Copyright Warning & Restrictions

The copyright law of the United States (Title 17, United States Code) governs the making of photocopies or other reproductions of copyrighted material.

Under certain conditions specified in the law, libraries and archives are authorized to furnish a photocopy or other reproduction. One of these specified conditions is that the photocopy or reproduction is not to be "used for any purpose other than private study, scholarship, or research." If a, user makes a request for, or later uses, a photocopy or reproduction for purposes in excess of "fair use" that user may be liable for copyright infringement,

This institution reserves the right to refuse to accept a copying order if, in its judgment, fulfillment of the order would involve violation of copyright law.

Please Note: The author retains the copyright while the New Jersey Institute of Technology reserves the right to distribute this thesis or dissertation

Printing note: If you do not wish to print this page, then select "Pages from: first page # to: last page #" on the print dialog screen

Order Number 9409118

Medium access control mechanisms for high-speed metropolitan area networks

Papamichail, Michail, Ph.D.

New Jersey Institute of Technology, 1993

U·M·I
300 N. Zeeb Rd.
Ann Arbor, MI 48106

MEDIUM ACCESS CONTROL MECHANISMS FOR HIGH SPEED METROPOLITAN AREA NETWORKS

by
Michail Papamichail

A Dissertation
Submitted to the Faculty of
New Jersey Institute of Technology
in Partial Fulfillment of the Requirements for the Degree of
Doctor of Philosophy

Department of Computer and Information Science

October 1993

APPROVAL PAGE

Medium Access Control Mechanisms for
High Speed Metropolitan Area Networks

Michail Papamichail

Dr. Dionysios Kargiotis, Dissertation Advisor (date)
Assistant Professor of Computer and Information Science, NJIT

Dr. Peter Ng, Committee Member (date)
Chairperson and Professor of Computer and Information Science, NJIT

Dr. James McHugh, Committee Member (date)
Associate Chairperson and Professor of Computer and
Information Science, NJIT

Dr. Daniel Chao, Committee Member (date)
Assistant Professor of Computer and Information Science, NJIT

Dr. Sotirios Ziavras, Committee Member (date)
Assistant Professor of Electrical and Computer Engineering, NJIT

Dr. Xiuli Chao, Committee Member (date)
Assistant Professor of Industrial Engineering, NJIT

ABSTRACT

Medium Access Control Mechanisms for High Speed Metropolitan Area Networks

by
Michail Papamichail

In this dissertation novel Medium Access Control mechanisms for High Speed Metropolitan Area networks are proposed and their performance is investigated under the presence of single and multiple priority classes of traffic. The proposed mechanisms are based on the Distributed Queue Dual Bus network, which has been adopted by the IEEE standardization committee as the 802.6 standard for Metropolitan Area Networks, and address most of its performance limitations. First, the Rotating Slot Generator scheme is introduced which uses the looped bus architecture that has been proposed for the 802.6 network. According to this scheme the responsibility for generating slots moves periodically from station to station around the loop. In this way, the positions of the stations relative to the slot generator change continuously, and therefore, there are no favorable locations on the busses. Then, two variations of a new bandwidth balancing mechanism, the NSW_BWB and ITU_NSW are introduced. Their main advantage is that their operation does not require the wastage of channel slots and for this reason they can converge very fast to the steady state, where the fair bandwidth allocation is achieved. Their performance and their ability to support multiple priority classes of traffic are thoroughly investigated. Analytic estimates for the stations' throughputs and average segment delays are provided. Moreover, a novel, very effective priority mechanism is introduced which can guarantee almost immediate access for high priority traffic, regardless of the presence of lower priority traffic. Its performance is thoroughly investigated and its ability to support real time traffic, such as voice and video, is demonstrated. Finally, the performance under the presence of erasure nodes of the various mechanisms that have been proposed in this dissertation is examined and compared to the corresponding performance of the most prominent existing mechanisms.

BIOGRAPHICAL SKETCH

Author: Michail Papamichail

Degree: Doctor of Philosophy in Computer Science

Date: October 1993

Undergraduate and Graduate Education:

- Doctor of Philosophy in Computer Science,
  New Jersey Institute of Technology, Newark, NJ, 1993

- Diploma of Engineering in Electrical Engineering,
  National Technical University of Athens (NTUA), Athens, Hellas, 1989

Major: Computer Science

Presentations and Publications:

- D.E. Karvelas and M. Papamichail, "The No Slot Wasting Bandwidth Balancing Mechanism for Dual Bus Architectures", accepted and will appear in *IEEE Journal Selected Areas on Communications;* also CIS-92-15 Technical Report, New Jersey Institute of Technology.

- D. Karvelas, M. Papamichail, "Performance Analysis of a New Bandwidth Balancing Mechanism under the Presence of Erasure Nodes", *Proceed. INFOCOM '93, San Francisco, pp. 1091-1098, March 30 - April 1, 1993.*

- D. Karvelas, M. Papamichail, "DQDB: A Fast Converging Bandwidth Balancing Mechanism that Requires No Bandwidth Loss", in *Proceed. ICC '92,* Chicago, pp. 142-146, June 14-18, 1992.

- D. Karvelas, M. Papamichail, "Performance Study of a New Bandwidth Balancing Mechanism under a Single and Multiple Priority Classes of Traffic", in *Proceed. of First International Conference on Computer Communications and Networks,* San Diego, California, pp. 102-107, June 8-10, 1992.

- D. Karvelas, M. Papamichail, and G. Polyzos, "Performance Analysis of the Rotating Slot Generator Scheme" in *Proceed. INFOCOM '92,* Florence, Italy, pp. 794-803, May 6-8 1992.

- D. Karvelas, M. Papamichail, and G. Polyzos, "The Rotating Slot Generator Dual Bus", in *Proceed. of Twenty-third Pittsburgh Conference on Modeling and Simulation,* Pittsburgh, pp. 2337-2344, April 30-May 1, 1992.

- M. Papamichail and D. Karvelas, "A Simulation Study of DCP", in *Proceed. of Twenty-third Pittsburgh Conference on Modeling and Simulation,* Pittsburgh, pp. 2495-2502, April 30-May 1, 1992.

- D. Karvelas and M. Papamichail, "A Simulation Model for DQDB", *Proceed. of Twenty-Second Annual Pittsburgh Conference on Modeling and Simulation,* Pittsburgh, pp. 2265-2272, May 1991.

This dissertation is dedicated to
the memory of my father
Spyros Papamichail

ACKNOWLEDGMENT

The author wishes to extend his sincere gratitude to his supervisor, Dr. Dionysios Karvelas, for his encouragement, guidance and help throughout the course of this work.

Special thanks go to Professors Peter Ng, James McHugh, Sotirios Ziavras, Xiuli Chao and Daniel Chao for serving as members of the committee.

The financial support of the CIS Department is highly appreciated.

Furthermore, the author would like to thank his parents, Spyros and Maria Papamichail, and his sister, Leda Papamichail, for their love and encouragement over the years.

Finally, he would like to thank Anthoula Trombouki for her continuous encouragement and moral support.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Figure **Page**

Figure **Page**

CHAPTER 1

INTRODUCTION

The proliferation of Local Area Networks (LANs) has established high speed data networking in the local area environment. The success of LANs, combined with the advances in fiber optic technology, which can provide huge bandwidths, and computing technology, which have prompted a rapidly growing use of even more powerful PCs and Workstations, portend an emerging market for multi-megabit communication services in both the local and metropolitan area environments.

A new generation of High Capacity Local Area Networks (HCLANs) has already been on the way with aggregate bit rates ranging from 100 Mbps to 1 Gbps. The availability of high bandwidths enables integration of services and has changed drastically our view of computer networks. It is expected that future networks, in addition to transferring massive amount of data between supercomputers, will also be capable of supporting voice, video and other types of real, or non real, time services. In fact, although we can easily envision a large number of applications that can be supported by the Gbps networks, it is certain that there will be others which we cannot currently foresee. Such diversity of services is going to generate flows of information with very different characteristics and delay requirements. Consequently, one of the major challenges that the designer of this new generation of high capacity networks encounters is to find ways for efficiently sharing the enormous bandwidth among a large number of users, meeting at the same time the delay requirements imposed by the presence of time critical types of traffic.

There have been many interesting proposals on how to efficiently access and share a high capacity channel in the local area environment. Among them EXPRESSNET [1], FASNET [2] and FDDI [3,4,5,6] have received a great amount of attention and have inspired proposals for other high speed networks. However, these networks cannot be directly extended to cover larger distances because their performance will deteriorate.

The main problem is that their operation is based on cycles. These cycles are separated by an intercycle gap which is equal to the round-trip propagation delay and is used to: **a)** establish that all stations had a transmission opportunity during each cycle, **b)** initiate a new cycle. In this way no station can monopolize the channel with its own transmissions, and fairness is introduced by providing each station with the same number of transmission opportunities. Consequently, as the size of the network increases, the round-trip propagation delay, the intercycle gap, the time the channel remains idle and the amount of bandwidth which is wasted, all increase. The higher the channel rate the higher the bandwidth loss. Hence, as the size and the capacity of the network increase, the performance of the above networks degrades.

In order to provide a quantitative feeling on the performance degradation of the cyclic operation, we show in Fig. 1.1 the effect that the various system parameters have on the maximum system utilization ρ_{max}. We define as utilization the percentage of time, during each cycle, that the system performs useful work, i.e. transmits information. We consider a high speed network of channel capacity C_r, connecting N stations. Each station, during each cycle of operation, can transmit up to l_m bits. Let t_{ov} be the overhead introduced by the cyclic operation to establish, each time, the beginning of a new cycle. It is then evident that the maximum utilization will be given by the following equation:

$$\rho_{max} = \frac{N\, l_m / C_r}{N\, l_m / C_r + t_{ov}} = \frac{1}{1 + \frac{t_{ov}\, C_r}{N\, l_m}} \tag{1.1}$$

We can easily see from (1.1) that as the transmission rate C_r and/or the size (switch-over overhead per cycle t_{ov}) of the network increase the denominator in the above expression also increases and the maximum system utilization decreases.

