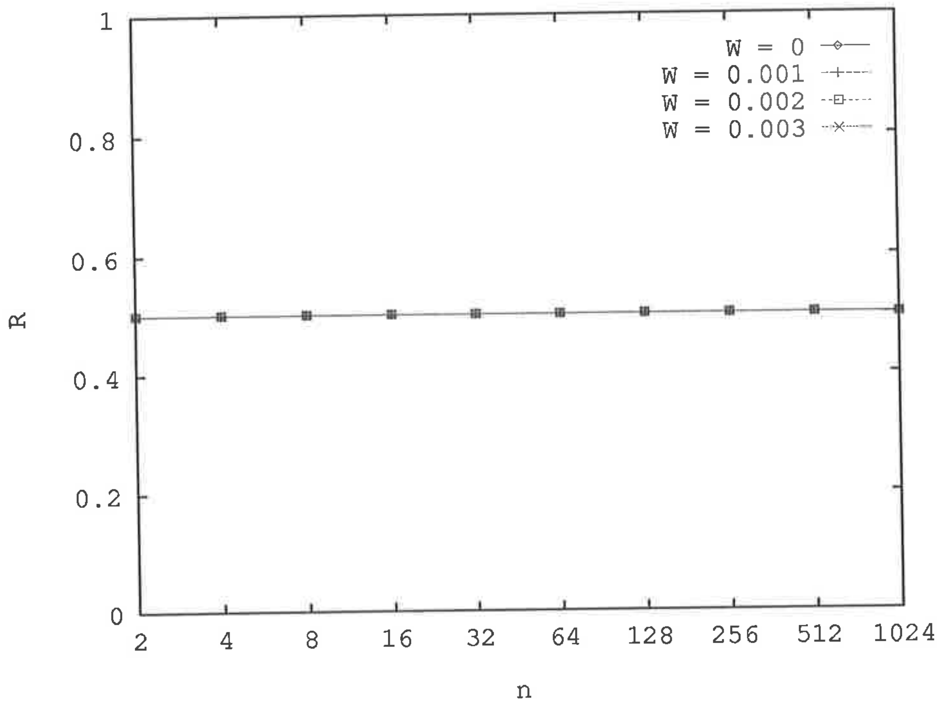


Figure 3.1. Worst Case Performance Ratio of *TP*

3.4 Numerical Results and Discussions

From Lemma 3.3.1, we see that the worst case performance ratio of the token passing protocol is always less than one. The following factors contribute to the performance loss.

- The token passing protocol implements the nearest-neighbor-first transmission policy. In the worst case, it sends messages in the latest-deadline-first order, which is in contrast to the CEDF protocol that always sends the earliest deadline message first. This nearest-neighbor-first transmission policy does not consider individual message deadlines, as a result it has a severe performance impact. To see this more clearly, we consider the worst case performance ratio given in (3.72) when the token node-to-node delay w is 0,

$$R(TP, w, n) \ \leq \ \frac{\lfloor \frac{n+1}{2} \rfloor}{n} \approx \frac{1}{2}. \qquad (3.86)$$

That is, given the same perfect environment where the contention overhead is assumed to be zero, the token passing protocol can send only half the messages sent by the CEDF protocol. This is solely due to the non-EDF transmission policy used by the token passing protocol. Figure 3.1 shows the result. We see that in the plotted ranges of n and w, the worst case performance ratio is as low as 0.5. It also shows that the performance of the token passing protocol is relatively insensitive to parameters w and n, which implies that the dominant cause for its poor worst case performance is the non-EDF transmission policy used.

- The token passing protocol requires a non-zero contention overhead, which is equal to the token node-to-node delay. To send a message, it takes at least w units of time for the token to travel from the current node to the node where the next sending message resides. This amount of time is the contention overhead, which is also responsible for degrading the protocol performance. However, we see that its impact is negligible as compared with that of the transmission policy.

Chapter 4

The Priority-Driven Protocol

In the last chapter, we analyzed a simple token ring protocol which schedules message transmission using the nearest-neighbor-first policy which does not consider individual message timing requirements. Because of this, the token passing protocol is not commonly used for real-time communications. Instead the predominant approach taken by many proposed standard or non-standard token ring protocols is *priority-driven* [1, 8, 11, 17, 40, 44] as discussed in Chapter 1. Under these token ring protocols, messages are assigned different access priorities based on their service classes. At any time the message with the highest priority is sent first. Although discriminating messages based on service classes is adequate in a non real-time environment, these protocols may perform poorly in supporting real-time message transmission as they do not explicitly address individual message deadlines.

In this chapter, we propose and study a modified *priority-driven* (PD) token ring protocol which explicitly addresses individual message deadlines [24, 47, 49, 51]. It uses a prioritized access mechanism together with a priority assignment function to support time-constrained message transmission. Different access priority levels are used to differentiate messages with different deadlines. We show that when the number of priority levels is sufficient, the protocol implements the exact EDF transmission policy; otherwise, it only approximates the EDF policy. Consequently, in the worst case, the protocol performance may not be desirable.

4.1 Protocol Description

In this protocol, each node implements a *priority assignment function* which assigns an access priority to each message based on its deadline upon its arrival. Each node maintains a prioritized message queue where messages are kept in the decreasing order of the access priority. Messages whose deadlines have already expired are discarded. At any time the head of a message queue at a node represents the highest priority message queued at that node, and thus is considered for transmission before any other messages in that queue.

The token contains a *priority field* to facilitate the prioritized access to the ring. When a free token arrives at a node, the node examines the token priority field and inserts the highest priority of its pending messages (if any) if this priority is higher than the one currently indicated in the token. This enables the token priority field to represent the highest priority of messages waiting in the system after the token has completed at least one full circulation around the ring. A node is allowed to capture the token and transmit its message only when the token returns with the node's claimed priority after passing through all other nodes. When the sending node completes the message transmission, it issues a new token with the token priority field set to the lowest priority. The protocol continues this way.

As this priority-driven protocol incorporates message deadlines into priorities, it is expected to perform better than the simple token passing protocol analyzed in the last chapter. On the other hand, we can see from the above description that the contention overhead of this protocol is substantially higher than that of the token passing protocol. This is because after a message transmission, it takes at least w (at most $(n-1)w$) units of time for the token to reach the node where the message with the next highest priority resides; it then takes another nw units of time for the token to return to this node. At that time the node starts transmitting the message. We will show that this higher contention overhead may degrade the protocol performance such that the priority-driven protocol performs even worse than the simple token passing protocol in some situations.

4.2 Priority Assignment Function

As already mentioned, to incorporate message deadlines into this priority-driven protocol, a priority assignment function is used to map message deadlines to access priorities. Before introducing a specific priority assignment function, we first identify the characteristics of deadline-to-priority mapping functions in general.

- Non-decreasing

  A message with a smaller deadline should always be assigned a higher priority. Formally, for two messages M and M' with deadlines being d and d' respectively, if $d \leq d'$, then we have

$$Pri_M = f(d) \leq Pri_{M'} = f(d'). \tag{4.87}$$

- Finite range

  In practice, the number of priorities provided by the token is limited by the token length. This implies that the priority assignment function must have a *finite range.*

- Many-to-one

  As the number of message deadlines is theoretically infinite, given a finite priority range, the priority assignment function must map an infinite number of deadlines to a finite number of priorities. The impact of the many-to-one property is that no matter what form the mapping function has, more than one message with different deadlines may be assigned the same priority. As a result, the one with a larger deadline may be sent first. Hence, the priority-driven protocol only approximates the EDF policy.

These properties are common to any mapping functions implementing the EDF policy. Following is a general form of priority assignment functions.

$$\mathrm{Pri}_M = f(d,q) = \begin{cases} \lceil \frac{d}{q} \rceil & \lceil \frac{d}{q} \rceil \leq m, \\ \\ m & \lceil \frac{d}{q} \rceil > m, \end{cases} \tag{4.88}$$

where q is the *length of priority assignment function.* We see that $f(d,q)$ defined in (4.88) is a *non-decreasing many-to-one* function, which has the properties discussed above.

Theoretically, it is difficult to decide the optimal length of the priority assignment function as it depends on network and protocol parameters, and the message deadline distribution of the applications. However, in Chapter 7 which deals with the average case performance of the protocol, we will examine the impact of the length of priority assignment function q on the protocol performance.

To facilitate our worst case analysis, we now introduce a simple priority assignment function. Assume that there are m priority levels indexed by $1, 2, \cdots, m$, where 1 is the highest priority and m is the lowest. Let integer d be the deadline of message M and Pri_M be the priority of M determined by the priority assignment function, we have

$$\mathrm{Pri}_M = f(d, 1) = \begin{cases} d & d \leq m, \\ m & d > m. \end{cases} \tag{4.89}$$

Clearly, this priority assignment function is non-decreasing and many-to-one. Furthermore, it is a special case of the priority assignment function defined in (4.88) with $q = 1$.

It should be pointed out that although many other mapping functions are feasible, this assignment function is chosen for the purpose of reducing the complexity of the worst case performance analysis.

Figure 4.1 shows this priority assignment function when the number of priorities m is 64 and 256 respectively. For the curve of $m = 64$, we see that a message with a deadline smaller than 64 is assigned a distinct priority[1]. The smaller the deadlines, the higher the priorities (i.e. the lower numerical value). In this case, when a message is transmitted, it must have the highest priority, thus the earliest deadline, among all pending messages at that time. Clearly, the protocol is implementing the EDF transmission policy exactly.

[1]Similar observations can be made for the curve of $m = 256$ except that the turning point is now at $n = 256$ instead of $n = 64$.

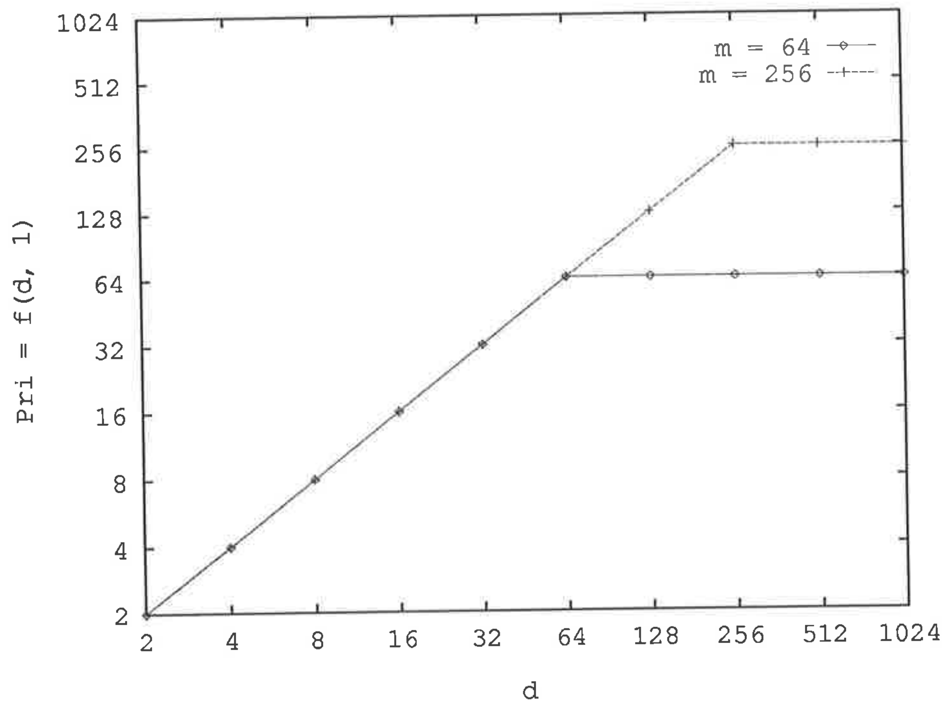


Figure 4.1. A Priority Assignment Function

On the other hand, messages with deadlines equal to or greater than 64 are assigned the same priority of 64. These messages are not differentiated for transmission. As a result, messages with smaller deadlines may have to wait while a message with a larger deadline is being transmitted. Evidently, in this case the protocol only implements the EDF policy approximately. Therefore, the EDF policy is observed by the priority-driven protocol only among different priority levels, but not necessarily within the same priority level. Consequently, more messages may be lost as compared with the case where the EDF policy is implemented exactly. The following example illustrates this point.

Example 4.2.1 *Suppose there are two priorities levels are available: Pri 1 and Pri 2. The priority assignment function is defined as*

$$Pri_M = f(d,1) \quad = \begin{cases} d & d \leq 2, \\ \\ 2 & d > 2. \end{cases} \tag{4.90}$$

Suppose there are four messages residing on different nodes with deadlines of 2, 3, 4 and 6 respectively. Each message transmission is assumed to take 1.1 units of time.

With the priority assignment function defined in (4.90), each message is assigned the same priority upon its arrival:

$$Pri_{M_1} = Pri_{M_2} = Pri_{M_3} = Pri_{M_4} = 2.$$

As messages have the same priority, the node that captures the token first will send its message first. Suppose the node with M_4 seizes the token first and completes transmission of M_4 at time 1.1. Then the node with M_3 captures the token and the transmission of M_3 completes at time 2.2. At that time the deadline of M_1 has already expired. The remaining message M_2 will also be lost, since a message transmission from time 2.2 will be completed at time 3.3 by when the deadline of M_2 has already expired. Thus, in this case, only two messages are sent. Figure 4.2(a) shows the time diagram for message transmission in this case.

However, if the EDF transmission policy is strictly observed, message transmission sequence in terms of message deadline should be 2, 3, 4 and 6. The times when the corresponding message transmission completes are 1.1, 2.2, 3.3, and 4.4. Clearly, all messages can make their deadlines as illustrated in Figure 4.2(b).

It is obvious that the priority assignment function defined in (4.89) maps a deadline to a priority *staticly*, i.e. for a given deadline, its corresponding priority is determined

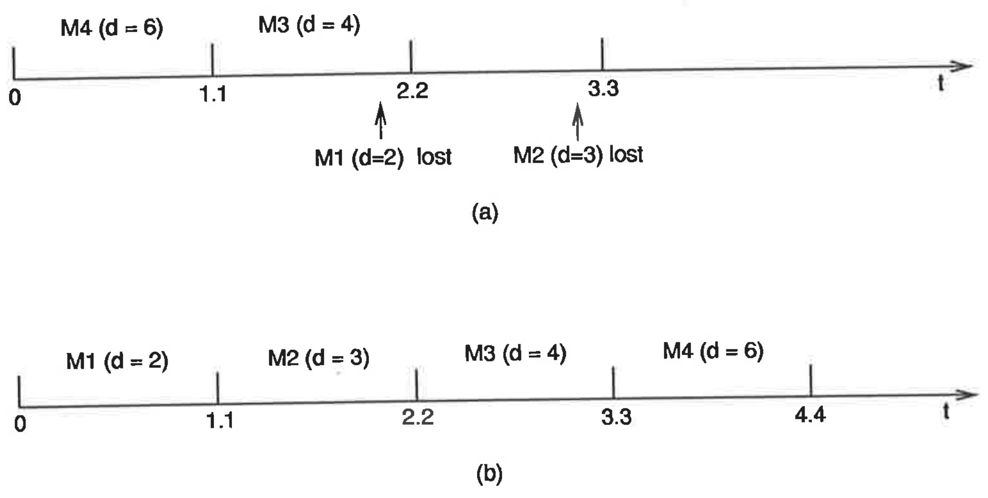


Figure 4.2. Time Diagram of Message Transmission

and fixed. However, if priorities are assigned *dynamically* rather than *staticly*, then fewer messages would have been lost. That is, upon a message arrival or a transmission, all message deadlines are re-ordered in the increasing order and then assigned priorities in the decreasing order (starting from the highest priority). In the above example, if priorities are assigned dynamically, then initially the message with a deadline of 2 would be assigned *Pri* 1 and hence be sent first, followed by messages with *Pri* 3, 4 and 6 respectively. No messages would be lost. However, it is not difficult to see that the implementation of such dynamic priority assignment function requires a centralized controller which has the complete knowledge of explicit deadlines of all messages waiting in the system. Evidently, it is not possible for each node in a token ring network, which only knows its own message deadlines, to obtain explicit global message deadline information in a distributed manner.

It may also be argued that if sufficient priorities are available, then the EDF policy may be observed. In the above example, if four priority levels are available, then each message would be assigned a distinct priority. As a result, all messages would be sent successfully. In the worst case analysis of the protocol, we will examine the impact of the number of priorities on the protocol performance.

In the following, we wish to derive the worst case performance ratio of the priority-driven protocol for a set of messages with arbitrary deadlines. Let d and D denote respectively the earliest and latest deadline of a set of messages. we use $PD_{m>D}$, $PD_{m\leq d}$ and $PD_{d<m<D}$ to indicate the following.

- $PD_{m>D}$: the priority-driven protocol is used to transmit a set of messages whose deadlines are smaller than m.
- $PD_{m\leq d}$: the priority-driven protocol is used to transmit a set of messages whose deadlines are equal to and greater than m.
- $PD_{d<m<D}$: the priority-driven protocol is used to transmit a set of messages whose deadlines are smaller than, equal to and greater than m.

As the protocol behaves differently in these three cases, we deal with them separately in order to reduce the complexity of the analysis. However, it will become clear that $PD_{m>D}$ and $PD_{m\leq d}$ are special cases of $PD_{d<m<D}$, thus the worst case performance analysis of the first two cases will provide some insights and results for the third case.

4.3 Worst Case Performance of $PD_{m>D}$

We now analyze the worst case performance of the priority-driven protocol when used to transmit messages whose deadlines are smaller than the number of priorities. With the priority assignment function given in (4.89), each message is assigned a distinct priority level since its deadline is smaller than m.

In the following, we first present several lemmas which characterize the protocol properties associated with $PD_{m>D}$. With these properties, we then derive the bounds of the number of messages sent in the worst case and obtain the bounds of the worst case performance ratios. Using these bounds, we arrive at an estimation of the worst case performance ratio. Finally, we give a quantitative measurement of the maximum error of the estimation.

We begin with the examination of the impact of message deadlines on the number of messages sent by $PD_{m>D}$.

Lemma 4.3.1 *Given two message sets of size n*

$$\begin{aligned} A(n) &= \{(d_1,p_1),(d_2,p_2),\cdots,(d_{n-1},p_{n-1}),(d_n,p_n)\}, \\ A'(n) &= \{(d'_1,p_1),(d'_2,p_2),\cdots,(d'_{n-1},p_{n-1}),(d'_n,p_n)\}, \end{aligned} \tag{4.91}$$

where $d_i \le d_j$ $(i \le j)$, $d'_i = d_i$ $(1 \le i \le n-1)$ and $d_n < d'_n$, if the EDF policy is used to transmit these two message sets, in the worst case,

$$Send(EDF, N(n,w), A'(n)) \ge Send(EDF, N(n,w), A(n)). \tag{4.92}$$

Proof: As messages are sorted in the deadline order and the EDF policy is implemented, thus (d_n, p_n) and (d'_n, p'_n) must be the last message sent from $A(n)$ and $A'(n)$ respectively if they are sent. We see that message deadlines in $A'(n)$ are the same as those in $A(n)$ except for d_n. Clearly, the message transmission sequence remains the same up to (d_{n-1}, p_{n-1}) for both $A(n)$ and $A'(n)$.

Let t be the time when message transmission of the first $n-1$ messages is completed. It follows that the message transmission sequence remains the same up to time t. Let

τ denote the message transmission time plus the contention overhead. There are two possible cases:

- d_n is sent in $A(n)$.

  That is, $d_n \geq t + \tau$. Hence

$$d'_n > d_n \geq t + \tau. \tag{4.93}$$

  Thus, d'_n is sent too. We have

$$Send(EDF, N(n,w), A'(n)) = Send(EDF, N(n,w), A(n)). \tag{4.94}$$

- d_n is lost in $A(n)$.

  This implies $d_n < t + \tau$. If $d_n < d'_n < t + \tau$, then d'_n is also lost, thus

$$Send(EDF, N(n,w), A'(n)) = Send(EDF, N(n,w), A(n)); \tag{4.95}$$

  otherwise, if $d'_n > d_n > t + \tau$, then d'_n is sent, hence

$$Send(EDF, N(n,w), A'(n)) = Send(EDF, N(n,w), A(n)) + 1. \tag{4.96}$$

In summary, we have

$$Send(EDF, N(n,w), A'(n)) \geq Send(EDF, N(n,w), A(n)). \tag{4.97}$$

This concludes the proof. Q.E.D.

Lemma 4.3.2 *If the EDF transmission policy is used to transmit n messages, the message set with a maximum message deadline of n minimizes the number of messages sent.*

Proof: From Lemma 4.3.1, we know that the smaller the maximum deadline of a message set, the smaller the number of message sent. For n messages, the deadline of the last message must satisfy $d_n \geq n$ to ensure the feasibility of the message set. Hence a message set with $d_n = n$ minimizes the number of messages sent. This concludes the proof. Q.E.D.

We now compute the number of messages sent from a set of n messages with a maximum deadline of n.

Lemma 4.3.3 *If the EDF policy is used to transmit n messages with a maximum deadline of $d_n = n$ and each transmission takes τ ($\tau > 1$) units of time (including the overhead), then the total number of messages sent, denoted by s, is given by*

$$s = \lfloor \frac{n}{\tau} \rfloor, \tag{4.98}$$

where $\lfloor x \rfloor$ is defined as the largest integer which is smaller than x.

Proof: Let these n messages be

$$A(n) = \{(d_1, p_1), (d_2, p_2), \cdots, (d_{n-1}, p_{n-1}), (d_n, p_n)\}. \tag{4.99}$$

If $s \neq \lfloor \frac{n}{\tau} \rfloor$, then either $s > \lfloor \frac{n}{\tau} \rfloor$ or $s < \lfloor \frac{n}{\tau} \rfloor$. In the following, we wish to prove that neither of them is true.

- $s > \lfloor \frac{n}{\tau} \rfloor$

  Without loss of generality, let

$$s = \lfloor \frac{n}{\tau} \rfloor + i, \qquad \text{where } i \geq 1. \tag{4.100}$$

The time t when the transmission of s sent messages completes is

$$\begin{aligned} t &= s \times \tau \\ &= (\lfloor \frac{n}{\tau} \rfloor + i) \times \tau. \end{aligned} \tag{4.101}$$

Let $n = \lfloor \frac{n}{\tau} \rfloor \tau + \varepsilon$, where $0 \leq \varepsilon < \tau$, hence $\lfloor \frac{n}{\tau} \rfloor = \frac{n-\varepsilon}{\tau}$. It follows that

$$\begin{aligned} t &= (\frac{n-\varepsilon}{\tau} + i) \times \tau \\ &= n - \varepsilon + i\tau. \end{aligned} \tag{4.102}$$

Rearranging the above, we arrive at

$$\begin{aligned} t - n &= i\tau - \varepsilon \\ &> 0, \qquad \text{as } \tau > \varepsilon \text{ and } i \geq 1. \end{aligned} \tag{4.103}$$

It follows that $t > n$, which implies that there must exist a message having a deadline greater than n. This is impossible as $d_n = n$ is the largest deadline and messages and messages are sent in the EDF order. Hence $s > \lfloor \frac{n}{\tau} \rfloor$ does not hold.

- $s < \lfloor \frac{n}{\tau} \rfloor$

  Similarly, let

$$s = \lfloor \frac{n}{\tau} \rfloor - i, \qquad \text{where } i \geq 1. \tag{4.104}$$

Using the same approach, we have

$$\begin{aligned} t &= s \times \tau \\ &= (\lfloor \frac{n}{\tau} \rfloor - i) \times \tau \\ &= (\frac{n-\varepsilon}{\tau} - i) \times \tau \\ &= n - \varepsilon - i\tau \\ &\leq n - \varepsilon - \tau, \qquad \text{as } i \geq 1 \\ &< n - 1, \qquad \text{as } \tau > 1 \text{ and } \varepsilon > 0. \end{aligned} \tag{4.105}$$

That is, the message transmission is completed before time $n-1$. This implies that message (d_n, p_n) with a deadline of $d_n = n$ must have been sent. As the protocol implements the EDF policy and $d_n = n$ is the largest deadline, it must be the last sent message. It follows that message (d_{n-1}, p_{n-1}) must have be sent before (d_n, p_n) and its transmission completes at

$$n - 1 - \tau \ < \ n - 2, \qquad \text{as } \tau > 1. \tag{4.106}$$

That is, transmission of message (d_{n-1}, p_{n-1}) is completed before time $n-2$. Repeat this process for messages $(d_{n-2}, p_{n-2}), (d_{n-3}, p_{n-3}), \cdots, (d_2, p_2)$ with the same argument, it is easy to see that transmission of message (d_2, p_2) must be completed before time $2 - \tau < 1$. This is impossible because it takes at least τ units of time to complete the first message transmission since $\tau > 1$. Hence, the hypothesis $s < \lfloor \frac{n}{\tau} \rfloor$ can not be true.

In summary of the above two cases, we must have $s = \lfloor \frac{n}{\tau} \rfloor$. Q.E.D.

The above lemma can be readily extended to more general cases.

Corollary 4.3.1 *Given a network of n nodes and h $(h < n)$ messages with a maximum deadline of h, if messages are transmitted in the EDF order starting at time T_s and each transmission takes τ $(\tau > 1)$ units of time (including overhead), then the total number of messages sent, denoted by s, is given by*

$$s = \lfloor \frac{h - T_s}{\tau} \rfloor. \tag{4.107}$$

Proof: This lemma can be easily proved by substituting $t = s \times \tau$ with $t = T_s + s \times \tau$ in (4.101) and (4.105). Q.E.D.

Let us now consider the contention overhead involved in a message transmission under $PD_{m>D}$ protocol.

Lemma 4.3.4 *Given a token ring of n nodes, under $PD_{m>D}$ messages are sent in the EDF order and the maximum and minimum contention overhead of a message transmission is $(2n-1)w$ and nw respectively.*

Proof: As all message deadlines are smaller than m under $PD_{m>D}$, with the priority assignment function defined in (4.89), messages with different deadlines are assigned distinct priorities. As $PD_{m>D}$ sends messages in the order of decreasing priority, thus it sends messages in the EDF order.

Assume that at time 0, the token is released from node i moving toward node $i+1$. Suppose that the highest priority message resides on node j $(j \neq i)$.

With definition (2.5.3), it takes the token $g_{i,j}w$ units of time to reach node j $(i \neq j)$ which holds the current highest priority message. It then takes another nw units of time for the token to pass through the network and return to node j so that it can start its message transmission. The contention overhead c of a message transmission can thus be seen as the sum of two components. One is the time c_1 needed for the token to travel from the last sending node (or the initial token node) to the node having the current highest priority message. The other is the amount of time c_2 required for the token to complete a full token circulation to confirm that this is indeed the highest priority message currently in the network. Formally,

$$\begin{aligned} c &= c_1 + c_2 \\ &= g_{i,j}w + nw. \end{aligned} \tag{4.108}$$

From Lemma 2.5.2, the maximum and minimum value of $g_{i,j}$ is $n-1$ and 0 respectively. Therefore, the maximum and minimum contention overhead, denoted by c_{max} and c_{min} respectively, is obtained as follows.

$$\begin{aligned} c_{max} &= \forall_i \forall_j \max(g_{i,j}w + nw) \\ &= (n-1)w + nw \\ &= (2n-1)w. \end{aligned} \tag{4.109}$$

$$\begin{aligned} c_{min} &= \forall_i \forall_j \min(g_{i,j}w + nw) \\ &= 0 + nw \\ &= nw. \end{aligned} \tag{4.110}$$

This concludes the proof. Q.E.D.

From Lemma 4.3.4, it seems that if messages are distributed in such a way that each transmission always incurs a maximum contention overhead of $(2n-1)w$, then we would have the worst case performance of PD$_{m>D}$. However, the following example demonstrates that this is not always true.

Example 4.3.1 *Given a network of 10 nodes and the token node-to-node delay w of 0.027. A message set $A(n)$ contains 10 messages whose deadlines are 1, 2, $\cdots$, 9 and 10 respectively. Let us now examine if it is possible that each message transmission incurs a maximum contention overhead of (2n-1)w, i.e. 0.513 in this case.*

If so, the first message to be sent must reside on node 9 and its deadline must satisfy the following:

$$\begin{aligned} b_1 &= \lceil (2n-1)w + 1 \rceil \\ &= \lceil 0.513 + 1 \rceil \\ &= \lceil 1.513 \rceil \\ &= 2. \end{aligned} \tag{4.111}$$

In the above, $\lceil x \rceil$ is defined as the smallest integer which is greater than x. That is, message (2, 9) is the first sent message. In the following, we show that this is indeed the case. First, message $(1, p_1)$, where $p_1 \neq n-1$, has a deadline of 1 and is on a node which is visited by the token before node 9, thus message $(1, p_1)$ is considered before message $(b_1, 9)$. From Lemma 4.3.4, the overhead of a message transmission is at least nw. Thus, the transmission of message $(1, p_1)$ will complete no earlier than time $nw + 1 = 1.59 > 1$. This indicates that at that time, the deadline of the message has expired. Thus, the node does not write the priority of message $(1, p_1)$ into the token. For other messages, their deadlines are larger than b_1. Hence, message $(b_1, 9)$ is the first sent message.

The transmission of the first message (2, 9) completes at time 1.513 and message $(1, p_1)$ *has been lost. Similarly, if the transmission of the second sent message incurs a contention overhead of 0.53, then the message must reside on node 8 and have a deadline of*

$$\begin{aligned} b_2 &= \lceil 2 \times ((2n+1)w+1) \rceil \\ &= \lceil 2 \times 1.513 \rceil \\ &= \lceil 3.026 \rceil \\ &= 4. \end{aligned} \tag{4.112}$$

Let message $(4, p_4)$ *be on node 8, thus message* $(3, p_3)$ *is on a node other than node 8, say node 7. We wish to show that message (3, 7) is the second sent message rather than message (4, 8). Clearly, It takes the token 8w units of time to reach node 7 from node 9. The node writes its priority into the token. It takes the token another 10w units of time to return to node 7. That is, at time*

$$\begin{aligned} t &= 1.513 + (8+10) \times 0.027 \\ &= 1.999, \end{aligned} \tag{4.113}$$

the transmission of message (3, 7) starts. It is completed at time 2.999. At that time, the deadline of message (3, 7) has not expired yet. So message (3, 7) is the second sent message which has an overhead of (2n-2)w rather than (2n-1)w.

In the example, we see that message (3, 7) is the message that makes it impossible for the second sent message to have the contention overhead of $(2n-1)w$. For the transmission of message (3, 7), the maximum contention overhead is $(2n-2)w$.

4.3.1 Lower Bound of $R(PD_{m>D}, w, n)$

The above example implies that given a n-node token ring, not every message transmission incurs a contention overhead of $(2n-1)w$, but $(2n-1)w$ is indeed the maximum possible overhead incurred. We now derive the lower bound of the number of messages sent by $PD_{m>D}$ based on this overhead.

Lemma 4.3.5 *Given a token ring of n nodes, if $PD_{m>D}$ is used to transmit n messages, the lower bound of the number of messages sent is*

$$Send_{low}(PD_{m>D}, N(n,w), A(n)) = \lfloor \frac{n}{(2n-1)w+1} \rfloor. \tag{4.114}$$

Proof: Lemma 4.3.2 indicates that for n messages, if the EDF policy is employed to transmit messages, the message set with the maximum message deadline being n minimizes the number of messages sent.

Lemma 4.3.4 states that $(2n-1)w$ is an upper bound of the contention overhead for a message transmission under $PD_{m>D}$, a lower bound of $Send(PD_{m>D}, N(n,w), A(n))$ can be obtained by substituting $\tau = (2n-1)w+1$ in Lemma 4.3.3. Q.E.D.

With this lemma and the definition of the worst case performance ratio, we can readily obtain the following theorem.

Theorem 4.3.1 *The lower bound of the worst case performance ratio of $PD_{m>D}$ is*

$$R_{low}(PD_{m>D}, w, n) = \frac{\lfloor \frac{n}{(2n-1)w+1} \rfloor}{n}. \tag{4.115}$$

With Corollary 4.3.1, the above theorem can be extended to the case where the number of messages is less than n and the protocol is invoked at time $T_s > 0$.

Corollary 4.3.2 *Given a network of n nodes and h $(h < n)$ messages, if $PD_{m>D}$ is invoked at time T_s to transmit these messages, then the lower bound of the worst case performance ratio is*

$$R_{low}(PD_{m>D}, w, n, h, T_s) \quad = \quad \frac{\lfloor \frac{h-T_s}{(2n-1)w+1} \rfloor}{h}. \tag{4.116}$$

4.3.2 Upper Bound of $R(PD_{m>D}, w, n)$

As the upper bound for the message transmission overhead is $(2n-1)w$, hence, $nw, (n+\ 1)w, \cdots, (2n-2)w$ are the possible lower bounds. To obtain the upper bound on the worst case performance ratio, we first derive a lower bound on the contention overhead for a message transmission.

Lemma 4.3.6 *When $PD_{m>D}$ is used to transmit messages from*

$$A(n) = \{(1, p_1), (2, p_2), \cdots, (n-1, p_{n-1}), (n, p_n)\}, \tag{4.117}$$

the messages can be positioned in the ring such that the contention overhead for each message transmission is at least $(2n-2)w$.

Proof: Let $\mu_i = (b_i, n_i)$ be the i-th sent message, where b_i is its deadline and n_i is its position. Let $n_0 = n$ and t_i be the time when the i-th message transmission starts. The lemma is proved if we can show the following statement is true: messages in $A(n)$ are positioned in such a way that for the i-th sent message $\mu_i = (b_i, n_i)$, where

$$\begin{aligned} b_i &= \lceil t_{i-1} + (2n-1)w + 1 \rceil, & (4.118) \\ n_{i-1} - 2 &\leq n_i \leq n_{i-1} - 1, & (4.119) \\ t_{i-1} + (2n-2)w + 1 &\leq t_i \leq t_{i-1} + (2n-1)w + 1. & (4.120) \end{aligned}$$

That is, for the message set given in (4.117), the i-th sent message μ_i has a deadline of $t_{i-1} + \lceil (2n-1)w + 1 \rceil$ and resides either on node $n_{i-1} - 1$ or on node $n_{i-1} - 2$. The transmission of message μ_i completes either at $t_{i-1} + (2n-1)w + 1$ or $t_{i-1} + (2n-2)w + 1$. It is clear that the deadline and position of message μ_i depends on its predecessors due to the uncertainty that each message transmission incurs an overhead of either $(2n-1)w$ or $(2n-2)w$.

We prove this by induction. First, we show that $\mu_1 = (b_1, n_0 - 1)$, where

$$\begin{aligned} b_1 &= \lceil t_0 + (2n-1)w + 1 \rceil \\ &= \lceil (2n-1)w + 1 \rceil, \\ n_1 &= n_0 - 1 \\ &= n - 1, \\ t_1 &= t_0 + (2n-1)w + 1 \\ &= (2n-1)w + 1. \qquad (4.121) \end{aligned}$$

As at time 0 the token is moving from node n to node 1, the time when the token arrives at node $n-1$ for the first time is $(n-1)w$. It then takes the token another nw units of time to return to node $n-1$. Thus, the transmission of message $(b_1, n-1)$ starts at time $(n-1)w + nw$. Message $(b_1, n-1)$ is sent successfully as there is sufficient time to complete its transmission before its deadline expires. That is,

$$\begin{aligned} & (n-1)w + nw + 1 \\ = \; & (2n-1)w + 1 \\ \leq \; & \lceil (2n-1)w + 1 \rceil \\ = \; & b_1. \end{aligned} \tag{4.122}$$

We need to demonstrate that messages on other nodes are either lost or have lower priorities than message $(b_1, n-1)$. Assume that a message M' is on node j and has a deadline of d'. Two cases need to be considered.

- $d' > b_1 = \lceil (2n-1)w + 1 \rceil$. In this case, when the token arrives at node j, the node writes this message's priority into the token. However, the priority field in the token will eventually be modified to a priority of $\lceil (2n-1)w + 1 \rceil$ by node $n-1$. Therefore, M' is not the first sent message.

- $d' < b_1 = \lceil (2n-1)w + 1 \rceil$. As $PD_{m>D}$ implements the EDF policy, messages with deadlines of d', such that $1 \leq d' < b_1$, have higher priorities than message $(b_1, n-1)$, thus they may be considered before message $(b_1, n-1)$. However, as $d' < b_1$ and both d' and b_1 are integers, we have

$$\begin{aligned} d' & \leq b_1 - 1 \\ & = \lceil (2n-1)w + 1 \rceil - 1 \\ & = \lceil (2n-1)w \rceil. \end{aligned} \tag{4.123}$$

On the other hand, it takes the token jw units of time to travel from node n to node j for the first time. Then it takes the token another nw units of time to return to node j. Hence, the time when node j can start transmitting its message M' is

$$t_j = jw + nw. \tag{4.124}$$

If message M' is sent, its transmission completes at time

$$t'_j + 1 \quad = \quad jw + nw + 1. \tag{4.125}$$

With (4.123) and (4.125), we have

$$d' - (t'_j + 1) \quad = \quad \lceil (2n-1)w \rceil - (jw + nw + 1). \tag{4.126}$$

From Lemma 2.5.2, we know that $0 \le j \le n-1$. It follows that

$$\begin{aligned} & d' - (t'_j + 1) \\ & \le \lceil (2n-1)w \rceil - (n-1)w - nw - 1 \\ & = \lceil (2n-1)w \rceil - (2n-1)w - 1 \\ & < (2n-1)w + 1 - (2n-1)w - 1 \\ & = 0. \end{aligned} \tag{4.127}$$

Note that the algebric inequality $\lceil (2n-1)w \rceil < (2n-1)w + 1$ is used in the derivation of the above formula. Therefore, we have $d' < t'_j + 1$, which implies that the deadline of M' would expire before the transmission of M' completes. Hence, even though M' has a higher priority than message $(b_1, n-1)$, there is not enough time to send M'. Thus, node j does not write its message priority into the token and message M' will be eventually lost.