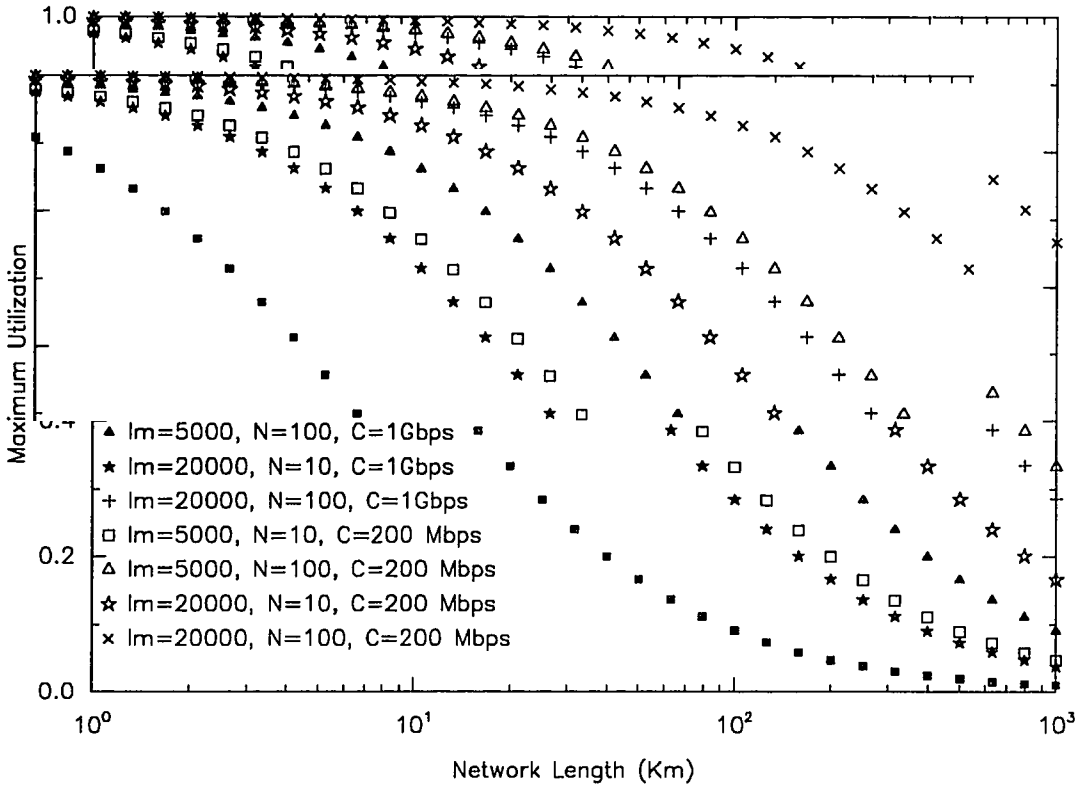

Fig.1.1: Effect of capacity and network length on the maximum throughput. Propagation delay 5 microsec/Km.

In Fig.1.1 we use equation (1.1) to plot the maximum utilization versus the network size characteristics for various values of the system parameters. We mention that in the computation of t_{ov} we have considered only the signal propagation delay, which is the major component of t_{ov} in high capacity long distance networks. We have assumed a value of 5 $\mu sec/Km$ for the signal propagation speed inside the medium. Fig.1.1 shows the strong negative effect that both network size and channel bandwidth have on the maximum utilization. For instance, an 100 Km network running at 1 Gbps and connecting 10 stations (i.e. a backbone network connecting 10 gateways) which can transmit 20,000 bits during each cycle has a maximum utilization of about 0.3. We also see that ρ_{max} increases as the number of connected stations or the maximum number of bits that each station can transmit per cycle increases. However, the higher the value of l_m the

more unfair the system becomes to lightly loaded stations. On the other hand, the larger the number of connected stations the higher the medium access delay.

The previous discussion clearly demonstrates the need of new Medium Access Control (MAC) mechanisms whose performance will not deteriorate as the network size and channel bandwidth increase. The following three criteria seem to be appropriate for characterizing the suitability of a MAC protocol for networks with high latency-bandwidth product.

a) Simplicity of implementation. This is a very important property since MAC protocols will be required to operate at Gbps.

b) Minimum bandwidth loss due to the scheduling mechanism. In addition, the amount of bandwidth loss should not be affected by the system parameters.

c) Fairness, in terms of both bandwidth allocation and medium access delay, among the competing for the medium stations.

A new class of MAC mechanisms has been recently proposed that tries to meet the above objectives. A discussion on these mechanisms follows in chapter 2 of this dissertation. The most prominent among them is the Distributed Queue Dual Bus (DQDB) MAC mechanism [7,8,9]. DQDB has been recently adopted by the IEEE 802.6 standards committee as the IEEE standard for Metropolitan Area Networks (MANs). Since the contributions of this dissertation, as well as the most of the recent research work in the area of Metropolitan Area Networks, has been motivated by DQDB, we provide in the sequel a brief description of its Medium Access Control (MAC) mechanism.

1.1 DQDB MAC Mechanism

DQDB consists of two unidirectional busses on which information travels in opposite directions. The stations are connected to both busses, as shown in Fig.1.2. The first station in each bus, station "0" for bus A and station "N-1" for bus B in Fig.1.2, generates slots that are traveling downstream. A slot is the basic data unit and consists of the

Access Control Field (ACF), which is one byte, and the 52 bytes segment. The segment is further divided into the segment header, which is 4 bytes, and the segment payload, which is 48 bytes. Finally, the segment payload consists of a header (2 bytes), a segmentation unit (44 bytes) and a trailer (2 bytes). We see that the maximum amount of information that can be carried by each segment is 44 bytes. Therefore, if the size of a data packet is greater than 44 bytes, it will be fragmented into blocks (segmentation units) of 44 bytes.

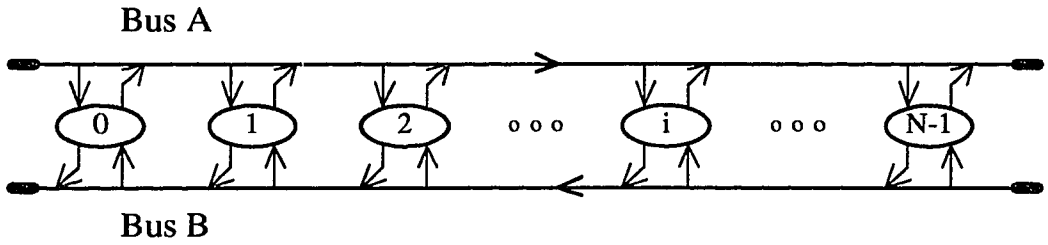

Fig.1.2 DQDB dual bus architecture.

It is evident from Fig.1.2 that if a station wants to send data to another station located to its right, it will transmit onto bus A. Otherwise it will transmit onto bus B. In the sequel we focus our attention to the transmissions on bus A. Similar is the operation on bus B. We mention that in some cases we use the terms forward bus or forward channel for bus A, and reverse bus or reverse channel for bus B. Finally, we say that a station "j" is upstream of a station "i" when a slot written by "j" can be read by "i", that is, the location of station "j" is closer to the slot generator on the forward channel.

The operation of DQDB is based on two bits in the Access Control Field (ACF) of the slot: the *Busy Bit* (BB), indicating whether a slot on the forward channel has already been written by an upstream station, and the *Request Bit* (RB), indicating whether a slot on the reverse channel carries a reservation made by a downstream station; for its queued segment. Furthermore two counters, the Request Counter (RQ_CTR) and the Count Down Counter (CD_CTR), are needed per station. Their operation is as follows. When the station is idle RQ_CTR increases by one for every *request* bit (RB=1) seen on the reverse bus and decreases by one (if RQ_CTR >0) for every idle slot (BB=0) seen on the

forward bus. Therefore, RQ_CTR keeps track of the number of queued segments in the downstream stations. When a segment arrives at a station, the content of RQ_CTR is transferred to CD_CTR and RQ_CTR is reset to 0. At the same time a *request* is sent on the reverse B to notify the upstream stations of the new queued segment. From this instant, the station decrements CD_CTR for every empty slot seen on the forward bus and increments RQ_CTR for every *request* bit seen on the reverse bus. When CD_CTR becomes 0 the station transmits its segment in the first empty slot on bus A. We point out that in DQDB a station is allowed to send a *request* only for the first segment in its queue.

The main advantage of DQDB is that eliminates the t_{ov} overhead required by the cyclic operation and, in this way, can achieve a maximum utilization of 1 regardless of the values of the system parameters. However, DQDB has serious fairness problems [10,11,12,13,14]. That is, the locations of the stations on the bus drastically affect their throughput as well as the delays their messages will encounter. It is evident that new algorithms are needed that will provide fairness and minimize the scheduling overhead introduced by their operation. Due to the large distances involved, this is a very challenging and difficult problem and has been the subject of most of the recent research work in the area of MANs. In chapter 2 a rather detailed description of the related work in the area is presented. However, all the current research effort has only provided very limited results. The reason is that the proposed algorithms are either not robust, that is, they are fair only under certain types of loading, or too complex to implement in a high speed network.

A very refreshing approach to the issue of fairness, which does not suffer from the problems of other proposed mechanisms, has been recently introduced and investigated in [15]. This mechanism, called Bandwidth Balancing (BWB) mechanism, achieves fairness by allowing each station to receive a multiple M of the unused channel bandwidth. The stations create this idle bandwidth in the following way. Every time a station

transmits its M_{th} segment on the forward channel, it increases its RQ_CTR by one and allows a free slot to pass by. This slot can then be written by the first active downstream station with CD_CTR=0. Then, the transmitting station has the opportunity of sending an additional *request* upstream, if it has another segment waiting. In this way, it decreases even more the transmission opportunities of the upstream stations.

It has been shown in [15] that the BWB mechanism (for which from now on we will use the term BWB_DQDB) can provide the requested throughput to the lightly loaded stations and evenly distribute the remaining bandwidth among the overloaded stations. For his reason it has been recently included in the IEEE 802.6 standard. Yet, it presents three problems: **a)** it slowly converges to steady state where fair bandwidth allocation is achieved, **b)** its operation requires to waste some channel bandwidth, the greater the bandwidth wastage the higher the convergence speed, **c)** it may become ineffective when more than one priority traffic classes are present in the system, i.e. the performance characteristics of the low priority traffic may become, in some cases, better than the ones of the high priority traffic.

The limitations of DQDB and BWB_DQDB, along with the importance of supporting a large variety of services over high speed MANs, have motivated our research interest in this area. The first step in our research effort has been to build a simulator of DQDB and use it to carry out an extensive analysis of its performance that would enhance our understanding of its behavior. We have looked at different variations of the DQDB medium access mechanism and examined their effect on the stations' throughput and delay. The results of our analysis, some of which have been reported in [10], clearly demonstrate the fairness problem of DQDB. They show that regardless of the MAC mechanism variation, the station location on the bus drastically affects its throughput as well as the delays that its messages will encounter. They also indicate the great sensitivity of the stations delay even on slight modifications of the MAC mechanism. This result points out that in order for analytic estimates of throughput or delay to be accurate,

minor variations (such as independence or correlation of the operation of the transmission queue on one bus with the corresponding queue of requests for the other bus) should not be neglected, but taken seriously into consideration. We finally mention that our simulator of DQDB has provided the basis from which various novel medium access control mechanisms, presented in this dissertation, have been developed. In the next section we provide a brief description of the main contributions of the dissertation.

1.2 Contributions of the Dissertation

There are four main contributions in this dissertation. The first is a geommetric solution to the fairness problem of DQDB. In DQDB, the stations which are closer to the slot generator see the idle slots first and in this way are favored. For this reason we have introduced the **Rotating Slot Generator (RSG)** scheme for dual bus architectures. This scheme uses the looped dual bus architecture of DQDB in which slot generator hardware has been incorporated in all stations. Its difference from DQDB is that during its operation the responsibility for generating slots moves periodically from station to station around the loop. Thus, the station positions relative to the slot generator change continuously, and therefore, there are no favorable locations on the busses. We provide a thorough investigation of the performance of different variations of RSG under single and multiple priority classes of traffic. We also compare RSG with DQDB and BWB_DQDB.

The second contribution of this dissertation is the introduction of a new bandwidth balancing mechanism for DQDB. The new mechanism, called **No Slot Wasting BWB (NSW_BWB)** mechanism, has similar complexity to that of BWB_DQDB, but much better performance. Its main advantage is that its operation does not require the wastage of channel slots and for this reason it can converge very fast to the steady state, where the fair bandwidth allocation is achieved. In this dissertation, we look at two variations of NSW_BWB. We thoroughly examine their performance and investigate their ability to

support multiple priority classes of traffic. Furthermore, we provide analytic estimates of their throughput performance under overload traffic conditions and develop a queuing analytic method that provides very good estimates for the delay in the case of underload conditions. The advantage of the proposed queueing model is that it tries to take into account the correlation between busy and request bits, as well as the distance between stations. Such kind of interdependencies between the various DQDB components have not been considered by previous analytic methods. Thus, the proposed queueing approach for NSW_BWB has also the potential of serving as a basis for developing a more accurate model for the DQDB.

The third contribution of the dissertation is the introduction of a novel, very effective priority mechanism which can support real time applications on high speed MANs. We mention here that all the current priority mechanisms that have been proposed for DQDB, including the ones we have proposed for NSW_BWB, are capable of favoring certain classes of traffic under overload conditions. Nevertheless, they cannot react fast enough to changes of the traffic load. As a result, temporary overloads of lower priority traffic can drastically affect the performance of higher priority traffic. The proposed new priority mechanism can guarantee almost immediate access for the high priority traffic, regardless of the presence of lower priority. Consequently, it can satisfy not only the throughput but also the delay constraints of real time traffic. We investigate the effectiveness of the new priority mechanism under various load configurations, as well as under the transmission of data, voice and video. Furthermore, we compare its performance with the other priority mechanisms that have been proposed for DQDB.

Finally, the last contribution of this dissertation is a performance investigation of the previous mechanisms under the presence of erasure nodes. Erasure nodes are special nodes that can reset slots which have already been read by their destinations. In this way, the same slot travelling on the bus can be used to transfer information from more than one station, significantly increasing the aggregate throughput of the system.

1.3 Dissertation Outline

This dissertation is organized as follows. In chapter 2 we provide a rather detailed literature review of the recent research work in the area of high speed MANs. In chapter 3 we introduce the RSG scheme. We present the system architecture and different variations of the RSG protocol. We demonstrate its fairness, with respect to throughput and delay, and investigate its performance under various load configurations, as well as single and multiple priority classes of traffic. In chapter 4 we propose the NSW_BWB scheme. We investigate its throughput and delay performance under one traffic class and examine its capacity to support multiple priority classes of traffic. We also discuss a queueing analytic model which provides good estimates for the average delay of NSW_BWB. the station behavior of NSW_BWB. Finally, we introduce a variation of NSW_BWB, called the Immediate Transmission NSW (ITU_NSW) mechanism, which in many cases can significantly improve the performance of the various stations on the bus. In chapter 5 we introduce a novel, very effective priority mechanism that can meet the strigent delay requirements of real time traffic. We also apply the principles of the new mechanism on BWB_DQDB and demonstrate that it can significantly improve its performance. Furthermore, we investigate the effectiveness of the proposed priority mechanism in a "real world" environment, that is, under the presence of data, voice and video. In chapter 6 we discuss the effect of the presence of erasure nodes on throughput and the average delay of the previous mechanisms under single and multiple priority classes of traffic. Finally, in chapter 7, we provide our conclusions along with some interesting open issues for further research.

CHAPTER 2

RELATED RESEARCH WORK IN THE FIELD

2.1 Introduction

The investigation of high capacity MANs has become a very active research area. Most of the research work is related to the DQDB network. However, there have been a few other interesting approaches. In this chapter we provide a literature review of the recent research work in the area.

DQDB was initially introduced as Queued Packet and Synchronous Exchange (QPSX) [8] but now is referred to as DQDB in order to distinguish it from the development company [16]; QPSX Communications, a subsidiary of Telecom Australia. In [8] the QPSX network architecture is presented. In [17] and [18] the QPSX MAC mechanism is described. Furthermore, QPSX is compared with CSMA/CD and the Token Passing networks. These results demonstrate the superior performance of QPSX when the transmission of independent segments by the stations is considered.

One of the objectives of QPSX is the support of isochronous traffic such as digital voice of 64 Kbps. For this reason, a frame structure has been introduced which is similar to the one of FDDI-II [6]. The frame structure is divided into slots with the size of each slot equal to one segment. These slots are divided into pre-arbitrated (PA) slots, which are allocated to isochronous traffic, and queued-arbitrated (QA) slots which are allocated to asynchronous traffic. In [17,18,19,20,21] mechanisms for allocation and de-allocation of the PA slots to circuit switching traffic have been proposed and investigated. The same frame structure of QPSX has been passed over to DQDB that can also support asynchronous and isochronous traffic. In both cases the MAC mechanism runs on the QA slots of the frame; in the same way the Timed Token Protocol (TTP) of FDDI-II runs on the asynchronous bytes of the frame.

In chapter 1 we described the DQDB and the BWB_DQDB mechanisms. We have also mentioned their major limitations. Extensive simulation studies of DQDB and BWB_DQDB have been presented in [9,10,22]. These studies have shown that BWB_DQDB, contrary to DQDB, can provide all the requested bandwidth to the lightly loaded stations and evenly distribute the remaining bandwidth among the overloaded stations. However, it wastes $1/(1+N\,M)$ amount of bandwidth, where N is the number of active stations. By increasing the value of M the amount of bandwidth that is wasted decreases, however, the mechanism now converges slower to the steady state where the fair bandwidth allocation is achieved. It has been shown in [14] that the length of the transient period depends on the length of the cable, the number of active stations, the traffic statistics, and the value of M. Furthermore, due to the bandwidth loss required for its operation, the average delay experienced by the stations can become considerably higher that the average delay in the case of the basic DQDB. It has been shown in [23] that by providing the various stations with different values of M (M_i for station "i") the channel bandwidth can be distributed at will among the overloaded stations. The work in [23] has been extended in [24] to show how the channel bandwidth can be allocated on a per user basis, rather on a per station basis. In this way, if more than one users access the DQDB network through the same station, they can still receive the desired bandwidth.

In the sequel, in section 2.2, we provide a sample of simulation results which are very indicative of the performance of DQDB and BWB_DQDB and can facilitate the discussion on the related research work presented in section 2.3. Section 2.4 refers to other approaches which have been proposed for medium sharing in high capacity MANs.

2.2 Performance of DQDB and BWB_DQDB Schemes

In this section some indicative simulation results on the performance of DQDB and its BWB mechanism (BWB_DQDB) are presented. The objective is to provide the reader with an early flavor of the positive and negative characteristics of the behavior of both

DQDB and BWB_DQDB schemes. A more detailed performance investigation of the two mechanisms is given in the next chapter, where we introduce the RSG scheme and compare its performance against DQDB and BWB_DQDB. A high capacity network of 155.520 Mbps is considered, which connects N=40 stations, uniformly distributed on the busses. The slot size is 53 bytes and the signal propagation delay 5 μsec/Km. Both a short network with $t_p=.5t_{sl}$, and a long network with $t_p=4t_{sl}$ are considered, where t_p is the propagation delay between two adjacent stations and t_{sl} is the slot time. With the above values of slot size and channel capacity the physical distance between the first and the last station is 10.63 Km and 42.53 Km respectively. Fig.2.1 and Fig.2.2 show the effect of the station location on its throughput and delay. It is assumed that segments arrive at the different stations according to a Poisson distribution. Furthermore, we have considered that the rate at which station "i" transmits segments to station "j" is constant and independent of "i" and "j".

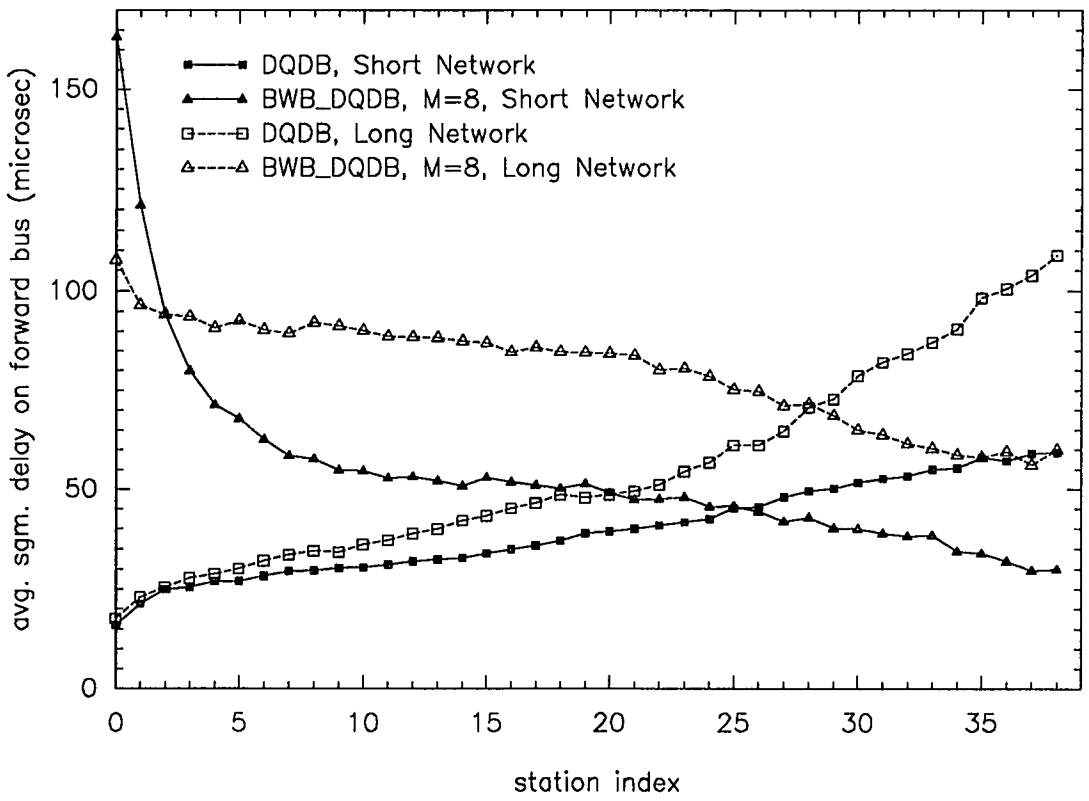

Fig 2.1: Effect of station location and network length on average segment delay. Bus utilization 0.95.

In Fig.2.1 both the short and long network configuration are considered and the effect of the station location on the average segment delay on bus A is shown. The average segment delay is defined as the average delay from the instant a segment arrives at a station, until the instant it starts its transmission on bus A. We see that the stations which are close to the slot generator are penalized in the case of BWB_DQDB, and favored in the case of DQDB. It must be noted here that for BWB_DQDB we have considered the version that uses only the RQ_CTR; and not the CD_CTR as well †. In the case where both the CD_CTR and the RQ_CTR are used, the delay unfairness among the stations becomes similar to the DQDB case. Nevertheless, in both variations of BWB_DQDB the average delay is considerably higher than in the case of DQDB. The reason this version of BWB_DQDB has been selected for this set of figures, is to show the effect that the presence of the CD_CTR has on the delay performance.