Therefore, none of other messages can be sent before message $(b_1, n-1)$, which implies that $(b_1, n-1)$ is the first sent message. Its transmission completes at time $t_1 = (2n-1)w + 1 = t_0 + (2n-1)w + 1$. Hence, Lemma holds for $i = 1$.

Suppose the lemma holds up to $i = k - 1$. We now prove the lemma is true for $i = k$. Specifically, we wish to show that message μ_k resides either on node $n_{k-1} - 1$ or on node $n_{k-1} - 2$ and that its deadline is $b_k = \lceil t_{k-1} + (2n-1)w + 1 \rceil$.

As the minimum contention overhead for a message transmission is nw, thus for a message whose deadline d is greater than t_{k-1} but smaller than b_k, it is considered for transmission before b_k only if

$$b_k > d \geq b_{\text{min}} = \lceil t_{k-1} + nw + 1 \rceil. \tag{4.128}$$

That is, a message whose deadline d is smaller than b_k and positioned before b_k is allowed to compete for transmission against b_k only if its deadline d would not expire before its transmission completes; otherwise, there is not sufficient time to complete the message transmission. Hence, it is lost after t_{k-1} and before the transmission of b_k is completed. The number of messages with deadlines satisfying (4.128) is

$$\begin{aligned} 0 \leq b_k - b_{\text{min}} &= \lceil t_{k-1} + (2n-1)w + 1 \rceil - \lceil t_{k-1} + nw + 1 \rceil \\ &\leq \lceil (t_{k-1} + (2n-1)w + 1) - (t_{k-1} + nw + 1) \rceil \\ &= \lceil (n-1)w \rceil \\ &\leq \lceil nw \rceil \\ &= 1, \qquad \text{as } 0 < nw < 1. \end{aligned} \tag{4.129}$$

Because messages in (4.117) have distinct deadlines, the above formula indicates that there may be either zero or one message competing for transmission against b_k. We deal with these two cases separately.

- $b_k - b_{\min} = 0$. In this case, every message with a deadline smaller than b_k has been lost after t_{k-1}. If we let $\mu_k = (b_k, n_{k-1} - 1)$, then the transmission of μ_k completes at $t_k = t_{k-1} + (2n-1)w + 1$. Hence the lemma holds in this case.

- $b_k - b_{\min} = 1$. In this case, there is only one message with a deadline of d $(d < b_k)$ that has not expired by time t_{k-1}. Hence, we have

$$\begin{aligned} d &< b_k \\ &\leq b_k - 1 \\ &= \lceil t_{k-1} + (2n-1)w + 1 \rceil - 1 \\ &= \lceil t_{k-1} + (2n-1)w \rceil. \end{aligned} \tag{4.130}$$

If the message with a deadline of d is on node $n_{k-1} - 1$ and the message with a deadline of b_k is on node $n_{k-1} - 2$, then there is not enough time to send $(d, n_{k-1} - 1)$. This is because

$$\begin{aligned} & t_{k-1} + (2n-1)w + 1 \\ &> \lceil t_{k-1} + (2n-1)w \rceil \\ &= d, \end{aligned} \tag{4.131}$$

which implies that the deadline d of message $(d, n_{k-1} - 1)$ would expire before its transmission completes. On the other hand, message μ_i is sent regardless it is on node $n_{k-1}-1$ or on node $n_{k-1}-2$ because

$$\begin{aligned}
& t_{k-1}+(2n-1)w+1 \\
& \quad \leq \lceil t_{k-1}+(2n-1)w+1 \rceil \\
& \quad = b_k. \\
& t_{k-1}+(2n-2)w+1 \\
& \quad \leq \lceil t_{k-1}+(2n-2)w+1 \rceil \\
& \quad \leq \lceil t_{k-1}+(2n-1)w+1 \rceil \\
& \quad = b_k. \qquad (4.132)
\end{aligned}$$

That is, message μ_k whose deadline is b_k is either on node $n_{k-1}-1$ or on node $n_{k-1}-2$. Its transmission completes at either $t_k = t_{k-1}+(2n-2)w+1$ or $t_k = t_{k-1}+(2n-1)w+1$ as required by the lemma.

This concludes the proof. Q.E.D.

From the above lemma we see that when $PD_{m>D}$ is used to transmit n messages with deadlines of $d_i = i$ $(i = 1, 2, \cdots, n-1, n)$, each transmission incurs an overhead of either $(2n-1)w$ or $(2n-2)w$. Hence, the following lemma can be established.

Lemma 4.3.7 *Given a token ring of n nodes, if $PD_{m>D}$ is used to transmit n messages, in the worst case the lower bound of the contention overhead of a message transmission is $(2n-2)w$.*

Proof: Lemma 4.3.4 and Example 4.2.1 have demonstrated that, given a token ring of n nodes, if $PD_{m>D}$ is used to transmit n messages, the possible lower bounds for the contention overhead are $(2n-2)w, (2n-3)w, \cdots$ and nw respectively. On the other hand, Lemma 4.3.6 show that under $PD_{m>D}$, there exists a message set such that each message transmission incurs an contention overhead of either $(2n-1)w$ or $(2n-2)w$. As we are concerned with the worst case performance ratio, we choose $(2n-2)w$ as the lower bound. This concludes the proof for the lemma. Q.E.D.

Lemma 4.3.8 *Given a token ring of n nodes, if* $PD_{m>D}$ *is used to transmit n messages, in the worst case the upper bound of the number of messages sent is*

$$Send_{up}(PD_{m>D}, N(n,w), A(n)) = \lfloor \frac{n}{(2n-2)w+1} \rfloor. \tag{4.133}$$

Proof: Using $(2n-2)w$ as the lower bound of the contention overhead, with Lemma 4.3.3, we concludes the proof. Q.E.D.

The upper bound of the worst case performance ratio can now be obtained.

Theorem 4.3.2 *An upper bound of the worst case performance ratio of* $PD_{m>D}$ *is*

$$R_{up}(PD_{m>D}, w, n) = \frac{\lfloor \frac{n}{(2n-2)w+1} \rfloor}{n}. \tag{4.134}$$

Similar to Corollary 4.3.2, the upper bound of the worst case performance ratio can be extended to more general cases.

Corollary 4.3.3 *Given a network of n nodes and h* ($h \leq n$) *messages, if* $PD_{m>D}$ *is invoked at time* T_s*, then the upper bound of the worst case performance ratio is*

$$R_{up}(PD_{m>D}, w, n, h, T_s) = \frac{\lfloor \frac{h-T_s}{(2n-2)w+1} \rfloor}{h}. \tag{4.135}$$

4.3.3 An Estimation of $R(PD_{m>D}, w, n)$

Based on the lower and upper bounds of the worst case performance ratio, we propose to use the following to estimate the worst case performance ratio $R(\mathrm{PD}_{m>D}, w, n)$.

$$\begin{aligned} & R_{\text{est}}(\mathrm{PD}_{m>D}, w, n) \\ = & \frac{R_{\text{low}}(\mathrm{PD}_{m>D}, w, n) + R_{\text{up}}(\mathrm{PD}_{m>D}, w, n)}{2} \\ = & \frac{\lfloor \frac{n}{(2n-1)w+1} \rfloor + \lfloor \frac{n}{(2n-2)w+1} \rfloor}{2n}. \end{aligned} \tag{4.136}$$

We now derive the maximum error for the estimation proposed above. For the convenience of our discussion, we let

$$\sigma = \lfloor \frac{n}{(2n-1)w+1} \rfloor. \tag{4.137}$$

It follows that

$$\sigma \leq \frac{n}{(2n-1)w+1} < \sigma + 1. \tag{4.138}$$

We start with deriving the difference between the upper and lower bounds of the number of messages sent by $PD_{m>D}$ in the worst case.

Lemma 4.3.9 *For given n and w, the difference between the lower and upper bound of the number of messages sent by $PD_{m>D}$ is either 1 or 0. That is,*

$$\begin{aligned} Send_{up}(PD_{m>D}, N(n,w), A(n)) &- Send_{low}(PD_{m>D}, N(n,w), A(n)) \\ &= \begin{cases} 1 & w \le \frac{n-(\sigma+1)}{(\sigma+1)(2n-2)}, \\ \\ 0 & otherwise. \end{cases} \end{aligned} \tag{4.139}$$

Proof: We deal with the two cases separately.

- If $w \le \frac{n-(\sigma+1)}{(\sigma+1)(2n-2)}$, then by solving for $\sigma + 1$, we have

$$\sigma + 1 \le \frac{n}{(2n-2)w+1}. \tag{4.140}$$

Multiplying $1 = \frac{(2n-2)w+1}{(2n-2)w+1}$ to the right half of (4.138), we obtain

$$\begin{aligned} \sigma + 1 &> \frac{n}{(2n-1)w+1} \times \frac{(2n-2)w+1}{(2n-2)w+1} \\ &= \frac{n}{(2n-2)w+1} \times \frac{(2n-2)w+1}{(2n-1)w+1} \\ &= \frac{n}{(2n-2)w+1} \times (1 - \frac{w}{(2n-1)w+1}). \end{aligned} \tag{4.141}$$

Rearranging (4.141), we arrive at

$$\frac{n}{(2n-2)w+1} < \sigma + 1 + \frac{n}{(2n-2)w+1} \times \frac{w}{(2n-1)w+1}. \tag{4.142}$$

For $n > 0$ and $w > 0$, we have $0 < \frac{n}{(2n-2)w+1} \le n$ and $0 < \frac{w}{(2n-1)w+1} < w$. It follows that

$$0 < \frac{n}{(2n-2)w+1} \times \frac{w}{(2n-1)w+1} < nw. \tag{4.143}$$

As $nw < 1$, we arrive at

$$0 < \frac{n}{(2n-2)w+1} \times \frac{w}{(2n-1)w+1} < 1. \tag{4.144}$$

Hence, (4.142) becomes

$$\frac{n}{(2n-2)w+1} < \sigma + 2. \tag{4.145}$$

From (4.140) and (4.145), we have

$$\sigma + 1 \leq \frac{n}{(2n-2)w+1} < \sigma + 2. \tag{4.146}$$

With the definition of the floor function $\lfloor\ \rfloor$, the above inequality implies

$$\sigma + 1 = \lfloor \frac{n}{(2n-2)w+1} \rfloor. \tag{4.147}$$

Rearranging the above, we have

$$\lfloor \frac{n}{(2n-2)w+1} \rfloor - \sigma = 1. \tag{4.148}$$

Replacing σ by (4.137) in the above, we arrive at

$$\lfloor \frac{n}{(2n-2)w+1} \rfloor - \lfloor \frac{n}{(2n-1)w+1} \rfloor = 1. \tag{4.149}$$

With (4.114) and (4.133), we have

$$\begin{aligned}\text{Send}_{up}(\text{PD}_{m>D}, N(n,w), A(n)) &- \text{Send}_{low}(\text{PD}_{m>D}, N(n,w), A(n)) \\ &= \lfloor\frac{n}{(2n-2)w+1}\rfloor - \lfloor\frac{n}{(2n-1)w+1}\rfloor \\ &= 1. \end{aligned} \tag{4.150}$$

- If $w > \frac{n-(\sigma+1)}{(\sigma+1)(2n-2)}$, then we have

$$\sigma + 1 > \frac{n}{(2n-2)w+1}. \tag{4.151}$$

However, (4.137) leads to

$$\begin{aligned}\sigma &= \lfloor\frac{n}{(2n-1)w+1}\rfloor \\ &\leq \lfloor\frac{n}{(2n-2)w+1}\rfloor \\ &\leq \frac{n}{(2n-2)w+1}. \end{aligned} \tag{4.152}$$

From (4.151) and (4.152), we have

$$\sigma \leq \frac{n}{(2n-2)w+1} < \sigma + 1. \tag{4.153}$$

With the definition of the floor function $\lfloor\ \rfloor$, the above inequality implies

$$\sigma = \lfloor\frac{n}{(2n-2)w+1}\rfloor. \tag{4.154}$$

With (4.137) and (4.154), we have

$$\begin{aligned}\lfloor\frac{n}{(2n-2)w+1}\rfloor &- \lfloor\frac{n}{(2n-1)w+1}\rfloor \\ &= \sigma-\sigma \\ &= 0. \qquad (4.155)\end{aligned}$$

$$\begin{aligned}\text{Send}_{up}(\text{PD}_{m>D}, N(n,w), A(n)) &- \text{Send}_{low}(\text{PD}_{m>D}, N(n,w), A(n)) \\ &= \lfloor\frac{n}{(2n-2)w+1}\rfloor - \lfloor\frac{n}{(2n-1)w+1}\rfloor \\ &= 0. \qquad (4.156)\end{aligned}$$

In summary of the above two cases, the lemma is proved. Q.E.D.

Theorem 4.3.3 *The difference between the lower bound and upper bound of the worst case performance ratio of $PD_{m>D}$ is*

$$\begin{aligned}&R_{up}(PD_{m>D}, w, n) - R_{low}(PD_{m>D}, w, n) \\ &= \begin{cases}\frac{1}{n} & w \le \frac{n-(\sigma+1)}{(\sigma+1)(2n-2)}, \\ \\ 0 & \textit{otherwise,}\end{cases} \qquad (4.157)\end{aligned}$$

where σ is defined in (4.137).

Proof:

$$\begin{aligned}&R_{up}(\text{PD}_{m>D}, w, n) - R_{low}(\text{PD}_{m>D}, w, n) \\ &= \frac{\lfloor\frac{n}{(2n-2)w+1}\rfloor}{n} - \frac{\lfloor\frac{n}{(2n-1)w+1}\rfloor}{n} \\ &= \frac{\lfloor\frac{n}{(2n-2)w+1}\rfloor - \lfloor\frac{n}{(2n-1)w+1}\rfloor}{n}\end{aligned}$$

$$= \begin{cases} \dfrac{1}{n} & w \leq \frac{n-(\sigma+1)}{(\sigma+1)(2n-2)}, \\ \\ 0 & \text{otherwise.} \end{cases} \tag{4.158}$$

This conclues the proof of the lemma. Q.E.D.

This theorem indicates that the maximum difference between the bounds of the performance ratio of $PD_{m>D}$ is $\frac{1}{n}$, which is a decreasing function of n.

Theorem 4.3.4 *The maximum error of the estimation of the worst case performance ratio of $PD_{m>D}$ is*

$$R_{est}(PD_{m>D}, w, n) - R_{low}(PD_{m>D}, w, n)$$
$$= \begin{cases} \dfrac{1}{2n} & w \leq \frac{n-(\sigma+1)}{(\sigma+1)(2n-2)}, \\ \\ 0 & \textit{otherwise.} \end{cases} \tag{4.159}$$

Proof: With R_{low} and R_{est} given in Theorem 4.3.1 and (4.136), we have

$$\begin{aligned} & R_{\text{est}}(\text{PD}_{m>D}, w, n) - R_{low}(\text{PD}_{m>D}, w, n) \\ &= \frac{\lfloor \frac{n}{(2n-1)w+1} \rfloor + \lfloor \frac{n}{(2n-2)w+1} \rfloor}{2n} - \frac{\lfloor \frac{n}{(2n-1)w+1} \rfloor}{n} \\ &= \frac{\lfloor \frac{n}{(2n-2)w+1} \rfloor - \lfloor \frac{n}{(2n-1)w+1} \rfloor}{2n} \\ &= \begin{cases} \dfrac{1}{2n} & w \leq \frac{n-(\sigma+1)}{(\sigma+1)(2n-2)}, \\ \\ 0 & \text{otherwise.} \end{cases} \end{aligned} \tag{4.160}$$

This concludes the proof. Q.E.D.

4.3.4 Numerical Results and Discussions

We have obtained the worst case performance ratio of the priority-driven protocol when used to transmit messages whose deadlines are smaller than the number of priorities. Theorem 4.3.1 implies that with a sufficient number of priorities, this priority-driven protocol is able to implement the exact EDF policy and that the contention overhead is the only factor degrading the protocol performance. Figure 4.3 shows the result. If the token node-to-node delay w is 0, then $\mathrm{R}(\mathrm{PD}_{m>D}, w, n) = 1$ implying that if working in the ideal environment and having a sufficient number of priorities, the protocol can send all messages, achieving a worst case performance ratio of 1. If the token node-to-node delay w is non-zero, the worst case performance ratio is less than 1. This performance loss is due to the contention overhead, i.e. the time taken for the token to circulate around the ring to locate the highest priority message. Furthermore, we see that the worst case performance ratio is a decreasing function of the token node-to-node delay w.

As the worst case performance ratio obtained is an estimation, we have derived the maximum error of the estimation. In practice, parameter n is normally in the order of 100, the error bound given in (4.160) is thus very small. Figure 4.4 shows the curves of R_{up}, R_{low} and R_{est} vs. n. We see that the difference between R_{up} and R_{low} is invisible in most cases as the maximum difference between R_{low} and R_{up} is $1/n$. This implies that the bounds of the worst case performance ratio obtained are very tight. Consequently, the estimation which is the medium value of the bounds is acceptable.

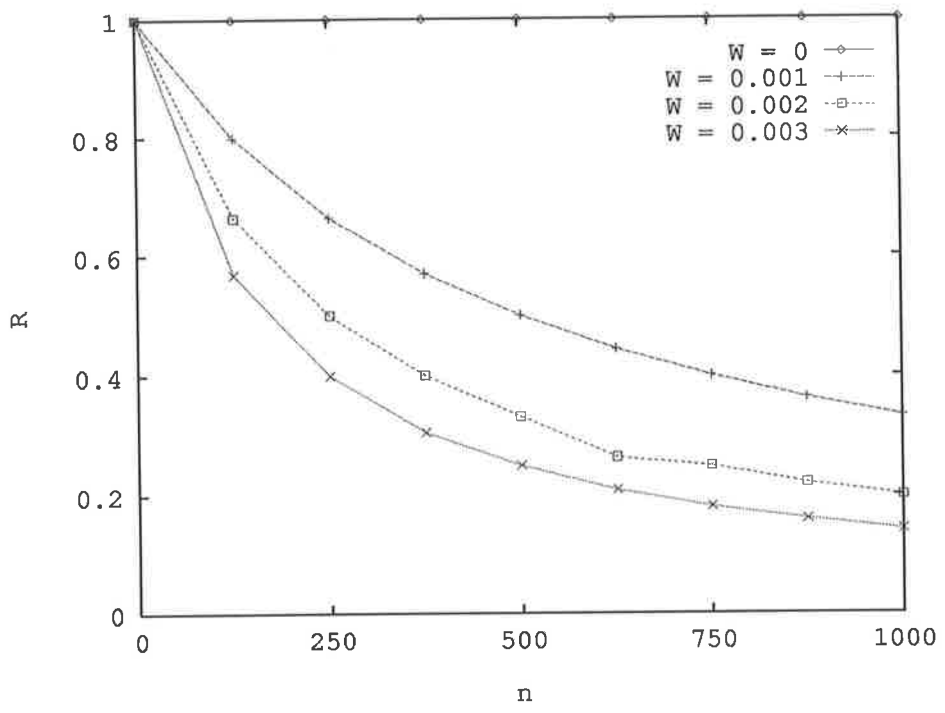


Figure 4.3. Worst Case Performance Ratio of $PD_{m>D}$

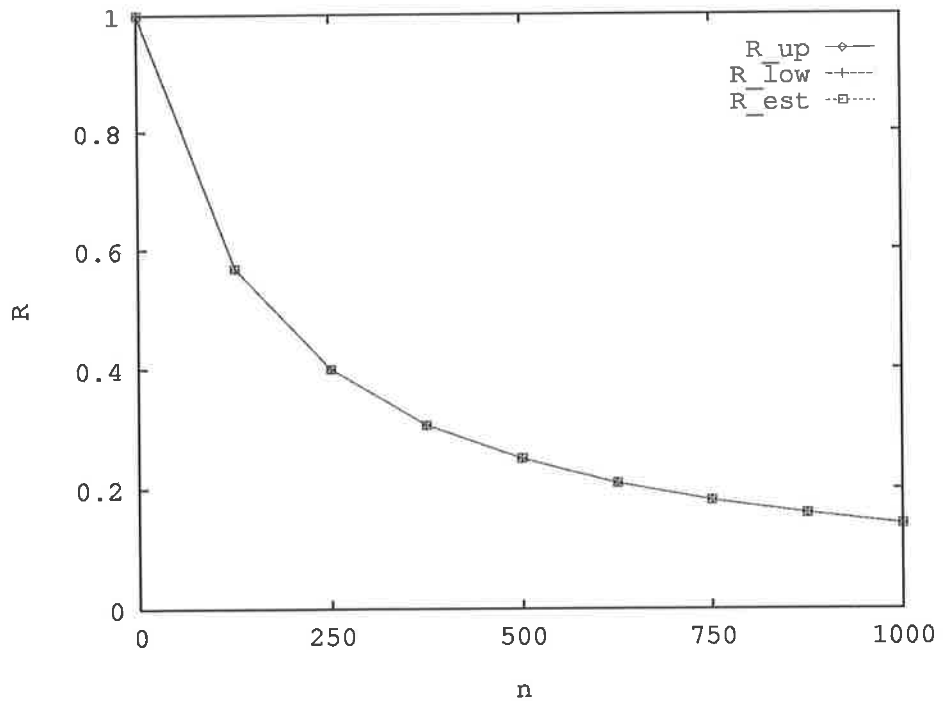


Figure 4.4. Comparison of R_{low}, R_{up} and R_{est}

4.4 Worst Case Performance of $PD_{m\leq d}$

In this section, we analyze the worst case performance of the priority-driven protocol when used to transmit messages whose deadlines are equal to and greater than the number of priorities m.

With the priority assignment function defined in (4.89), messages having deadlines equal to and greater than m are assigned the same priority m regardless of their actual deadlines. As $PD_{m\leq d}$ sends messages in the priority order, it is conceivable that messages with larger deadlines are sent before those having smaller deadlines (since they are of the same priority). As a result, $PD_{m\leq d}$ may not observe the EDF policy.

Lemma 4.4.1 *$PD_{m\leq d}$ is deadline monotonic.*

Proof: The proof of this lemma is similar to that of Lemma 3.2.1, except that under $PD_{m\leq d}$, the contention overhead is nw units of time more than that of the token passing protocol. Hence, replacing the term $t_i + 1$ with $t_i + nw + 1$ in the proof of Lemma 3.2.1, we can have this lemma proved. Q.E.D.

Let $A(n)$ be a message set where message deadlines are equal to and greater than m, we define its subsets Y and Z to consist of these two types of messages respectively. Formally, we have

$$\begin{aligned} A(n) &= Y \cup Z, \qquad \text{where} \\ Y &= \{(d_1,p_1),(d_2,p_2),\cdots,(d_j,p_j)\},\ d_i = m,\ 1 \leq i \leq j < n, \\ Z &= \{(d_1,p_1),(d_2,p_2),\cdots,(d_{n-j},p_{n-j})\},\ d_i > m,\ 1 \leq i \leq n-j. \end{aligned} \tag{4.161}$$

We now identify the message deadlines in the worst case message set.

Lemma 4.4.2 *Given a token ring of n nodes, if $PD_{m\leq d}$ is used to transmit messages from $A(n)$ given in (4.161), message deadlines in the worst case message set are*

$$\begin{aligned} A_{wc} &= Y_{wc} \cup Z_{wc}, \qquad \textit{where} \\ Y_{wc} &= \{d_i | d_i = m\}, \quad 1 \leq i \leq m, \\ Z_{wc} &= \{d_i | d_i = m + i\}, \quad 1 \leq i \leq n - m. \end{aligned} \tag{4.162}$$

That is, subset Y_{wc} consists of m messages that have identical deadlines of m, and subset Z has $n - m$ messages whose deadlines are $m + 1, m + 2, \cdots, n - 1$ and n respectively.

Proof: First, we show that for messages sets defined in (4.161), $A_{wc}(n) = Y_{wc} \cup Z_{wc}$ given in (4.162) is the smallest feasible set.

- To prove that the CEDF protocol can send all the messages from $Y_{wc} \cup Z_{wc}$.
  - As CEDF protocol always sends the earliest deadline message first, messages from Y_{wc} are sent before those from Z_{wc}. It takes the CEDF protocol one unit time to send a message, thus the transmission of m messages from Y_{wc} requires m units of time. It follows that all messages from Y_{wc} are sent as their deadlines m would not expire by time m.
  - It follows that the transmission of messages from Z_{wc} starts at time m. The transmission of the first message from Z_{wc} completes at time $m + 1$, by which time its deadline would not expire, hence it is sent. Similar, it is easy to see that the i-th message from Z_{wc} is sent at time $m + i$. Thus, the transmission of the $(n - m)$-th message, which is the last message from

Z_{wc}, completes at time $m + (n - m) = n$. Hence, all messages from Z_{wc} are also sent under the CEDF protocol.

Therefore, the CEDF protocol can send all the messages from $Y_{wc} \cup Z_{wc}$.

- To prove that the CEDF protocol can not send all the messages from $Y_{wc} \cup Z_{wc}$ if any message deadline is reduced.

  As we are dealing with the case of $PD_{m\leq d}$, thus no message deadlines are smaller than m. Hence, no message deadlines in Y can be reduced.

  - Suppose the deadline of the i-th message from Z_{wc} is reduced from $m+i$ to m. Thus there are now $m+1$ messages have deadlines of m. It is obvious that under the CEDF protocol the transmission of $m+1$ messages requires $m+1$ units of time, but the message deadline m would expire by time m. Therefore, only m messages are sent and one message is lost.
  - Suppose the deadline of the i-th message in Z_{wc} is reduced from $m + i$ to $m + j$, where $j < i$. Under the CEDF protocol, the transmission of m messages from Y_{wc} completes at time m. Hence, the transmission of the i-th message from Z_{wc} completes at time $m + i$, by which time its deadline $m + j$ would expire. Consequently the message is lost.

  The above demonstrates that if any message deadline in $Y_{wc} \cup Z_{wc}$ is reduced even the CEDF protocol can not send all messages, i.e. the new message set is not feasible.

In summary of the above, with Definition 2.12 the message set given in (4.4.2) is the smallest feasible set. Furthermore, as $PD_{m\leq d}$ is deadline monotonic, with Lemma 2.5.1 the message set given in (4.4.2) is the worst case set. Q.E.D.

We now examine how messages given in (4.162) are sent by $PD_{m\leq d}$ in the worst case.

Lemma 4.4.3 *Given a token ring of n nodes, if $PD_{m\leq d}$ is used to transmit messages from the worst case message set given in (4.162), it sends messages in the latest-deadline-first order and each message transmission incurs a contention overhead of $(n+1)w$.*

Proof: As all messages are assigned the same priority m under $PD_{m\leq d}$, they are sent in the order in which the token is captured.

As $PD_{m\leq d}$ is deadline monotonic, thus it behaves the same as the token passing protocol but with a different contention overhead. Replacing $x+1$ with $x+nw+1$ in the proof of Lemma 3.2.2 where x is t_i, t'_i, t_{i+1} or t'_{i+1}, we conclude that given a message set, if the deadlines of two messages on the neighboring nodes are interchanged so that the token visits the node with the smaller deadline message first, then the number of messages sent by $PD_{m\leq d}$ may increase. As a result, in the worst case $PD_{m\leq d}$ always sends the latest deadline message first.

For each message transmission, it takes w units of time for the token to move from the current sending node to the next node where the priority of the message is written into the token priority field. After another nw units of time, i.e. when the token completes a full circulation, the message is sent. Hence, the overhead for a message transmission is $(n+1)w$. Q.E.D.

With the worst case message set and the contention overhead given in Lemmas 4.162 and 4.4.3 respectively, it is straightforward to compute the number of messages sent in the worst case.

Lemma 4.4.4 *For the worst case message set given in (4.162), let S_Y and S_Z be the number of messages sent from subsets Y_{wc} and Z_{wc} respectively, we have*

$$S_Y = \begin{cases} 0 & m < (n-m+1)((n+1)w+1)), \\ \\ \lfloor \frac{m-(n-m)\times((n+1)w+1)}{(n+1)w+1} \rfloor & otherwise. \end{cases} \tag{4.163}$$

$$S_Z = min(n-m, \lfloor \frac{n+1}{(n+1)w+2} \rfloor) \tag{4.164}$$

$$\begin{aligned} & Send(PD_{m\leq d}, N(n,w), A(n)) \\ &= S_Y + S_Z \\ &= \begin{cases} min(n-m, \lfloor \frac{n+1}{(n+1)w+2} \rfloor) & m < (n-m+1)((n+1)w+1), \\ \\ \lfloor \frac{m}{(n+1)w+1} \rfloor & otherwise. \end{cases} \end{aligned} \tag{4.165}$$

Proof: Lemma 4.4.3 states that in the worst case $PD_{m\leq d}$ sends messages in the latest-deadline-first order, hence messages in Z are sent before those in Y because message deadlines in the former are greater than those in the latter.

- Message transmission from Z

  As messages in Z are sent in the latest-deadline-first order with a contention overhead of $(n+1)w$, with Theorem 3.3.1 the maximum number of messages sent from Z is $\lfloor \frac{n+1}{(n+1)w+2} \rfloor$, which can be obtained by replacing w with $(n+1)w$ in (3.74) in the proof of Theorem 3.3.1, On the other hand, the total number of messages in Z is $n-m$. Hence, we have

$$S_Z = \min(n-m, \lfloor\frac{n+1}{(n+1)w+2}\rfloor). \tag{4.166}$$

- Message transmission from Y

  Clearly, if any messages are sent from Y, all messages in Z must have been sent. This is because message deadlines in Z are larger than those in Y and $PD_{m\leq d}$ sends messages in the latest-deadline-first order. Thus, we have

$$S_Z = n - m. \tag{4.167}$$

  Transmission of messages from Z complete at $S_Z \times ((n+1)w+1)$.

  - No messages from Y are sent if

$$\begin{aligned} m &< S_Z \times ((n+1)w+1) + ((n+1)w+1) \\ &= (n-m+1)((n+1)w+1). \end{aligned} \tag{4.168}$$

    This is because by the time the transmission of messages from Z completes, the message deadlines m in Y would expire. Hence, all the messages in Y are lost, implying $S_Y = 0$.

  - Otherwise, all $n-m$ messages from Z are sent. The transmission of messages from Y starts at

$$\begin{aligned} T_s &= S_Z \times ((n+1)w+1) \\ &= (n-m) \times ((n+1)w+1). \end{aligned} \tag{4.169}$$

As all messages in Y have the same deadlines, sending them in the latest-deadline-first order is equivalent to sending them in the EDF order. With Corollary 4.3.1, we have

$$\begin{aligned} S_Y &= \lfloor\frac{m-T_s}{(n+1)w+1}\rfloor \\ &= \lfloor\frac{m-(n-m)\times((n+1)w+1)}{(n+1)w+1}\rfloor. \end{aligned} \tag{4.170}$$

In summary, $Send(PD_{m\leq d}, N(n,w), A(n)$ have two possible values.

- Only messages from Z are sent

$$\begin{aligned} &Send(PD_{m\leq d}, N(n,w), A(n) \\ &= S_Z + S_Y \\ &= \min(n-m, \lfloor\frac{n+1}{(n+1)w+2}\rfloor) + 0 \\ &= \min(n-m, \lfloor\frac{n+1}{(n+1)w+2}\rfloor). \end{aligned} \tag{4.171}$$

- All messages from Z are sent, followed by some from Y

$$\begin{aligned} &Send(PD_{m\leq d}, N(n,w), A(n) \\ &= S_Z + S_Y \\ &= (n-m) + \lfloor\frac{m-(n-m)\times((n+1)w+1)}{(n+1)w+1}\rfloor \\ &= \lfloor\frac{(n-m)\times((n+1)w+1)+m-(n-m)\times((n+1)w+1)}{(n+1)w+1}\rfloor \\ &= \lfloor\frac{m}{(n+1)w+1}\rfloor. \end{aligned} \tag{4.172}$$

This concludes the proof. Q.E.D.

4.4.1 Worst Case Performance Ratio $R(PD_{m\leq d}, w, n)$

From Lemma 4.4.4, the following is straightforward.

Theorem 4.4.1 *The worst case performance ratio of $PD_{m\leq d}$ is*

$$R(PD_{m\leq d}, w, n) = \begin{cases} \frac{min(n-m, \lfloor \frac{n+1}{(n+1)w+2} \rfloor)}{n} & m < (n-m+1)((n+1)w+1), \\ \\ \frac{\lfloor \frac{m}{(n+1)w+1} \rfloor}{n} & otherwise. \end{cases} \tag{4.173}$$

4.4.2 Numerical Results and Discussions

We have derived the the worst case performance ratio of the priority-driven protocol when used to transmit messages whose deadlines are equal to and greater than the number of priorities m. Theorem 4.4.1 indicates that the worst case performance ratio of $PD_{m\leq d}$ is less than 1. The following aspects contribute to the performance loss.

- Under $PD_{m\leq d}$, messages with different deadlines are assigned the same priority due to insufficient number of priorities. Our analysis has shown that in the worst case messages are sent in the latest-deadline-first order, which may lead to a low worst case performance ratio. To demonstrate this, we consider the worst case performance ratio given in Theorem 4.4.1 when $w = 0$,

$$R(PD_{m\leq d}, w, n) = \begin{cases} \frac{\min(n-m, \lfloor \frac{n+1}{2} \rfloor)}{n} & m < n-m+1, \\ \\ \frac{m}{n} & \text{otherwise.} \end{cases} \tag{4.174}$$

Furthermore, if

$$m < n - m + 1 \quad \text{and} \quad n - m > \lfloor \frac{n+1}{2} \rfloor, \tag{4.175}$$

then (4.174) becomes

$$R(PD_{m \leq d}, w, n) = \begin{cases} \frac{\lfloor \frac{n+1}{2} \rfloor}{n} & m < \frac{n+1}{2}, \\ \\ \frac{m}{n} & \text{otherwise.} \end{cases} \tag{4.176}$$

If n is sufficiently large such that $\frac{1}{n} \approx 0$, then we have

$$R(PD_{m \leq d}, w, n) \approx \begin{cases} \frac{1}{2} & m < \frac{n+1}{2}, \\ \\ \frac{m}{n} & \text{otherwise.} \end{cases} \tag{4.177}$$

This implies that if the contention overhead of each message transmission is zero, the limiting value of the worst case performance is determined by the number of message deadlines n and the number of priorities m.

- If $m < \frac{n+1}{2}$, then $PD_{m \leq d}$ can send only half of the messages sent by the CEDF protocol. This is because in this case $PD_{m \leq d}$ sends messages in the latest-deadline-first order since messages with distinct deadlines have the same priority. This is solely due to insufficient number of priorities.
- Otherwise, the worst case performance ratio is $\frac{m}{n}$, which is an increasing function of m. Figure 4.5 demonstrates the effect of m on the worst case performance ratio. We observe that for a given m, as n increases the worst

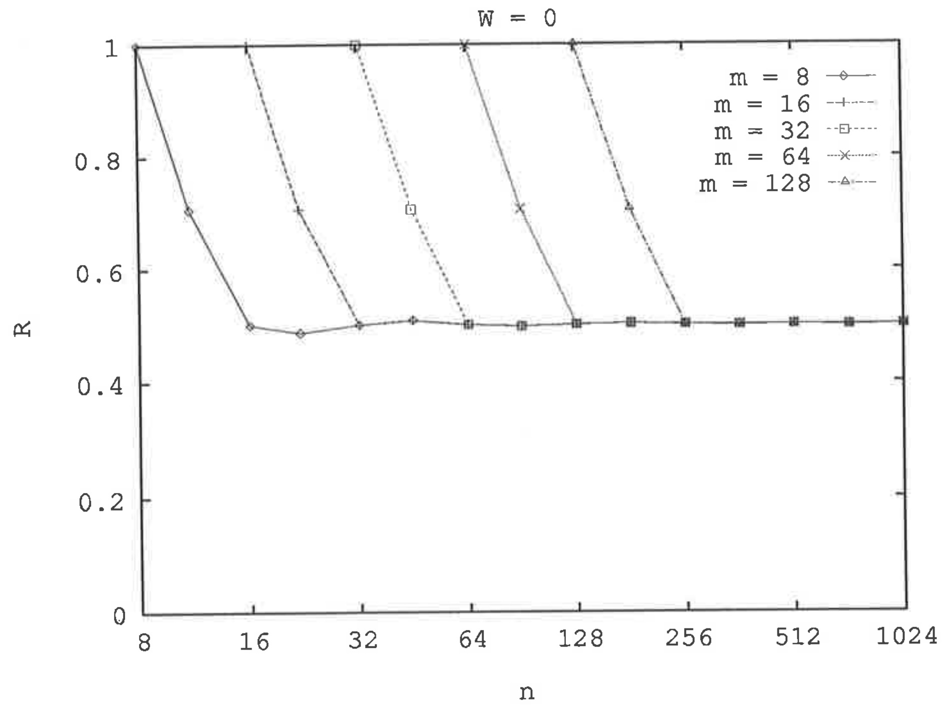


Figure 4.5. Effect of the Number of Priorities

case performance ratio decreases and eventually approaches 0.5. Hence, the number of priorities is one of the dominant factors that determines the worst case performance of $PD_{m\leq d}$.

- The contention overhead is also responsible for degrading the performance of $PD_{m\leq d}$. Figure 4.6 shows the result. We see that the worst case performance ratio is a decreasing function of the token node-to-node delay w. We also notice that when $n \geq m = 64$ the performance ratio is always less than 0.5 for $w > 0$. This is due to both the transmission policy and the contention overhead incurred.

In summary, if the number of message deadlines is large and the number of priorities available (determined by the number of bits used for the priority field in the token) is small, then the worst case performance of $PD_{m\leq d}$ is poor.