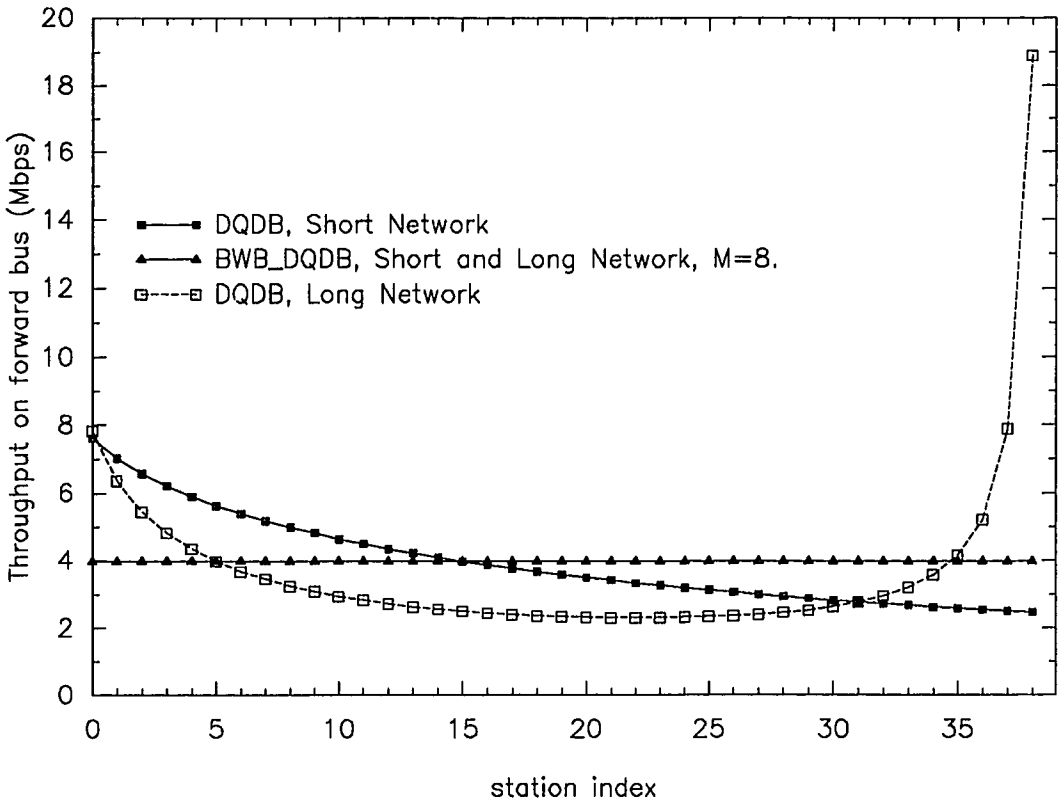

Fig 2.2: Effect of station location and network length on stations' throughput. All stations are overloaded.

† In this case the station can transmit in an empty slot if RQ_CTR is equal to 0.

In Fig.2.2 the bandwidth distribution among the various stations under overload conditions is shown . We see that for both networks BWB_DQDB provides the same bandwidth to all stations; only one curve is plotted. In the case of DQDB, the acquired bandwidth by the stations decreases as we move away from the slot generator in the case of the short network, and as we move towards the center of the bus in the case of the long network. We point out, however, that the distribution of the bandwidth among the stations, in the case of DQDB, depends on the initial loading configuration. In the case of Fig 2.2. all stations become overloaded at the same time. Nevertheless, regardless of the initial loading, DQDB remains unfair in terms of throughput distribution.

The above figures clearly demonstrate the unfairness of DQDB in terms of both access delay and bandwidth distribution. The intention of DQDB is to behave as a global distributed FIFO queue. Unfortunately, due to the significant propagation delay from station to station and the limited capacity of the reverse bus, the actual performance of DQDB is far from the intended one. The next section, briefly describes some of the numerous remedies that have been proposed for the DQDB scheme. Common characteristic of all these proposals is that, under certain load and network configurations, they behave better than DQDB or BWB_DQDB. However, when their overall performance is considered, their attractiveness fades significantly.

2.3 Research Work Related to the DQDB Scheme

We have classified the research work on DQDB in four categories. The first one includes the papers that consider one type of segments or messages present into the system. The second one includes the papers that address the issue of supporting multiple priority classes of traffic. The third consists of the papers that investigate the performance of DQDB under the presence of erasure nodes. Finally, the last category includes the various queueing models that have been proposed for the analysis of DQDB.

2.3.1 One Priority Class of Traffic

In [25] three MAC mechanisms have been proposed. The first is the Proportional Bandwidth Allocation (PBA) scheme which, similarly to [23] and [24], suggests the use of different values of M by the stations. The objective here is to distribute the available bandwidth among the stations in a way which is proportionate to their offered load. Therefore, unlike BWB_DQDB, the lightly loaded stations are penalized too. The second scheme, named Multiple Requests (MPR) scheme, allows a station to transmit consecutive requests on the request channel, and consecutive segments on the transmission channel. In order to achieve that, the operations of RQ_CTR and CD_CTR are slightly modified and an additional counter is used to keep track of the number of segments for which requests have not been sent. The third scheme, called the FCFS-message-queue strategy, tries to combine the advantages of the previous PBA and MPR schemes. Finally, an analytic model for DQDB is presented.

In [26] an application of BWB_DQDB has been considered. The performance of a system is investigated, in which the stations dynamically vary the value of M, in order to support variable bit rate video services.

In [27] a modification of the basic DQDB MAC mechanism has been presented in which each station has one RQ_CTR, more than one CD_CTRs, and the Access Control Filed (ACF) of the slot has one busy bit (BB) and more than one request (RQ) bits; all for the same priority level. Each station tries to transmit, on a passing slot, as many requests as possible for its outstanding segments for which requests have not been sent. Moreover, a station increases the value of its RQ_CTR by the number of RQ bits that have been set in the ACF of this slot. In the case where only some of the RQ bits in the ACF are set, it is possible for the station to increase the value of its RQ_CTR and at the same time set the remaining unset RQ bits. The objective of this mechanism is to increase the request channel bandwidth so that the stations have a more up-to-date knowledge of the number of segments in the system. At the same time by allowing each station to have more than

one CD_CTR, which means that a station can put simultaneously more than one segments into the transmission queue, the operation of the system approaches more closely that of a FCFS system. It is shown in [27] that by using two CD_CTRs in each station and two RQ bits in the ACF of the slot the fairness of the system significantly improves. However, additional increase in the numbers of CD_CTRs or RQ bits has only a minor effect on the performance.

In [28] the Reservation Request Control (RRC) mechanism has been proposed. According to this mechanism each station, in addition to the number of reservations made by the downstream stations, also knows the number of upstream stations that are active. This is achieved by introducing the Start Bit (SB) and the End Bit (EB) in the ACF of the slot. SB is set by a station that becomes active. EB is set by a station that transmits its last segment and becomes idle; SB and EB are set on the slots in the forward bus. Each station, by observing the SB=1 and EB=1 on the forward bus, knows the number of upstream active stations. If the number of upstream stations is "K", then a station cannot send requests at a rate which is greater than one every K slots. It is evident that the objective of this system is to introduce a round robbin service on a dual bus network. This in fact can be achieved in the case of saturated stations. However, the system becomes inefficient in the case of underload conditions, especially, when the transmitted messages consist of a small number of segments.

In [29] the Access Protection and Priority Control (APPC) scheme has been introduced. Its objective is to maintain the bandwidth which is allocated to a station within known limits. The upper and lower protecting limits are indicated by U and L. In the case of the upper protection scheme, when a segment arrives at a station, the station does not transfer the value of RQ_CTR to CD_CTR but rather the value of $\min(RQ\_CTR, U)$. In the case of the lower protection scheme the station transfers the value of $\max(RCQ\_CTR, L)$. Furthermore, after the transmission of a segment the CD_CTR is immediately initialized with the value of L. Additional mechanisms, using some thres-

holds, have also been proposed in [29] but the details on how should be controlled, to achieve fairness, have not been discussed.

In [30] a MAC mechanism has been proposed which aims at reducing the variation in the delay encountered by the different stations. According to this mechanism an additional bit, the pre-request (PRQ) bit, is used in each slot. Besides, each station has an additional counter, the pre-request counter (PRQC). When a station generates a segment for transmission on bus A, it sets PRQ=1 in the next slot on bus A. Stations that see PRQ=1 increase the value of PRQC by one. When the PRQ bit arrives at the end of the forward bus the slot generator will send a request on the reverse bus. The stations with PRQC>0 will decrease PRQC by one, while the stations with PRQC=0 will increase the value of RQ_CTR by one. Consequently, only the stations which were upstream of the station that generated the segment and sent the PRQ bit will increase their request counters and allow a free slot to go by. In order to prohibit the stations for using a free slot that has been reserved by another station, a segment is not allowed to be transmitted for a t_{e_e} interval after its generation, where t_{e_e} is the end-to-end propagation delay. The objective of this mechanism is to artificially delay, by a different amount, the instants at which the stations can insert their requests so that the stations which are closer to slot generators will not be favored. It is evident that this mechanism can significantly increase the delay encountered by all the stations even though the authors claim that will reduce the variability of their delays. Nevertheless, the authors do not provide any performance results to support their claim.

In [31] another access mechanism has been proposed according to which the stations do not transmit a separate request bit for each segment. Instead, a request bit is sent upstream by a station when this station becomes active. Another control bit, the Empty Bit (EB), is used by a downstream station to notify the upstream stations that has become idle. In this way each station keeps track of the number of downstream stations that are active. If a station has segments waiting for transmission and its estimate for the number

of the active downstream stations is K, it will transmit one segment every K+1 slots. It is evident that this mechanism will work very well under overload conditions or when the stations transmit very long messages consisted of a large number of segments. However, as the size of the message decreases or the size of the network increases, its performance will approach that of DQDB.

2.3.2 Multiple Priority Classes of Traffic

The importance of supporting priority services has been recognized from the early stages of the development of DQDB. A priority mechanism has been proposed that uses a separate pair of RQ_CTR and CD_CTR, one for each class, inside the station. Furthermore, the ACF of the slot has a separate request bit for each priority. The counters operate as before with the exception that an RQ_CTR at a particular level counts request bits at the same and higher priority levels. In this way, an RQ_CTR records all the queued segments at equal and higher priorities. Furthermore, a CD_CTR operating at a particular level is not only decremented for any empty slot that passes by on the forward channel. It is also incremented for any request bit of higher priority seen on the reverse channel. In this way higher priority segments can gain access ahead of already queued lower priority segments.

The objective of DQDB priority mechanism is to provide absolute priority to high priority segments. That is, the performance characteristics of the higher priority users must not be affected by the presence of the lower priority users. Although this is the case on a per station basis, performance investigations of the DQDB priority mechanism in [32,33,34] have shown that the station location has a very strong effect on its performance. Low priority users at stations closer to the slot generator can get more bandwidth and encounter significant lower delays than higher priority users located far from the slot generators. For this reason, although four levels of priority were initially supported, the number of priority levels has now been reduced to three.

A direct extension of the BWB mechanism has been proposed, according to which, every time M segments are transmitted by a station the request counters of all three priorities are increased by one. However, this extension of BWB can only guarantee that high priority users can receive at least as much bandwidth as the lowest priority users. The throughput that the highest priority users receive still depend on their position on the bus. Furthermore, it is possible for high priority users to receive less bandwidth than medium priority users. In order to improve the performance, three extensions of BWB have been presented in [35] that can allocate the channel bandwidth among the different classes at a predetermined ratio. Extensions of the ideas in [35] have been presented and investigated in [36,37,38]. We point out however, that all the proposed schemes, despite their ability to provide each priority level with the desired amount of bandwidth, cannot guarantee lower delay to the high priority messages. That is, the position of the station still has a strong effect on its delays.

In [39] a simple mechanism has been proposed that can achieve preemptive priorities in DQDB within a maximum round trip delay. The two bits of the access control field of the slot, currently reserved for future use, are used by the downstream stations to inform the upstream stations about the existence of additional high priority segments waiting for transmission. In this way upstream stations, having segments of lower priority, defer from transmitting. Although this priority mechanism can provide preemptive priority, the station position drastically affects the bandwidth that users of the same priority can receive. As a solution to this problem the authors suggest the use of BWB in combination with their mechanism. Another problem of this priority mechanism is that under certain loading conditions it may waste a significant amount of bandwidth.

2.3.3 Introduction of Erasure Nodes

In all papers we have discussed so far, the two unidirectional busses of DQDB have been considered to be passive. However, passive taps are not appropriate for fiber optic networks due to the limited power budget which is available with current sources and

detectors. As a result, only a few tens of passive taps can be connected to the network. For this reason active tapping, in which the signal is regenerated inside each station, has currently received great attention. With active tapping not only the physical size of the network can increase, almost without limit, but also the system performance can improve. This can be achieved by using the slots from source to destination and then releasing them from subsequent use by other downstream stations. The price that someone has to pay is: a) the increased system latency, since each station must delay each slot in order to read its destination field and decide whether to release it, and b) the added processing complexity at the station for deciding whether the slot should be released. The approach that has been proposed for DQDB is to have a few stations only, called erasure nodes, to be able to release written slots. A Previous Slot Read (PSR) bit is introduced to the ACF of the slot. The station which is the destination of a slot will set the PSR bit in the next slot. Each erasure node has a buffer, equal to one slot plus the ACF size of a slot, in which it delays the passing slot and decides whether to erase it. The introduction of erasure nodes can significantly increase the maximum network throughput without significantly increasing the network latency. Furthermore, the additional processing complexity is restricted to the erasure nodes only.

There is an ongoing research activity on the performance of DQDB under the presence of erasure nodes. In [40] and [41] the problem of the optimal placement of K erasure nodes on a bus with N active nodes has been considered. Furthermore, in [41] some modifications of the DQDB MAC mechanism have been considered which try to improve even more the system performance. In [42] a mechanism to implement one of the DQDB variations introduced in [42] has been proposed and investigated. In [43] and [44] two new MAC mechanisms, using a combination of the continuation-bit strategy with the erasure nodes approach, have been proposed and investigated. The continuation-bit strategy has been introduced in [45] and aims at reducing the large overhead associated with the transmission of long messages over slotted networks. The key

idea is to provide all the addressing information in the first packet only, and use a continuation-bit to indicate that the subsequent packets are continuation packets. The application of the continuation-bit approach eliminates the need of repeating the Message Identifier (MID) field in each slot which can be used to carry data. This will achieve a 4% increase in message throughput which can result to a significant reduction of the message delays at high loads. Although the erasure nodes can improve significantly the throughput and delay of the system, the unfairness problem may become worse. In [43] and [45] several methods for achieving different fairness criteria have been presented. In [45] a recursive algorithm has been presented that can compute, under overload conditions, the bandwidth allocation at each station.

2.3.4 Analytical Models

Most of the performance studies on DQDB are based on simulation. The main reason is the great degree of complexity that is required to provide an accurate queueing model of the system. For instance, it has been shown in [25] that an exact queueing model of DQDB consisted of two stations with one buffer per station and with inter-station distance of 6 slots will have 880,000 states, out of which 92,000 are valid. It is not therefore strange that approximate models, making simplifying assumptions, have been used to analyze its performance.

In [46] a multi-priority queueing system, consisted of a number of distributed stations and a processor sharing central server, is presented. The main two simplifying assumptions of the model are: a) when a station sends a request, immediately, all the other stations see it, and b) the order in which packets arrive into the system is random. With these two assumptions the effect of the station location on its performance is eliminated and the analysis becomes tractable. Using a variation of the Round Robbin Processor Sharing analysis in [47] the authors derive an (approximate) expression for the average waiting time of the n-segments message at priority "p". A comparison with simulation

results has shown that the model provides accurate delay estimates when the stations of the network are close to each other.

In [48] an approximate method for the distribution of the segment delay at a station is presented. Each station is assumed to have a buffer of size one that can hold at most one segment. After the transmission of this segment another one can be generated. In this method the network is partitioned into three parts. The left network (L_NET), consisted of all the upstream stations, the station whose average segment delay is to be computed (tagged station), and the right network (R_NET) consisted of all the downstream stations. This analysis can capture the effect of the station location on its performance.

In [49] another approximate method for evaluating the DQDB segment delay is presented. The total delay of a segment is decomposed into the following three parts: a) from its arrival instant until it reaches the head of the local queue, at which time the value of RQ_CTR is transferred to CD_CTR and a request is sent upstream, b) from the instant it reaches the head of the local queue until the CD_CTR becomes 0, and c) from the instant the CD_CTR becomes 0 until a free slot is seen and the segment is transmitted onto the channel. A hierarchical model is developed, consisted of M/G/1 systems, in which the estimated waiting time of the lower level is used as service time for the higher level. In [50] and [51] a single server system serving two classes of customers with one class having non-preemptive priority over the other is presented. It is then shown how this analysis can be used to evaluate the DQDB segment delay.

Finally, in [52] a three priority queueing policy is proposed which models exactly the service that is provided to the stations under the DQDB mechanism. Furthermore, approximate delay results are derived. The work in [52] is extended in [53] and an accurate analytic solution for the three priority queuing model is provided. This model, however, does not capture the effect that the network length has on the performance.

2.4 Additional Related Research Work

The afore-mentioned research work is directly related to DQDB. In this section we provide a brief discussion of some other work that has been conducted in the area of high capacity MANs. The main objective of most of the proposed schemes here is to achieve a token-ring like access fairness among the stations, eliminating or reducing significantly at the same time, the intercycle gap.

In [54] a new protocol called Load-Controlled Scheduling of Traffic (LOCOST) has been presented that can distribute the channel bandwidth among the stations in an almost arbitrary way. According to this protocol, the stations measure the traffic load on the medium and determine the rate at which they should transmit. Although a small amount of bandwidth may be wasted the performance of the system becomes independent of the physical size of the network and the channel capacity.

In [55] LIGHTNET, a slotted version of EXPRESSNET [1], has been presented. According to this protocol the reservation status of a slot that passes in front of a station in the outbound channel is determined by the status of the bumper of the slot that the station most recently read in the inbound channel. If the bumper was damaged the slot is reserved and the station cannot write a packet on it. There is no cyclic operation and each station can write on any passing slot that is free or has not been reserved. Although the performance of LIGHTNET is not sensitive to channel capacity or cable length, it may suffer severe bandwidth loss.

In [56] the Cycle Compensation Protocol (CCP) has been proposed which is based on the cyclic protocol of FASNET but tries to decrease the intercycle gap. In [57] the Distributed Control Polling (DCP) network has been proposed which introduces a cyclic operation on dual bus architectures without the need of an intercycle gap. As a result it can distribute the bandwidth among the stations in any desirable way and have a performance which is independent of the cable size or channel rate. The performance of DCP both under single and multiple priority classes of traffic has been investigated in [58].

In [59] the Cyclic-Reservation Multiple-Access scheme has been proposed which can be implemented in both folded and dual bus networks. In this scheme the stations reserve the slots before their transmissions. This significantly increases the message delay but: a) can achieve a token-ring like fairness in accessing the medium, b) does not require the intercycle gap and, c) allows stations to transmit consecutively all the segments of their messages facilitating, in this way, the message reassembly at the destination. Some results on the performance of CRMA have been presented in [60] along with another scheme, the Distributed-Queue Multiple-Access (DQMA).

In [61] the p_i*–persistent* protocol for multiaccess communication over unidirectional busses has been proposed and analyzed. In [62] the voice/data performance of the previous protocol has been investigated. In [63] a distributed control has been presented and investigated in which the stations dynamically adjust the values of p_i at the proper levels governed by the offered load. In [64] the authors have proved that in a system in which the stations have a single packet buffer, appropriate values for the p_i at the different stations can be found that can achieve fairness in terms of average delay, blocking, or throughput.

We conclude our discussion with Metaring [65,66,67], which is currently being implemented as part of the IBM participation in the CNRI/Aurora Gigabit Testbed [68]. Metaring uses the dual (bi-directional) ring topology and has two basic modes of operation: a) buffer insertion mode for long messages, and b) slotted mode for fixed size packets. The combination of destination release and transmission to the destination along the shortest path, enables Metaring to carry four times more throughput than a dual token ring configuration. In [65,66,67] a description of the Metaring architecture has been provided and some performance results demonstrating its superior performance have been presented. In [69] the principles for constructing a single Metaring from multiple rings have been described. We finally mention that Metaring basically uses a cyclic operation in its MAC mechanism. This mechanism is far more efficient than token passing and

may even eliminate completely the intercycle gap under certain types of loading. Under others, bandwidth may be lost. The channel bandwidth which maybe wasted becomes significant as the physical size of the network and the channel capacity increase.

CHAPTER 3

THE ROTATING SLOT GENERATOR SCHEME

3.1 Introduction

In chapter 2 the limitations of DQDB and BWB_DQDB were discussed. It was shown that the performance of a station is drastically affected by its location on the bus. In this chapter the **Rotating Slot Generator (RSG)** scheme is introduced. RSG uses the looped bus architecture of DQDB, in which slot generator hardware is included in every station. The key idea behind RSG is to periodically change the station location on the busses, in order to eliminate the effect of a favorable position. This is achieved by moving the Slot Generator (SG) from station to station, so that every station takes its turn as a slot generator. In this way, the location of each station relative to the SG changes continuously and the correlation between station location and its long term performance is eliminated.

The organization of chapter 3 is as follows. In section 3.2, we describe the RSG scheme. Furthermore, we propose and compare two variations for switching the slot generator. In section 3.3 we investigate the performance of the RSG scheme under underload and overload conditions. In section 3.4 we examine its ability to handle multiple priority classes of traffic. Finally, in section 5, we provide the conclusions.