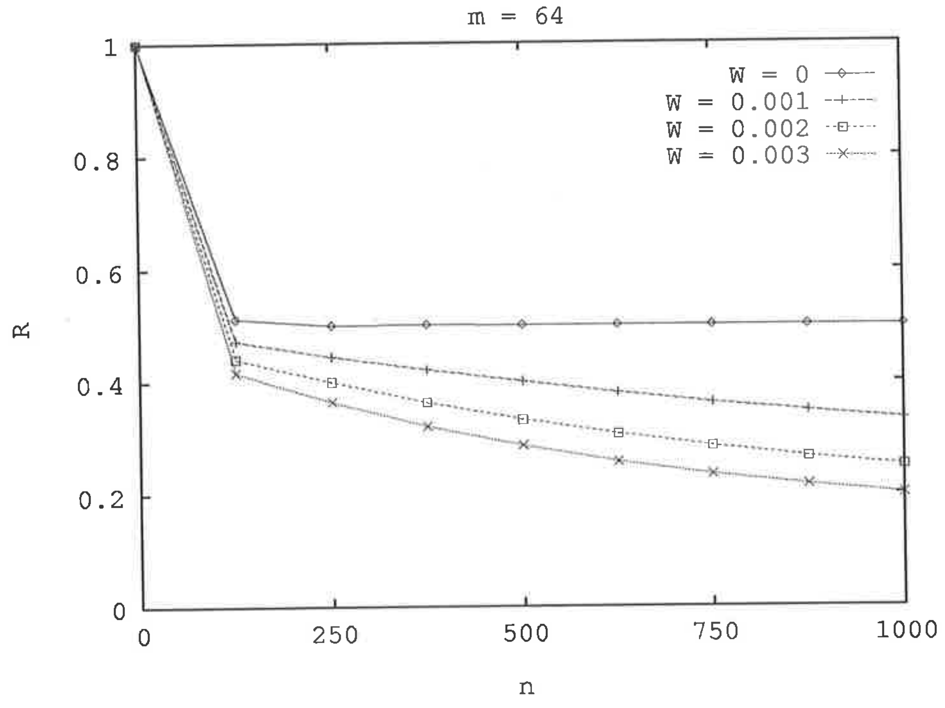


Figure 4.6. Worst Case Performance ratio of $PD_{m \leq d}$

4.5 Worst Case Performance of $PD_{d<m<D}$

In this section, we study the worst case performance of the priority-driven protocol when used to transmit messages whose deadlines are smaller than, equal to and greater than the number of priorities m.

To facilitate our analysis, we group the messages into three subsets according to their deadlines. We then derive the worst case performance ratio for each subset using the results obtained for $PD_{m>D}$ and $PD_{m \leq d}$. Finally, the worst case performance ratio of $PD_{d<m<D}$ is computed by summing up the worst case performance of each subset.

For given n and m, let $A(n)$ be a message set of size n and its message deadlines are smaller than, equal to and greater than m. Let $A^k(n)$ be a subset of $A(n)$, which has k $(0 \leq k \leq m)$ messages with deadlines smaller than m. Consequently, all message sets $\{A(n)\}$ of size n can be seen as a union of subsets $A^k(n)$ with k ranging from 0 to $m-1$. That is

$$\{A(n)\} = \{A^0(n)\} \cup \{A^1(n)\} \cup \cdots \cup \{A^{m-1}(n)\}. \tag{4.178}$$

Using a notation similar to that defined in (4.161), we have

$$\begin{aligned} A^k(n) &= X^k \cup Y^k \cup Z^k, \qquad \text{where} \\ X^k &= \{(d_1,p_1),(d_2,p_2),\cdots,(d_k,p_k)\},\ d_i < m,\ 1 \le i \le k, \\ Y^k &= \{(d_1,p_1),(d_2,p_2),\cdots,(d_j,p_j)\},\ d_i = m,\ 1 \le i \le j, \\ Z^k &= \{(d_1,p_1),(d_2,p_2),\cdots,(d_{n-k-j},p_{n-k-j})\},\ d_i > m,\ 1 \le i \le n-k-j. \end{aligned} \tag{4.179}$$

That is, messages are grouped into three subsets X^k, Y^k and Z^k, in which message deadlines are smaller than, equal to and greater than m respectively[2]. With Definition 2.6 given in Chapter 2, for w, n, m and k, the worst case performance ratio of $PD_{d<m<D}$ is

$$R(PD_{d<m<D}, N(n,w), n, k) = \min_{\forall A^k(n)} \left(\frac{\text{Send}(PD_{m<n}, N(n,w), A^k(n))}{n}\right), \tag{4.180}$$

where $A^k(n)$ is any message set of size n, which has k message whose deadlines are smaller than m. It follows that for given w, n and m, the worst case performance ratio of $PD_{d<m<D}$ is

[2]It is obvious that when $n = k$ both Y^k and Z^k are empty, hence $PD_{d<m<D}$ is equivalent to $PD_{m>D}$. When $k = 0$, X^k is empty, thus $PD_{d<m<D}$ is equivalent to $PD_{m\le d}$.

$$\begin{aligned} R(PD_{d<m<D}, w, n) &= \min_{\forall k \forall A^k(n)} \left(\frac{\text{Send}(PD_{m<n}, N(n,w), A^k(n))}{n}\right) \\ &= \min_{\forall k}(R(PD_{d<m<D}, N(n,w), n, k)). \end{aligned} \tag{4.181}$$

Therefore, to derive the worst case performance ratio of $R(PD_{d<m<D}, w, n)$, we first derive the bounds for the performance ratio of $R(PD_{d<m<D}, w, n, k)$.

4.5.1 Properties of $PD_{m\leq d}$

To facilitate the worst case analysis, we first outline some properties of $PD_{d<m<D}$ when used to transmit messages from $A^k(n)$.

Lemma 4.5.1 *When $PD_{d<m<D}$ is used to transmit messages from $A^k(n)$ defined in (4.179), in the worst case*

- *messages in X^k are transmitted first, followed by those from Z^k and Y^k respectively, and*
- *messages in X^k are sent in the EDF order and those in Y^k and Z^k are sent in the latest-deadline-first order.*

Proof: As message deadlines in X^k are smaller than m while those in Y^k and Z^k are equal to and greater than m respectively, with the priority assignment function defined in (4.89), messages in X^k are assigned priorities from 1 to $m-1$ and those in Y and Z are assigned the priority of m. As $PD_{d<m<D}$ always sends the highest-priority message first, thus messages in X^k are sent before those from Y^k and Z^k.

As all message deadlines in X^k are smaller than m, from Lemma 4.3.4 messages in X^k are sent in the EDF order. As message deadlines in Y^k and Z are equal to and greater than m, it follows from Lemma 4.4.3 that they are sent in the latest-deadline-first order. Q.E.D.

We now identify message deadlines in the worst case message set.

Lemma 4.5.2 *When $PD_{d<m<D}$ is used to transmit messages from $A^k(n)$ defined in (4.179), message deadlines in the worst case message set are*

$$\begin{aligned}
A_{wc}^k(n) &= X_{wc}^k \cup Y_{wc}^k \cup Z_{wc}^k, \qquad \textit{where} \\
X_{wc}^k &= \{d_i | d_i \geq i \textit{ and } d_j = k\}, \quad 1 \leq i \leq k-1 \textit{ and } j = k, \\
Y_{wc}^k &= \{d_i | d_i = m\}, \quad 1 \leq i \leq m-k, \\
Z_{wc}^k &= \{d_i | d_i = m+i\}, \quad 1 \leq i \leq n-m. \qquad (4.182)
\end{aligned}$$

Proof: For message sets consisting of k messages, it follows from Lemma 4.3.2 that the one with a maximum deadline of k minimizes the number of messages sent. Hence, X_{wc}^k given in (4.182) is the worst case set.

Given that there are k messages whose deadlines are smaller than m and the largest deadline of these k messages is k, Lemma 4.4.2 implies that for the other $n-k$ messages whose deadlines are equal to and greater than m, there must be $m-k$ messages with deadlines equal to m, which is indicated by Y_{wc}^k in (4.182). The number of the remaining messages is

$$(n-k)-(m-k) = n-m, \qquad (4.183)$$

and their deadlines are $d_i = m+i$ $(1 \leq i \leq n-m)$, which is denoted by Z_{wc}^k in (4.182). This concludes the proof. Q.E.D.

4.5.2 Performance bounds for $A^k(n)$

To facilitate the description, we denote $\text{Send}_{low}(PD_{d<m<D}, N(n,w), U)$ and $\text{Send}_{up}(PD_{d<m<D}, N(n,w), U)$ as the lower and upper bound of the number of messages sent from message set U respectively, where U is X^k_{wc}, Y^k_{wc} or Z^k_{wc}. Clearly,

$$\begin{aligned}
&\text{Send}_{low}(PD_{d<m<D}, N(n,w), A^k_{wc}(n)) \\
&= \text{Send}_{low}(PD_{d<m<D}, N(n,w), X^k_{wc} \cup Y^k_{wc} \cup Z^k_{wc}) \\
&= \text{Send}_{low}(PD_{d<m<D}, N(n,w), X^k_{wc}) + \text{Send}_{low}(PD_{d<m<D}, N(n,w), Y^k_{wc}) + \\
&\quad \text{Send}_{low}(PD_{d<m<D}, N(n,w), Z^k_{wc}), \qquad (4.184)
\end{aligned}$$

$$\begin{aligned}
&\text{Send}_{up}(PD_{d<m<D}, N(n,w), A^k_{wc}(n)) \\
&= \text{Send}_{up}(PD_{d<m<D}, N(n,w), X^k_{wc} \cup Y^k_{wc} \cup Z^k_{wc}) \\
&= \text{Send}_{up}(PD_{d<m<D}, N(n,w), X^k_{wc}) + \text{Send}_{up}(PD_{d<m<D}, N(n,w), Y^k_{wc}) + \\
&\quad \text{Send}_{up}(PD_{d<m<D}, N(n,w), Z^k_{wc}). \qquad (4.185)
\end{aligned}$$

To derive the bounds for the number of messages sent from $A^k_{wc}(n)$ in the worst case, we need to compute the bounds of the number of message sent from X^k_{wc}, Y^k_{wc} and Z^k_{wc} respectively.

Performance Bounds of Subset X^k_{wc}

Lemma 4.5.3 *Given a token ring of n nodes, in the worst case the bounds of the number of messages sent from X^k_{wc} are given by*

$$Send_{low}(PD_{d<m<D}, N(n,w), X^k_{wc}) = \lfloor \frac{k}{(2n-1)w+1} \rfloor, \tag{4.186}$$

$$Send_{up}(PD_{d<m<D}, N(n,w), X^k_{wc}) = \lfloor \frac{k}{(2n-2)w+1} \rfloor. \tag{4.187}$$

Proof: From Lemma 4.5.1, we know that messages in X^k_{wc} are sent in the EDF order. With Lemmas 4.3.4 and 4.3.7, in the worst case each message transmission from X^k_{wc} incurs a maximum and minimum contention overhead of $(2n-1)w$ and $(2n-2)w$ respectively. Thus with Lemma 4.3.3, the lemma is proved. Q.E.D.

Performance Bounds of Subset Z^k_{wc}

Let T^U denote the time when the transmission of the first message from set U starts, where U is Y^k_{wc} or Z^k_{wc}. Furthermore, let T^U_{up} and T^U_{low} denote the upper and lower bound of T^U respectively. It is apparent that T^U is one of the factors determining the number of messages sent from U.

As messages from Z^k_{wc} are sent before those in Y^k_{wc} in the worst case, we consider subset Z^k_{wc} first and derive the bounds for T^Z. Consequently, the bounds of the number of messages sent from Z^k_{wc} can be determined.

Lemma 4.5.4 *The bounds of T^Z are*

$$T^Z_{low} = \lfloor \frac{k}{(2n-1)w+1} \rfloor ((2n-2)w+1) + (n+1)w, \tag{4.188}$$

$$T^Z_{up} = \lfloor \frac{k}{(2n-2)w+1} \rfloor ((2n-1)w+1) + (2n-1)w. \tag{4.189}$$

Proof: For the lower bound T^Z, the right sides of (4.189) consists of the following three items.

- First, $\lfloor\frac{k}{(2n-1)w+1}\rfloor((2n-2)w+1)$ is the lower bound of the time when the last message transmission from X^k_{wc} completes. It is the product of the lower bound of the number of messages sent from X^k_{wc} and the minimum time needed for a message transmission including contention overhead. With Lemmas 4.3.7 and 4.5.3, it is $\lfloor\frac{k}{(2n-1)w+1}\rfloor((2n-2)w+1)$.

- Second, w is the minimum time needed for the token to reach the node having the first message from Z^k_{wc}. As all messages in Z^k_{wc} have the same priority of m, this first node is also the first sending node.

- Third, nw is the time taken for the token to complete a full circulation after the first node in Z^k_{wc} writes its message priority into the token.

Hence, the lower bound of the time when the first message transmission from Z^k_{wc} starts is obtained by summing up the above three items, i.e.

$$\lfloor\frac{k}{(2n-2)w+1}\rfloor((2n-1)w+1)+w+nw, \tag{4.190}$$

which leads to (4.189).

For the upper bound of T^Z, $\lfloor\frac{k}{(2n-2)w+1}\rfloor((2n-1)w+1)$ is the upper bound of the time when the transmission of messages from X^k_{wc} completes. $(n-1)w$ is the maximum time needed for the token to reach the first node in Z^k_{wc}. nw is the time needed for the token to complete a full rotation before the node can start its message transmission. Therefore, we have

$$\lfloor\frac{k}{(2n-2)w+1}\rfloor((2n-1)w+1)+(n-1)w+nw, \tag{4.191}$$

which yields (4.5.4). This concludes the proof. Q.E.D.

Using the bounds of T^Z, we now compute the bounds of the number of messages sent from Z^k_{wc} in the worst case.

Lemma 4.5.5 *Given a token ring of n nodes, in the worst case the bounds of the number of messages sent from Z^k_{wc} are given by*

$$\begin{aligned} &Send_{low}(PD_{d<m<D}, N(n,w), Z^k_{wc}) \\ &= \begin{cases} 0 & n < T^Z_{up}+1, \\ \min(n-m, \lfloor\frac{n+1+(n+1)w-T^Z_{up}}{(n+1)w+2}\rfloor) & \textit{otherwise.} \end{cases} \end{aligned} \tag{4.192}$$

$$\begin{aligned} &Send_{up}(PD_{d<m<D}, N(n,w), Z^k_{wc}) \\ &= \begin{cases} 0 & n < T^Z_{low}+1, \\ \min(n-m, \lfloor\frac{n+1+(n+1)w-T^Z_{low}}{(n+1)w+2}\rfloor) & \textit{otherwise.} \end{cases} \end{aligned} \tag{4.193}$$

where T^Z_{up} and T^Z_{low} are given in Lemma 4.5.4.

Proof: We only prove for the upper bound. The lower bound can be obtained with a similar approach.

From Lemma 4.5.4, we see that the first message transmission from Z^k_{wc} starts at time T^Z_{low}. From Lemmas 4.5.1 and 4.5.2, messages in Z^k_{wc} are transmitted in the latest-deadline-first order and the first message sent from Z^k_{wc} has a deadline of n. We have two cases to consider.

- If $n < T^Z_{low} + 1$, there is not enough time from time T^Z_{low} to send the first message from Z^k_{wc}, which has a deadline of n. Consequently, no other messages in Z^k_{wc} are sent as deadlines of all other messages in Z^k_{wc} are smaller than m.

- Otherwise, the message with a deadline of n is sent. From Lemma 4.4.3, each subsequent message transmission from Z^k_{wc} takes $(n+1)w+1$ units of time and messages are sent in the latest-deadline-first order, which is the same as the token passing protocol. Note that after the first message is transmitted from Z^k_{wc}, there are $n-m-1$ remaining messages, of which the maximum deadline is $n-1$. With Corollary 3.3.2, the number of messages sent from the remaining $n-m-1$ messages in Z^k_{wc} is

$$\min(n-m-1, \lfloor\frac{(n-1)+1-(T^Z_{low}+1)}{(n+1)w+2}\rfloor), \tag{4.194}$$

where $(T^Z_{low}+1)$ is the time when the transmission of the first message in Z^k_{wc} completes. Hence, the total number of messages sent from Z^k_{wc} is

$$\begin{aligned}
&\text{Send}_{low}(PD_{d<m<D}, N(n,w), Z^k_{wc}) \\
&= 1+\min(n-m-1, \lfloor\frac{(n-1)+1-(T^Z_{low}+1)}{(n+1)w+2}\rfloor) \\
&= \min(n-m, \lfloor 1+\frac{(n-1)+1-(T^Z_{low}+1)}{(n+1)w+2}\rfloor) \\
&= \min(n-m, \lfloor\frac{n+1+(n+1)w-T^Z_{low}}{(n+1)w+2}\rfloor).
\end{aligned} \tag{4.195}$$

Note that the algebraic equality $C + \min(A, B) = \min(A + C, B + C)$ is used in the derivation of the above formula.

This concludes the proof of the lemma. Q.E.D.

Performance Bounds of Subset Y_{wc}^k

We now derive the performance bounds associated with subset Y_{wc}^k.

Lemma 4.5.6 *If any message(s) from Y_{wc}^k is sent, the bounds of T^Y are*

$$T_{low}^Y = T_{low}^Z + (n-m)((n+1)w+1), \tag{4.196}$$

$$T_{up}^Y = T_{up}^Z + (n-m)((n+1)w+1). \tag{4.197}$$

Proof: We only prove for the upper bound. The lower bound can be obtained in a similar way.

If any message(s) from Y_{wc}^k is sent, all $n-m$ messages from Z_{wc}^k must have been sent. This is because messages in Y_{wc}^k have smaller deadlines than those in Z_{wc}^k, and with Lemma 4.4.3, messages in Z_{wc}^k are sent before those in Y_{wc}^k in the latest-deadline-first order.

As pointed out in the proof of Lemma 4.5.5, the upper bound of the time when the first message transmission from Z_{wc}^k completes is

$$t_1 = T_{up}^Z + 1. \tag{4.198}$$

There are $n-m-1$ messages remaining from Z^k_{wc}. With Lemma 4.4.3, each message transmission from Z^k_{wc} incurs a contention overhead of $(n+1)w$. Thus, it requires

$$t_2 = (n-m-1)((n+1)w+1) \tag{4.199}$$

units of time to transmit the remaining messages. Then, it takes another

$$t_3 = (n+1)w \tag{4.200}$$

units of time for the token to reach the first node in Y^k_{wc} and to complete a full circulation round the ring before it can start its message transmission. With (4.198), (4.199) and (4.200), the upper bound of the time when the first message transmission from Y^k_{wc} starts at

$$\begin{aligned} T^Y_{up} &= t_1 + t_2 + t_3 \\ &= T^Z_{up} + 1 + (n-m-1)((n+1)w+1) + (n+1)w \\ &= T^Z_{up} + (n-m)((n+1)w+1). \end{aligned} \tag{4.201}$$

This conclude the proof. Q.E.D.

Now the number of messages sent from Y^k_{wc} in the worst case can be determined.

Lemma 4.5.7 *Given a token ring of n nodes, in the worst case the bounds of the number of messages sent from Y^k_{wc} are given by*

$$Send_{low}(PD_{d<m<D}, N(n,w), Y^k_{wc})$$
$$= \begin{cases} 0 & m < T^Y_{up} + 1, \\ \\ \lfloor \frac{m+(n+1)w-T^Y_{up}}{(n+1))w+1} \rfloor & otherwise, \end{cases} \tag{4.202}$$

$$Send_{up}(PD_{d<m<D}, N(n,w), Y^k_{wc})$$
$$= \begin{cases} 0 & m < T^Y_{low} + 1, \\ \\ \lfloor \frac{m+(n+1)w-T^Y_{low}}{(n+1)w+1} \rfloor & otherwise. \end{cases} \tag{4.203}$$

where T^Y_{up} and T^Y_{low} are given in Lemma 4.5.6.

Proof: Again we only prove for the upper bound. As the first message transmission from Y^k_{wc} starts at T^Y_{low}, it is clear that if $m < T^Y_{low} + 1$, then there is not enough time to send any message from Y^k_{wc} since its message deadlines m would expire before the message transmission completes; if $m \geq T^Y_{low} + 1$, then the first message transmission from Y^k_{wc} completes at $T^Y_{low} + 1$. With Lemma 4.4.3, we know that each subsequent message transmission from Y^k_{wc} takes $(n+1)w+1$ units of time. Furthermore, the latest time when the message transmission must stop is m, as all message deadlines in Y^k_{wc} are equal to m. Thus, with Corollary 4.3.1, the upper bound of the number of message sent from Y^k_{wc} is

$$\begin{aligned} 1 + \lfloor \frac{m-(T^Y_{up}+1)}{(n+1)w+1} \rfloor &= \lfloor 1 + \frac{m-(T^Y_{up}+1)}{(n+1)w+1} \rfloor \\ &= \lfloor \frac{m+(n+1)w-T^Y_{up}}{(n+1)w+1} \rfloor. \end{aligned} \tag{4.204}$$

This concludes the proof. Q.E.D.

Performance Bounds of $A^k(n) = X^k_{wc} \cup Y^k_{wc} \cup Z^k_{wc}$

Combining the results from Lemmas 4.5.3, 4.5.5 and 4.5.7, we obtain the bounds for the total number of messages sent from $A^k_{wc} = X^k_{wc} \cup Y^k_{wc} \cup Z^k_{wc}$ as follows.

Lemma 4.5.8 *Given a token ring of n nodes, the bounds of the total number of messages sent from $A^k_{wc}(n)$ are given by*

$$Send_{low}(PD_{d<m<D}, N(n,w), A^k_{wc}(n))$$
$$= \begin{cases} \lfloor \frac{k}{(2n-1)w+1} \rfloor & n < T^Z_{up}+1, \\ \lfloor \frac{k}{(2n-1)w+1} \rfloor + \min(n-m, \lfloor \frac{n+1+(n+1)w-T^Z_{up}}{(n+1)w+2} \rfloor) & n \geq T^Z_{up}+1 \text{ and} \\ & m < T^Y_{up}+1, \\ \lfloor \frac{k}{(2n-1)w+1} \rfloor + \lfloor \frac{m+(n+1)w-T^Z_{up}}{(n+1)w+1} \rfloor & \text{otherwise,} \end{cases} \quad (4.205)$$

$$Send_{up}(PD_{d<m<D}, N(n,w), A^k_{wc}(n))$$
$$= \begin{cases} \lfloor \frac{k}{(2n-2)w+1} \rfloor & n < T^Z_{low}+1, \\ \lfloor \frac{k}{(2n-2)w+1} \rfloor + \min(n-m, \lfloor \frac{n+1+(n+1)w-T^Z_{low}}{(n+1)w+2} \rfloor) & n \geq T^Z_{low}+1 \text{ and} \\ & m < T^Y_{low}+1 \\ \lfloor \frac{k}{(2n-2)w+1} \rfloor + \lfloor \frac{m+(n+1)w-T^Z_{low}}{(n+1)w+1} \rfloor & \text{otherwise.} \end{cases} \quad (4.206)$$

where T^Z_{up}, T^Z_{low}, T^Y_{up} and T^Y_{low} are given in Lemmas 4.5.4 and 4.5.6 respectively.

Proof: We only prove for the upper bound. The lower bound can be derived in a similar way.

Combining the cases discussed in the proofs of Lemmas 4.5.3, 4.5.5 and 4.5.7, we have the following.

- Only messages from X^k_{wc} are sent[3]

  This is the case when $n < T^Z_{low} + 1$. With Lemmas 4.5.3, the number of message sent is $\lfloor \frac{k}{(2n-2)w+1} \rfloor$.

- Only messages from X^k_{wc} and Z^k_{wc} are sent

  The fact that no messages from Y^k_{wc} are sent implies that both conditions $n \geq T^Z_{low} + 1$ and $m < T^Y_{low} + 1$ are satisfied. A summation of (4.187) and (4.193) yields the result for this case.

- Messages from X^k_{wc}, Y^k_{wc} and Z^k_{wc} are sent

  As $PD_{d<m<D}$ sends messages in the latest-deadline-first order and messages in Y^k_{wc} have smaller deadlines than those in Z^k_{wc}, hence, the fact that messages from Y^k_{wc} are sent implies that all the $n - m$ messages from Z^k_{wc} must be sent. Adding the number of messages sent from each subset, we arrive at

$$\begin{aligned} &\text{Send}_{up}(PD_{d<m<D}, N(n,w), A^k_{wc}(n)) \\ &= \lfloor \frac{k}{(2n-1)w+1} \rfloor + (n-m) + \lfloor \frac{m + (n+1)w - T^Y_{low}}{(n+1)w+1} \rfloor. \end{aligned} \tag{4.207}$$

  Replacing T^Y_{low} with (4.196) in the above, we have

[3]When we say that messages in a subset are sent, we do not necessarily imply that all of them are sent.

$$
\begin{aligned}
&\mathrm{Send}_{up}(PD_{d<m<D}, N(n,w), A^k_{wc}(n)) \\
&= \lfloor \frac{k}{(2n-1)w+1} \rfloor + (n-m) + \\
&\quad \lfloor \frac{m+(n+1)w-T^Z_{low}-(n-m)((n+1)w+1)}{(n+1)w+1} \rfloor \\
&= \lfloor \frac{k}{(2n-1)w+1} \rfloor + \\
&\quad \lfloor \frac{(n-m)((n+1)w+1)+m+(n+1)w-T^Z_{low}-(n-m)((n+1)w+1)}{(n+1)w+1} \rfloor \\
&= \lfloor \frac{k}{(2n-1)w+1} \rfloor + \lfloor \frac{m+(n+1)w-T^Z_{low}}{(n+1)w+1} \rfloor. \qquad (4.208)
\end{aligned}
$$

This concluses the proof. Q.E.D.

Now the bounds of the worst case performance ratio of $A^k(n)$ can be derived.

Theorem 4.5.1 *Given a token ring of n nodes and for w, n, m and k, the bounds of the worst case performance ratio of $PD_{d<m<D}$ are given by*

$$
\begin{aligned}
&R_{low}(PD_{d<m<D}, N(n,w), n, k) \\
&= \begin{cases}
\frac{\lfloor \frac{k}{(2n-1)w+1} \rfloor}{n} & n < T^Z_{up}+1, \\
\frac{\lfloor \frac{k}{(2n-1)w+1} \rfloor + \min(n-m, \lfloor \frac{n+1+(n+1)w-T^Z_{up}}{(n+1)w+2} \rfloor)}{n} & n \geq T^Z_{up}+1 \text{ and } \\
 & m < T^Y_{up}+1, \\
\frac{\lfloor \frac{k}{(2n-1)w+1} \rfloor + \lfloor \frac{m+(n+1)w-T^Z_{up}}{(n+1)w+1} \rfloor}{n} & \text{otherwise,}
\end{cases} \qquad (4.209)
\end{aligned}
$$

$$R_{up}(PD_{d<m<D}, N(n,w), n, k)$$
$$= \begin{cases} \frac{\lfloor \frac{k}{(2n-2)w+1} \rfloor}{n} & n < T^Z_{low} + 1, \\ \frac{\lfloor \frac{k}{(2n-2)w+1} \rfloor + \min(n-m, \lfloor \frac{n+1+(n+1)w-T^Z_{low}}{(n+1)w+2} \rfloor)}{n} & n \geq T^Z_{low} + 1 \text{ and } \\ & m < T^Y_{low} + 1, \\ \frac{\lfloor \frac{k}{(2n-2)w+1} \rfloor + \lfloor \frac{m+(n+1)w-T^Z_{low}}{(n+1)w+1} \rfloor}{n} & \text{otherwise.} \end{cases} \quad (4.210)$$

Similar to the case of $PD_{m>D}$, we use the medium value of the performance bounds as an estimation of the worst case performance ratio. That is,

$$R_{est}(PD_{d<m<D}, N(n,w), n, k)$$
$$= \frac{R_{low}(PD_{d<m<D}, N(n,w), n, k) + R_{up}(PD_{d<m<D}, N(n,w), n, k)}{2}, \quad (4.211)$$

where $R_{low}(PD_{d<m<D}, N(n,w), n, k)$ and $R_{up}(PD_{d<m<D}, N(n,w), n, k)$ are given in Theorem 4.5.1. It follows that for given w, n and m, the worst case performance ratio of $PD_{d<m<D}$ can be obtained by numerically exhausting k ($0 \leq k \leq m-1$) as indicated in (4.181).

4.5.3 Numerical Results and Discussions

We have derived the worst case performance ratio of the priority-driven protocol when used to transmit messages whose deadlines are smaller, equal to and greater than the number of priorities m. Theorem 4.5.1 and (4.211) show that the worst case performance ratio of $PD_{d<m<D}$ is a function of n, w, m and k. We now examine the effect of these parameters on the worst case performance ratio.

- We have shown that under $PD_{d<m<D}$ messages whose deadlines are greater than m are sent in the latest-deadline-first order. This has taken place because those messages are assigned the same priority due to the insufficient number of priorities. To see this more clearly, we consider the worst case performance ratio given in (4.211) when the node-to-node delay w is zero[4].

$$R(PD_{d<m<D}, N(n,w), n, k) = \begin{cases} \frac{k}{n} & n < T^Z + 1, \\ \frac{k+\min(n-m, \lfloor \frac{n+1-k}{2} \rfloor)}{n} & n \geq T^Z + 1 \text{ and } m < T^Y + 1. \\ \frac{m}{n} & \text{otherwise.} \end{cases} \quad (4.212)$$

Figure 4.7 shows the impact of the number of priorities m on the worst case performance ratio. We observe that for any given n the larger the number of priorities, the higher the worst case performance ratio. This is can be explained as follows. When n is smaller than m all messages are assigned distinct priorities and messages are sent in the priority order, hence the EDF policy is implemented exactly. Thus, the worst case performance ratio is m/n, implying that for a given n the larger the m, the higher the worst case performance ratio. When n increases to be greater than m, the worst case performance ratio drops to 0.5. This is because in this case messages whose deadlines are greater than m

[4]Note that in this case $T^Z = T^Z_{low} = T^Z_{up}$, $T^Y = T^Y_{low} = T^Y_{up}$ and $R(PD_{d<m<D}, N(n,w), n, k) = R_{est}(PD_{d<m<D}, N(n,w), n, k) = R_{low}(PD_{d<m<D}, N(n,w), n, k) = R_{up}(PD_{d<m<D}, N(n,w), n, k)$.

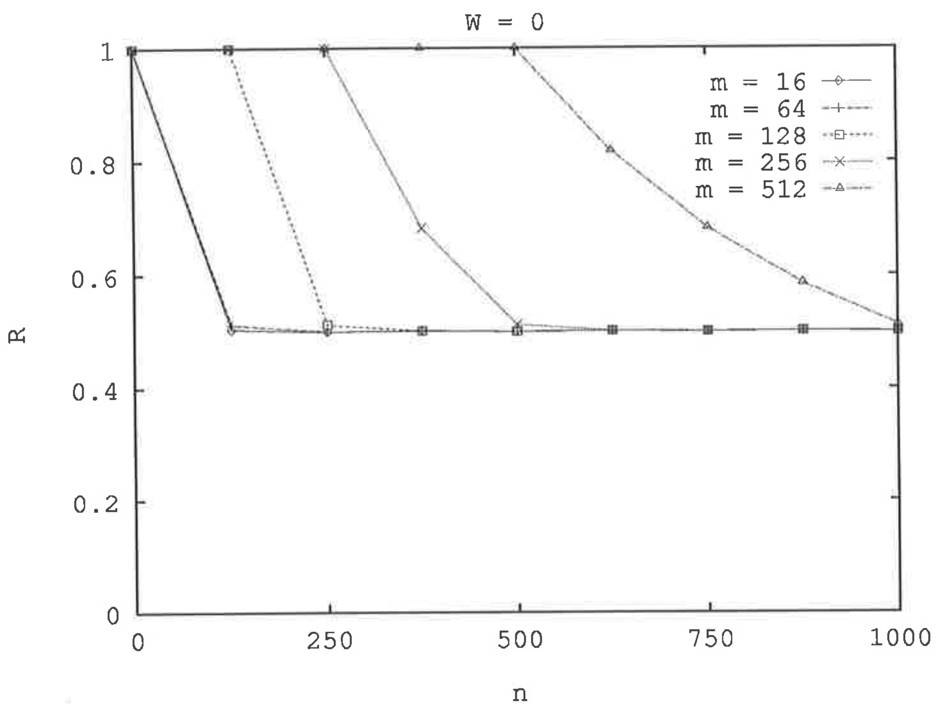


Figure 4.7. Effect of the Number of Priorities

are assigned the same priority regardless of their actual deadlines. As a result, messages are sent in the latest-deadline-first order in the worst case.

- Figure 4.8 shows the effect of parameter k on the protocol performance. We observe that for given w, n and m, increasing k does not result in monotonic increase in the worst case performance ratio. This is because a smaller k causes more messages to be sent in the latest-deadline-first order with a contention overhead of $(n+1)w$, while a larger k enables more messages to be sent in the EDF order, but with a larger contention overhead of at least $(2n-2)w$. As a result, its impact on the worst performance ratio is a trade-off of the transmission policy used and the contention overhead incurred.

- Theorem 4.5.1 also indicates that the worst case performance ratio is a decreasing function of w. Figures 4.9 shows that for $w > 0$ even when the number of

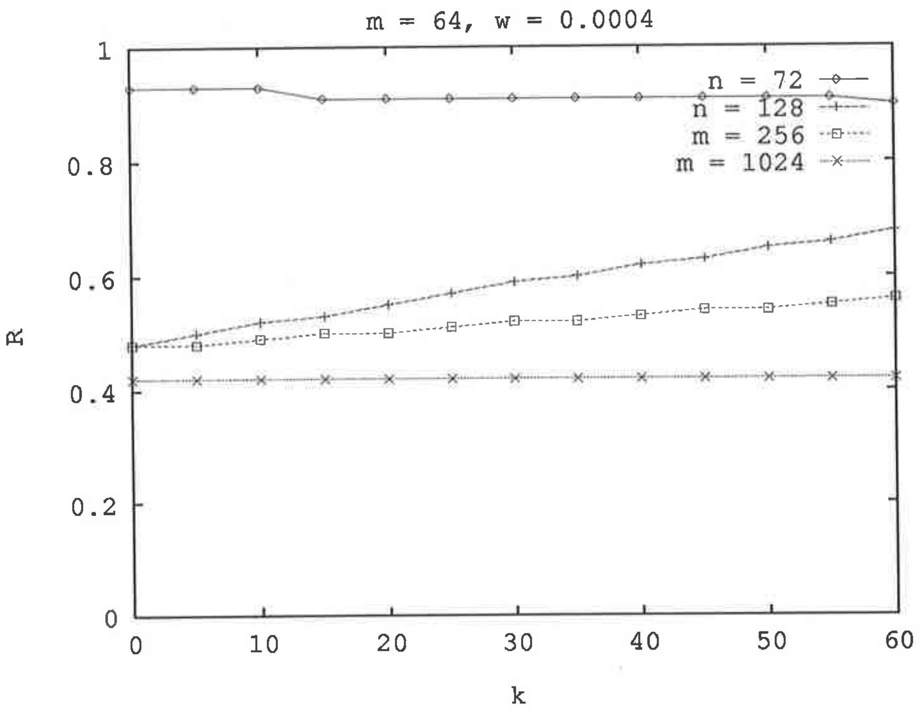


Figure 4.8. Effect of Parameter k

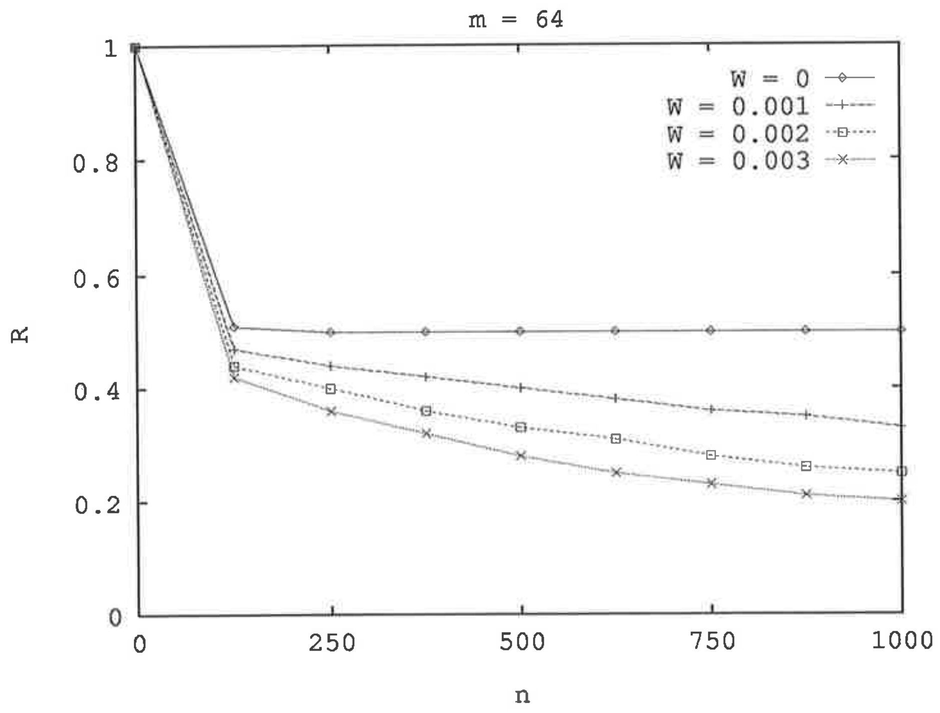


Figure 4.9. Worst Case Performance of $PD_{d<m<D}$

priorities is sufficient, i.e. $n < m$, the worst case performance ratio is still less than 1. This is solely due to the contention overhead. Furthermore, if the number of priorities is not sufficient, i.e. $n > m$, the worst case performance ratio of the protocol drops below 0.5. This is caused by both the contention overhead incurred and the insufficient number of priorities used.

4.6 Enhancements and Modifications

In this section, we propose and discuss several modifications to enhance the previously described priority-driven protocol in order to make it more flexible and adaptive in handling urgent messages.

- Suppose At time t, if the token holder has a message with deadline d and length l, such that $t + l = d$, then the message should be sent immediately; otherwise, the message will eventually be lost since it takes at least another token circulation before the message transmission can start. By then it is too late for the message to make its deadline. This modification can be further extended so that a node sends a message immediately upon capturing the token if the deadline of this message is smaller than a pre-defined threshold.

- Upon capturing the token, a node sends a message immediately if the message has the highest priority. This is because the protocol does not allow a node to overwrite the token priority field with its message priority if the two are the same. This implies if a node writes a message priority, which is the highest, to the token priority field, the token will return with this priority and the node will send the message. To improve efficiency, the protocol should allow a node to send a message of the highest priority immediately upon the token arrival.

Chapter 7 will investigate the average case performance of the priority-driven protocol that has incorporated the above modifications.

Chapter 5

The Window Protocol

In the preceding chapters, we have shown that the worst case performance ratio of the simple token passing protocol and the modified priority-driven protocol can be less than 0.5. Two factors are responsible for their poor performance, namely the transmission policy employeed and the contention overhead incurred. We have demonstrated that the impact of the transmission policy is dominant in determining the protocol performance. Hence, the key for a protocol to achieve high performance ratio is to implement the exact EDF policy.

In this chapter, we propose and analyze for the first time a new *window* (WD) protocol for token ring networks, which implements an exact network-wide EDF transmission policy [24, 47, 48, 50]. We first describe in detail the proposed protocol, together with the window setting, token format and data structure at each node. We then derive the worst case performance ratio and discuss the results. In the third part of this chapter, we propose and examine various encoding schemes for realization of the protocol. Finally, we propose and discuss a number of possible enhancements, aiming to make the proposed window protocol more flexible, adaptable and efficient.

5.1 Basic Concepts

As mentioned in Chapter 1, there exist several window protocols designed for CSMA/CD networks. In those protocols, each node maintains a data structure called *window*, which is a pair of numbers defining an interval on the axis of a message parameter, such as laxity or deadline. Each node continually monitors the channel state and keeps the window information updated. When a node senses the channel idle, it transmits its waiting message if the laxity or deadline of the message falls in the current window. If a collision occurs, all transmitting nodes abort the transmission and the current window is divided into two or more windows which are stored in a stack. Subsequently, the top of the stack is popped to become the new window in which the above procedure repeats until a successful message transmission takes place.

It is clear that in those models, at any time only one window is used to regulate message transmission. This is because in a CSMA/CD network a node obtains the system state information by monitoring the channel state, consequently the search for the earliest deadline message is performed in a binary split manner which is slow and inefficient. In a token ring network, the token circulates around the ring, which makes it possible to convey the global message deadline information to a certain extent, so that the search can be conducted more efficiently. Hence, we propose a novel *multiple window based* token ring protocol. It differs from the existing window protocols for CSMA/CD networks in the window setting and the window splitting, in the search for the earliest deadline message and in an additional token.

The basic idea of our window protocol is as follows. The message deadline axis is partitioned into multiple *windows*. While the token is circulating around the ring for the first time, information about the number of messages in the first non-empty window (i.e. the one contains the earliest deadline message) is collected. After the first token circulation, if the first non-empty window contains only one message, then this message has the earliest deadline and will be sent; otherwise, that window is further split into many smaller windows and the protocol recursively uses the token to locate the earliest deadline message. This way, when a message transmission takes place, the message involved is always the earliest deadline message amongst those currently waiting in the network. Therefore, this window protocol implements the EDF transmission policy.

5.2 Data Structures

Before describing the new window protocol, we first introduce the various data structures associated with the protocol. They are the *window setting*, *token format*, and *data structure at a node.*

5.2.1 Window Setting

In our model, the interval $[t, \infty)$ on the message deadline axis is partitioned into s windows: W_1, W_2, ..., W_s, where t is the current time. Each window defines a half closed interval on the message deadline axis:

$$W_i = [L_i, U_i) \quad \text{and} \quad U_i = L_{i+1} \quad \text{for } 1 \leq i \leq s-1. \tag{5.213}$$

Let α (> 1) denote the size of windows $W_2, W_3, \cdots, W_{s-1}$. There are two special windows W_1 and W_s whose sizes differ from α. The size of W_1 is δ and its lower bound is always set to the current time which enables any newly arrived messages to

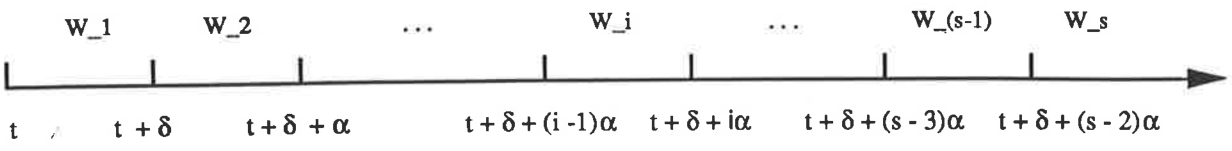


Figure 5.1. Initial Window Boundaries

be considered for transmission immediately. The size of W_s is unbounded because its upper bound is always set to be ∞ to accommodate messages with arbitrarily long deadlines. For convenience, in the following we call the upper bound of W_1 and W_{s-1} the *window lower bound* and *window upper bound* respectively.

The exact size of each window changes from time to time, depending on the stage of the protocol operation. In general, window boundaries at time t can be expressed in terms of t, s, α and δ as follows:

$$\begin{aligned}
W_1 &= [t, t+\delta) \\
W_2 &= [t+\delta, t+\delta+\alpha) \\
W_3 &= [t+\delta+\alpha, t+\delta+2\alpha) \\
&\vdots \\
W_i &= [t+\delta+(i-2)\alpha, t+\delta+(i-1)\alpha) \\
&\vdots \\
W_{s-1} &= [t+\delta+(s-3)\alpha, t+\delta+(s-2)\alpha) \\
W_s &= [t+\delta+(s-2)\alpha, \infty). \qquad (5.214)
\end{aligned}$$

Figure 5.1 shows these non-overlapping windows partitioning the interval $[t, \infty)$. Assume that the protocol is invoked at time 0, then the initial window lower and upper bound is δ and $\delta + (s-2)\alpha$ respectively.

We say that a message M is in window W_k at time t if its deadline falls in the interval defined by the boundaries of W_k at time t, i.e.

$$L_k \leq d < U_k. \tag{5.215}$$

Suppose messages M_i and M_j, with deadlines d_i and d_j, are in windows W_h and W_k respectively. With window boundaries defined in (5.214), if M_i has a smaller deadline than M_j, then W_h must be in front of W_k. That is, if $d_i < d_j$, then we must have $h \leq k$. Therefore, at any time the earliest deadline message must be in the first non-empty window. There are two possible cases.

- The earliest deadline message under search is unique.

  In this case, the earliest deadline message is located only if the number of messages in the first non-empty window is one.

- There exists a *deadline tie* in the first non-empty window.

  That is, more than one message in the first non-empty window have identical deadlines. As a result there are more than one earliest deadline message. In this case, the earliest deadline messages are located (i.e. the deadline tie is detected) if the size of the first non-empty window becomes one.

It is apparent that in either case we only need to be concerned with the message information in the first non-empty window in order to locate the earliest deadline message. This leads to a simple and unique way to search for the earliest deadline message in the new window protocol.

Send Enable (SE)	Split Window (SW)	Previous Window (PW)	Window Counter (WC)	Current Window (CW)

Figure 5.2. Proposed Token Access (AC) Field

5.2.2 Token Format

To implement the EDF transmission policy, we need to know the deadline information about all messages in the network. As the token is the only means to deliver information, we propose the token Access Control (AC) field to contain several information fields to facilitate the protocol operation. They are *Send Enable* (SE), *Split Window* (SW), *Previous Window* (PW), *Window Counter* (WC) and *Current Window* (CW) as shown in Figure 5.2.

We now explain the meanings of these fields. Their usages will be detailed in the next section where the protocol is presented and described.

- *SE* is a binary flag. A token with *SE* set indicates that the earliest deadline message has been located and that the node with the message should capture the token and transmit the message.

- *SW* is also a binary flag. It signals whether window boundaries have been changed since the last token circulation. $SW = 0$ implies that window boundaries retain their initial values; otherwise, current windows are derived from the previous window, indicated by *PW*, as we shall see next.

- *PW* is an integer in the range from 0 to s, where s is the number of windows. In an initial token, this field is set to 0. When flag *SW* is set, it implies that the previous window indexed by *PW* (i.e. W_{PW}) has been split for further search.

Each node uses this information as well as its knowledge of the previous window boundaries to derive the new window boundaries.

- *WC* is also an integer. It counts the number of messages in the first non-empty window. No message counters for other windows are required because we only need to know the number of messages in the first non-empty window which contains the earliest deadline message. For example, if there are two messages in window W_3 and one message in window W_6, then we only need to know that the message counter for the first non-empty window (W_3) is equal to 2 (i.e. $WC = 2$) in order to locate the earliest deadline message.

- *CW* is another integer in the range from 0 to s. If *CW* is zero, it indicates that no message has been registered yet. A non-zero *CW* is the index of the current first non-empty window. This information is used by nodes to determine whether they should register their messages. If a node has a message in a window with a higher index than *CW*, then this message is not a potential earliest deadline message; otherwise, the node registers its message as a potential earliest deadline message. In the above example, only the fact that the W_3 is the current first non-empty window (i.e. $CW = 3$) is needed to eventually locate the earliest deadline message.

In summary, the token conveys three types of information: whether the earliest deadline message has been located (*SE*), whether the window boundaries have been changed (*SW, PW*), and the message deadline information (*WC, CW*). They are essential to the protocol operation. Initially, all token fields are set to 0.

5.2.3 Data Structures on a Node

To support the protocol operation, each node maintains a message queue where messages are kept in the increasing order of their deadlines. Messages with identical deadlines are placed according to the arrival order. Hence, at any time the first message in a node's message queue has the earliest deadline among all pending messages on that node, thus is considered for transmission first. A message is discarded when its deadline has expired.

Initially, all nodes are notified of the values of system parameters s, α and δ. Using these parameters, each node derives and stores the values of the initial window boundaries as shown in (5.214).

5.3 Protocol Description

In this section, we describe the operation of the new window protocol, which is divided into two parts: one for the monitor node and the other for non-monitor nodes.

The *monitor* node is a special node, which is the controller in the process of searching for the earliest deadline message. Initially, it is selected by the management function. After each message transmission, the sending node becomes the new monitor node.

The monitor node is responsible for

- collecting the global message deadline information,
- informing nodes if any window has been split for the current search,
- notifying nodes when the earliest deadline message has been located.

As will become clear later, the use of the token is the key for the monitor node to carry out the above tasks successfully. The monitor node is allowed to modify all the AC fields in the token.

A *non-monitor* node[1] differs from the monitor node by its local role in the search for the earliest deadline message. That is, a non-monitor node is only responsible for notifying the monitor node of its message deadline information if it is relevant to the search for the current earliest deadline message, but does not participate in the decision making as whether the earliest deadline message has been located or the window needs to be split. At any time, a non-monitor node has only partial knowledge of the deadline information about messages residing on other nodes[2]. Upon receiving the token, a non-monitor node examines the information fields in the token and is allowed to modify only three fields in the token: *PW*, *WC* and *CW*.

Next, we describe in detail the operations performed by the monitor and a non-monitor node respectively.

5.3.1 Monitor Node

We define the protocol operation by a state machine transition diagram as shown in Figure 5.3. It consists of three states: *Search*, *Split & Search* and *Send Enable*. The condition for the protocol operation to be in a particular state is listed as follows:

- *Search*: initial search for the earliest deadline message and the initial window boundaries are used,

[1]The monitor node can be viewed as a pair of concentrated virtual nodes, one is the monitor node and the other the non-monitor node. Initially, it is assumed that the token is moving from the monitor virtual node to the non-monitor virtual node.

[2]Specifically, only those upstream nodes.

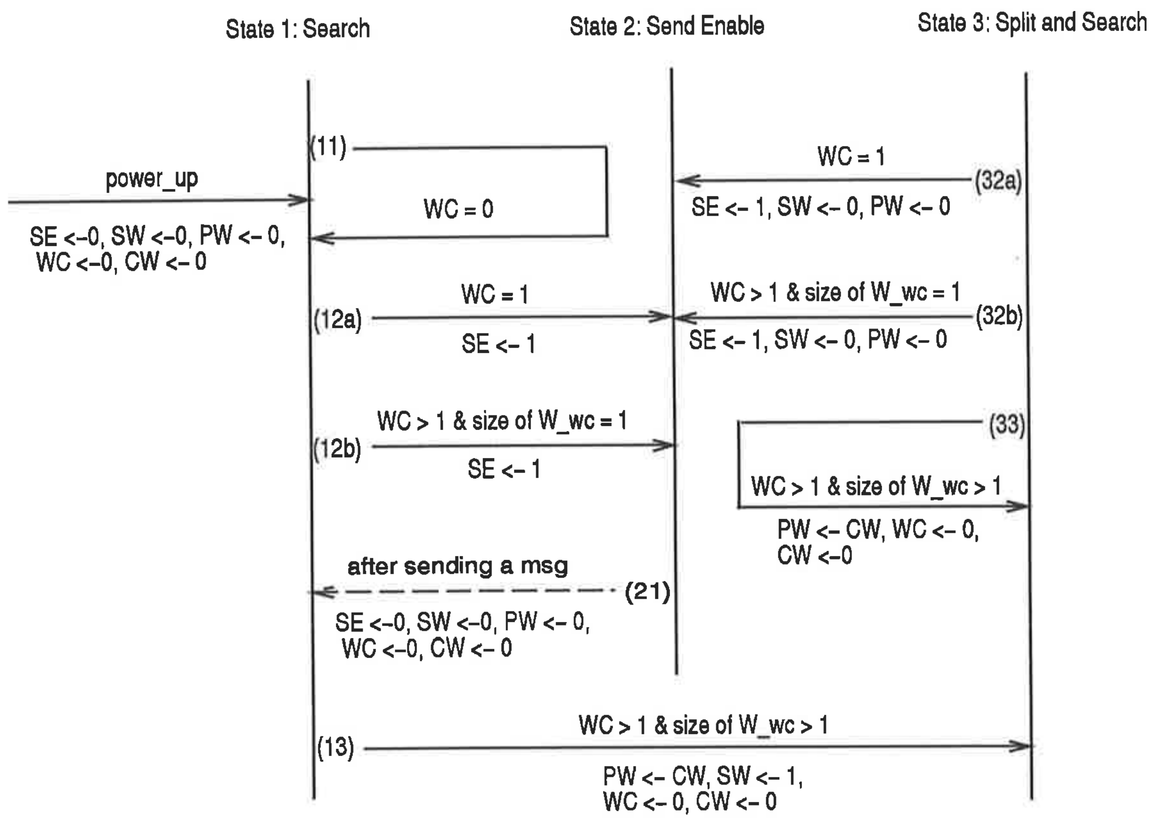

State 1: Search
State 2: Send Enable
State 3: Split and Search
power_up
SE <-0, SW <-0, PW <- 0, WC <-0, CW <- 0
(11)
WC = 0
WC = 1
(32a)
SE <- 1, SW <- 0, PW <- 0
(12a)
WC = 1
SE <- 1
WC > 1 & size of W_wc = 1
(32b)
SE <- 1, SW <- 0, PW <- 0
(12b)
WC > 1 & size of W_wc = 1
SE <- 1
(33)
WC > 1 & size of W_wc > 1
PW <- CW, WC <- 0, CW <-0
after sending a msg
(21)
SE <-0, SW <-0, PW <- 0, WC <-0, CW <- 0
(13)
WC > 1 & size of W_wc > 1
PW <- CW, SW <- 1, WC <- 0, CW <- 0

Figure 5.3. Protocol State Machine Transition Diagram

- *Split & Search*: continued search for the earliest deadline message and window boundaries have been changed, and

- *Send Enable*: the earliest deadline message is located.

In a state transition diagram, a transition from one state to another occurs if certain condition is satisfied. In Figure 5.3, writings above and below each transition line show respectively the token content seen (i.e. the condition) and the operations performed by the monitor node.

We now describe in detail the transitions associated with each state.

- **State 1: Search**

  The protocol operation first enters *Search* state when the network is powered up.

The protocol operation remains in this state if the monitor node receives the token back and finds $WC = 0$. This implies that currently the ring is idle and there are no messages waiting. The monitor node simply passes the token without any change. This is labeled by *transition 11.*

If $WC = 1$, the monitor node knows that the earliest deadline message has been located in window *CW* and the search should now be terminated. The monitor node sets flag *SE* to notify the prospective node to capture the token and send the earliest deadline message. *Transition 12a* shows this case.

If $WC > 1$ and the size of W_{CW} is 1, the monitor node recognizes that there exists an unresolvable ***deadline tie*** in window *CW*. It sets flag *SE* to 1 to notify a node having one of these messages to send the message upon the token arrival. This is indicated by *transition 12b.*

If $WC > 1$ and the size of W_{CW} is larger than 1, the monitor node detects that there are more than one message in the first non-empty window W_{CW} and the search for the earliest deadline message should continue. It sets flag *SW* to 1 and copies *CW* to *PW* to inform other nodes that the potential earliest deadline message is in window *PW*, which must now be split. Fields *WC* and *CW* are reset to 0 for collecting message information regarding to the new windows. The monitor node also computes the new window boundaries and updates its copy of the current window information[3]. This case is indicated by *transition* 13.

- **State 3: Send Enable**

  Denoted by a dashed line, *transition 21* is an indirect transition which may

[3]The procedure for computing the new window operations will be given in the next section when the procedure for non-monitor nodes are given and discussed.

involve two different nodes. This is the case that after the completion of a message transmission, the sending node becomes the new monitor node and releases an initial token with all fields reset to 0.

- **State 2: Split & Search**

  When the monitor node receives the token back, similar to transitions 12a, 12b and 13, there are three possible transitions from this state. They are denoted by *transitions 32a, 32b* and *33*.

  If $WC = 1$, the monitor node recognizes the earliest deadline message has been located in W_{CW}. It carries out all operations as described for transition 12a, and in addition it turns off flag SW and resets PW to 0 to indicate no change in window boundaries. This case is shown as *transition 32a.*

  if $WC > 1$ and the size of W_{CW} is 1, the monitor node detects that there exists a deadline tie in W_{CW}. It carries out all operations as described for *transitions 12b* and in addition it resets flag SW and PW to 0 to indicate no change in window boundaries. This case is denoted by *Transition 32b.*

  If $WC > 1$ and the size of W_{CW} is larger than 1, the monitor node knows that there are more than one message in the first non-empty window W_{CW}. It performs all operations as indicated in transition 13 except setting SW to 1, since in this state SW must have already been 1. This is indicated by *transition 33.*

Table 5.1 summaries the token and window operations performed by the monitor node as described above. Rows 1 – 4 correspond to the transitions associated with *Search* state and Rows 5 – 7 are for the transitions originating from *Split & Search* state. Row 8 represents the case that a sending node becomes the new monitor node when it completes its message transmission.

Row	Token Content					Size of W_{CW}	Interpretation	Operations
	SE	SW	PW	WC	CW			
1	0	0	0	0	0	– –	no msg waiting	no change in token
2	0	0	0	1	x	–	1 msg in W_{CW}	$SE := 1$
3	0	0	0	> 1	x	1	deadline tie in W_{CW}	$SE := 1$
4	0	0	0	> 1	x	> 1	> 1 msg in W_{CW}	split W_{CW}, $SW := 1$, $PW := CW$, $WC := 0$, $CW := 0$
5	0	1	x	1	x	x	1 msg in W_{CW}	$SE := 1$, $SW := 0$, $PW := 0$
6	0	1	x	> 1	x	1	deadline tie in W_{CW}	$SE := 1$, $SW := 0$, $PW := 0$
7	0	1	x	> 1	x	> 1	> 1 msg in W_{CW}	split W_{CW}, $PW := CW$, $WC := 0$, $CW := 0$
8	–	–	–	–	–	–	*a non-monitor node becomes the new monitor after sending a message*	*reset all token fields to 0*

Table 5.1. Monitor Node Operations (Explanatory Scheme)

Note that in whatever follows, for *PW* or *CW*, symbol 'x' represents a positive integer in the range from 1 to s, where s is the number of windows. Symbol '–' means that the value is of no significance in determining the subsequent operations.

5.3.2 Non-Monitor Nodes

We now describe the procedures associated with a non-monitor node upon the token arrival. Table 5.2 shows the following information:

- the possible token content seen by a non-monitor node,

Row	Token Content					Interpretation	Operations
	SE	SW	PW	WC	CW		
1	1	0	0	1	x	*Send Enable*, ED msg in W_{CW}	if $k = CW$, $WC := 0$, $CW := 0$, send msg, become new monitor
2	1	0	0	> 1	x	*Send Enable*, deadline tie in W_{CW}	if $k = CW$, $WC := 0$, $CW := 0$, send msg, become new monitor
3	0	0	0	0	0	*Search*, no msg so far	$WC := 1$, $CW := k$
4	0	0	0	≥ 1	x	*Search*, ≥ 1 msg in W_{CW}	if $k < CW$, $WC := 1$, $CW := k$
5	0	0	0	≥ 1	x	*Search*, ≥ 1 msg in W_{CW}	if $k = CW$, WC:= WC + 1
6	0	0	0	≥ 1	x	*Search*, ≥ 1 msg in W_{CW}	if $k > CW$, no change in token
7	0	1	x	0	0	*Split & Search*, no msg in W_{CW}	split W_{PW} + Row 3
8	0	1	x	≥ 1	x	*Split & Search*, ≥ 1 msg in W_{CW}	split W_{PW} + Row 4
9	0	1	x	≥ 1	x	*Split & Search*, ≥ 1 msg in W_{CW}	split W_{PW} + Row 5
10	0	1	x	≥ 1	x	*Split & Search*, ≥ 1 msg in W_{CW}	split W_{PW} + Row 6

Table 5.2. Non-Monitor Node Operations (Explanatory Model)

- the interpretation with regard to the current state of the protocol operation and message deadline information about upstream nodes, and
- the token and window operations performed by the node before it releases the token.

Upon the arrival of the token at a non-monitor node, if a node has no messages waiting, it only updates the window boundaries if necessary and passes the token without any change; otherwise, the node computes k such that the first message in its message queue is in W_k. That is,

$$L_k \leq d < U_k, \tag{5.216}$$

where d is the deadline of this message, and L_k and U_k are defined in (5.214). The node then takes different actions depending on the values of the token fields and k. In the following, we divide the description into three parts with regard to the value of SE and SW.

- **SE = 1 (cf. Row 1 – 2 in Table 5.2)**

  Once the node sees SE set, $WC \geq 1$ and $CW > 0$, it knows that the protocol operation is now in *Send Enable* state. Counter WC may have different values.

  Case 1: $WC = 1$ (*cf.* Row 1)

  The node knows that the earliest deadline message has been located in W_{CW}.

  Case 2: $WC > 1$ (*cf.* Row 2)

  The node realizes that there exists a deadline tie in W_{CW}.

  In both cases, if the node does not have a message that is in W_{CW}, it simply passes the token without any change; otherwise, the node resets fields WC and CW to 0. This is to inform the downstream nodes that this is a data frame as the combination of $SE = 1$, $PW = 0$, $WC = 0$ and $CW = 0$ is not valid for a free token. The node then sends out the message immediately. After the completion of the message transmission, the node becomes the new monitor node and releases the token with all fields reset to 0.

- **SE = 0 and SW = 0 (cf. Row 3 – 6 in Table 5.2)**

  When the node sees that SE is 0, it knows that the search for the earliest deadline message continues and that the initial window boundaries are used. There are four cases to consider regarding to different values of WC, CW and k.

Case 1: $WC = 0$ (*cf.* Row 3)

This indicates that no message has been registered by upstream nodes yet. The node simply writes the window number k in CW field and sets counter WC to 1 to indicate that there is now a message in W_{CW}.

Case 2: $WC > 0$ and $k < CW$ (*cf.* Row 4)

This means that the node has a message with a deadline smaller than the one denoted on the token, thus its message is the potential earliest deadline message. Hence, it overwrites CW with k and resets counter WC to 1.

Case 3: $WC > 0$ and $k = CW$ (*cf.* Row 5)

This means that upstream nodes have message(s) in W_{CW} and that the node also has a message in W_{CW}. Thus, it simply increments counter WC by 1.

Case 4: $WC > 0$ and $k > CW$ (*cf.* Row 6)

The node knows that its message is not the potential earliest deadline message, as there are other messages having smaller deadlines. The node simply passes the token without any change.

- **SE = 0 and SW = 1 (cf. Row 7 – 10 in Table 5.2)**

As flag SW is set, the node recognizes that the previous window indexed by PW has been split and that the window boundaries have changed since the last token visit. In the following, we only describe how a node derives the new window boundaries and how to compute the window index k for its message given the newly computed window boundaries. Operations on the token fields are exactly the same as those described for Rows 3 – 6, thus are omitted.

– The node first computes the sizes of the new windows. Figure 5.4 shows the splitting of window PW, for $PW = 1$, $2 \leq PW \leq s-1$ and $PW = s$ respectively. We consider these three cases separately.

Case 1: the split window is the first window, i.e. $PW = 1$.

The previous W_1 is now split into $s-1$ smaller equal-sized windows $W'_1, W'_2, \ldots, W'_{s-1}$ and their sizes are derived as

$$\delta' = \alpha' = \lceil \frac{\alpha}{s-1} \rceil. \tag{5.217}$$

The new window boundaries are

$$\begin{aligned} L'_i &= t' + (i-1)\alpha', \quad 1 \leq i \leq s-1 \\ U'_i &= t' + i\alpha', \quad 1 \leq i \leq s-1 \\ L'_s &= t' + (s-1)\alpha' \\ U'_s &= \infty, \end{aligned} \tag{5.218}$$

where t' is the current time.

Case 2: the split window is neither the first nor the last window, i.e. $2 \leq PW \leq s-1$.

In this case, previous windows $W_1, W_2, \ldots, W_{PW-1}$ are now merged to become new window W'_1. This is to accommodate dynamic arrivals so that any newly arrived messages with deadlines smaller than that

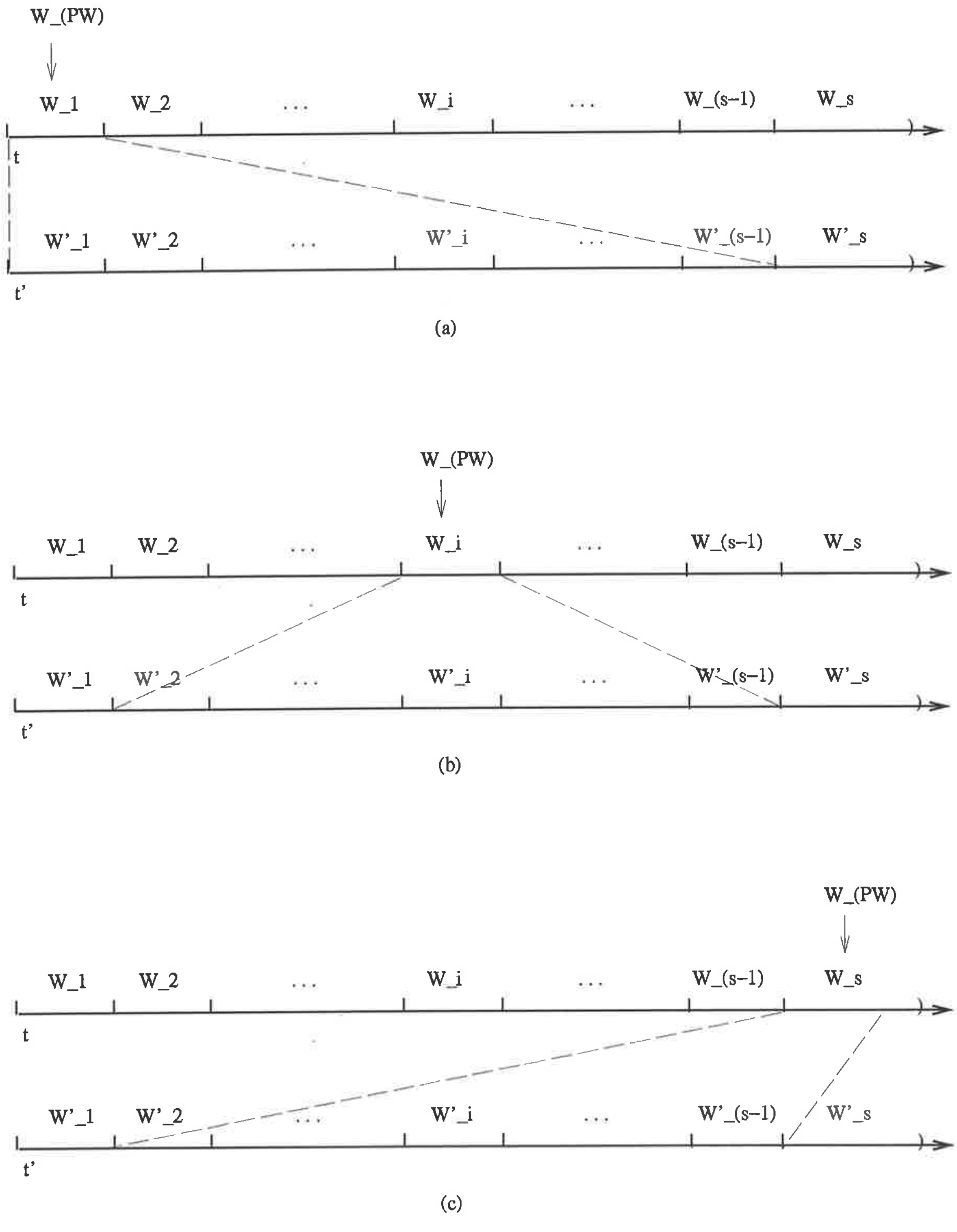


Figure 5.4. Splitting of Window W_{PW}

intended by the current search can still be considered for transmission. The size of the new window W_1' becomes

$$\delta' = \delta + (PW - 2)\alpha, \tag{5.219}$$

where *t'* is the current time. The previous window W_{PW} is split into $s-2$ equal-sized windows $W_2', W_3', \ldots, W_{s-1}'$ whose sizes are

$$\alpha' = \lceil \frac{\alpha}{s-2} \rceil \tag{5.220}$$

The new window boundaries become

$$\begin{aligned} L_1' &= t' \\ U_1' &= t' + \delta' \\ L_i' &= t' + \delta' + (i-2)\alpha', \quad 2 \leq i \leq s-1 \\ U_i' &= t' + \delta' + (i-1)\alpha', \quad 2 \leq i \leq s-1 \\ L_s' &= t' + \delta' + (s-2)\alpha' \\ U_s' &= \infty, \end{aligned} \tag{5.221}$$

where t' is the current time.

Case 3: the split window is the last window, i.e. $PW = s$.

As the upper bound of window W_s is ∞, thus, we need to choose a fraction of W_s for splitting. Let φ denote its size[4]. The previous windows $W_1, W_2, \ldots, W_{s-1}$ are now merged to become the new first

[4]In practice, its value can be set differently according to the message deadline characteristic. The default value can be set to multiples of α.

window W_1'. Hence, the size of the new first window W_1' is

$$\delta' = \delta + (s-2)\alpha. \tag{5.222}$$

Part of the previous W_s is now split into $s-2$ equal-sized windows $W_2', W_3', \ldots, W_{s-1}'$, whose sizes are

$$\alpha' = \lceil \frac{\varphi}{s-2} \rceil. \tag{5.223}$$

The new window boundaries are

$$\begin{aligned} L_1' &= t' \\ U_1' &= t' + \delta' \\ L_i' &= t' + \delta' + (i-2)\alpha', \quad 2 \le i \le s-1 \\ U_i' &= t' + \delta' + (i-1)\alpha', \quad 2 \le i \le s-1 \\ L_s' &= t' + \delta' + (s-2)\alpha' \\ U_s' &= \infty, \end{aligned} \tag{5.224}$$

where t' is the current time.

– With newly computed window boundaries, the node locates window W_k which contains the first message in node's message queue. That is,

$$L_k' \le d < U_k', \tag{5.225}$$

where d is the deadline of the first message.

- The node then takes different actions according to the value of k and the token fields. The operations are exactly the same as those described for Rows 3 – 6 in Table 5.2.

This concludes the description of the new window protocol. Note that the protocol described is not intended to be implemented directly and is in need of refinement in two ways. Firstly, an encoding scheme of the token AC field is needed to efficiently realize the protocol. Secondly, a variety of performance enhancements can be incorporated to make the protocol more flexible, adaptive and efficient. We will discuss these two issues in detail in Sections 5.7 and 5.8.

5.4 Worst Case Performance Analysis

In this section, we examine the worst case performance of the new window protocol.

Lemma 5.4.1 *Suppose messages M_i and M_j, having deadlines of d_i and d_j, are in windows W_h and W_k respectively. With the window setting given in (5.214) in section 5.2.1, we must have*

$$h \leq k, \quad \text{if} \ \ d_i < d_j. \tag{5.226}$$

Proof: As M_i is in W_h, we have

$$L_h \leq d_i < U_h, \tag{5.227}$$

where L_h and U_h denote the lower and upper bound of W_h. There are two cases to consider.

- $L_h \leq d_i < d_j < U_h$

  In this case, M_j is also in window W_h. Thus, $h = k$.

- $L_h \leq d_i < U_h \leq d_j$

  In this case, M_j is in one of the windows with higher index than h. Hence, $h < k$.

In summary, we have $h \leq k$. The concludes the proof. Q.E.D.

Lemma 5.4.2 *The proposed window protocol implements the EDF transmission policy. That is, at time t if a message M is transmitted, it must be the earliest deadline message waiting at that time.*

Proof: Lemma 5.4.1 implies that the earliest deadline message must be in the window with the smallest index, i.e. the first non-empty window. The operations given in Tables 5.1 and 5.2 ensure that the earliest deadline message is found only if the number of messages in the first non-empty window is one or the size of the first non-empty window is one. Hence, if a node is allowed to send a message, it must be the earliest deadline message. Q.E.D.

Lemma 5.4.3 *If the window protocol is used to transmit n messages, the message set with a maximum message deadline of n minimizes the number of messages sent.*

Proof: Lemma 4.3.2 states that if the EDF policy is employed to transmit n messages, the message set with a maximum message deadline of n minimizes the number of messages sent in the worst case. As the new window protocol implements the EDF policy, we have the lemma proved. Q.E.D.

We now examine the maximum contention overhead involved in locating the earliest deadline message in the window protocol. Recall that the upper bound of the last window W_s is always set to ∞. If all messages are in the last window, then according to the protocol part of the last window is split again and again until at least one message is found not in the last window. As long as message deadlines are bounded, it is straightforward to calculate the number of splittings required. Hence, without loss of generality we assume that the actual window upper bound, i.e. the upper bound of W_{s-1}, is chosen such that it is greater than the largest message deadline. This way, no splitting occurs in the last window.

Lemma 5.4.4 *Given a token ring of n nodes, the maximum contention overhead c incurred in locating the earliest deadline message in the window protocol is*

$$c_{max} = \lceil \log_s \Delta \rceil nw + w, \tag{5.228}$$

where s is the number of windows and Δ is the initial value of the actual window upper bound (i.e. the upper bound of the initial window W_{s-1}).

Proof: According to the protocol, if there are messages waiting in the network, then there are two cases to consider when the token returns to the monitor node after one complete circulation.

- Only one message is found in the first non-empty window, implying that it is the earliest deadline message. In this case, it takes nw units of time for the token to complete one full circulation and iw $(1 \leq i \leq n)$ units of time for the token to reach the node having the earliest deadline message. Hence, the total overhead is $nw + iw \leq 2nw$.

- Two or more messages are found in the first non-empty window. In this case, the first non-empty window is further split into many smaller windows and the protocol continues recursively. Evidently, if the first non-empty window contains two messages having identical deadlines, then the search for the earliest deadline message involves a deadline tie. Consequently, it needs $\lceil \log_s \Delta \rceil$ token circulations, thus $\lceil \log_s \Delta \rceil nw$ units of time to reduce the window size to 1 to solve the deadline tie. After the earliest deadline message is located, it takes another w units of time for the token to reach the next node downstream to inform the node to send its message[5]. Therefore, the total contention overhead for a message transmission is $\lceil \log_s \Delta \rceil nw + w$.

Clearly, $\lceil \log_s \Delta \rceil nw + w > 2nw$ since $\Delta > s$ for $\alpha > 1$. Thus, in the worst case all messages have the same deadlines and each message transmission incurs a maximum overhead of $\lceil \log_s \Delta \rceil nw + w$. This concludes the proof. Q.E.D.

The worst case message set can now be identified.

Lemma 5.4.5 *Given a token ring of n nodes, if the window protocol is used to transmit n messages, the worst case set is*

$$A_{wc}(n) = \{(n,1),(n,2),\cdots,(n,n-1),(n,n)\}. \tag{5.229}$$

Proof: As the number of messages deadlines is n, there must be at least one message with a deadline of n; otherwise, the message set is infeasible. On the other hand, it

[5]As all message have the same deadlines, the node capturing the token next is the next sending node.

follows from Lemma 5.4.3 that in the worst case the maximum deadline of messages must be n. Furthermore, Lemma 5.4.4 states that when there exists a deadline tie, the message transmission incurs a maximum overhead. Thus we conclude that in the worst case message set all messages have the same deadlines of n so that every search involves a deadline tie. As a result, each message transmission incurs a maximum overhead. Q.E.D.

With Lemmas 5.4.4 and 5.4.5, the number of messages sent from message set $A_{wc}(n)$ can be determined.

Lemma 5.4.6 *Given a token ring of n nodes, if the window protocol is used to send n messages, then in the worst case the number of messages sent is*

$$\mathit{Send}(WD, N(n,w), A_{wc}(n)) = \lfloor \frac{n}{(\lceil \log_s \Delta \rceil n + 1)w + 1} \rfloor . \tag{5.230}$$

Proof: With Lemma 5.4.5, we know that when the window protocol is used to transmit n messages, all messages in the worst case message set have the same deadlines. Hence, any time the search for the earliest deadline message involves a deadline tie.