3.2 The Rotating Slot Generator MAC mechanism

RSG is based on an extension of DQDB's architecture, the DQDB looped bus architecture shown in Fig.3.1. In this configuration, the end points of the busses are co-located and the slot generators for both busses are "inside" station "0". Data, however, is not allowed to flow through this station. This can be achieved by introducing, for instance, an AND gate between the read and write connections of the station's Access Control Unit (ACU), as it is shown in Fig. 3.2. In this figure $R_{A,i}$, $W_{A,i}$, $I_{A,i}$ and $R_{B,i}$, $W_{B,i}$, $I_{B,i}$ indicate the read, write, and input to AND lines of station's "i" ACU on busses A and B respectively.

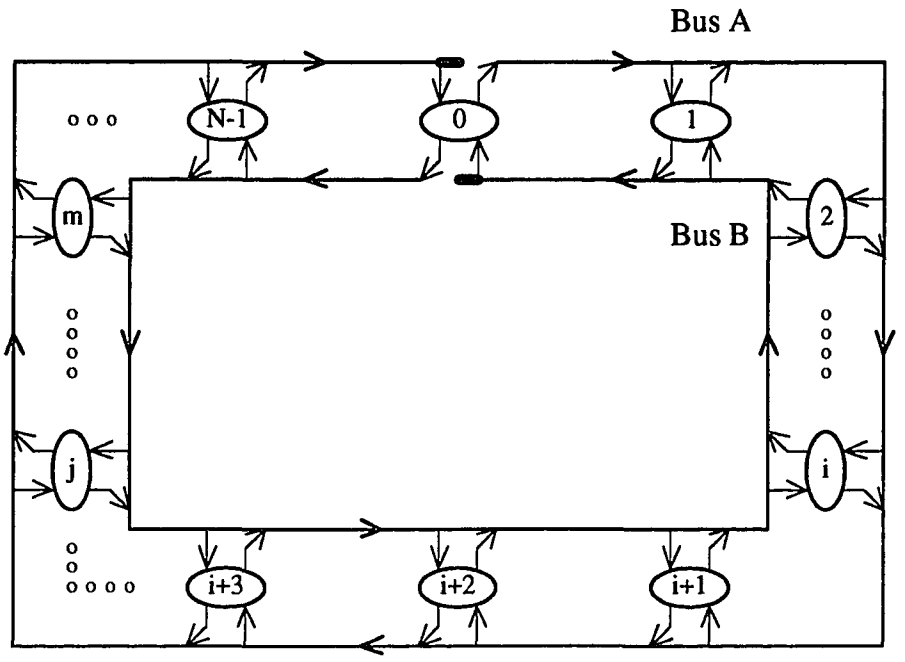

Fig.3.1: DQDB looped bus architecture.

The main advantage of the DQDB looped bus architecture is that it increases significantly the reliability of the system. Consider, for instance, the case of a bus fault between stations "i+1" and "i+2". Station "0" can close both busses by setting $I_{A,0}=I_{A,0}=1$, and stations "i+1" and "i+2" can open the busses by setting $I_{A,i+1}=I_{A,i+2}=I_{B,i+1}=I_{B,i+2}=0$. The configuration becomes now that of Fig.3.3 with station "i+2" generating the slots on bus A and station "i+1" generating the slots on bus B. It is evident that in order for this reconfiguration to be possible all the stations must have the capabilities of station "0" and be connected to both busses using the realization of Fig.3.2, i.e. all stations should have the capability to become SG.

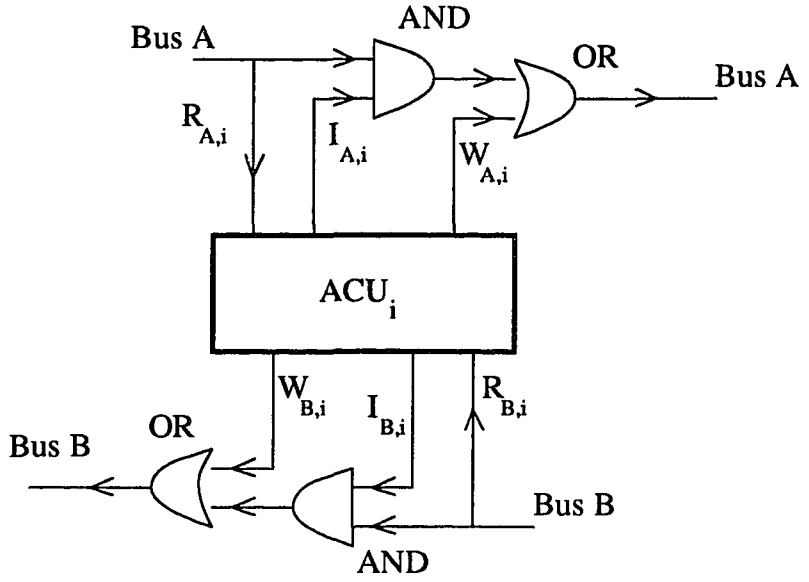

Fig. 3.2: Station "i" Access Control Unit connections when the station has the capability of interrupting the flow of information in both busses.

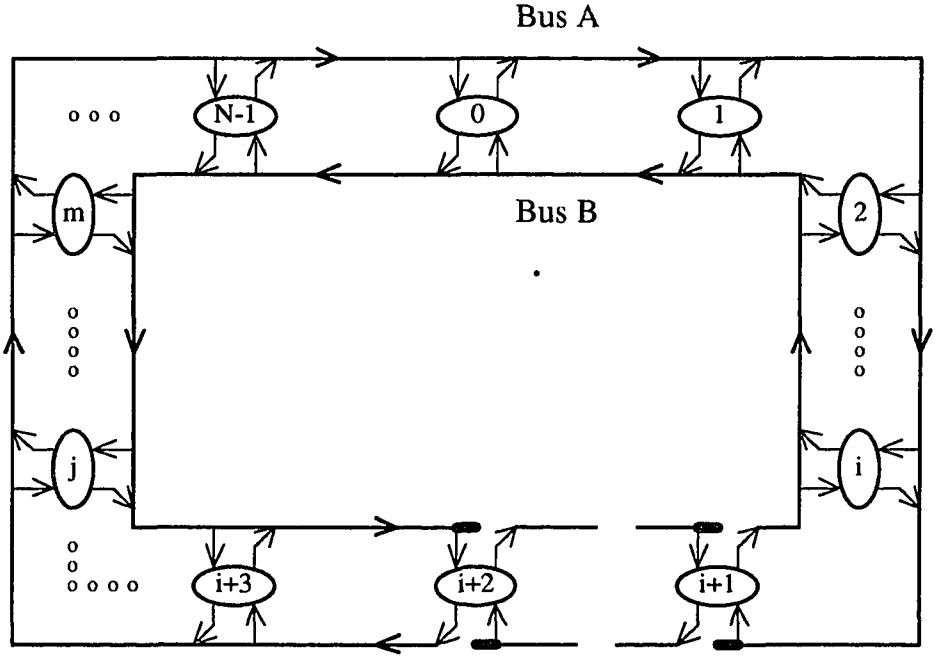

Fig. 3.3: DQDB reconfiguration after a bus fault between stations "i+1" and "i+2".

The RSG scheme uses the looped bus architecture of Fig. 3.1 with all stations having the capability of becoming slot generators for either bus. During initialization, station "0" becomes the slot generator for both busses. After some agreed upon time, N_{switch}, station "0" passes a control signal to station "1", which then becomes the slot generator. The same procedure is followed by station "1", and eventually by all other stations. Thus, the slot generator rotates around the loop. In this way the position of the stations relative to the slot generator changes periodically. It is therefore expected, since there are no favorable locations on the busses, that throughput and delay fairness will be achieved. In the sequel, two alternative mechanisms for switching the SG, the Immediate Switching and the Simultaneous Switching, are described in detail.

3.2.1 The Immediate Switching of the Slot Generator (IS_SG) Mechanism

The process of transferring the responsibility for generating slots from the current slot generator to the next station is initiated by the current SG ("old"), which informs all stations of its decision through a control message. This message can be implemented by setting a bit in the ACF of a slot. Hence, a new control bit, the Move Slot Generator (MSG) bit, must be introduced in the ACF of the slot. Note that there are two available *reserved* bits in the current standard, one of which could be used as the MSG bit.

We now describe the sequence of actions that implement the transfer of the SG responsibility from one station, the "old" SG, to the next, the "new" SG. We consider that SG responsibility is transferred from station "0" to "1","2",...,"N" in a cyclic order. We can see from Fig.3.1 that the SG can write to and read from either of the two busses. In RSG, for reasons that will become evident in the sequel, each SG transmits its own segments into the opposite direction of the movement of the slot generator. For instance in Fig.3.1, where the SG responsibility will be transferred from station "0" to station "1", station "0" transmits its segments on bus B and receives segments from the other stations on bus A.

Let us now assume that N_{switch} slots have been generated on bus A since the time instant the "old" SG has become slot generator. The "old" SG sets MSG=1 in the first slot that generates on bus A and closes bus A after the last bit has been generated. It also sets MSG=1 in the first slot generated on bus B that follows the transmission of MSG=1 on bus A; our timing assumption is that the generation of slots on bus A precedes the generation of slots on bus B. When the "new" SG sees MSG=1 on a slot on bus A, it will allow this slot to pass by, open bus A, and become the SG for this bus. † However, it will continue sending its segments on bus A. The closing of the bus A by the "old" SG will not affect the writing of the slots on this bus, since the busy slots that pass, by the "old" SG, will be blocked at the "new" SG. Notice that during the switching of the SG, and until the MSG message which was transmitted on bus B returns to the "old" SG, the "new" SG will be the SG for bus A and the "old" SG will remain the SG for bus B. We see that during this transition period the "new" SG can receive segments from the other stations on both busses, which, as we will see, greatly facilitates the operation of the system.

As the two MSG messages travel on the two busses the other stations are notified about the change in the position of the SG. Since the SG must receive segments on bus A, the only modification that these station should make is to start sending their segments for the "new" SG on bus A. However, because the "new" SG can now receive segments from both busses, the stations do not have to do that immediately. In fact, a station with a segment for the "new" SG, that has already sent a request on bus A to reserve a slot on bus B, must first transmit this segment on bus B before it can switch to bus A (for its transmissions to the "new" SG). In this way, it is certain that there are no slots wasted due to the switching of the SG. Of course, in both the RSG and DQDB schemes there is

† If the station reaction time becomes a problem (e.g., in the case of very high capacity networks), the stations may agree to open and close the busses k slots after the instant they have seen and/or generated an MSG control message.

always a possibility that a reserved slot may not be used by a station because the station has already sent its segment on an empty slot that has arrived before the reserved slot.

It is evident from our discussion that some stations will see the MSG message first on bus A and then on bus B. Other stations will see these control messages in the reverse order. When a station sees the first of the two MSG messages, regardless of bus, it starts transmitting its segments for the "new" SG on bus A. When a station sees the second MSG message, it simply ignores it ‡ .

Finally, when the MSG message that has been transmitted on bus B arrives at the "new" SG, this station will wait for the last bit of this slot to pass, open bus B, and start transmitting its segments on bus B. When the "old" SG sees the MSG message on bus B it will complete its current transmission, reset MSG bit and close bus B. The "new" SG has now become the SG for both busses. It transmits its segments on bus B and receives segments from the other station on bus A. It is evident from the previous discussion that the incoming slot from the "new" SG on bus B will be able to follow immediately the last slot generated by the "old" SG only if the total cable latency is an integer multiple of the slot size; a latency buffer can be used for this purpose.