Lemma 5.4.4 shows that for a token ring of n nodes, the maximum overhead in detecting a deadline tie is $\lceil \log_s \Delta \rceil nw$. Then, it takes another w units of time for the token to travel from the monitor node to its nearest neighbor, which is the node having the earliest deadline message. Hence, the total time needed for a message transmission is $(\lceil \log_s \Delta \rceil n + 1)w + 1$, where 1 is the message transmission time. As the window protocol implements the EDF policy and the maximum message deadline is n, it follows from Lemma 4.3.3 that the number of messages sent from (5.229) is $\lfloor \frac{n}{(\lceil \log_s \Delta \rceil n+1)w+1} \rfloor$. Q.E.D.

With the above lemma, the following can be readily established.

Theorem 5.4.1 *The worst case performance ratio of the window protocol is given by*

$$R(WD, w, n) = \frac{\left\lfloor \frac{n}{(\lceil \log_s \Delta \rceil n + 1)w + 1} \right\rfloor}{n} \tag{5.231}$$

5.5 Numerical Results and Discussions

As the window protocol implements the exact EDF transmission policy, the contention overhead is the only cause for the performance degradation. We can see this more clearly when w approaches to zero.

$$\lim_{w \to 0} R(WD, w, n) = 1. \tag{5.232}$$

That is, if working in an ideal environment where $w = 0$, the window protocol can send all the messages as a result of implementing the EDF policy.

Figures 5.5 shows the protocol performance when the number of windows is 64. We observe that when $w = 0$, the protocol obtains a performance ratio of 1 for any given n. This implies that the window protocol would have the same behavior as the CEDF protocol if the contention overhead is assumed to be zero. On the other hand, the protocol performance degrades significantly as w and/or n increases. This is because in the worst case the contention overhead of the window protocol is $(\lceil \log_s \Delta \rceil n + 1)w$.

Theorem 5.4.1 also indicates that the worst case performance ratio is a function of s and n. Figures 5.6 shows the impact of s and n on the protocol performance.

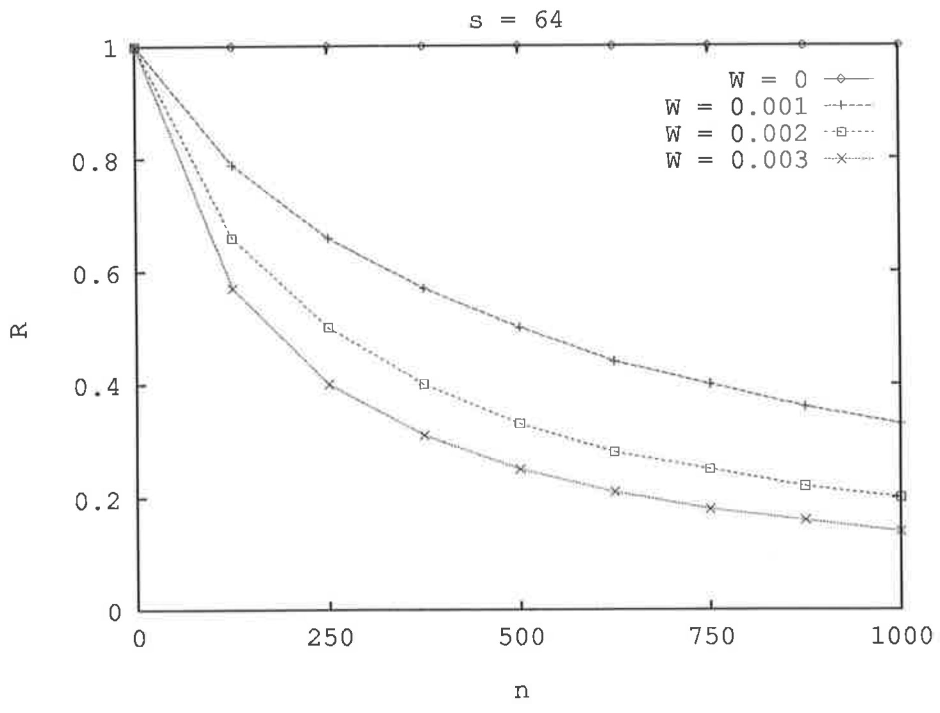


Figure 5.5. Worst Case Performance Ratio of *WD*

Clearly, for a given n the larger the number of windows, the higher the sent ratio[6]. Likewise, for a given s the larger the n, the lower the sent ratio. This is because when $w > 0$, the contention overhead incurred in a message transmission is an increasing function of $\lceil Log(s, \Delta) \rceil$. Obviously, increasing s results in a smaller overhead, hence a higher sent ratio. On the other hand, increasing n gives rise to a larger overhead, and thus a smaller sent ratio.

5.6 Major Advantages of the Window Protocol

We now summarize the major advantages of the proposed window protocol.

[6] We also note that increasing s may not result in any increase in sent ratio, this is due to the property of function $\lceil\ \rceil$.

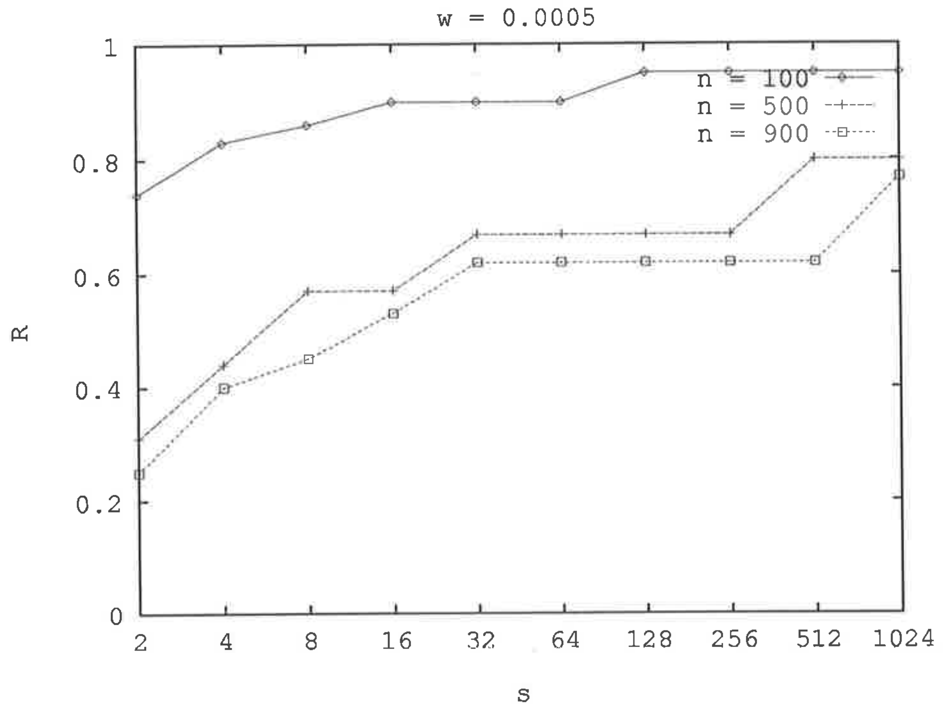
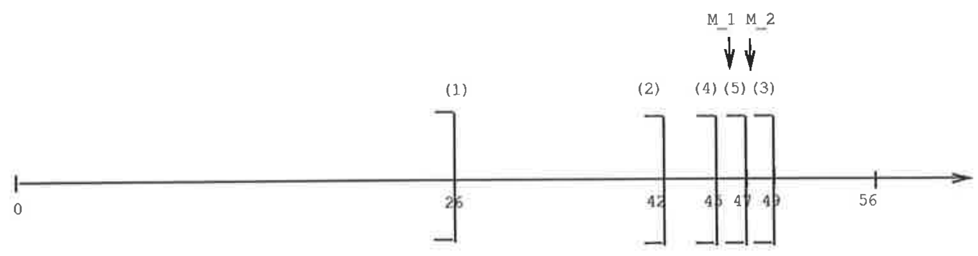


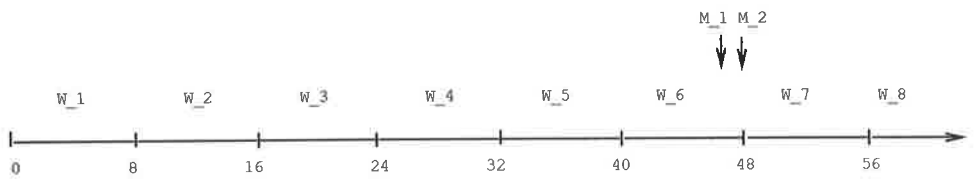
Figure 5.6. Effect of Number of Windows

- The protocol implements the network-wide EDF transmission policy. That is, at any time if a message is sent, it is the earliest deadline message.

- The use of multiple windows offers faster convergence in locating the earliest deadline message as compared with the existing window protocols designed for CSMA/CD networks. In those protocols, whenever a collision occurs the window upper bound is moved in a binary manner. The protocols continue in this way until either a message transmission is successful or the window size becomes one. This inefficient strategy of the window operation is due to the nature of CSMA/CD networks where nodes are not able to obtain sufficient deadline information about messages residing on other nodes. Therefore, contention can only be detected upon a collision, but the actual message deadlines involved are not known. The new window protocol takes the advantage of the token



(a) Binary Search in the Window Protocols for CSMA/CD Networks



(b) Multiple Window Search in the New Window Protocol for Token Ring Networks

Figure 5.7. Time Diagram for Message Transmission

circulating around the ring and benefits from the multiple window structure, so that the monitor node is able to gain useful knowledge of the global message deadline information. As a result, the search for the earliest deadline message is more efficient. Consequently, the new window protocol offers much faster convergence in locating the earliest deadline message. Figure 5.7 demonstrates this by a simple example. Suppose two messages have deadlines of 47 and 48 respectively. With an initial window size of 56, the binary search in the existing window protocols starts with window [0,56] and needs to move the window upper bound at least 5 times before M_1 can be sent successfully. By contrast, in the new window protocol (with 8 windows), after one token circulation, the monitor node finds that M_1 is the only message in W_6. Hence, only one token rotation is needed to locate the earliest deadline message. Evidently, the search for the desired message in the new window protocol is significantly faster.

5.7 Protocol Realization

In this section, we discuss how to efficiently realize the previously described window protocol. We start with the direct realization of the protocol and show how the information fields on the token can be reduced. This leads to an optimal encoding scheme using a minimum 5-bit AC field, while the full functionality of the protocol is correctly preserved. We then derive a general formula identifying the quantitative relationship between the number of windows obtainable and the number of bits needed, using the optimal encoding scheme. Finally, we give an alternative encoding scheme using an 8-bit AC field which conforms to the IEEE 802.5 standard.

5.7.1 Direct Realization

In the window protocol described in Section 5.3, one bit is needed for binary flags SE and SW respectively. Fields PW and CW are of equal length and are in the range from 0 to s (the number of windows), hence the number of bits needed for each of them is $\lceil \log_2 s \rceil$. Finally, $\lceil \log_2 n \rceil$ bits are required for counter WC, where n is the number of nodes. This is because in the most unfavorable case where each node has a message waiting and all these messages are in the same window, counter WC has to record these n messages. The IEEE 802.5 token ring network needs to support up to 500 nodes, which requires 9 bits for counter WC. Summing up the above, the total number of bits needed in the token AC field is

$$2 + 2 * \lceil \log_2 s \rceil + 9. \qquad (5.233)$$

This means that at least 13 bits (since $s \geq 2$) are needed for the token AC field if the window protocol is implemented directly.

5.7.2 Optimal Realization

In this section, we show how to use a minimum length AC field to implement the previously described window protocol with its full functionality preserved.

Before we present the encoding scheme, we first discuss some possible reduction in information needed for locating the earliest deadline message and examine ways in which the length of the AC field may be minimized.

- Recall in Table 5.2, when a non-monitor node finds that there is at least one message in W_{CW} (*cf.* Rows 5 and 9 in Table 5.2) and its own message is in the same window, it increments counter *WC* by 1. This way, after one token circulation the monitor node has an accurate account of the number of messages in the first non-empty window W_{CW}. However, as the earliest deadline message is located only if one message is found in the first non-empty window (when no deadline tie exists), we only need to know whether **one** or **more than one**[7] message is in W_{CW}. This implies that counter *WC* can now be simplified to a binary flag. Together with field *CW*, three cases reflecting the number of messages in the first non-empty window can be encoded:

  $WC = 0$ and $CW = 0$: no message is found in the network so far.

  $WC = 0$ and $CW > 0$: one message is in the first non-empty window so far.

  $WC = 1$ and $CW > 0$: more than one message are in the first non-empty window so far.

[7]The exact number is not relevant to the current search of the earliest deadline message.

Note that with this compound encoding, only one bit is needed for counter *WC*, which is a significant reduction from $\lceil \log_2 n \rceil$ bits required by the direct realization.

- When a non-monitor node sees $SW = 1$ (*cf.* Rows 7–10 in Table 5.2), it knows that one of the previous windows has been split since the last token visit. The index of the split window is denoted by PW (> 0). Clearly, we can use field PW alone to indicate whether any previous window has been split.

  $PW = 0$: initial window boundaries are used.

  $PW > 0$: previous window W_{PW} has been split.

- We also notice that when a unique earliest deadline message is found or a deadline tie is recognized (*cf.* Row 1 – 2 in Table 5.1), the monitor node sets flag *SE* to 1 and resets field PW to 0. This implies that when a non-monitor node receives the token with flag *SE* set, PW must be 0. if flag *SE* is off, then PW must be positive. Hence, it is feasible not to use flag *SE* in the token at all. This is done by introducing another local binary flag at each node, so that the node having the earliest deadline message is able to determine whether it is allowed to transmit its message upon the arrival of the token if PW is 0.

  Each node now has a binary flag called *Message Registered* (*MR*) which is set to 0 initially. When the system is in *Search* state, a node checks whether it should register its message in the token as previously described. Only if the node overwrites CW on the token with its own message window index, the node sets flag *MR* to 1. When the token arrives at a node with $PW > 0$ (window boundaries have been changed), the node resets flag *MR* if it is on. It then carries out the procedure described in Table 5.2 to see whether it should register its message in relation to the new window boundaries. If the token arrives with $PW = 0$ and its flag *MR* is on, the node knows that the protocol operation is now in *Send Enable* state. It compares its message window index with *CW* in the token. If they are the same, then the node resets *MR* and sends its message; otherwise, the node simply resets flag *MR* and passes the token without any change.

The operations on flag *MR* described above ensures the following:

– flag *MR* on a node is set only if the node overwrites *CW* with its own message window index;

– flag *MR* on a node is reset before any other operations when the node sees $PW > 0$;

– when the monitor node starts *Send Enable*, flags *MR* on all nodes between the node having the earliest deadline message and the monitor node are off;

– when the monitor node starts *Send Enable*, the node with the earliest deadline message is the last node having flag *MR* set;

– after a non-monitor node becomes the monitor node and enters *Search* state, no node has flag *MR* set.

The above properties of the operations on flag *MR* guarantee that these local flags are reset once a new search starts. Clearly, the test for a node to check if the system is in *Send Enable* state is $PW = 0$ and $MR = 1$. As a result, the condition that a node is allowed to send a message is $PW = 0$, $MR = 1$ and $k = CW$.

Row	Token Content			Size of W_{CW}	Interpretation	Operations
	PW	WC	CW			
1	0	0	0	–	no msg waiting	no change in token fields
2	0	0	x	–	1 msg in W_{CW}	no change in token fields
3	0	1	x	1	deadline tie in W_{CW}	no change in token fields
4	0	1	x	> 1	> 1 msg in W_{CW}	split W_{CW}, $PW := CW$, $WC := 0$, $CW := 0$
5	x	0	x	–	1 msg in W_{CW}	$PW := 0$
6	x	1	x	1	deadline tie in W_{CW}	$PW := 0$
7	x	1	x	> 1	> 1 msg in W_{CW}	split W_{CW}, $PW := CW$, $WC := 0$, $CW := 0$
8	*0*	–	> 0	–	*a non-monitor node becomes the monitor after sending the msg*	*reset all token fields to 0*

Table 5.3. Encoding for Monitor Node Operations (Optimal Scheme)

It should be pointed out that when using nodal flags *MR* to replace the explicit binary flag *SE* on the token, idle nodes or nodes that have not registered their messages during the previous token circulation(s) are not aware of the transition of the system state from *Search* to *Send Enable* as their flags *MR* are off in both cases. Thus the condition specified above is always evaluated false. Nevertheless, the protocol operation remains correct.

The optimal encoding schemes incorporating the changes described above are given in Tables 5.3 and 5.4, for the monitor node and non-monitor nodes respectively. We see that the token AC field is now reduced to include only three information fields. In Table 5.4, fields *SE* and *SW* no longer appear explicitly on the token, instead their values shown are computed from the three token fields and the additional node-based flag *MR* as discussed above. Note that for *MR*, symbol 'x' means that its value (either on or off) has no significance in determining the node's operations. For *PW* and *CW*, symbol 'x' denotes a positive integer in the range from 0 to s.

Row	Token Content			Nodal	Computed		Interpretation	Operations
	PW	WC	CW	MR	SE	SW		
1	0	0	x	1	1	0	*Send Enable*, ED msg in W_{CW}	reset *MR*, if $k = CW$, $WC := 0$, $CW := 0$, send msg, become new monitor
2	0	1	x	1	1	0	*Send Enable*, deadline tie in W_{CW}	reset *MR*, if $k = CW$, $WC := 0$, $CW := 0$, send msg, become new monitor
3	0	0	0	0	0	0	*Search*, no msg	$CW := k$, set *MR*
4	0	0	x	0	0	0	*Search*, 1 msg in W_{CW}	if $k < CW$, $CW := k$, set *MR*
5	0	0	x	0	0	0	*Search*, 1 msg in W_{CW}	if $k = CW$, $WC := 1$
6	0	0	x	0	0	0	*Search*, 1 msg in W_{CW}	if $k > CW$, no change in token
7	0	1	x	0	0	0	*Search*, > 1 msg in W_{CW}	if $k < CW$, $WC := 0$, $CW = k$, set *MR*
8	0	1	x	0	0	0	*Search*, > 1 msg in W_{CW}	if $k = CW$, no change in token
9	0	1	x	0	0	0	*Search*, > 1 msg in W_{CW}	if $k > CW$, no change in token
10	x	0	0	x	0	1	*Split & Search*, no msg	reset *MR* + split W_{PW} + Row 3
11	x	1	x	x	0	1	*Split & Search*, 1 msg in W_{CW}	reset *MR* + split W_{PW} + Row 4
12	x	1	x	x	0	1	*Split & Search*, 1 msg in W_{CW}	reset *MR* + split W_{PW} + Row 5
13	x	1	x	x	0	1	*Split & Search*, 1 msg in W_{CW}	reset *MR* + split W_{PW} + Row 6
14	x	1	x	x	0	1	*Split & Search*, > 1 msg in W_{CW}	reset *MR* + split W_{PW} + Row 7
15	x	1	x	x	0	1	*Split & Search*, > 1 msg in W_{CW}	reset *MR* + split W_{PW} + Row 8
16	x	1	x	x	0	1	*Split & Search*, > 1 msg in W_{CW}	reset *MR* + split W_{PW} + Row 9

Table 5.4. Encoding for Non-Monitor Node Operations (Optimal Scheme)

Evidently, the above encoding scheme replaces the explicit flags SE and SW on the token with a compound encoding. Furthermore, each possible value of a field is fully utilized. As a result, the explicit information needed on the token is minimized, while the protocol functionality is fully reserved. Therefore, this encoding scheme is optimal.

Token Length

Under the above optimal encoding scheme, we now derive a general formula to identify the relationship between the number of bits b needed in the token AC field and the number of windows s implemented.

One bit is needed for counter WC field. To implement s windows, $\lceil \log_2 s \rceil$ bits are needed for PW and CW respectively. Thus, in total we have

$$b = 1 + 2 * \lceil \log_2 s \rceil. \tag{5.234}$$

Suppose we have multiple windows ($s \geq 2$), then a minimum of 5 bits are needed to realize the protocol.

On the other hand, for a given b-bit long token AC field, it follows that the minimum number of windows that can be implemented is[8]

$$s = 2^{(b-1)/2}. \tag{5.235}$$

Evidently, the number of windows which can be realized by the optimal encoding method increases *exponentially* to the length of the token AC field. Consequently,

[8]In the following formula, we assume that b is an odd number. If b is an even number, the result should be rounded to the nearest integer.

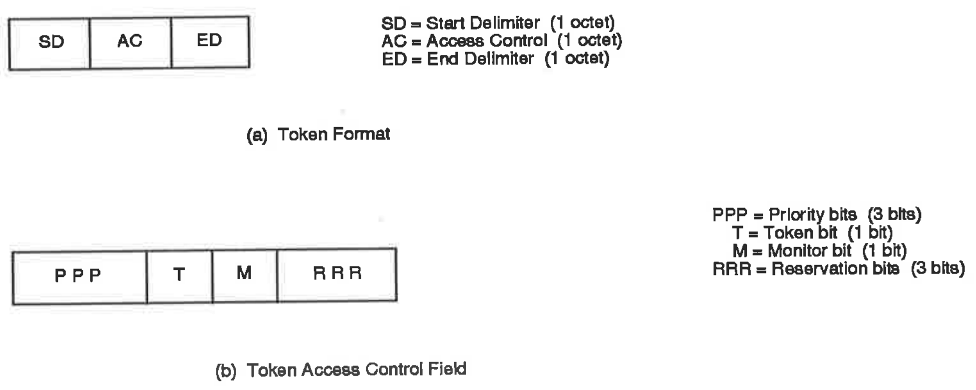

SD
AC
ED
SD = Start Delimiter (1 octet)
AC = Access Control (1 octet)
ED = End Delimiter (1 octet)
(a) Token Format
PPP
T
M
RRR
PPP = Priority bits (3 bits)
T = Token bit (1 bit)
M = Monitor bit (1 bit)
RRR = Reservation bits (3 bits)
(b) Token Access Control Field

Figure 5.8. IEEE 802.5 Token Format

a small increase in the number of bits results in a large increase in the number of windows, which in turn reduces the contention overhead and thus improves the protocol performance.

5.7.3 Practical Realization

We have given an optimal encoding scheme which requires a 5-bit token AC field to realize the protocol,

In practice, the encoding scheme is constrained by the actual token length, hence we now consider realizing the protocol using an 8-bit token AC field which conforms to the IEEE 802.5 standard. Figure 5.8 depicts the token format used in an 802.5 token ring network.

We see that the AC field contains three priority bits, three reservation bits and two binary flag bits. The priority and reservation bits are protocol related, hence they not relevant to the window protocol. The *Token bit* which distinguishes a data frame from the free token and the *Monitor bit* which prevents a packet from endlessly circulating around the ring are replaced by the compound encoding in our encoding scheme.

Hence, we have a total of 8 bits which can be used to realize the previously described window protocol. One method is to use the optimal encoding scheme described above. That is, 3 bits are allocated to fields *PW* and *CW* respectively, and 1 bit is used for counter *WC*, which gives a total of 7 bits.

This leaves us one spare bit, which can be used in different ways. In the following, we give one such scheme. Let this spare bit to be used for flag *SE*. That is, we let *Send Enable* to be explicitly indicated on the token rather than to be computed from the values of *PW*, *CW*, *WC* and local flag *MR*. As a result, there is no need for nodal flag *MR*.

Tables 5.5 and 5.6 give the detailed encoding for operations performed by the monitor node and non-monitor nodes respectively. Note that 'x' denotes a positive integer in the range from 1 to s.

5.8 Enhancements and Modifications

In this section, we propose various possible enhancements and modifications which can be incorporated into the previously described window protocol. We then examine and discuss their feasibility, utility, advantages and disadvantages in terms of making the proposed window protocol more adaptive, flexible and efficient. These modifications aim to preserve in principle the EDF transmission policy, while minimizing the overhead to achieve the best possible protocol performance.

5.8.1 Urgent Pre-emption

The basic version of the proposed window protocol may not be flexible enough in handling urgent messages which have very tight deadlines. The following simple

Row	Token Content				Size of W_{CW}	Interpretation	Operations
	SE 1 bit	PW 3 bits	WC 1 bit	CW 3 bit			
1	0	0	0	0	– –	no msg waiting	no change in token fields
2	0	0	0	x	–	1 msg in W_{CW}	$SE := 1$
3	0	0	1	x	1	deadline tie in W_{CW}	$SE := 1$
4	0	0	1	x	> 1	> 1 msg in W_{CW}	split W_{CW}, $PW := CW$, $CW := 0$, $WC := 0$
5	0	x	0	x	–	1 msg in W_{CW}	$SE := 1$, $PW := 0$
6	0	x	1	x	1	deadline tie in W_{CW}	$SE := 1$, $PW := 0$
7	0	x	1	x	> 1	> 1 msg in W_{CW}	split W_{CW}, $PW := CW$, $CW := 0$, $WC := 0$
8	*1*	–	–	–	–	*a non-monitor node becomes the monitor after sending a msg*	*reset all token fields to 0*

Table 5.5. Encoding for Monitor Node Operations (Practical Scheme)

example demonstrates the possibility. Suppose a node receives the token at time t and has a message with a deadline of $d = t + l_M$, where l_M is the message transmission time plus the overhead. Obviously, if the node registers this message on the token, it takes at least another token circulation before the node can start the transmission, by which time it would be too late for the message to make its deadline. Thus, a straightforward modification is to allow a node having a message with a deadline of d, such that $d = t + l_M$, to send out the message immediately even if flag $SE = 0$. This example shows a very extreme case. In practice, the condition can be relaxed to $d \leq t + l_M + threshold$. The value of the threshold can be chosen small enough so that the EDF policy can be well approximated.

Row	Token Content				Interpretation	Operations
	SE 1 bit	PW 3 bits	WC 1 bits	CW 3 bit		
1	1	0	0	x	*Send Enable*, ED msg in W_{CW}	if $k = CW$, $WC := 0$, $CW := 0$, send msg, become new monitor
2	1	0	1	x	*Send Enable*, deadline tie in W_{CW}	if $k = CW$, $WC := 0$, $CW := 0$, send msg, become new monitor
3	0	0	0	0	*Search*, no msg	$CW := k$
4	0	0	0	x	*Search*, 1 msg in W_{CW}	if $k < CW$, $CW := k$
5	0	0	0	x	*Search*, 1 msg in W_{CW}	if $k = CW$, $WC := 1$
6	0	0	0	x	*Search*, 1 msg in W_{CW}	if $k > CW$, no change in token
7	0	0	1	x	*Search*, > 1 msg in W_{CW}	if $k < CW$, $CW := k$, $WC = 0$
8	0	0	1	x	*Search*, > 1 msg in W_{CW}	if $k = CW$, no change in token
9	0	0	1	x	*Search*, > 1 msg in W_{CW}	if $k > CW$, no change in token
10	0	x	0	0	*Split & Search*, no msg	split W_{PW} + Row 3
11	0	x	0	x	*Split & Search*, 1 msg in W_{CW}	split W_{PW} + Row 4
12	0	x	0	x	*Split & Search*, 1 msg in W_{CW}	split W_{PW} + Row 5
13	0	x	0	x	*Split & Search*, 1 msg in W_{CW}	split W_{PW} + Row 6
14	0	x	1	x	*Split & Search*, > 1 msg in W_{CW}	split W_{PW} + Row 7
15	0	x	1	x	*Split & Search*, > 1 msg in W_{CW}	split W_{PW} + Row 8
16	0	x	1	x	*Split & Search*, > 1 msg in W_{CW}	split W_{PW} + Row 9

Table 5.6. Encoding for Non-Monitor Node Operations (Practical Scheme)

An alternative approach is to allow a node to send a message immediately if the message is in the first window W_1. Recall that the upper bound of this window changes during the search. After a window is split, W_1 may be formed by merging all the previous windows up to the first non-empty windows. Thus, if a node has a message in this new first window after the window split, it must be a newly arrived message with a deadline smaller than the one that is intended by the current search. Thus this modification dynamically takes care of newly arrived urgent messages. If the probability of two or more message arrivals during a token circulation is very low, then the EDF policy is mostly observed. To maximize the adherence to the EDF policy, a small initial size of the first window should be used. This modification also implies that a node is allowed to pre-empt the token with flag SE set on the way from the monitor node to a node having the previously located the earliest deadline message.

Clearly, the above urgent pre-emption schemes allow urgent messages to be sent sooner. Under the previously described protocol, a non-monitor node becomes the monitor node after it transmits a message. However, in the case where the token with SE set is pre-empted on the way to a non-monitor node, when the pre-empting node becomes the monitor node after its transmission, it resets the token fields and the search for the previously located the earliest deadline message starts again. Obviously, it is wasteful to do so. The solution is that after the pre-empting node completes the message transmission, it issues the token identical to the one it received and does not become the monitor node. This way, the non-monitor node having the earliest deadline message can still send its message upon the token arrival. Consequently, the overhead involved is minimized.

5.8.2 Faster Deadline Tie Handling

Recall when a deadline tie is recognized, the monitor node sets flag SE to 1. Then the first node having such a message captures the token and sends the message first. It then releases the token with all fields reset to 0 and the window boundaries are reset to their initial values. However, reseting the token fields also means discarding the information of the deadline tie. In the worst case where remaining messages involved in the tie are still the earliest deadline messages[9], it requires further token circulations and window splittings to locate the remaining earliest deadline messages that involved in the tie. Clearly, this process continues until the message set involved in the tie is reduced to the last one (no more deadline tie). It is obviously wasteful to do so and this inefficiency is most pronounced when the number of messages involved in a deadline tie is large.

The way to overcome the above problem is to modify the protocol so that the first sending node does not reset the token fields, but releases the same token (i.e. $SE = 1$, $WC = 1$ and $CW > 0$). The purpose of this is to let other messages involved in the deadline tie be transmitted one after another once the tie is detected. This way, when the first sending node receives the token back with $SE = 1$, it knows that all messages involved in the tie have been transmitted. It then issues a token with all fields reset to 0 and the initial window boundaries are resumed.

Clearly, the above modification observes the EDF policy when there are no new arrivals or newly arrived messages have larger deadlines than those involved in the deadline tie. If a node, having a newly arrived message with a smaller deadline, captures the token before the second node involved in the deadline tie, to preserve the

[9]If there are no new message arrivals or if all newly arrived message have deadlines greater than those involved in the tie.

EDF policy, the protocol should allow this node to send the newly arrived message immediately. This is similar to the urgent pre-emption as discussed before.

5.8.3 Faster Resolution

It is obvious that when there are messages having deadlines very close to the earliest deadline message (the extreme case would be a deadline tie as discussed above), many window splits may be needed to reduce the window size to very small. This is especially true when the initial window size is large and the number of windows is small. For any messages with identical deadlines or close deadlines, in practice it is perhaps not beneficial to further differentiate them[10]. Hence, if the protocol can predict or estimate in advance and stop further splitting windows, then the overhead can be reduced significantly. Indeed, in the following we examine three such modifications.

One solution is to terminate the search once the monitor node detects that there is enough time to send all messages in the first non-empty window. Specifically, the monitor checks to see if the following holds:

$$L_{CW} \geq t + i * l_M, \tag{5.236}$$

where i is the number of messages in W_{CW}[11] and l_M is the message transmission time plus the overhead. The reason that L_{CW}, the lower bound of W_{CW}, is used in (5.236) rather than the actual message deadline is that the monitor node only knows the number of messages in W_{CW}, but not the exact message deadlines involved. However, if a message is in W_{CW}, its deadline must be at least equal to the lower bound of

[10]Unless to differentiate them on the basis of their service classes.

[11]In this case counter WC needs to count more than one message.

W_{CW}, but less than the upper bound of W_{CW}. If all messages in the first non-empty window can make their deadlines, then it does not matter in which order they are sent. To understand this better, consider the case where the current time t is 4, and L_{CW} and U_{CW} is 16 and 32 respectively. Suppose four messages, with deadlines of 22, 25, 27 and 30 respectively, are found in W_{CW}. Hence, each of them is the potential earliest deadline message as far as nodes are concerned currently. The monitor node checks if $16 \geq 4 + 4 * 1$ holds. As it is true in this case, the monitor node sets *SE* to 1, terminating the search. Note that this estimation is conservative as the actual message deadlines can be much larger than the lower bound of W_{CW}. For example, in the same scenario, if the current time is 15, the test of (5.236) would be evaluated false, though in fact all these four messages can still make their deadlines.

A less conservative solution is not to be concerned with the number of messages in the window, but only the window size. That is, when the monitor node detects the size of W_{CW} is less than a threshold, $WC > 0$ and $CW > 0$, it terminates the search and sets *SE* to 1. A node having a message in W_{CW} and capturing the token first sends its message. If the threshold is set to 1, then this solution is equivalent to resolving a tie in the previously described window protocol. When the threshold is chosen to be greater than 1, it becomes an imprecise solution and the degree of imprecision depends on the value of the threshold, WC and CW. Although the alteration of the order in which messages are sent may result in different messages to be lost, the performance of the protocol may be improved as a result of considerably reduced overhead.

Another approach is that when the number of messages in the first non-empty window is less than a threshold, the monitor node sets flag *SE*. This approach also reduces the overhead, though it may also violate the EDF policy, the degree of which depends on the window size α and the value of counter WC. In practice, a small value of the threshold may produce significant improvement in performance ratio.

5.8.4 Choice of Threshold

We see all the modifications proposed and discussed above aim to offer fast response time to urgent messages and to reduce the overhead incurred in the search for the earliest deadline message. They exhibit a good trade-off between the level of implementing the optimal EDF transmission policy and the overhead incurred in the implementation.

Although it is difficult to determine analytically the optimal values of thresholds involved in these schemes as they are subject to network load, message arrival pattern, and message deadline and position distributions. In practice, a threshold can use either a fixed pre-defined value or a heuristic value assigned dynamically according to the load and message characteristics.

5.8.5 Possible Realization

These modifications are also simple to be implemented, each of which requires only one or two additional entries in Tables 5.3, 5.4, 5.5 and 5.6.

The only complication of some of these modifications is the need for additional bits for counter WC to record the exact number of messages (may be more than one) in the first non-empty window. Thus, more bits and more encoding are needed to incorporate the modifications. In practice, a realistic figure may be chosen as a compromise.

Chapter 6

Worst Case Performance Comparison

In this chapter, we use the results obtained from the previous three chapters to compare the worst case performance of the three token ring protocols [24]. We first outline the comparison method, and then present and discuss the results. Finally, we examine the implication and applicability of the comparison results. This investigation provides a guideline for the design of distributed scheduling algorithms and communication protocols.

6.1 Comparison Method

Our objective is to identify the conditions (in terms of network parameters) under which one protocol may perform better than the others. Specifically, we wish to partition the parameter space into several regions, in each of which one protocol outperforms the others.

First, we compare the worst case performance of the three protocols on a pair-wise basis. As for each protocol, the worst case performance ratio is a function of network parameters n and w, thus each pair-wise comparison is made by first equating the worst case performance ratios of the two chosen protocols. We then solve w in terms of n. However, due to the nature of $\lfloor\ \rfloor$ function contained in the worst case performance ratios, there may be multiple values of w that satisfy the equation. Therefore, for a given n, we define w_{min} and w_{max} to be the minimum and the maximum value of w that satisfy the equation. Thus, w_{min} and w_{max} are functions of n, i.e.

$$\begin{aligned} w_{min} &= f_{min}(n), \\ w_{max} &= f_{max}(n). \end{aligned} \tag{6.237}$$

The above equations can be presented as two curves on the n-w plain. We call the area between the two curves the ***equal-performing band***, as the two protocols perform the same in this area.

To further clarify the concept of the equal-performing band and its usage, let us consider comparing the token passing protocol with the priority-driven protocol. First, we set up the following equation:

$$R(TP, w, n) = R(PD, w, n), \tag{6.238}$$

where $R(TP, w, n)$ is given in (3.3.1), while $R(PD, w, n)$ is given in (4.136), (4.4.1) and (4.3.1) respectively for different m. We then numerically solve (6.238) to obtain $w_{min} = f_{min}(n)$ and $w_{max} = f_{max}(n)$. The two corresponding curves are depicted in Figure 6.1(a). The area between the two curves is the equal-performing band. Clearly, the equal-performing band partitions the parameter space into three parts:

- the band itself in which $R(TP, w, n) = R(PD, w, n)$, indicating that two protocols perform the same;

- the lower left part (below the curve of $f_{min}(n)$)) in which $R(TP, w, n) < R(PD, w, n)$, suggesting that the priority-driven protocol performs better than the token passing protocol;
- the upper right part (above the curve of $f_{max}(n)$) in which $R(TP, w, n) > R(PD, w, n)$, implying that the token passing protocol outperforms the priority-driven protocol.

Other two pairs of protocols are compared in the same manner and their equal performing bands are shown in Figures 6.1(b) and (c) respectively. The relevant protocol parameters used for Figure 6.1 are $m = 16$ for the priority-driven protocol and $s = 16$ for the window protocol.

When we compare the three protocols, the three pair-wise equal-performing bands are used. However, superimposing all of them together may not produce a graphical presentation which is clear and easy to understand. Hence, to simplify the comparison without lossing the essence of the problem, we reduce an equal-performing band to an *equal-performing curve* as follows: for a given equal-performing band, i.e. $w_{min} = f_{min}(n)$ and $w_{max} = f_{max}(n)$, its corresponding equal-performing curve is obtained as follows:

$$w = f(n) = \frac{f_{min}(n) + f_{max}(n)}{2}. \tag{6.239}$$

That is, for each n we choose the medium value of w_{min} and w_{max}. In the following, we use the equal-performing curves to compare the three protocols.

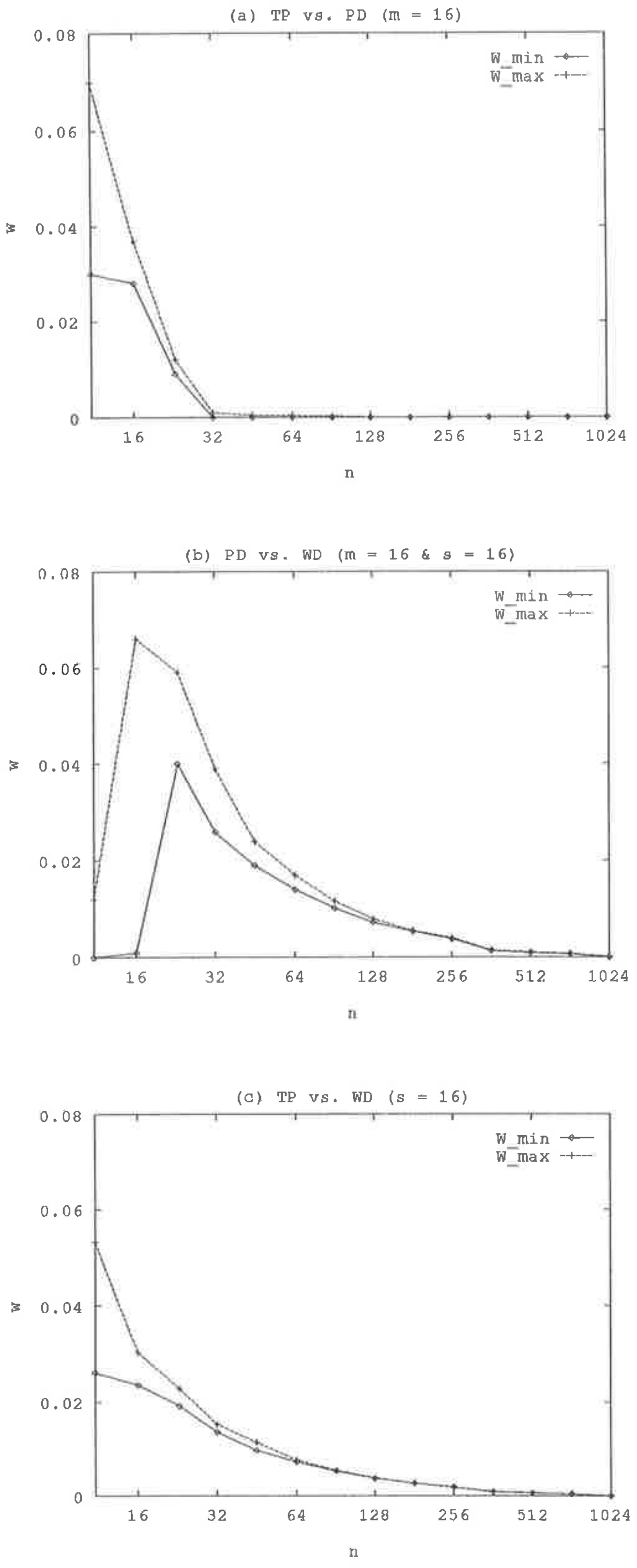


Figure 6.1. Pair-Wise Equal Performing Bands

6.2 Pair-Wise Comparisons

The equal-performing curves are depicted in Figure 6.2.

- **Token passing protocol versus priority-driven protocol**

  From Figure 6.2(a), we observe that the priority-driven protocol performs better than the token passing protocol when both n and w are small. This is because the token passing protocol does not observe the EDF policy, while the priority-driven protocol implements the exact EDF policy when the number of message deadlines is smaller than the number of priorities. Consequently, the priority-driven protocol produces better performance for small n and w. On the other hand, as the contention overhead of the priority driven protocol and the token passing protocol is in the order of nw and w respectively, the impact of increasing n or/and w on the performance of the priority-driven protocol is greater than that on the token passing protocol. Hence, the token passing protocol eventually outperforms the priority-driven protocol as n or/and w increases.

- **Priority-driven protocol versus window protocol**

  The equal-performing curve is shown in Figure 6.2(b). Recall that when the number of message deadlines n is smaller than the number of priorities m, both the priority-driven and the window protocols implement the EDF policy. Hence, in this case the performance of the two protocols is determined entirely by their contention overheads, which are functions of parameters n, w, m and s. If the contention overhead of the priority-driven protocol is smaller, then it outperforms the window protocol; otherwise, the window protocol yields better performance. As n becomes greater than m, the performance of the priority-driven protocol deteriorates as it no longer implements the exact EDF policy. Nevertheless,

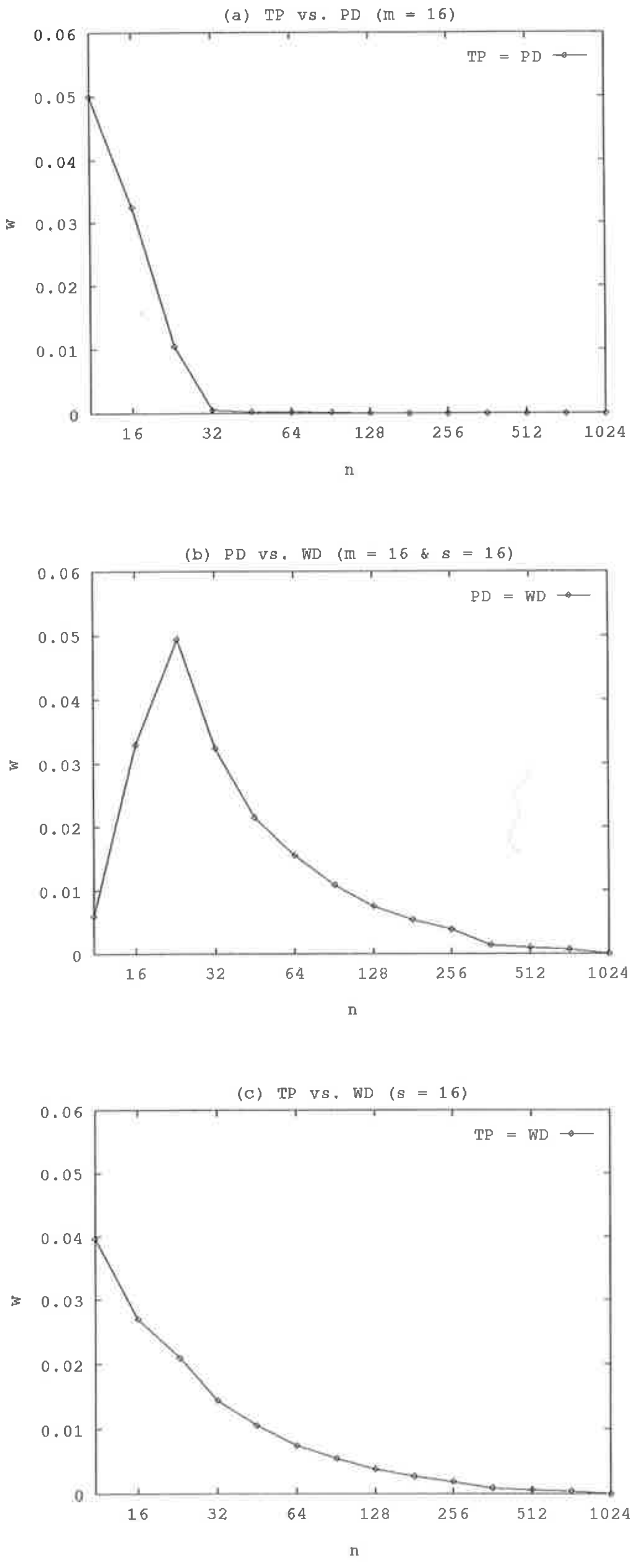


Figure 6.2. Pair-Wise Equal Performing Curves

because of the large contention overhead, the window protocol outperforms the priority-driven protocol only when w is sufficiently small.

- **Token passing protocol versus window protocol**

  The result is shown in Figure 6.2(c). We see that the window protocol exhibits its performance advantage when both n and w are small. This is because the token passing protocol does not consider message deadlines, while the window protocol always implements the EDF policy with a small contention overhead when n and w are small. As n and/or w increase, the window protocol suffers from a rapid growth in its contention overhead. As a result, the benefit of an accurate implementation of the EDF policy is nullified by the increase in its contention overhead, making it incapable of competing with the token passing protocol for large n and w.

6.3 Comparison of Three Protocols

To compare the three protocols, we superimpose the above three pair-wise equal-performing curves and the result is shown in Figure 6.3.

We see that the three curves corss each other resulting in a small triangle area near $n = 16$ and $w = 0.03$, in which the three protocols have the same performance. As our goal of the comparison is to partition the parameter space into several disjoint regions, thus to simplify the presentation we choose a point inside the triangle to replace the triangle. Figure 6.4(a) shows the result.

We see that the three curves partition the parameter space into six different regions, labeled as $I, II, \cdots$ and VI respectively. Each region represents an area where the performance of one protocol is the same or better than another. For example, in

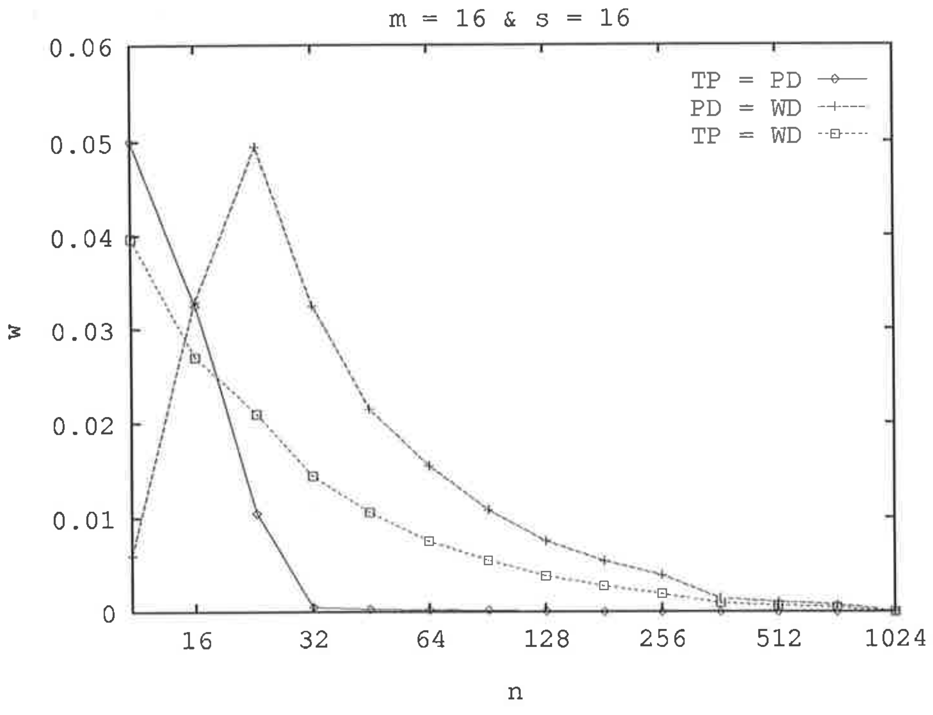
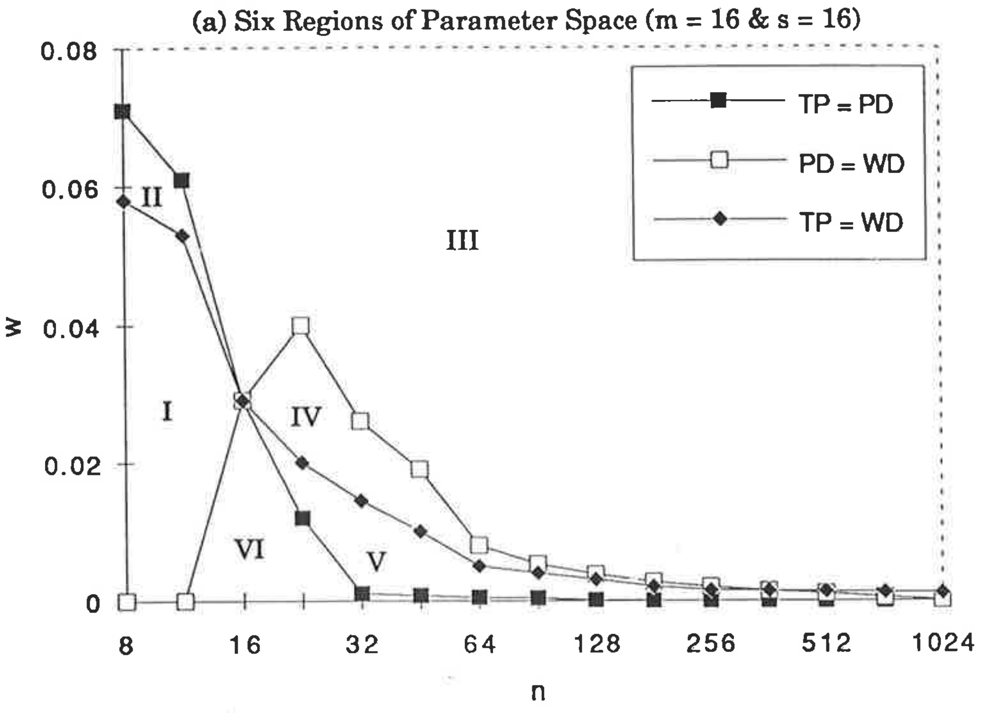


Figure 6.3. Comparison of Three Protocols

Region I, we have $R(PD, w, n) \geq R(WD, w, n)$ and $R(WD, w, n) \geq R(TP, w, n)$, implying that the performance of the priority-driven protocol performs either the same or better than the window protocol, and that the window protocol performs either the same or better than the token passing protocol. Table 6.1 lists the performance relationship for each region.

These six regions can be further grouped into three domains, such that one protocol achieves the best performance in one domain as shown in Figure 6.6(b). For instance, the domain labeled with "WD" covers regions V and VI, indicating that the window protocol is either the same or better than both the token passing and the priority-driven protocol. The three disjoint domains cover the entire parameter space, implying that no protocol can always dominate the others, and each protocol has its own applicable area in the parameter space defined by the network attributes



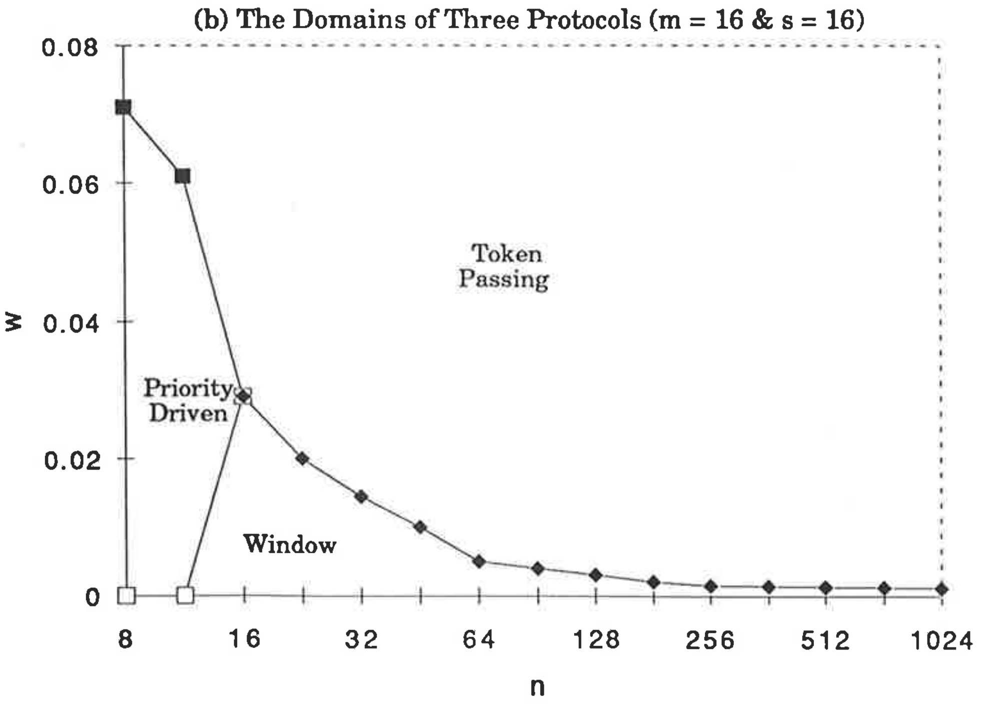


Figure 6.4. Comparison of Three Protocols

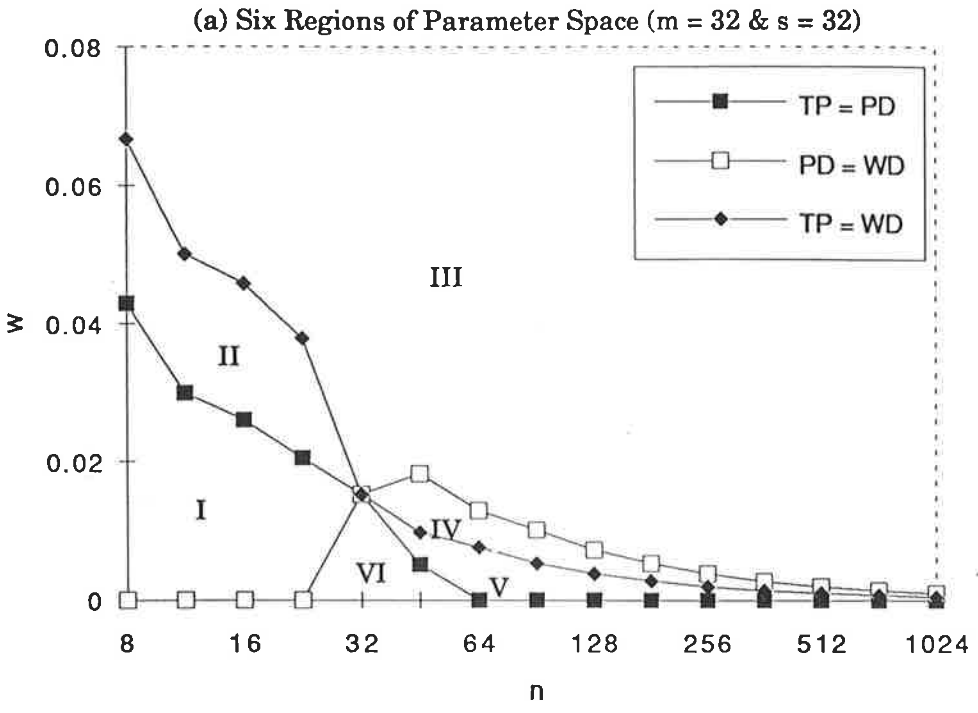
Region	Performance Relationship
I	$R(PD,w,n) \geq R(WD,w,n)$ and $R(WD,w,n) \geq R(TP,w,n)$
II	$R(PD,w,n) \geq R(TP,w,n)$ and $R(TP,w,n) \geq R(WD,w,n)$
III	$R(TP,w,n) \geq R(PD,w,n)$ and $R(PD,w,n) \geq R(WP,w,n)$
IV	$R(TP,w,n) \geq R(WD,w,n)$ and $R(WD,w,n) \geq R(PD,w,n)$
V	$R(WD,w,n) \geq R(TP,w,n)$ and $R(TP,w,n) \geq R(PD,w,n)$
VI	$R(WD,w,n) \geq R(PD,w,n)$ and $R(PD,w,n) \geq R(TP,w,n)$

Table 6.1. Performance Relationship in Six Regions

and application parameters. This indicates that the performance of a communication protocol is a trade-off between the optimality of the transmission policy employed and the contention overhead involved.

Although the comparison results presented and discussed above are generated with $m = 16$ and $s = 16$, many other comparison results have also been obtained under different values of protocol parameters. The general conclusion is that although the shape and the size of each domain differs from that in Figure 6.4, similar observations can be made. Figure 6.5 shows comparison result for $m = 32$ and $s = 32$.

The above observations and discussions have implied that in designing a real-time communication protocol, one should carefully assess the trade-off of implementing the optimal scheduling policy and minimizing the implementation overhead to satisfy the design objective. Furthermore, in determining a communication protocol for a particular network, it is important to conduct the worst case performance analysis of each candidate protocol and compare their performance in the parameter ranges associated with the network and applications. The desired protocol should be the one which yields the best performance within the projected parameter ranges.



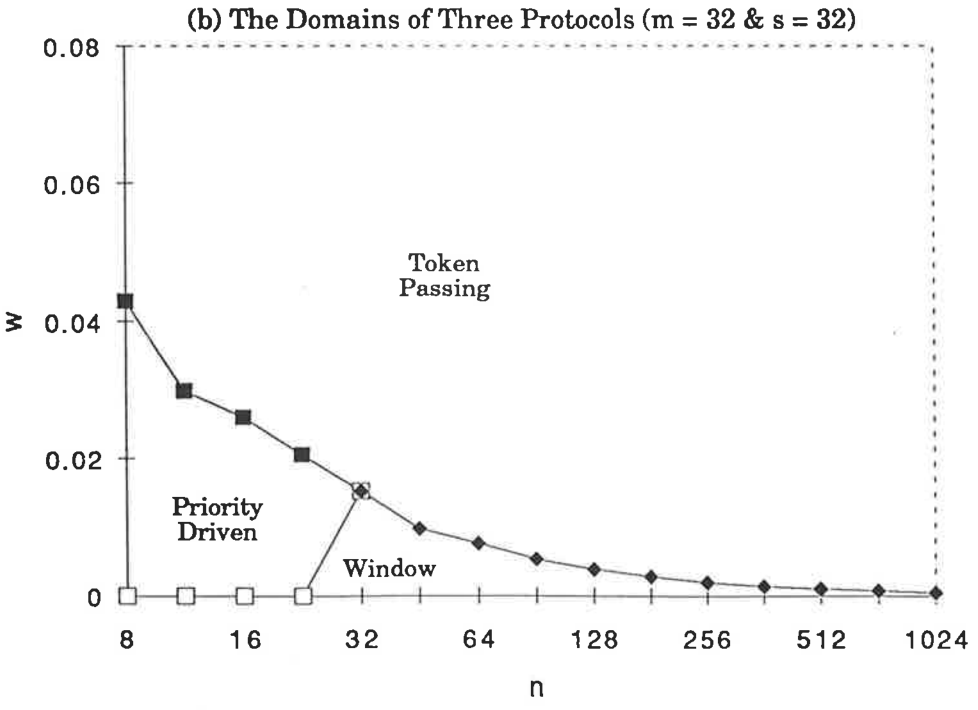


Figure 6.5. Comparison of Three Protocols

6.4 Applicability of Results

Our study would have little contribution in practice if the parameter values considered are not within the range supported by the current token ring networks in terms of parameters n and w. We validate this in relation to the IEEE 802.5 token ring standard.

The 802.5 token ring can operate at a speed c of 1, 4 or 16 *Mbit/s*. It can support a maximum number n of 500 stations. The bus propagation delay ρ is 5 μs/km. The token length L_t is 24 bits. From Chapter 2, the normalized token node-to-node delay w' is

$$w' = \frac{(l * \rho)/n + \theta/c}{(L_p + L_t)/c}, \tag{6.240}$$

Where θ is the station bit delay, l is the ring length, L_p and L_t are the packet length and the token length respectively.

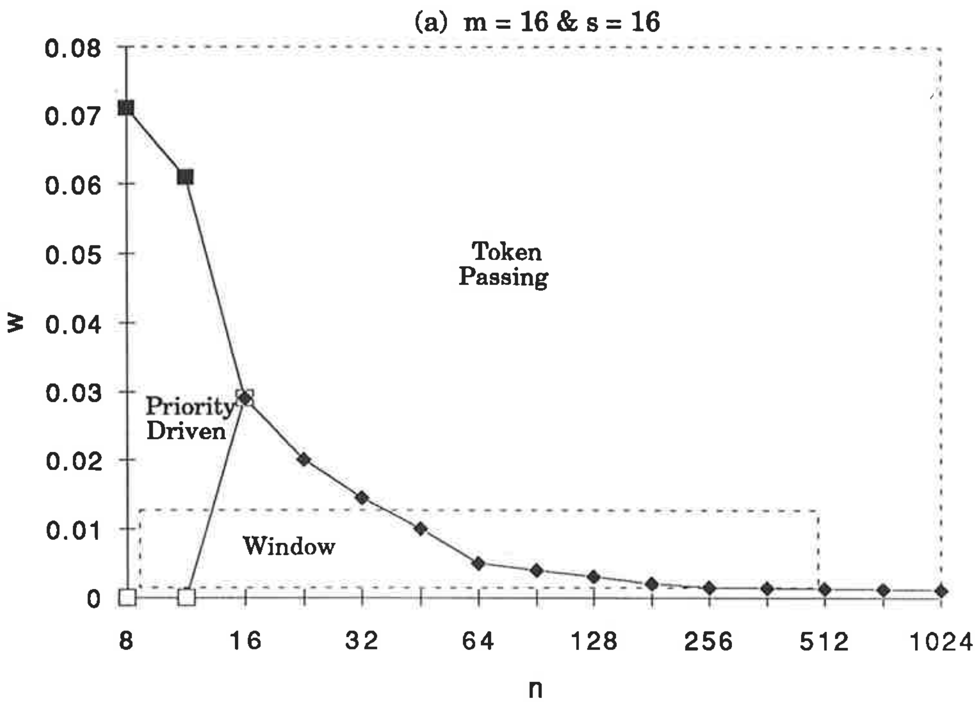
Table 6.2 lists the w' values for different combinations of system parameters. The ring length l is 1 *km* and the station bit delay θ is 4 *bits*. Although the 802.5 standard permits a maximum token holding time of 10 *ms* for message transmission by a token holder, in the following we use an average packet length of 1024 bits in order to derive the maximum possible value of w'. Hence, for ring speeds 1, 4 and 16 *Mbit/s*, the corresponding message transmission times are 1.024, 0.256 and 0.2256 *ms* respectively. Furthermore, we assume that each station only transmits one such frame whenever it holds the token.

In Table 6.2, we see that w' ranges from 0.00382 to 0.0119. The dotted line boxes in Figure 6.5(a) and (b) illustrate the ranges prescribed by the values of n and w. It

Ring speed (c)	# of nodes (n)	Normalized node-to-node delay (w')
1	5	0.00477
4	5	0.00763
16	5	0.0119
1	500	0.00382
4	500	0.00385
16	500	0.00396

Table 6.2. Range of Normalized Token Node-to-Node Delay

is obvious that given the current token ring networks no protocol among the three can always outperform the others. Hence, for a given environment one should carefully select a protocol in order to achieve the best possible performance. Specifically, when w and n are small, the priority-driven protocol should be used; otherwise, the window protocol should be used unless (n, w) is in the upper-right corner in which the token passing protocol is the best choice.



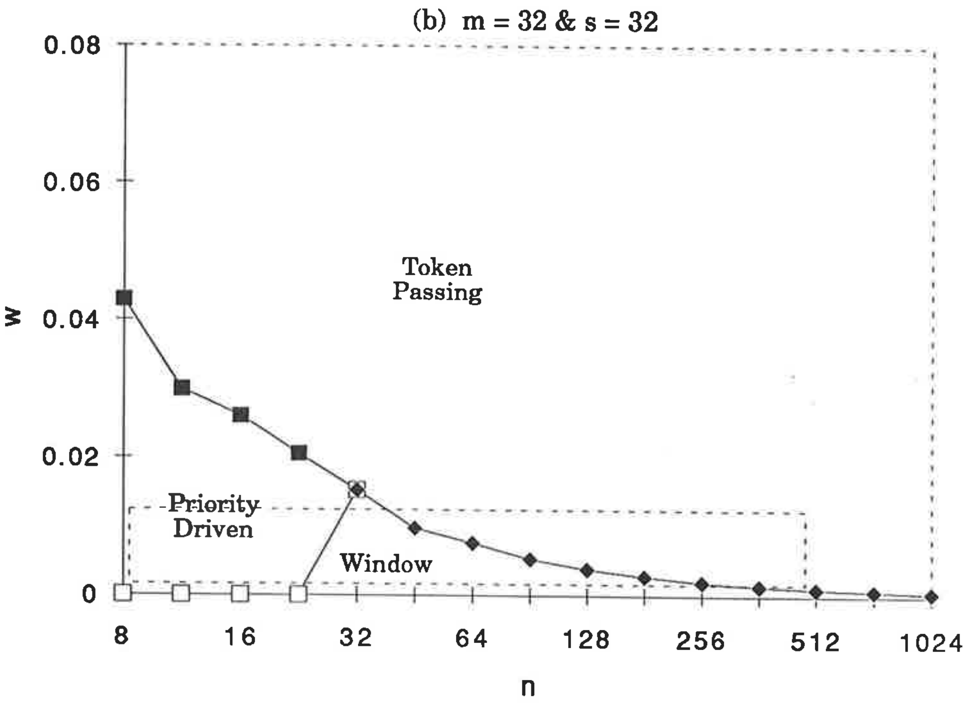


Figure 6.6. Parameter Range of Current Token Ring Networks

Chapter 7

Average Case Performance Comparison

The worst case performance of the three protocols analyzed in the previous chapters has deterministically established a lower bound of the performance ratio for each protocol. However, this performance ratio is pessimistic and characterizes only the protocol behavior under the worst case scenario. This chapter deals with the average case performance of the three protocols by means of simulation. The average case performance reflects the protocol performance under normal operational environments. Therefore, it forms an integral part of the performance evaluation of a protocol.

Our goal is to investigate the performance of each protocol in supporting real-time heterogeneous traffic under various network loads, traffic mix, and network and protocol parameters [47]. First, we introduce the simulation model and simulation language used. We then describe the traffic model on which the simulation experiments are conducted. Finally, we present, compare and discuss various simulation results.

7.1 Simulation Program

We have developed a discrete-event simulation program written in SIMSCRIPT II.5. SIMSCRIPT II.5 is a discrete event-driven simulation language and was selected in view of the following advantages:

- programming flexibility and portability,
- modularity and structured programming,
- built-in data collection and analysis,
- real-time event scheduling capacities, and
- on-line debugging.

Figure 7.1 is a simplified flow chart of the simulation program. At the beginning of the simulation, network, protocol, message and simulation parameters are read in. Then the initialization routine is called to activate all processes and to initialize all parameters used in the simulation. The simulation is initiated by the creation of message generators which schedule subsequent message arrivals according to the specified arrival distribution. The control then enters the token ring emulation program. At the end of each simulation run, various statistics are collected and written to an output file.

The simulation program can be divided into eight major modules defined as either *processes* or *routines* in SIMSCRIPT. The functionalities of these processes and routines are described as follows.

- Routine *read_data* reads in network, protocol, message and simulation parameters specified by users. They include the simulation time, number of simulation

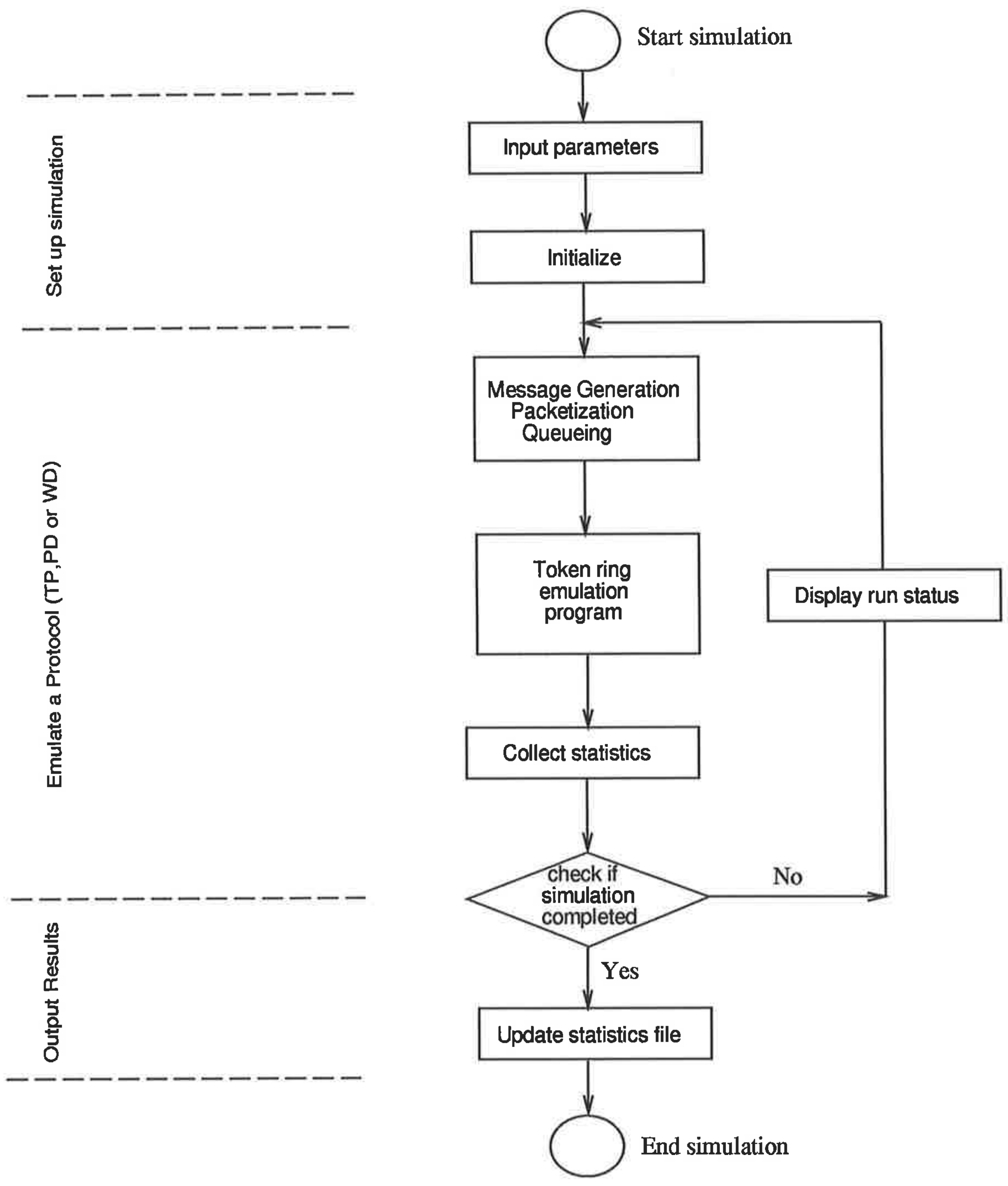


Figure 7.1. Simulation Program Flow Chart

runs for a given set of parameter values, debugging level, number of nodes, ring length and propagation delay, system offered load, message length, position and deadline distributions, message arrival rate, etc.

- Routine *initialization* activates all the processes, creates all the permanent entities and initializes all the system variables used in the simulation. This routine is first called at the time when simulation starts and is called repeatedly at the end of ecah simulation run to reset the system variables and reactivate processes.

- Process *msg_generator* is responsible for generating messages throughout the simulation. The process schedules the next message generation according to the given message inter-arrival time distribution and assigns each message an arrival time, a message class, a deadline and a length. In our model, each node has a message generator which generates five classes of messages.

- Process *ring_mgr* is responsible for managing the ring states and co-ordinating the activities of all the nodes in the network. They include passing token from one node to the next, removing messages whose deadlines have expired, calling designated routines to update statistics, etc.

- Routine *access_control* is called whenever a node sees a token. It checks whether the node should capture the token and send a message, or only modify the token fields, or simply let the token pass by without any change.

- Routine *msg_trans* is entered whenever a message transmission takes place. It updates various ring, message and node variables.

- Routine *update_states* updates relevant statistics, e.g. number of messages sent, ring utilization, etc. whenever a successful message transmission has taken place.

- Process *get_results* awakes at the end of each simulation run to record statistics. The overall results for the runs are displayed or saved in the output files. This process is first called at the end of the warm-up period to reset the statistical variables and then at the end of each simulation run.

7.2 Traffic Model and Parameters

To evaluate and compare the average case performance of the three protocols, we have chosen a traffic model based on a local area network operating in a typical real-time manufacturing environment [60].

The profile of this traffic model is given in Table 7.1. Traffic generated by sources includes voice, data and various control messages. Each message class is assumed to have an independent Poisson arrival process. In our model, messages are packetized before they are transmitted. Packets of length 240, 1024 and 8192 bits are used for different messages classes. The reason to use short and long packets is to maximize the ring efficiency for lengthy data transfer and to minimize the loss for short and urgent messages. Message deadlines range from 0.9 *msec*[1] for short alarm messages to 50 *msec* for lengthy file transfer messages.

Note that the packet length shown in Table 7.1 includes both payload and an overhead of 13 *bytes* as in the IEEE 802.5 standard. A packet with a smaller information field is padded with zeros, so that there are no partial packets. Figure 7.2

[1]This is a relative deadline which indicates that the message needs to be received within 0.9 *ms* from its arrival at a node.

Class Id	**Traffic Type**	**Msg Length (bits)**	**Pkt Length (bits)**	**# of Pkts**	**% of Traffic**	**Deadline (msec)**
1	File Transfer	16000 – 32000	8192	2 – 4	0.27	50
2	File Transaction	1600	1024	2	5	20
3	Telephone	2000	1024	2	37	15
4	Sensor	240	240	1	57	5
5	Alarm	240	240	1	0.73	0.9

Table 7.1. Traffic Profile

shows the token AC field used for each protocol. Although it is different for each protocol, in the simulation a token length of 3 *bytes* is assumed for all protocols.

With the above five traffic classes, we define the network *offered load* as

$$\rho = \sum_{i=1}^{n} \sum_{j=1}^{5} \lambda_{i,j} \times 1/\mu_j, \tag{7.241}$$

where n is the total number of nodes in the network, $\lambda_{i,j}$ is the arrival rate of message class j on node i, and $1/\mu_j$ represents the transmission time of message class j. In the simulation, it is assumed that the arrival rate of class j message is the same for each node, that is,

$$\lambda_{1,j} = \lambda_{2,j} = \ldots = \lambda_{n,j}. \tag{7.242}$$

Nodes are uniformly distributed along a *1-km* long ring which operates at a speed of 1, 4 and 16 *Mbit/s*. We assume 4 *bits* latency delay at each node and a medium propagation delay of 5 *μs/km*.