We now elaborate on some of the characteristics of the RSG operation. The transmission of the SG segments in the opposite direction of that of the RSG rotation is justified by the two previously mentioned reasons. That is, no reserved slots are wasted and the stations do not have to switch bus immediately, for the transmission of their segments with destination the "new" SG. This behavior is not possible if the SG transmits its segments in the same direction with that of the SG rotation. Consider for instance the case where SG is station "0", transmits its segments on bus A and receives segments on bus B. When station "N-1" sees the MSG message on bus B it must immediately switch to bus A for the transmission of its messages with destination station "0" because,

‡ The second MSG message can be used as a verification that the slot generator has indeed changed position. If a station does not see it within a propagation delay, an error will be reported and the recovery mechanisms will be activated.

otherwise, its transmission on bus B (following the MSG=1 bit) will be blocked at station "1" and will never reach station "0". Also notice that if station "N-1" has indeed a segment for station "0" and has sent a request on bus A, when the reserved slot arrives on B, station "N-1" will not use it because its segment must be transmitted now on bus A. We point out that this type of unused reserved slots are due to the switching of SG and are different from the unused reserved slots in the case of DQDB. In order to clarify that, consider that neighbor stations are located one slot away from each other and that the instant station "0" generates the MSG message for bus B, all the slot on bus B are busy and bus A is full of requests for segments with destination station "0". It is then evident that all the idle slots that are generated by station "0" and follow the MSG=1 bit on bus B will not be used by the stations since their segments must be transmitted on bus A. This type of slot wastage does not appear in the case of DQDB. Therefore, the implementation of the RSG mechanism with the current SG sending segment in the same direction with the SG rotation introduces the potential of higher slot wastage than DQDB. We point out that in practise however, as simulation results have shown, this kind of wastage is negligible and usually the reserved slots are written by other downstream stations on bus B. This version of RSG, where the SG transmits on bus A, has been presented and analyzed in [70] and [71].

We now elaborate on the transmission of MSG messages by the "old" SG on both busses. Let us assume, again, that station "0" is the "old" SG which sends an MSG message only on bus A and closes bus A. Station "1" sees the MSG message and opens bus A. When the MSG message returns to station "0" it must close bus B and allow the MSG message to arrive at station "1" which will open bus B. If this is the case, then all the possibly busy slots on bus B which are between stations "1" and "0" at the time station "0" closes bus B, as well as all the busy slots that pass by station "1" on bus B during the transition of the MSG message from station "0" to station "1" on bus A, will complete another rotation on bus B before they return to station "1" to be blocked. Therefore, in

this case bandwidth will be wasted and the destinations of these slots will see them twice. A similar behavior will be observed if station "0" closes both busses when it sends the MSG message on bus A. A better approach would be the following. At the time the MSG message returns to station "0" on bus A this station simply allows it to pass by. When this message arrives at station "1", station "1" sets MSG=1 in the next slot on bus B and opens bus B. When station "0" sees the MSG bit it closes bus B. We see that in this last case, again, we have to send MSG messages on both busses. We prefer the approach where station "0" generates the two MSG messages on both busses because the stations are informed earlier about the switching of the SG and have more time to adjust their behavior.

Another important issue is the recovery of the mechanism in the case where the "new" SG suddenly crashes just before the switching of the SG. One way of dealing with this type of problem is to allow the "old" SG to send one only MSG message on bus B, but to continuously set to 1 all the MSG bits on bus A, until it receives the first slot on bus A that follows the MSG message(s) with MSG=0. The reason for this approach is the following. Consider that station "0" is initially the SG and that decides to pass the responsibility of generating slots to station "1". Station "1" knows that it is its turn, allows MSG=1 to pass by and opens bus A. All the subsequent slots with MSG=1 will consequently be blocked at station "1", and all other stations will see one only MSG=1 bit. Assume now that during the transition of the SG station "1" crashes and it cannot become the new SG and therefore will not open bus A. Station "2" will see the first MSG=1 bit and will realize that station "1" has now become SG. However, when it sees the second MSG=1 will realize that station "1" did not become an SG and that it (station "2") should become SG. Therefore, immediately after the slot carrying the second MSG=1 bit it will open bus A. All the other stations will see two consecutive MSG=1 bits and realize that "new" SG is the station that is second next to the "old" SG. In general if the "i" next stations of the "old" SG have crashed "i+1" consecutive MSG=1 bits

will appear on the channel, and all the active stations will know that the "i+1" station following the "old" SG will become the "new" SG.

Counters Update

In this subsection we investigate how the various stations should modify the values of their RQ_CTR and CD_CTR when they see the first MSG message in either of the two busses. We consider bus A first. The "new" SG for bus A has now become the last station on that bus (and first on bus B, where it transmits all its traffic). Thus, it does not transmit any segments on bus A, and its counters that govern its operation on this bus do not play any role (so, they might as well be reset to 0). The stations that were downstream from the "new" SG on bus A see the same downstream stations as before, except that the "new" SG has now been added to the list. Their counters have an accurate view of the requests on bus B, since the requests sent from the new SG, when it was the first station on bus A, were not counted by any RQ_CTR. Therefore, all these stations should not modify the values of their RQ_CTR and CD_CTR, which control their operation on bus A.

We now concentrate on the counters that control the operation of the stations on bus B. Let us first consider all stations, other than the "new" SG. For all these stations the only segments that change bus of transmission are those which are destined for the "new" SG; they are now transmitted through bus A. However, during the switching of the SG the stations will continue transmitting segments destined for the "new" SG on bus B, unless there is no pending request. Consequently, all the requests on bus A remain valid and the counters should not be altered.

Finally, we consider the counters controlling the operation on bus B of the "new" SG. The "new" SG becomes the first station on bus B and starts transmitting all its segments on bus B. We may assume that the "new" SG was the first station on bus B, always idle, during the period that the "old" SG was generating slots. Notice that the RQ_CTR of the "new" SG, controlling the operation on bus B, can be at most 1 at the instant of the

switching of the SG † . The approach we take in setting the RQ_CTR of the "new" SG is the following. At the instant the "new" SG opens bus B, it uses as value for the RQ_CTR the request bit seen on the most recent slot on bus A. Therefore, if the last slot carries a request, the "new" SG will allow an idle slot on bus B to go by.

3.2.2 The Simultaneous Switching of the Slot Generator (SS_SG) Mechanism

In the case of the IS_SG mechanism, as the MSG messages are traveling around the busses, they inform the various stations that they must switch bus of transmission for their messages destined to the "new" SG. Although the stations may not switch busses immediately, due to the outstanding request that maybe present, the change of bus of transmission will temporarily increase the offered load on bus A. This increase of the offered load on bus A is temporary because as soon as the MSG message sent by the "old" SG on bus B arrives at the "new" SG, this station will also switch bus for its transmissions to all other stations (it will now transmit on bus B). However, this temporary increase of the load consistently happens on bus A. It is evident that the smaller the value of N_{switch}, the more frequently this transient phenomenon is observed and the stronger the effect it has on the average delay. The objective of the Simultaneous Switching of the SG (SS_SG) mechanism is to eliminate this transient behavior, by forcing all stations to change bus of transmission at the same time. Furthermore, under the SS_SG mechanism, the network segment between the "old" and the "new" SG becomes isolated during the switching of the SG. The SS_SG mechanism enables the utilization of this bus segment. This point is clarified in the next few paragraphs, where the SS_SG procedure is described in detail.

We now focus on the sequence of actions that implement the transfer of the SG responsibility from one station to the next in the case of the SS_SG mechanism. The

† The "new" SG did not transmit any segments on bus B. Therefore, for each incoming request on bus A that increases its RQ_CTR by one, the next empty slot generated on bus B returns this RQ_CTR to 0.

"old" SG sets MSG=1 in the first slot that generates on bus A but leaves bus A open; contrary to the IS_SG mechanism. The "old" SG also sets MSG=1 in the first slot generated on bus B that follows the transmission of MSG=1 on bus A. When the "new" SG sees the MSG message on bus A, it will allow this slot to pass by and then open both busses; under the IS_SG mechanism the "new" SG opens only bus A. Notice that now the segment of the network between the "old" and the "new" SG is isolated from the other stations. Consequently, it can be utilized by both the "old" and the "new" SG. Indeed, after the transmission of the MSG bit, the "old" SG will start transmitting its segments for the "new" SG on bus A. Similarly, immediately after the opening of the busses by the "new" SG, the "new" SG will start sending its segments for the "old" SG on bus B. Since part of the offered load from the "old" and "new" SG is now transferred onto the isolated bus segment, the offered load on the rest of the busses will decrease and the performance of all stations will improve. Of course this improvement will be probably minor in the case of underloaded stations and more significant in the case of overloaded stations.

When the MSG bit on bus B arrives at the "new" SG, this station will forward it to the "old" SG. Notice that bus B is open at the "new" SG. This means that the slots coming on bus B will be stopped at the "new" SG. However, the "new" SG, using the AND gate, can reset (block) all the bits of the incoming slots on bus B but the MSG bit. In this way the MSG=1 bit will be immediately forwarded to the "old" SG. Then, after the "new" SG has observed the MSG=1 bit on bus B, it will reset all the bits of the passing slots. When the MSG bit arrives at the "old" SG, this station will erase it and close bus B. Furthermore, when the MSG bit on bus A returns to the "old" SG, this station will reset it, and close bus A. Now, the "new" SG is generating slots for both busses.

The objective of the SS_SG mechanism is to enable stations to rearrange their transmission queues simultaneously †. This is implemented in the following way. We

† With exception of the segments of the "old" SG and "new" SG which are destined to each other. These segments have already been rearranged in order to utilize the isolated network segment.

assume that all stations know the ring latency, L_{at}. This information may be provided to the station at the instant it is connected to the network. Let us also assume that the distance of station "i" from the "old" SG on bus A is d_i slots; its distance on bus B is $L_{at}-d_i$ slots. It is evident that the interarrival time, $INTR_i$, of the two MSG bits that are issued by the "old" SG is equal to $|L_{at}-2d_i|$; where $|x|$ is the absolute value of x. Each station "i" can calculate the value of $INTR_i$ by the use of a counter which starts counting the passing slots when it sees the first MSG bit on either bus and stops when it sees the MSG bit on the other bus. The final value of this counter will be equal to $INTR_i$. Eventually, the MSG bit (either on bus A or bus B) will return to the "old" SG after L_{at} slots from the time it was issued by the "old" SG or $(Lat-INTR_i)/2$ slots after the second MSG bit was observed by station "i". Consequently, the stations may rearrange their transmission queues at the moment the MSG bits returns to the "old" SG, that is, after $(Lat-INTR_i)/2$ slots from the time they observed the second MSG bit. The following example may clarify the above mechanism.