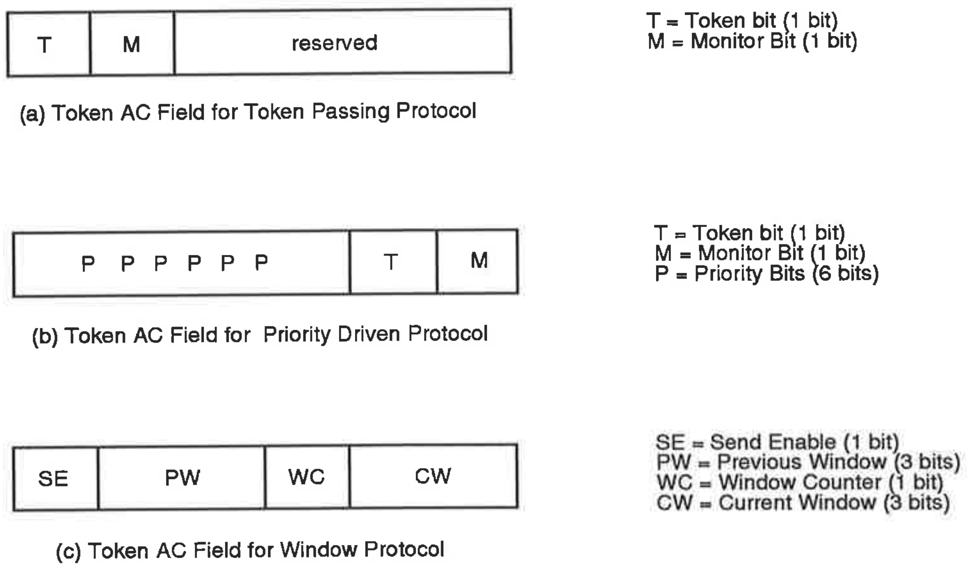

T
M
reserved
T = Token bit (1 bit)
M = Monitor Bit (1 bit)
(a) Token AC Field for Token Passing Protocol
P P P P P P
T
M
T = Token bit (1 bit)
M = Monitor Bit (1 bit)
P = Priority Bits (6 bits)
(b) Token AC Field for Priority Driven Protocol
SE
PW
WC
CW
SE = Send Enable (1 bit)
PW = Previous Window (3 bits)
WC = Window Counter (1 bit)
CW = Current Window (3 bits)
(c) Token AC Field for Window Protocol

Figure 7.2. Token Access Control Field Format

7.3 Performance Metrics

As in the worst case performance analysis, the performance metric to be used is the *sent ratio* which is defined as the percentage of messages successfully meeting their deadlines.

- For a class i messages, the sent ratio is defined as

$$r(i) = \frac{sent(i)}{lost(i) + sent(i)}, \qquad i = 1, 2, \cdots, 5, \tag{7.243}$$

  where sent(i) and lost(i) denote respectively the number of class i sent and lost messages. Note that for a messages consisting of multiple packets, unless all packets make their deadlines, the message is considered lost.

- To give an indication of the overall protocol performance for m message classes, we define the *average sent ratio* as

$$r(avg) = \frac{\sum_{i=1}^{5} sent(i)}{\sum_{i=1}^{5}(sent(i) + lost(i))}. \tag{7.244}$$

Using these metrics to measure the protocol performance, simulation results for the three protocols can now be analyzed.

7.4 Simulation Results

Message characteristics are summarized in Table 7.1. In each simulation run, statistics are reset after the transient phase and then collected after an additional 50,000 average message arrivals. Multiple runs of simulations are conducted to ensure that the maximum range (diameter) of 95% confidence intervals are kept within 10%.

In the following sections, we discuss simulation results obtained under various network, protocol and load parameters for the three protocols. Although we examine the sent ratios of all message classes, our discussions may focus more on the the sent ratio of class 5 messages, which indicates how well a protocol handle urgent messages. The simulation results are presented using tables and/or graphs whenever appropriate. This is to facilitate our discussion, as tables give accurate numerical numbers while graphs show the change in protocol performance more clearly.

7.4.1 Effect of Offered Load

Results in Tables 7.2(a)–(e) show the impact of the load change on the protocol performance for a 50-node ring operating at a speed of 1 *Mbit/s*. Each table compares the sent ratio of a particular message class for the three protocols under various offered loads.

(a): Class 1 Message Sent Ratio			
Offered Load	Token Passing	Priority-Driven	Window Based
0.50	0.992	0.991	0.934
1.00	0.924	0.910	0.664
1.50	0.821	0.781	0.611
2.00	0.754	0.695	0.588

(b): Class 2 Message Sent Ratio			
Offered Load	Token Passing	Priority-Driven	Window Based
0.50	0.996	0.995	0.996
1.00	0.966	0.955	0.927
1.50	0.881	0.841	0.801
2.00	0.803	0.750	0.722

(c): Class 3 Message Sent Ratio			
Offered Load	Token Passing	Priority-Driven	Window Based
0.50	0.992	0.992	0.997
1.00	0.946	0.936	0.925
1.50	0.833	0.807	0.798
2.00	0.749	0.726	0.723

(d): Class 4 Message Sent Ratio			
Offered Load	Token Passing	Priority-Driven	Window Based
0.50	0.983	0.986	0.992
1.00	0.907	0.921	0.959
1.50	0.770	0.825	0.882
2.00	0.695	0.776	0.823

(e): Class 5 Message Sent Ratio			
Offered Load	Token Passing	Priority-Driven	Window Based
0.50	0.888	0.927	0.945
1.00	0.685	0.817	0.868
1.50	0.558	0.760	0.831
2.00	0.540	0.726	0.801

Table 7.2. Effect of Offered Load

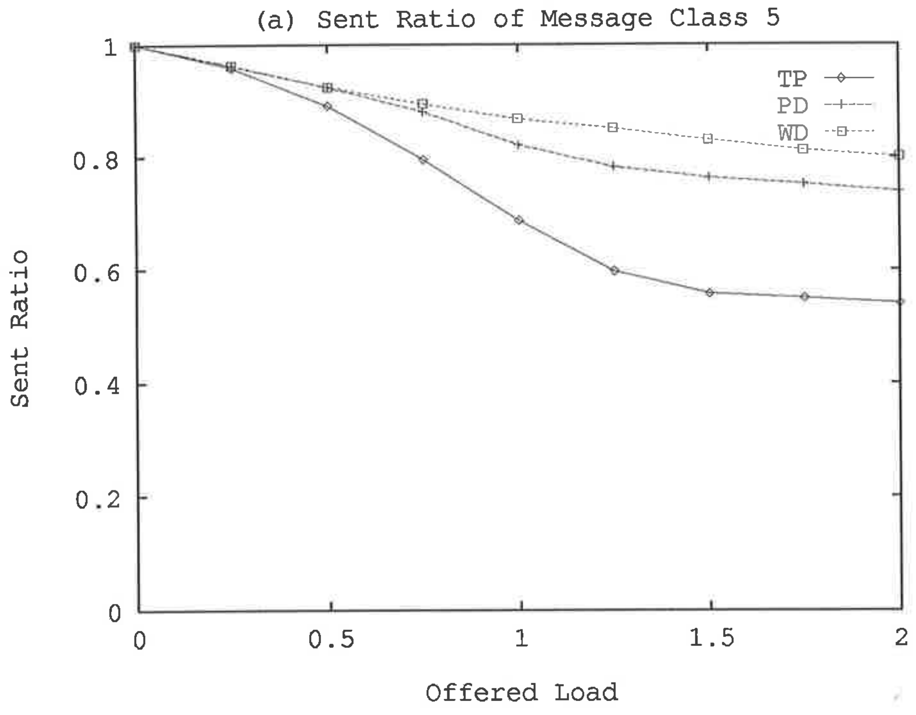
We see that for all protocols and message classes, the message sent ratios decrease as load increases, however the degree of the reduction varies for different protocols and message classes.

When the network is lightly loaded, each protocol achieves very high sent ratios for all message classes and the performance difference between one protocol and another is minimal. This is because under the light load there are few messages in the network and almost all of them are successfully transmitted. Hence, the effect of transmission policy is not obvious. On the other hand, we see that for each protocol message class 5 has the lowest sent ratio as compared with other message classes. This is because message class 5 has the most urgent deadlines which may not be met even under light load.

When the offered load increases from light to heavy, and eventually to sustained overload, all protocols have degraded performance. This is because as load increases, there are more messages competing for transmission. However, the load change affects each protocol and message class differently. In particular, message classes 1 and 5 are of interest to the following discussion.

When under sustained overload, the token passing protocol has a relatively high sent ratios for message class 1–4 while its sent ratio of message class 5 is as low as 0.540. This is in a sharp contrast with the picture of the window protocol. The latter maintains a high sent ratio of 0.801 for message class 5, though the sent ratio of its class 1 messages is 0.588 which is much lower than that of of token passing protocol. This can be explained as follows: although class 5 messages have the most stringent deadlines while class 1 messages have the largest deadlines, they are not differentiated by the token passing protocol. That is, in the token passing protocol messages are sent in the order of the token circulation regardless of their deadlines, while in the window protocol class 1 messages are not considered by the protocol if other class messages with smaller deadlines are waiting for transmissions. For the priority-driven protocol, its performance ranks between that of the token passing and the window protocols, due to the fact that it only implements the EDF policy approximately.

Figure 7.3(a) shows that the sent ratio of class 5 messages under the window protocol is substantially higher than those of the token passing and priority-driven protocols. The effect of different transmission policies can be readily seen. On the other hand, in Figure 7.3(b) we notice that the window protocol only outperforms the other two protocols marginally in terms of the average sent ratio. This implies that the token passing protocol may achieve a similar average sent ratio as any other protocols implementing the EDF transmission policy. However, the difference lies in whether or not a protocol can differentiate messages with regard to their deadlines, which is precisely the consequence of the transmission policy employed by a protocol. Figure 7.3(a) has revealed that token passing protocol yields a substantially low sent ratio for message class 5 which represents short, urgent and critical messages. In practice, a high loss ratio of these messages may impose severe penalty on the system performance, hence may not be acceptable to a real-time system which requires very high success rate in delivering critical control messages. Thus, in a network of mixed traffic the average message sent ratio alone is not sufficient as a performance index and it is imperative to grant the transmission right to urgent messages to ensure good quality service for this message class. Therefore, a good real-time communication protocol should maximize both the average message sent ratio and the sent ratio of urgent messages. In this respect, the proposed window protocol achieves better performance and quality of service than both the token passing and priority-driven protocols.



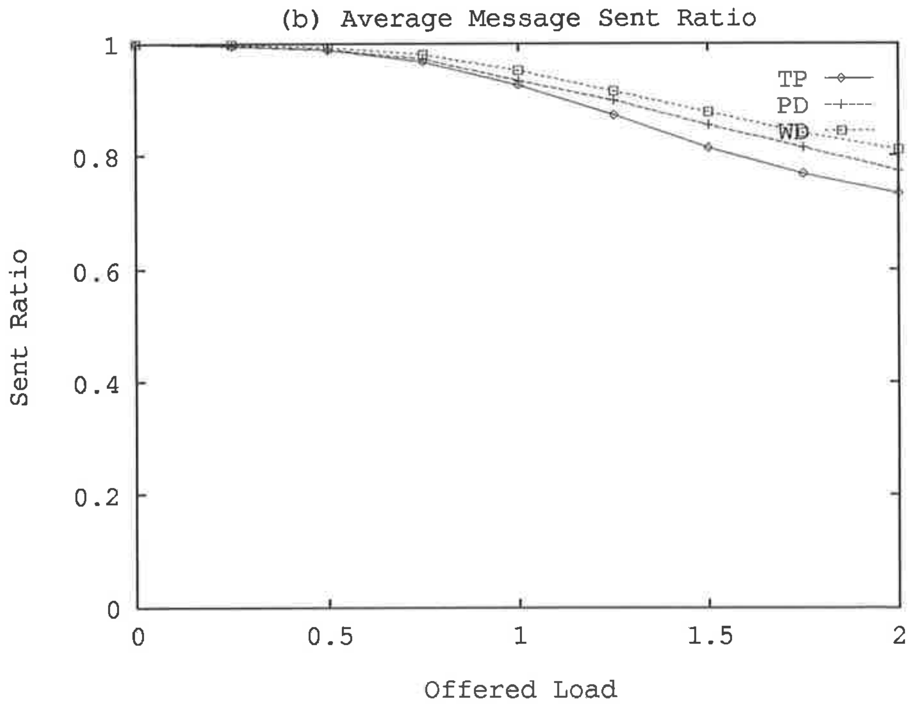


Figure 7.3. Effect of Offered Load

7.4.2 Effect of Ring Speed

Another important aspect of performance evaluation is to examine the protocol performance when operating at different ring speeds. With the advent of the optical fiber technology and its falling price, optical fiber will become the dominant transmission medium for future high-speed LANs/MANs. Hence, it is important to examine the suitability of the protocol for future high-speed networks.

Tables 7.3(a)–(e) show sent ratios of individual message classes for the three protocols. The ring has a population of 50 and an offered load of 1.25.

We see that the increase in ring speed results in substantial increase in the sent ratios for all message classes. This is because the increase in the ring speed is translated into reduction not only in message transmission time, but also in the delay introduced at each node when the node relays the token or a message. This eventually leads to a smaller contention overhead for each message transmission, and hence higher sent ratios.

We observe that as the ring gets faster, the sent ratio of each message class increases in general due to the reduction in message transmission time and contention overhead. However, the absolute values of the sent ratios vary from one protocol to another.

It is obvious that when the ring speed reaches 16 *Mbit/s* the token passing protocol achieves high sent ratios for message classes 1–4, but its sent ratio of message class 5 is still as low as 0.731. In contrast, the window protocol obtains very high message sent ratios for all message classes except for class 1 messages which have largest deadlines. This is because although the reduction in message transmission time and contention overhead brings in increase in the sent ratios for the token passing protocol, the penalty of not implementing the EDF policy is still a dominant factor. The impact

(a): Class 1 Message Sent Ratio			
Ring Speed	Token Passing	Priority-Driven	Window Based
1	0.873	0.861	0.620
4	0.934	0.893	0.673
16	0.934	0.799	0.699
100	0.934	0.799	0.999

(b): Class 2 Message Sent Ratio			
Ring Speed	Token Passing	Priority-Driven	Window Based
1	0.928	0.917	0.856
4	0.930	0.904	0.880
16	0.933	0.909	0.907

(c): Class 3 Message Sent Ratio			
Ring Speed	Token Passing	Priority-Driven	Window Based
1	0.892	0.875	0.856
4	0.905	0.882	0.876
16	0.926	0.908	0.904

(d): Class 4 Message Sent Ratio			
Ring Speed	Token Passing	Priority-Driven	Window Based
1	0.836	0.817	0.920
4	0.864	0.871	0.926
16	0.921	0.968	0.979

(e): Class 5 Message Sent Ratio			
Ring Speed	Token Passing	Priority-Driven	Window Based
1	0.597	0.611	0.851
4	0.607	0.735	0.920
16	0.731	0.852	0.976

Table 7.3. Effect of Ring Speed

of transmission policy is most pronounced in its low sent ratio of class 5 messages, which have the most urgent deadlines. On the other hand, apart from the reduced message transmission time the window protocol has the advantage of implementing the exact EDF policy under which class 5 messages are always privileged and class 1 messages discriminated against. Furthermore, the increase in ring speed brings down the contention overhead to a higher extent in the window protocol than that in the token passing protocol. Hence, the window protocol performs much better than the token passing protocol in terms of supporting transmission of urgent messages. Finally, the priority-driven protocol also has much improved performance in terms of the sent ratio of message class 5. Although its performance is better than that of the token passing protocol, it is still much lower than that of the window protocol. This is because the priority-driven protocol only implements the EDF policy approximately and the reduction in the contention overhead is not as much as that in the window protocol.

Figure 7.4 shows the effect of high speeds on the sent ratio of message class 5 for each protocol. We see that when the ring operates at 100 *Mbit/s*, the window protocol achieves a sent ratio as high as 0.980 while the token passing protocol still has a low sent ratio of 0.766. The priority-driven protocol comes very close to that of the window protocol. This demonstrates that although increasing the ring speed improves the protocol performance, within a certain speed range the fundamental factor in determining the protocol performance in supporting urgent message transmission is the transmission policy used. However, when the ring speed increases further all three protocols are able to achieve a sent ratio of 1. This is because under very high speeds, the reduction in message transmission time and contention overhead as a result of speed increases overshadows the impact of transmission policy.

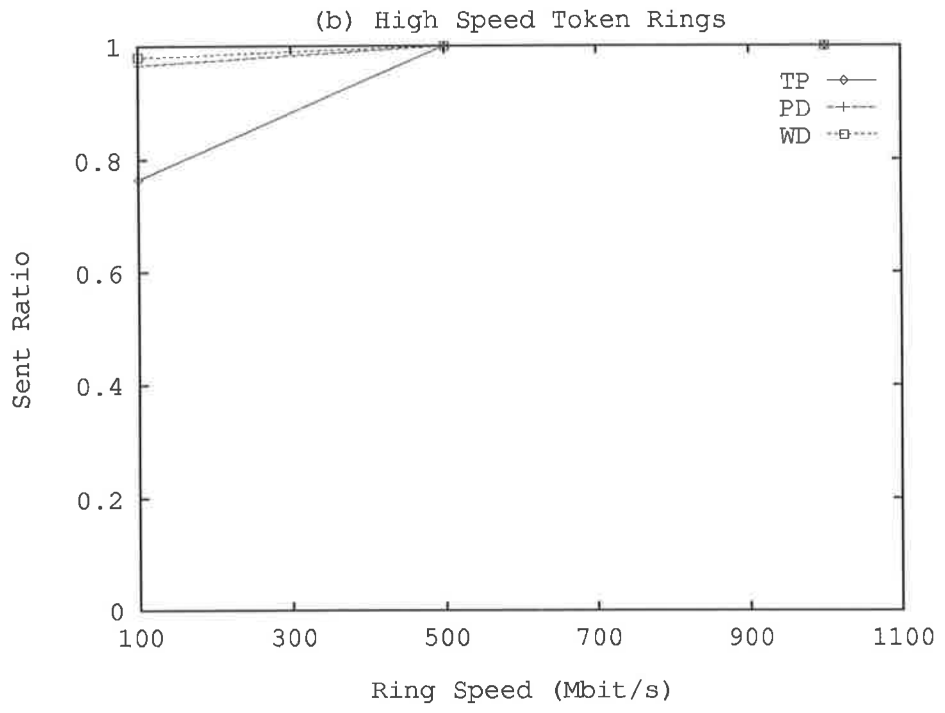


Figure 7.4. Effect of Ring Speed

7.4.3 Effect of Ring Population

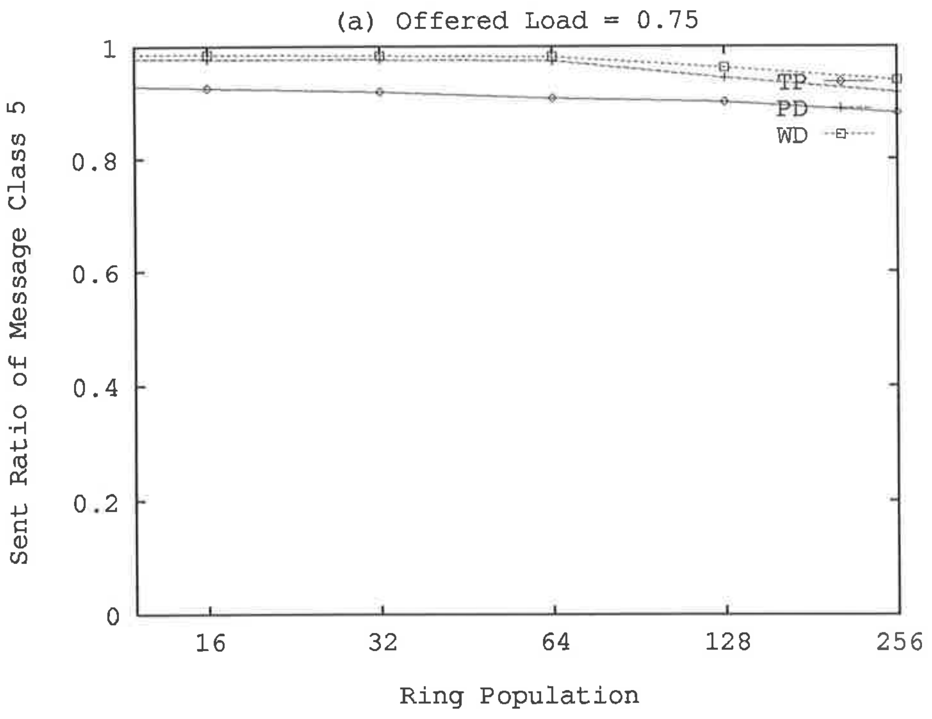
In the worst case performance analysis, we see that the ring population (i.e. the number of nodes in the ring) is an important factor affecting the protocol performance. In this section, we examine the impact of the increase in the ring population on the protocol performance. Figures 7.5(a) and (b) compare the sent ratio of message class 5 for each protocol when the offered loads are 0.75 and 1.5 respectively. The ring operates at a speed of 4 *Mbit/s*. We see in both figures that the window protocol performs the best and is least sensitive to the increase of the number of nodes, while the token passing protocol is most sensitive to the change. It is also clear that under lighter load the change in ring population affects the protocol performance to a lesser extent. Hence, in the following we concentrate our discussion on the case where the offered load is 1.5.

The results are shown in Tables 7.4(a)–(e). We observe that the common trend is that the sent ratio of each message class decreases as the network population increases. This is because the increase in the number of nodes yields an increased token walk time, which leads to a larger contention overhead of message transmission. However, this increase may not be uniform across each protocol as the contention overhead incurred in each protocol is different.

For the token passing protocol, the contention overhead of a message transmission is the delay between the nodes where two consecutive message transmissions take place. The larger the number of nodes, the higher contention overhead it may incur. We see that this increase in contention overhead causes the sent ratio of class 5 messages to drop close to 20%. However, the sent ratios of message classes 1–3 are less sensitive to the increase in the ring population as they have much larger deadlines than class 5 messages.

The picture for the window protocol is quite different. Increasing the number of nodes has yielded only a moderate drop in the sent ratio of message class 5. This is because for the window protocol the contention overhead of a message transmission is a function of the ring population. Hence, the increase in ring population translates into an increase in the contention overhead and consequently a higher message loss. However, message class 5 still has relatively high sent ratio. This can be explained by the fact that the window protocol implements the EDF policy which always give privileges to messages with smaller deadlines.

The performance of the priority-driven protocol is similar to that of the window protocol except that the sent ratio of message class 5 is much lower. This is because the priority-driven protocol only approximates the EDF transmission policy.



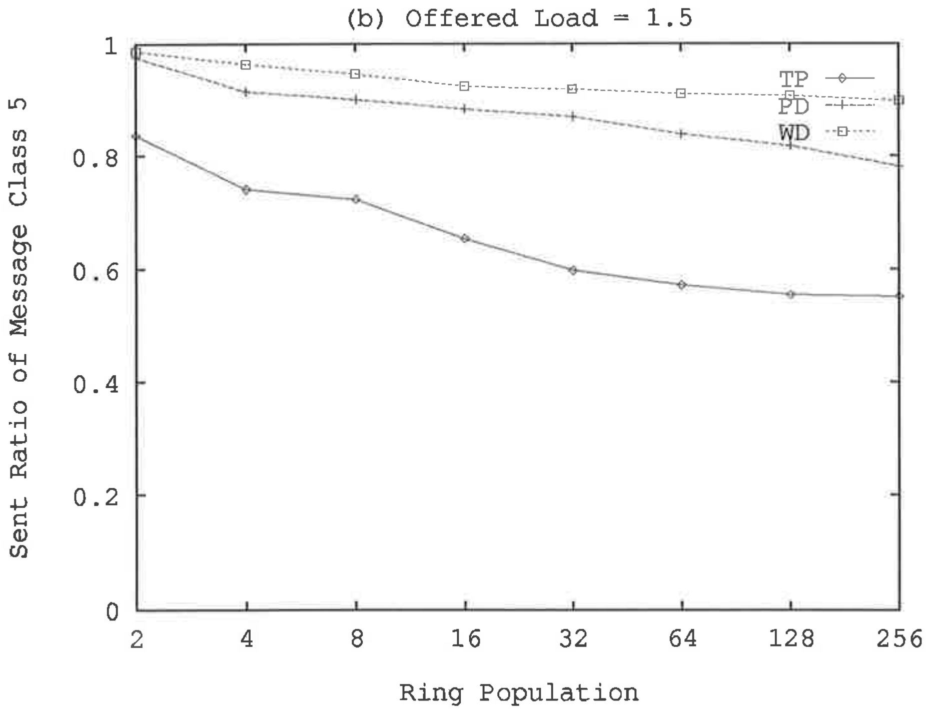


Figure 7.5. Effect of Ring Population

(a): Class 1 Message Sent Ratio			
Number of Nodes	Token Passing	Priority-Driven	Window Based
4	0.924	0.708	0.657
16	0.923	0.678	0.647
64	0.922	0.644	0.635
256	0.903	0.611	0.605

(b): Class 2 Message Sent Ratio			
Number of Nodes	Token Passing	Priority-Driven	Window Based
4	0.875	0.835	0.841
16	0.864	0.809	0.829
64	0.855	0.776	0.754
256	0.846	0.711	0.657

(c): Class 3 Message Sent Ratio			
Number of Nodes	Token Passing	Priority-Driven	Window Based
4	0.867	0.835	0.836
16	0.841	0.806	0.823
64	0.837	0.762	0.778
256	0.833	0.728	0.716

(d): Class 4 Message Sent Ratio			
Number of Nodes	Token Passing	Priority-Driven	Window Based
4	0.834	0.914	0.946
16	0.799	0.906	0.925
64	0.789	0.863	0.909
256	0.732	0.756	0.897

(e): Class 5 Message Sent Ratio			
Number of Nodes	Token Passing	Priority-Driven	Window Based
4	0.741	0.915	0.964
16	0.654	0.884	0.925
64	0.572	0.839	0.911
256	0.552	0.782	0.899

Table 7.4. Effect of Ring Population

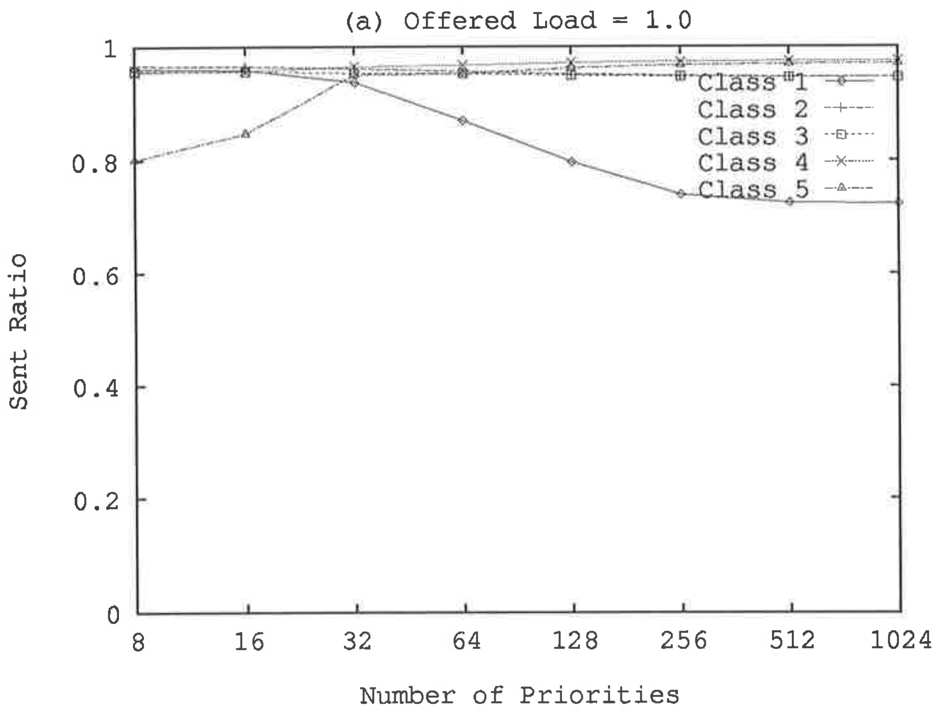
7.4.4 Effect of Protocol Parameters

In the preceding sections, we have examined the impact of three system parameters, namely the offered load, the ring speed and the ring population, on the protocol performance. From the worst case performance analysis, we know that the protocol performance is also affected by parameters associated with a particular protocol. For example, in the priority-driven protocol, message sent ratios may vary considerably for different number of priorities or priority assignment functions. For the window protocol, we have also predicted that the initial window size and the initial window lower bound could affect the protocol performance. In this section, we examine the impact of these protocol parameters on the protocol performance. This investigation should give us insights into whether or not protocol parameters need to be fine-turned in order to achieve the best possible performance.

7.4.4.1 Effect of Number of Priorities

Figures 7.6(a) and (b) show message sent ratios of the priority-driven protocol. The ring has 30 nodes, operates at 4 *Mbit/s* and has an offered load of 1.0 and 1.25 respectively.

It is obvious that the increase in the number of priorities affects the sent ratio of each message class differently. In both figures we observe that when the number of priorities increases, there is a significant increase of sent ratio for class 5 messages, while the sent ratio of class 1 message drops sharply. This is because when there are fewer priorities, more class 5 messages are assigned same priorities as other class messages though the former have smaller deadlines. Consequently, messages with larger deadlines such as class 1 messages may be transmitted before class 5 messages.



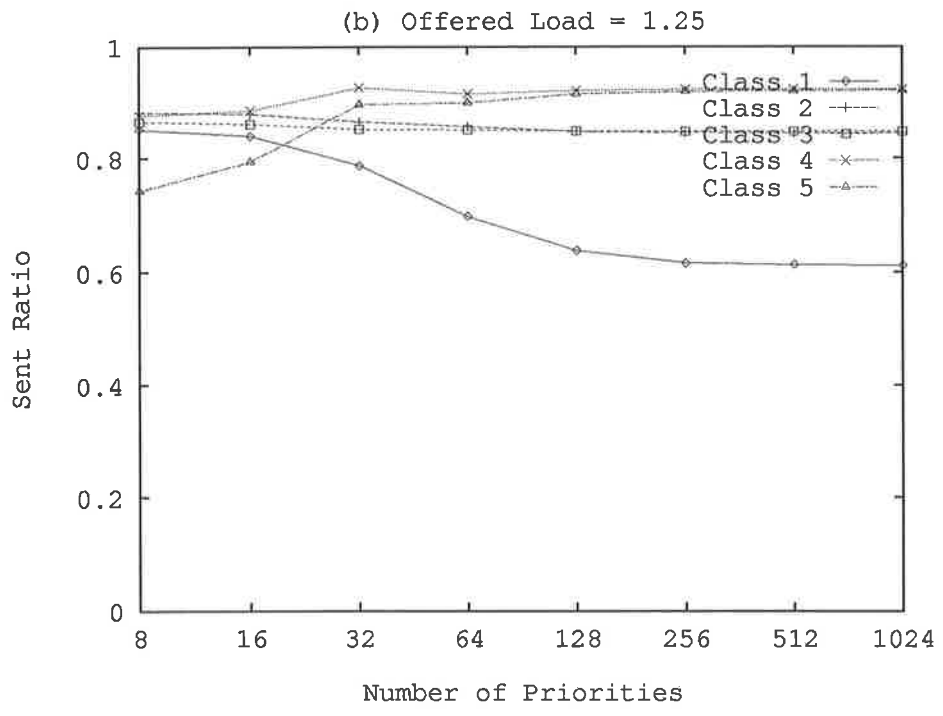


Figure 7.6. Effect of Number of Priorities

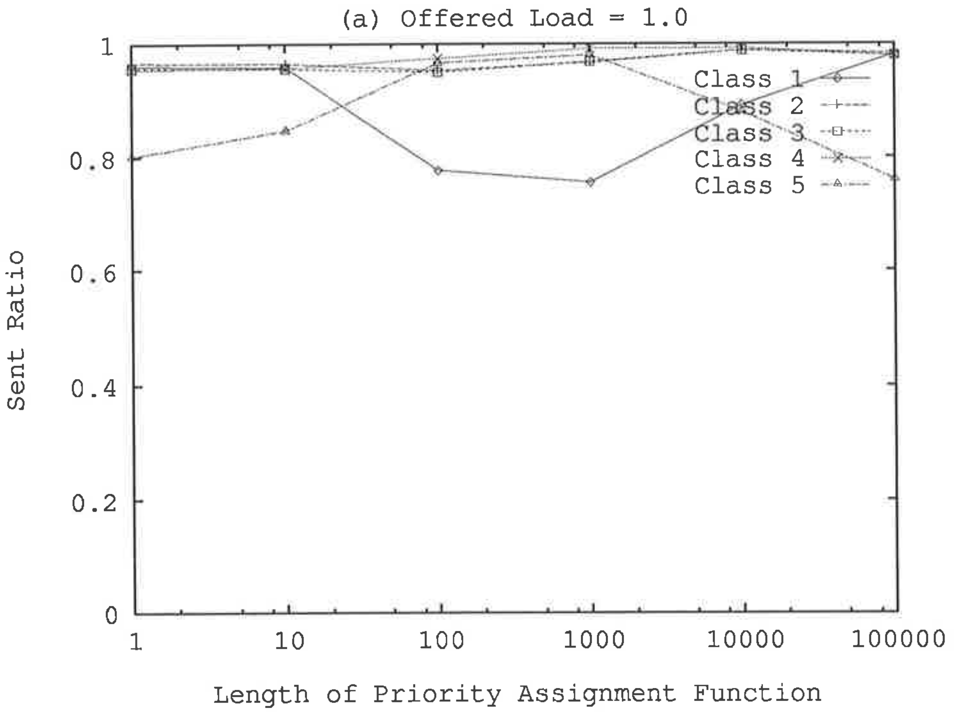
On the other hand, increasing the number of priorities results in greater accuracy in implementing EDF policy, hence more class 5 messages are distinguished from other class messages and meet their deadlines successfully. As a result, their sent ratios increase while sent ratios of message class 1 drops significantly.

We notice that when the number of priorities increases to 512, the sent ratio of message 5 reaches as high as 0.970 and 0.921 respectively, however further increasing the number of priorities to 1024 does not result in much improvement in sent ratios for all message classes. This demonstrates that when operating with a sufficiently large number of priorities, the priority-driven protocol can achieve a reasonable performance under certain load condition in supporting real-time message transmission.

7.4.4.2 Effect of Length of Priority Assignment Function

Although various forms of priority assignment functions are feasible provided they are non-decreasing and many-to-one, the form defined in (4.88) is the most common one whose argument is the length of the mapping function. In the following, we examine the impact of this length.

Figures 7.7(a) and (b) show the results. The ring has a population of 30, operates at a speed of 4 *Mbit/s* and employs 16 priority levels. We see that changing the length of the priority assignment function does not cause a monotonic increase of sent ratios. In both figures, we observe that the sent ratios of message classes 1 and 5 are most affected by the increase in the length. More specifically, the curve corresponding to the sent ratio of message class 5 goes from low to high and then from high to low, reaching the highest when the length is near 1000. On the other hand, the sent ratio of class 1 messages takes the reverse trend, dropping to its minimum when the length is near 1000. This demonstrates that for a given network configuration and message



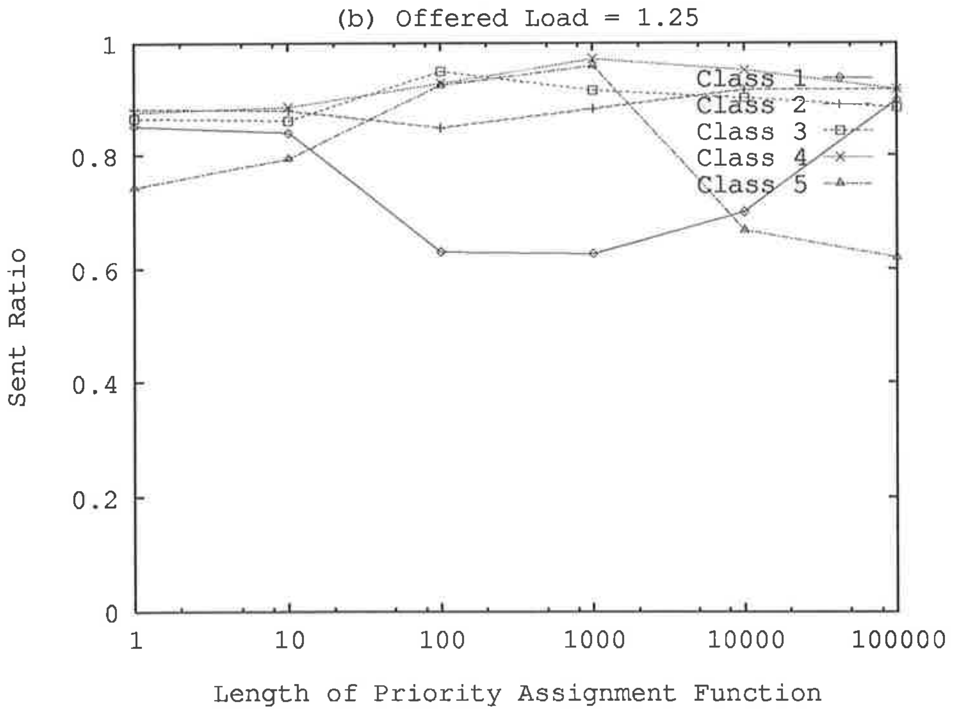


Figure 7.7. Effect of Length of Priority Assignment Function

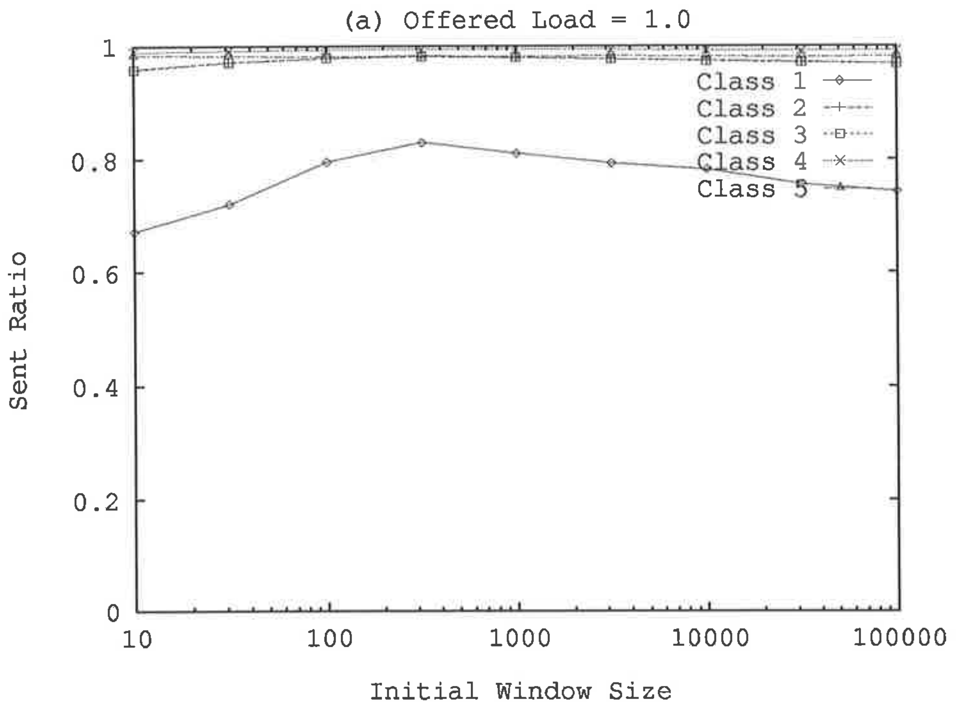
deadline distribution, there exists an "optimal" length with which the protocol can achieve the highest sent ratio for urgent messages. This is because when the length is too small or too large, many class 5 messages are assigned the same priorities as other class messages. Consequently, critical messages can not be distinguished in terms of transmission, which gives rise to lower sent ratios of class 5 messages.

7.4.4.3 Effect of Initial Window Size

Recall that in the window protocol described in Chapter 5, parameter α is the initial size of windows $W_2, W_3, \ldots, W_{s-2}, W_{s-1}$. All subsequent sizes of these windows are partly derived from this initial value which may affect the rate of the search for the earliest deadline message and the protocol performance. Figures 7.8(a) and (b) show the impact of α on message sent ratios when the offered load is 1.0 and 1.25 respectively. The ring has a population of 30 and operates at 4 *Mbit/s.* The number of windows used is 32.

In both figures, we observe that message sent ratios are not very sensitive to α except that of class 5 messages in Figure 7.8(a). We see that as α increases to 1000, the sent ratio of class 5 messages has a significant improvement, however when α increases further the trend is now reversed. We notice that the sent ratio of class 5 messages is decreasing while other messages classes have the improved performance. Clearly, there exists an "optimum" value of α (around 1000 in this case) with which the sent ratio of class 5 messages reaches its maximum.

We now investigate why the protocol displays such behavior as described above. Recall that after transmitting the earliest deadline message, the window is set to the initial size α before being used to locate the next message. Hence, one may prefer to use a smaller initial window size to reduce the number of window partitions in order



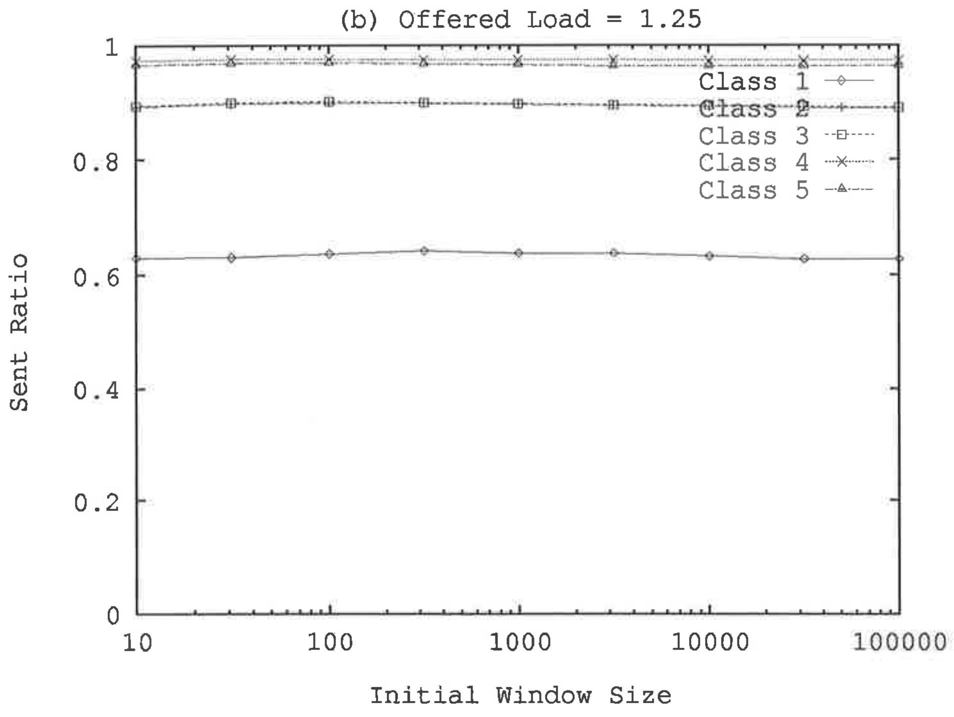


Figure 7.8. Effect of Initial Window Size

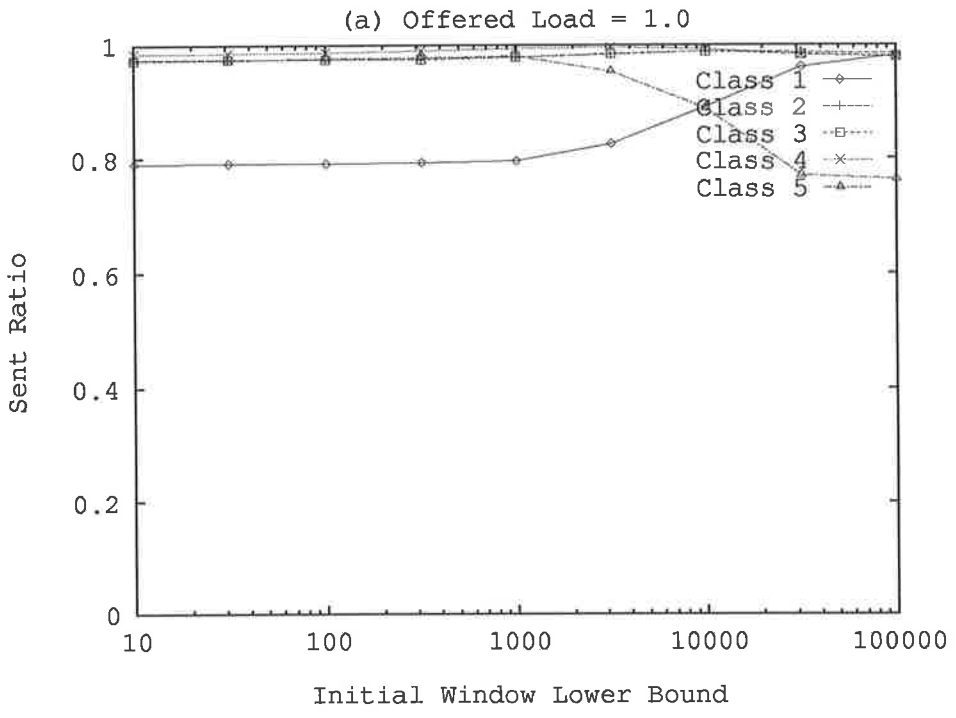
to speed up the search for the earliest deadline message, however this is not always the case. For example, when all messages are in the last window, a smaller window size would result in more (partial) partitions of the last window and slower convergence rate. This explains why an optimum value of α exists. However, we also noticed that in a relatively wide range the network performance stays very close to the optimum. This indicates that it is not difficult to select an α value which yields a good and stable protocol performance under various load conditions.

7.4.4.4 Effect of Initial Window Lower Bound

In Section 5.8, we see that in the enhanced window protocol a node is allowed to send a message immediately if the message deadline is smaller than the window lower bound. Therefore, the window lower bound reflects how the enhanced window protocol deals with urgent messages. Consequently, the choice of its initial value δ determines the trade-off between the degree of the approximation of the EDF policy and the reduction in the contention overhead.

Figures 7.9(a) and (b) display message sent ratios when the offered load is 1.0 and 1.25 respectively. The ring has 30 nodes and operates at 4 *Mbit/s*. The number of windows used is 32 and the initial window size is 100.

It is obvious that for each message class, there exists an "optimum" value of δ which maximizes its sent ratio. In both figures, the sent ratio of message class 5 reaches its maximum when δ increases to near 1000 and then drops sharply when δ increases further. On the other hand, the sent ratio of class 1 messages increases monotonically as δ increases. This is because when δ is too small only few urgent messages are transmitted immediately upon the token arrival, as a result the contention overhead may be high which leads to a slightly low message class 5 sent ratio. When δ is too



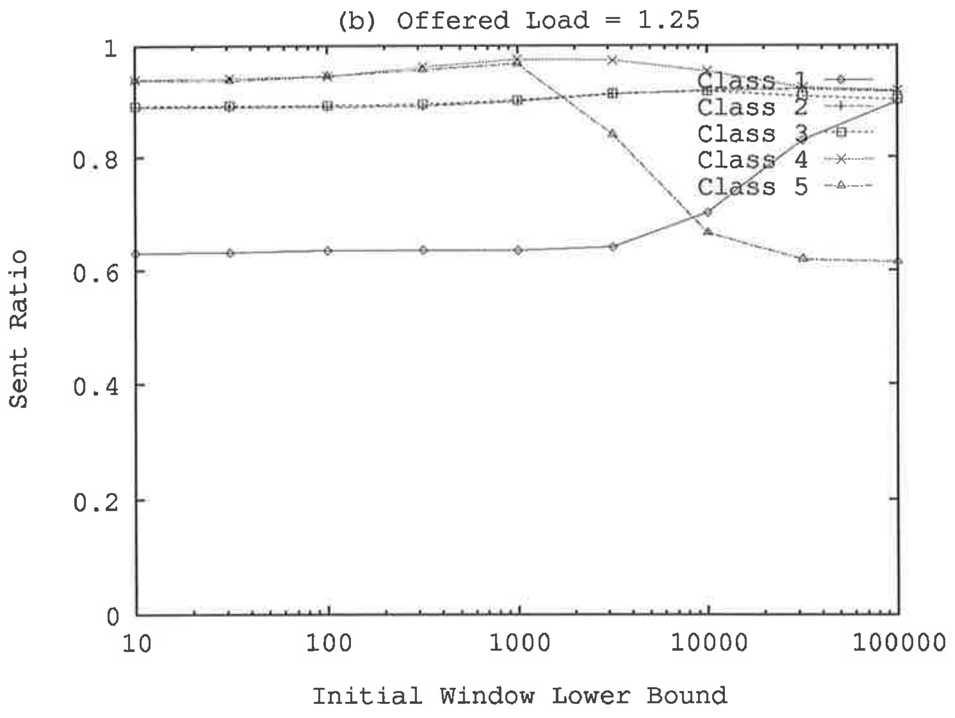


Figure 7.9. Effect of Initial Window Lower Bound

Message Sent Ratios					
δ	Class 1	Class 2	Class 3	Class 4	Class 5
10	0.921	0.991	0.992	0.995	0.994
100	0.917	0.992	0.992	0.997	0.996
1000	0.933	0.996	0.996	0.999	0.999
10000	0.997	0.999	0.999	1.000	0.991
100000	1.000	0.999	0.999	1.000	0.966

Table 7.5. Effect of Initial Window Lower Bound

large, the majority of messages are sent in the token rotation order rather than in the EDF order, In this case, messages with larger deadlines may take the advantage at the cost of urgent messages, which leads to a low message class 5 sent ratio.

Table 7.5 shows the results for the same configuration except now the ring speed is 16 *Mbit/s.* We see that the higher speed greatly reduces the impact of δ on the protocol performance. This suggests that when operating in high-speed environment, the choice of δ is not as critical as in a low-speed environment.

7.5 Discussions

In this chapter, we have evaluated the average case performance of the three protocols through simulation. The experiments have given us insights into the protocol behavior under various network and protocol parameters, message characteristics and load conditions. We noted that the average case performance of a protocol was more complex and less predicatable than its counterpart in the worst case scenario.

The simulation results shown and discussed have demonstrated quantitatively how each protocol performs in terms of message sent ratios. We conclude that for a real-time system required to maximize the sent ratio of critical and urgent messages, the proposed window protocol is the best choice. The priority-driven protocol may also be acceptable provided the number of priorities is sufficient and the length of the priority assignment function is adequate. The simple token passing protocol is not suitable for supporting real-time urgent message transmission, but can achieve comparable average message sent ratio.

Finally, it is important to note that when the ring gets faster, the difference in the performance of the three protocols is reduced.

Chapter 8

Conclusions and Recommendations for Future Research

In this final chapter, we summarize the results obtained from this research, outline its significance and contribution and propose future research direction and agenda in the field of real-time communications.

8.1 Summary of Results

This thesis has addressed some important issues involved in real-time communications in token ring networks. We began our study by defining the problem of real-time communications. We identified that the most important objective of real-time message transmission was to meet individual message deadlines. We observed that the predominant approach taken by many existing token ring protocols for real-time communications was priority-driven in which real-time synchronous messages are given higher priorities at the cost of real-time asynchronous messages. As a result, most of these protocols could handle real-time synchronous messages well, but their performance in supporting real-time asynchronous messages might be poor. Thus, we argued that a protocol suitable for real-time message transmission must address the individual message deadlines explicitly. From the centralized scheduling theory we knew that the EDF policy was optimal, however we noted that implementing the EDF policy in a distributed system, such as a token ring network, not only may incur a non-negligible overhead, but also may be approximate. Hence, we concluded that in designing a distributed scheduling algorithm such as a communication protocol, issues in both achieving 'optimal' (e.g. EDF) scheduling policy and minimizing the scheduling overhead must be addressed.

In Chapter 2, we first introduced the network and message models together with their parameters and attributes. We then defined the protocol notations and performance metrics. Finally, we described the methodology to analyze the worst case performance of MAC protocols for real-time communications.

In Chapters 3, 4 and 5, we studied in detail three token ring protocols implementing variations of the EDF transmission policy respectively. First, we described a simple token passing protocol which sent messages in the nearest-neighbor-first order. As a result, this simple token passing protocol does not adhere to the EDF policy.

Next, we proposed a modified priority-driven protocol in an attempt to incorporate message deadlines. In this protocol, each message was assigned a priority based on its deadline and the highest priority message was always sent first. We observed that this protocol would approximate the EDF policy when the number of priorities was less than the number of deadlines.

We analyzed the worst case performance of the token passing and the priority-driven protocols for real-time message transmission. The analytical results demonstrated that the worst case performance of the two protocols was poor. Specifically, given a message set the two protocols could send only half of the messages even if the contention overhead was assumed to be zero. This implied that the dominant factor in deciding the protocol performance in supporting real-time message transmission was whether or not the transmission policy used considered the individual message deadlines explicitly. Thus, we concluded that it was imperative to design a token ring protocol which could implement the exact EDF policy.

As a result, in Chapter 5 we proposed a novel window token ring protocol which could implement the exact network-wide EDF policy. In our approach, the message deadline axis was divided into a number of non-overlapping windows. Once the earliest deadline message was located in the first non-empty window, the message was sent. This new window protocol for token ring networks differed significantly from the previously proposed window protocols for CSMA/CD networks in that it employed multiple windows and that the search for the earliest deadline message converged much faster. The worst case performance analysis of the new window protocol demonstrated that if the contention overhead is assumed to be 0, this protocol could achieve a performance ratio of 1. Thus, the contention overhead was the only factor degrading its performance.

In Chapter 6, we compared the worst case performance of the three protocols in the dimension prescribed by the number of nodes and the normalized token node-to-node delay, hoping to provide a guideline for selecting the best protocol for given operating parameters. We noted that no protocol always outperformed the others for the entire parameter ranges considered and that each protocol had its own applicable region where its performance was the best. These observations implied that the performance of a distributed scheduling algorithm or a communication protocol was determined not only by the transmission policy employed, but also by the contention overhead incurred when implementing such a policy. As a result, there might not exist a communication protocol for real-time message transmission which would be optimal for all operating environment. Hence, in designing a distributed scheduling algorithm, such as a communication protocol, one should seek a balance in achieving an optimal scheduling policy and minimizing the scheduling overhead.

In Chapter 7, we studied the average case performance of the three protocols through simulation. From the simulation results, we observed that the new window protocol was predominantly superior to the other two under the cases studied. The token passing protocol, as a result of not considering message deadlines, had the lowest sent ratio, while the priority-driven protocol obtained a sent ratio in between as it approximated the EDF policy in some cases. This meant that under the current technologies the window protocol was the best choice in terms of the average case performance. Furthermore, we also noted that when the ring gets faster, the difference in the performance of the three protocols is reduced. It implied that under very high speeds the benefit of implementing the EDF policy by the window protocol was overshadowed by its relatively high contention overhead and that using the simple token ring protocol might be sufficient.

8.2 Significance and Contribution

This research has provided a systematic investigation of real-time communications in token ring networks. The work has advanced the state of the art of research in the distributed scheduling and real-time communications in two distinct respects. Firstly, we addressed and revealed the fundamental difference between distributed real-time scheduling and centralized real-time scheduling. Secondly, we proposed and studied three token ring protocols implementing variations of EDF transmission policy to support real-time communications. In particular, the new window protocol offers much needed integration in supporting both real-time synchronous and asynchronous messages by considering individual message deadlines explicitly. Furthermore, we carried out performance evaluation and comparison of the three protocols and showed how to select the best protocol for given operating parameters. The following outlines the major contributions made by this research.

- For the first time, we have addressed in depth the issue that a distributed scheduling environment is fundamentally different from a centralized scheduling system and revealed the importance of seeking a balance in implementing the "optimal" scheduling algorithm and minimizing the scheduling overhead.

- We have proposed a modified priority-driven token ring protocol which can be easily incorporated to the existing 802.5 token ring standard to support real-time communications. We have shown that with a sufficient number of priorities, this modified priority-driven protocol can achieve a reasonable performance in transmitting real-time messages.

- We have proposed a new window protocol which is the first of its kind for the token ring network. It takes a novel approach using multiple non-overlapping windows offering fast convergence to the earliest deadline message. This new window protocol provides a much needed integrated approach for supporting real-time message transmission. We have also devised an effective coding scheme which can realize the proposed protocol with a token of limited length. Simulation results have demonstrated that this protocol achieves an excellent performance under current network technologies.

- Much of the research in real-time communications uses simulation to evaluate the protocol performance. While simulation is an effective tool, it does not provide insight into protocol performance under the most unfavorable operating conditions, which is crucial for a real-time system. This research studied not only the average protocol performance by simulations, but also the worst case performance which provides a lower bound to the performance ratio. The results can be readily used to predict the worst case protocol performance once the various parameters are given, and hence can serve as a guide to the protocol selection.

- This research has also developed a simple but effective methodology to analyze the worst case performance of protocols. The importance of this methodology is that it can be effectively applied to the worst case analysis of general communication protocols. Therefore, it is a valuable tool which can be used for future research in this field.

8.3 Recommendations for Future Research

Currently, we are studying one extension of this work, which is an adaptive token ring protocol combining the advantages of the token passing, priority-driven and window protocols. This is because if the size of all windows is 1, then the window protocol is equivalent to the priority-driven protocol and if the size of the first window is larger than the maximum message deadline, then the window protocol is equivalent to the token passing protocol. The transition between the window protocol and the other two protocols can be easily achieved by adding two control functions to the window protocol, namely *traffic monitor* and *configuration monitor*. Whenever the traffic monitor detects the offered load reaches a certain threshold or the network configuration changes, the network enters the corresponding protocol operation mode by changing the protocol parameters. Such an adaptive protocol should give the best possible performance according to the offered load and network configuration.

Although our study concentrated on real-time communications in token ring networks, the methodology developed in this study can be extended to many other network environments. Currently, there is great interest in using high-speed networks, such as FDDI, DQDB and ATM networks, to support real-time applications. We now briefly discuss the issues to be addressed when using these network for real-time applications.

- FDDI is an ANSI standard for 100 Mbits/s optical fiber token ring based on a timed token access method. FDDI supports both synchronous and asynchronous messages. The key to supporting transmission of real-time synchronous messages is the *synchronous capacity allocation scheme*, which determines how long a node is allowed to transmit its synchronous messages every time it captures the token. However, how to deal with asynchronous real-time messages in FDDI networks is still an open question and is a very important research issue.

- DQDB is the IEEE 802.6 MAN subnetwork standard. It consists of two counter-flow point-to-point unidirectional optical fiber buses and is based on the distributed queue concept with prioritized access. DQDB differs from FDDI in its distributed control of message transmission and information collection. However, like FDDI, it provides guaranteed bandwidth for real-time synchronous messages, but real-time asynchronous messages are still disadvantaged. In addition, due to the unfairness evidenced in DQDB under heavy load, some real-time asynchronous messages may suffer from positional discrimination. Hence, DQDB alone may not be adequate to provide timely delivery of real-time messages.

- ATM has been specified as the transfer mode for the future B-ISDN. It uses fixed-size cell and is envisaged to provide both telecommunication and LAN type services. The Generic Flow control (GFC) protocol is intended to regulate multiple terminals within the Customer Premises Network (CPN) accessing the B-ISDN network through the User Network Interface (UNI). If the CPN is required to support real-time applications, the GFC protocol must have the ability to ensure the timely delivery of these real-time messages. So far, this aspect has not been addressed in the design of the GFC protocol.

One impact of high speed networking is that the message transmission time will be greatly reduced, whereas the relative contention overhead incurred will increase dramatically. Therefore, while the goals in designing real-time communication protocols remain the same, the philosophy and approach to solving the problem may be totally different. This will impose a new challenge, dimension and direction for future research in real-time communications.

Appendix A

Publications and Presentations

P. Potter, M. Zukerman, L. Wedding and **L. Yao**, "A Multi-Service Generic Flow Control Protocol", *Proc. The 1993 IEEE International Conference on Communications (ICC'93)*, Geneva, Switzerland, May 1993.

Zukerman, **L. Yao** and P. Potter, "DQDB Performance Under Sustained Overload with BWB and MRO", *Computer Communications*, Vol 16, No 1, January 1993.

Z. L. Budrikis, G. Mercankosk, M. Blasikievicz, M. Zukerman, **L. Yao**, and P. Potter, "Access Protocol for a Shared Medium", *Australian Teletraffic Research Journal*, Vol 26, No 2, November 1992.

L. Yao, W. Zhao and C.C. Lim, "An Efficient Window Protocol for Real-time Communications in Token Ring Networks", *Proc. The 2nd International Computer Conference, Data and Knowledge Engineering: Theory and Applications*, Hong Kong, December 1992.

C.C. Lim, **L. Yao** and W. Zhao, "Transmitting Time-Dependent Multimedia Data in FDDI Networks", *Proc. SPIE's International Symposium, OE/FIBERS'92*, Boston, September 1992.

P. Potter, M. Zukerman, L. Wedding and **L. Yao**, "A Proposed Generic Flow Control Protocol", *Proc. Australian Broadband Switching and Services Symposium'91*, Melbourne, July 1992.

Z. L. Budrikis, G. Mercankosk, M. Blasikievicz, M. Zukerman, **L. Yao**, and P. Potter, "A Generic Flow Control Protocol for B-ISDN", *Proc. The IEEE 11th International Conference on Computer Communications (INFOCOM'92)*, Florence, Italy, May 1992.

M. Zukerman, **L. Yao** and P. Potter, "Performance under Sustained Overload of DQDB with Bandwidth Balancing and Multiple Requests Outstanding", *Proc. The 5th IEEE Workshop on MANs*, Taormina, Italy, May 1992.

P. Potter, **L. Yao** and M. Zukerman, "A Revised Generic Flow Control Protocol for B–ISDN", Australian Contribution to CCITT Study Group XVIII, Working Party 8, Melbourne, Australia, December 1991.

P. Potter, **L. Yao** and L. Wedding, "Simulation Results of the Proposed GFC Protocol," Australian Contribution to CCITT Study Group XVIII, Working Party 8, Melbourne, Australia, December 1991.

M. Zukerman, **L. Yao** and P. Potter, "Performance of DQDB under Sustained Overload with Bandwidth Balancing and Multiple Requests Outstanding", *Proc. The 6th Australian Teletraffic Research Seminar*, Wollongong, Australia, November 1991.

M. Zukerman, **L. Yao** and P. Potter, "A Generic Flow Control Protocol for B–ISDN", *Proc. Australian Broadband Switching and Service Symposium'91*, Sydney, Australia, July 1991.

M. Zukerman, **L. Yao** and P. Potter, "A Generic Flow Control Protocol for B–ISDN", Australian Contribution to CCITT Study Group XVIII, Working Party 8, Geneva, Switzerland, June 1991.

C.C. Lim, **L. Yao** and W. Zhao, "A Comparative Study of Three Token Ring Protocols for Real-Time Communications", *Proc. The 11th International Conference on Distributed Computing Systems*, Arlington, Texas, May 1991.

L. Yao and W. Zhao, "Performance of an Extended IEEE 802.5 Protocol", *Proc. The IEEE 10th International Conference on Computer Communications (INFOCOM'91)*, Miami, April 1991.

L. Yao and W. Zhao, "Token Ring Protocols for Transmission of Time Constrained Messages", *Proc. The 5th Australian Fast Packet Switching Workshop*, Melbourne, Australia, July 1990.

L. Yao and W. Zhao, "Implementing the Minimum-Laxity-First Transmission Policy in a Real-Time Token Ring Network", *Proc. The 4th Australian Teletraffic Research Seminar*, Bond University, Australia, December 1989.

Bibliography

[1] IEEE Standard 802.5-1989, *Token Ring Access Method and Physical Layer Specifications*, 1989.

[2] ANSI/IEEE Standard 802.4-1985(ISO/DIS 8802-4), *Token-Passing Bus Access Method and Physical Layer Specification*, 1985.

[3] D. W. Andrews and G. D. Schulz, "A Token Ring Architecture for Local-Area Networks: an update", *Proc. COMPCON F82*, 1982.

[4] K. Arvind, K. Ramamritham and J. A. Stankovic, "A Local Area Network Architecture for Communication in Distributed Real-Time Systems", *The Journal of Real-Time Systems*, 3(2), May 1991.

[5] S. Bakry, B. El-Redaisy and M. Al-Turaigi, "Computer Simulation of a Packet Switching Computer Network", *Computer Communications*, Vol 11, No 3, June 1988.

[6] G. Carlow, "Architecture of the Space Shuttle Primary Avionics Software System", *Tutorial, Hard Real-Time Systems*, IEEE Press, 1988.

[7] W. Y. Cheng and J. Liu. "Performance of ARQ Schemes in Token Ring Network", *IEEE Transactions on Computers*, 37, July 1988.

[8] V. Cherkassky, H. Lari-Najafi and N. Lawrie, "Performance of a New LAN for Real-Time Traffic", *Computer Communications*, Vol 13, No 5, June 1990.

[9] S. Casale, V. Catania, A. Faro and N. Parchenkov, "Design and Performance Evaluation of an Optical Fibre LAN with Double Token Rings", *Computer Communications*, Vol 12, No 3, June 1989

[10] E. G. Economou, D. J. Mouzakis and G. Philokyprou, "Skipnet: A Two Channel Token Access Scheme", *Computer Communications*, Vol 12, No 5, October 1989

[11] A. Goyal and D. Dias, "Performance of Priority Protocols on High Speed Token Ring Networks", *Data Communication Systems and Their Performance*, Elsevier Science Publishers B. V (North-Holland), IFIP, 1988.

[12] D. T. Green and D. T. Marlow, "SAFENET – A LAN for Navy Mission Critical Systems", *Proc. The 14th Conference on Local Computer Networks*, Minneapolis, Minnesota, October 1989.

[13] J. Hong, X. Tan, and D. Towsley, "A Performance Analysis of Minimum Laxity and Earliest Deadline Scheduling in Real-Time System", *IEEE Transactions on Computers*, 38(12):1736–1744, December 1989.

[14] D. Towsley and G. Venkatesh, "Window Random Access Protocols for Local Computer Networks", *IEEE Transactions on Computers*, Vol C-31, No 8, 1982.

[15] J. H. Huang, C. W. Chen and M. C. Lee, "A Distributed, Fair and Efficient Protocol for Integrated Voice/Data Services on Token Ring Networks", *Proc. International Conference on Communications*, ICC'91, June 1991.

[16] E. D. Jensen, C. D. Locke, and H. Tokuda, "A Time-Driven Scheduling Model for Real-Time Operating Systems", *Proc. IEEE Real-Time Systems Symposium*, December 1985.

[17] C. S. Kang, E. K. Park and J. H. Herzog, "Hybrid Token Ring: A Load Sharing Local Area Network," *Computer Communications*, Vol 14, No 9, November 1991

[18] H. Kasahara and S. Narita, "Parallel Processing of Robot-Arm Control Computation on a Multimicroprocessor System", *Tutorial, Hard Real-Time Systems*, IEEE Press, 1988.

[19] B. G. Kim and D. Towsley, "Dynamic Flow Control Protocols for Packet-Switching Multiplexers Serving Multipacket Messages", *IEEE Transaction on Communications*, Vol COM-34, Nov 4, 1986.

[20] C. M. Krishna and Y. H. Lee, "Special Issue on Real-Time Systems", *IEEE Computer*, 24(5), May 1991.

[21] J. F. Kurose, M. Schwartz and Y. Yemini, "Multiple-Access Protocols and Time-Constrained Communications", *Computing Serveys*, Vol 16, No 1, March 1984.

[22] J. F. Kurose, M. Schwartz and T. Yemini, "Controlling Window Protocols for Time-Constrained Communications in Multiple Access Environment", *Proc. The 8th IEEE Data Communication Synposiums*, 1983.

[23] J. P. Lehoczky and L. Sha, "Performance of Real-Time Bus Scheduling Algorithms", *ACM Performance Evaluation Review, special issue*, Vol 14, No 1, May 1986.

[24] C. C. Lim, L. Yao and W. Zhao, "A Comparative study of Three Token Ring Protocols for Real-Time Communications", *Proc. The 11th International Conference on Distributed Computing Systems*, Arlington, Texas, May 1991.

[25] C. L. Liu and J. W. Layland, "Scheduling Algorithms for Multiprogramming in Hard Real-Time Environment", *Journal of ACM*, Vol 20(1), 1973.

[26] N. Malcolm, W. Zhao and C. J. Barter, "Guarantee Protocols for Communication in Distributed Real-Time Systems", *Proc. IEEE INFOCOM '90*, San Francisco, June 1990.

[27] N. Malcolm and W. Zhao, "Version Selection Schemes for Hard Real-Time Communications", *Proc. The 12th IEEE Real-Time Systems Symposium*, San Antonio, December 1991.

[28] D. Marinescu, "A Protocol for Multiple Access Communication with Real-Time Delivery", *Proc. IEEE INFOCOM'90*, San Francisco, June 1990.

[29] A. K. Mok and M. L. Dertouzos, "Multiprocessor Scheduling in a Hard Real-Time Environment", *Proceeding, The 7th Texas Conference on Computer Systems*, November 1978.

[30] J. K. Y. Ng and J. W. S. Liu, "Performance of Local Area Network Protocols for Hard Real-Time Applications", *Proc. The 11th International Conference on Distributed Computing Systems*, Arlington, Texas, May 1991.

[31] J. Schoeffler, "Distributed Computer Systems for Industrial Process Control", *Tutorial, Hard Real-Time Systems*, IEEE Press, 1988.

[32] K. C. Sevcik and M. J. Johnson, "Cycle Time Properties of the FDDI Token Ring Protocol", *IEEE Transactions on Software Engineering*, March 1987.

[33] K. G. Shin, "Special Issue on Real-Rime Systems", *IEEE Transactions on Computers*, 36(8), 1987.

[34] K. G. Shin and C. J. Hou, "Analysis of Three Contention Protocols in Distributed Real-Time Systems", *Proc. The 11th IEEE Real-Time Systems Symposium*, Florida, December 1990.

[35] J. A. Stankovic, "Misconceptions about Real-Time Computing: A Serious Problem for Next Generation Systems", *IEEE Computer*, 21(10), October 1988.

[36] J. A. Stankovic and K. Ramamritham, Editors, *Tutorial, Hard Real-Time Systems*, IEEE Press, 1988.

[37] J. K. Strosnider, T. Marchok, and J. Lehoczky, "Advanced Real-Time Scheduling Using the IEEE 802.5 Token Ring", *Proc. IEEE Real-Time Systems Symposium*, December 1988.

[38] J. K. Stronsnider, T. Marchok, "Deterministic IEEE 802.5 Token Ring Scheduling", *The Journal of Real-Time Systems*, Vol 1, Nov 2, September 1989.

[39] T. Suda and T. T.Bradley, "Packetized Voice/Data Intergrated Transmission on a Token Ring Local Area Network", *IEEE Transaction on Communications*, 37(3), March 1989.

[40] B. Tangney and D. O'Mahony, *Local Area Networks and Their Applications*, Prentice-Hall International, 1st edition, 1988.

[41] A. S. Tanenbaum, *Computer Networks*, Prentice-Hall International, 2nd edition, 1988.

[42] H. Tokuda, C. W. Mercer, Y. Ishikawa, and T. E. Marchok, "Priority Inversions in Real-time Communication", *Proc. The 10th IEEE Real-Time Systems Symposium*, Santa Monica, California, December 1989.

[43] D. Towsley and G. Venkatesh, "Window Random Access Protocols for Local Computer Networks", *IEEE Transactions on Computers*, Vol C-31, No 8, 1982.

[44] Z. Tsai and I. Rubin, "Performance of Token Schemes Supporting Delay-Constrained Priority Traffic Streams", *IEEE Transactions on Communications*, 38(11): 1194–2003, November 1990.

[45] A. Valenzano, C. Demartini, L. Ciminiera, "MAP and TOP communications: Standards and Applications", Addison-Wesley Publication, 1992.

[46] J. W. Wong and P. M. Gopal, "Analysis of a Token Ring Protocol for Voice Transmission", *Computer Networks and ISDN System*, 8(4), August 1984.

[47] L. Yao, W. Zhao and C. C. Lim, "Perforamnce of Three Token Ring Protocols for Real-Time Communications", *To appear in Proc. 1994 International Conference on Communication Technology*, Shanghai, China, June 1994.

[48] L. Yao, W. Zhao and C. C. Lim, "An Efficient Window Protocol for Real-time Communications in Token Ring Networks", *Proc. The 2nd International Computer Conference, Data and Knowledge Engineering: Theory and Applications* , Hong Kong, December 1992.

[49] L. Yao and W. Zhao, "Performance of an Extended IEEE 802.5 Protocol," Proceedings, *The IEEE 10th International Conference on Computer Communications (INFOCOM'91)*, Miami, April 1991.

[50] L. Yao and W. Zhao, "Token Ring Protocols for Transmission of Time Constrained Messages Proceedings, *The 5th Australian Fast Packet Switching Workshop*, Melbourne, July 1990.

[51] L. Yao and W. Zhao, "Implementing the Minimum-Laxity-First Transmission Policy in a Real-Time Token Ring Network," Proceedings, "The 4th Australian Teletraffic Research Seminar", Bond University, December 1989.

[52] J. Zhang and E. Coyle, "The Transient Performance Analysis of Voice/Data Integrated Networks", *Proc. IEEE International Conference on Computer Communications*, June 1990.

[53] W. Zhao and J. A. Stankovic, "Performance Analysis of FCFS and Improved FCFS Scheduling Algorithms for Dynamic Real-Time Computer Systems", *Proc. the 10th Real-Time Systems Symposium*, Santa Monica, California, December 1989.

[54] W. Zhao, K. Ramamritham and J. A. Stankovic, "Scheduling Tasks with Resource Requirements in Hard Real-Time Systems", *IEEE Transactions on Software Engineering*, Vol SE-13, No 5, May 1987.

[55] W. Zhao and K. Ramamritham, "Virtual Time CSMA Protocols for Hard Real-Time Communications", *IEEE Transactions on Software Engineering*, Vol SE-13, No 8, August 1987.

[56] W. Zhao, K. Ramamritham and J. A. Stankovic, "Preemptive Scheduling under Time and Resource Constraints", *IEEE Transactions on Computers*, Vol C-36, No. 8, 1987.

[57] W. Zhao, C. Barter and N. Malcolm, "Virtual Time CSMA Protocols with Two Version Message Model for Real-Time Communications", *Proc. 1989 IEEE International Conference on Networks*, Singapore, July 1989.

[58] W. Zhao, J. A. Stankovic and K. Ramamritham, "A Window Protocol for Transmission of Time Constrained Messages", *IEEE Transactions on Computers*, Vol 39, No 9, September 1990.

[59] W. Zhao, "Special Issue on Real-Time Operating Systems", *ACM Operating Systems Review*, 23(3), 1989.

[60] T. Znati, "A Minimum-Laxity-First Window Protocol for Transmission of Real-Time Traffic", *Proc. 10th Annual IEEE International Pheonix Conference on Computers and Communications*, Pheonix, March 1991.