Assume that Lat=80 slots and that the distance on bus A of a tagged station from the SG, for instance station "0", is 24 slots. Its distance from the SG on bus B will then be 56 slots. Assume that at t=0 station "0" sends an MSG bit on both busses. The first MSG bit will arrive at the tagged station at t=24 on bus A and the second at t=56 on bus B. The interarrival time of the two MSG bits is INTR=56-24=32 slots. The MSG bit will return to station "0" at t=80, i.e (80-32)/2=(Lat-INTR)/2=24 slots after the instant the tagged station saw the second of the MSG bits. At this instant all stations may switch bus for the transmission of their segments for station "1". Station "1" also switches bus for the rest of its segments and from now on transmits its entire load on bus B. Recall that station "1", at the instant it saw the MSG bits and opened the busses, switched the transmission of its segments for station "0" to bus B.

Although, the intention of the SS_SG mechanism is to enable stations to switch bus of transmission simultaneously, it is not desirable to waste any bandwidth due to the

switching of the SG. For preventing the wastage of any slots we have adopted the approach of the IS_SG mechanism. Notice that the "new" SG can receive segments from both busses. Therefore, a station does not have to switch bus immediately in the case where it has already sent a request on bus A to reserve a slot on bus B for its segment with destination the "new" SG. In this case, the station may wait to transmit the first segment (for which it has reserved a slot) before it switches bus. Thus, no slot will be wasted due to the switching of SG. Moreover, the recovery of the mechanism when the next in line SG crashes is identical to the one described in the IS_SG case. Finally, the counters of the stations can be updated in a similar way as in the case of the IS_SG mechanism.

3.2.3 Transmission Queue Management

One issue that the RSG protocol has to deal with is the management of its transmission queues. The reason is that the location of each station with respect to SG and each other is not fixed, but depends on the current position of SG. Therefore, segments from one station destined for another station must be transmitted sometimes on bus A and sometimes on bus B. Furthermore, segments from the same long message may have to be transmitted over different busses because the location of SG has changed. It is evident that new functionality should be added to the DQDB layer in order to be able to deal with the rotation of SG. A queue rearrangement is required every time the responsibility of generating slots is transferred to the next station. Notice, however, that during the switching of SG the only messages that need reassignment are the following:

1. For the "new" SG, all messages.
2. For all other stations, only the messages destined to the "new" SG.

Under the IS_SG mechanism, type "1" messages do not need any rearrangement, since all of them should simply change bus of transmission; from A to B. When the SS_SG mechanism is used, type "1" messages are rearranged in two steps. First, the messages with destination the "old" SG should change bus of transmission when the first

MSG message is seen by the "new" SG. Notice, however, that this switching does not have to be immediate since the "old" SG can receive segments from both busses. The rest of the type "1" messages should change bus of transmission when the MSG bit arrives at the "old" SG. In this case, all messages which were being transmitted on bus A change bus of transmission and thus no actual rearrangement takes place. For the type "2" messages, for both switching mechanisms, there will be plenty of time for their rearrangement since each station will not have to switch bus immediately; we remind the reader that the "new" SG can receive segments from both busses. We finally point out that the possible reception of segments from the same message on different busses will not confuse the receiver, since the segments arriving on bus A must always precede the segments arriving on bus B.

3.2.4 Basic and Standby RSG

We have already mentioned that DQDB may waste some slots, under certain loading configurations. This is the case when a station transmits its segment in an unreserved slot which precedes the one for which a reservation has been made. A slight modification of the Basic DQDB, the Standby DQDB, attempts to minimize the number of slots that can be wasted by reducing the number of requests a station can send for its queued segments. According to Standby DQDB, if at the instant a station becomes busy, its RQ_CTR and CD_CTR are 0, and the next slot on the bus is empty, then the station will transmit its segment and will not insert any request on the reverse bus; since there is no reason for reserving a slot for a segment that has already been transmitted. As a result, Standby DQDB can provide lower delays to the stations than Basic DQDB. However, the delay variation among the stations is higher.

Similarly, for the RSG scheme we have considered two different versions, the Basic RSG and Standby RSG, which use the Basic DQDB and the Standby DQDB mechanisms respectively. In most of the comparative performance results, presented in the next

section, the Standby RSG has been considered, since it provides lower delays to the stations. However, for completeness, in some figures the Basic RSG has also been included.

3.3 RSG Performance and Fairness

In this section we investigate the effects of various system parameters on the performance and fairness of the RSG scheme. We also compare RSG with Basic DQDB, Standby DQDB (STB_DQDB), and DQDB with the BWB mechanism (BWB_DQDB). We have considered both Standby and Basic RSG, as well as the two switching mechanisms, IS_SG and SS_SG. The switching mechanism that is used is indicated in parenthesis, i.e. STB RSG (SS) is the Standby RSG with Simultaneous Switching. We consider a high capacity network of 155.520 Mbps connecting 40 stations uniformly distributed over the busses. We assume a slot (or segment) size of 53 bytes, a propagation delay of 5 μsec/Km, and a distance between neighbor stations of 2 slot times, i.e., 5.45 μsec, or 1.09 Km. The total bus latency is then 80 slots, or equivalently, 43.6 Km. In all of our figures $N_{switch} = 160$ slots. This is the minimum sojourn time for which a station can be SG; 80 slots to complete a full rotation of the MSG bit and 80 slots (at maximum) to serve all pending requests, before the station changes bus of transmission. In the case of underload conditions we consider that each station sends the same amount of traffic to any other station. In this way, stations which are closer to SG on a bus see more downstream stations on this bus, relative to more remote stations, and therefore, generate more traffic on the bus. We call this type of loading *linear*. The corresponding value of total offered load used in this case refers to the total traffic generated by all stations on both busses, measured in segments per slot. We further assume that independent messages, consisting of one or more segments, arrive at each station according to a Poisson distribution. We finally mention that in the case of overload conditions we assume that the saturated stations become active simultaneously. An assumption about the initial state of the system is necessary because in the case of Basic DQDB the steady state

throughputs of the overloaded stations are drastically affected by the initial load conditions.

Note that in the case of the RSG scheme the same source station can transmit segments to the same destination station on either bus at different times, depending on the position of SG. Hence, it is reasonable to compare the various schemes considering the average segment delay over both busses. However, for completeness, we also present results for the average segment delay over bus A only. The average segment delay is defined as the mean time from the instant a segment arrives at a station until it starts transmission on either of the busses.

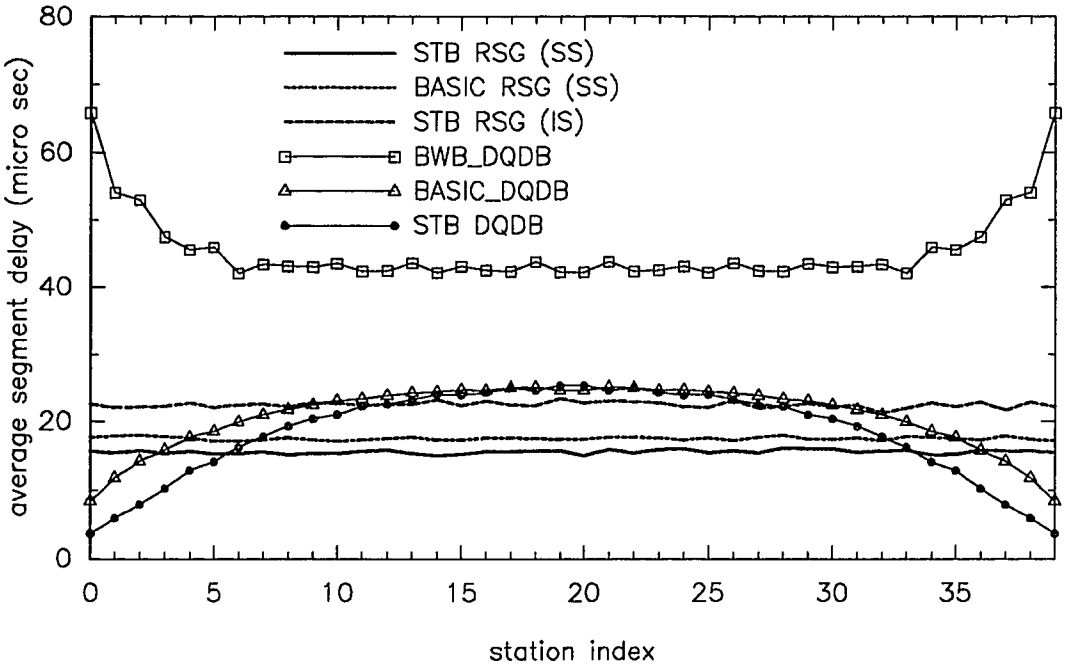

Fig.3.4:Delay comparison of Standby RSG(SS,IS), Basic RSG(SS), BWB_DQDB, Basic DQDB and Standby DQDB. Offered load per bus 0.9.

In Fig. 3.4 we show the effect of station location on the average segment delay of Standby RSG (IS), Standby RSG (SS), Basic RSG (SS), Standby DQDB, Basic DQDB and BWB_DQDB when the total offered load per bus is 0.9. In the case of BWB_DQDB we have selected $M = 10$, that is, each station increases the value of its RQ_CTR by one

every time it has transmitted 10 segments.† We see from Fig. 3.4 that all RSG variations are the most fair among the compared schemes. Furthermore, in the case of Standby RSG (SS), the stations encounter lower average delays. Standby RSG (IS) provides all stations with similar delays, but higher than the corresponding delays under the SS_SG mechanism. The reason for this behavior is the consistent overloading of bus A every time the SG switches from one station to the other. Finally, Basic RSG (SS) has higher delays than the corresponding Standby RSG because of the slots that can be wasted by the Basic DQDB mechanism. Basic DQDB and Standby DQDB favor significantly the end stations whereas BWB_DQDB penalizes significantly the end stations. However, the delay variation among the stations is smaller for the Basic DQDB. The same behavior is observed under higher and lower system utilizations, with the unfairness becoming more severe at higher loads.

The average delay of the segments transmitted on one bus provides a very good insight into the effect of the station location on performance. For this reason, in Fig. 3.5 we show the average delay of segments transmitted on bus A assuming the system parameters used for Fig. 3.4. A comparison of the delay curves of Figs. 3.4 and 3.5 shows that in the case of Standby and Basic RSG (SS) the average segment delay of the segments transmitted by a station on bus A is almost identical to the average segment delay of all segments transmitted by the station on both busses and is independent of the station location on the busses. Nevertheless, this is not the case for Standby RSG (IS), since bus A is constantly overloaded during the switching of the SG. The average segment delay on bus A is higher than the average delay over both busses; of course the average delay on bus B is lower than the average delay over both busses. Basic DQDB and Standby DQDB favor the first stations whereas the BWB_DQDB favors the last sta-

† For small values of M more bandwidth is wasted, but, under overload conditions, the system converges faster to a steady state that provides all stations with a fair share of the transmission bandwidth. The choice of M =10 provides a good trade-off between bandwidth loss and convergence speed to the steady state.

tions. The reason that upstream stations in the case of Basic and Standby DQDB encounter lower delays is that they see the idle slots earlier than the remote from the SG stations. The remote stations encounter higher delays because the farther from the SG they are the larger number of busy slots they see. In addition, the request bits they transmit must travel longer upstream to reserve slots, which also increases their delays.

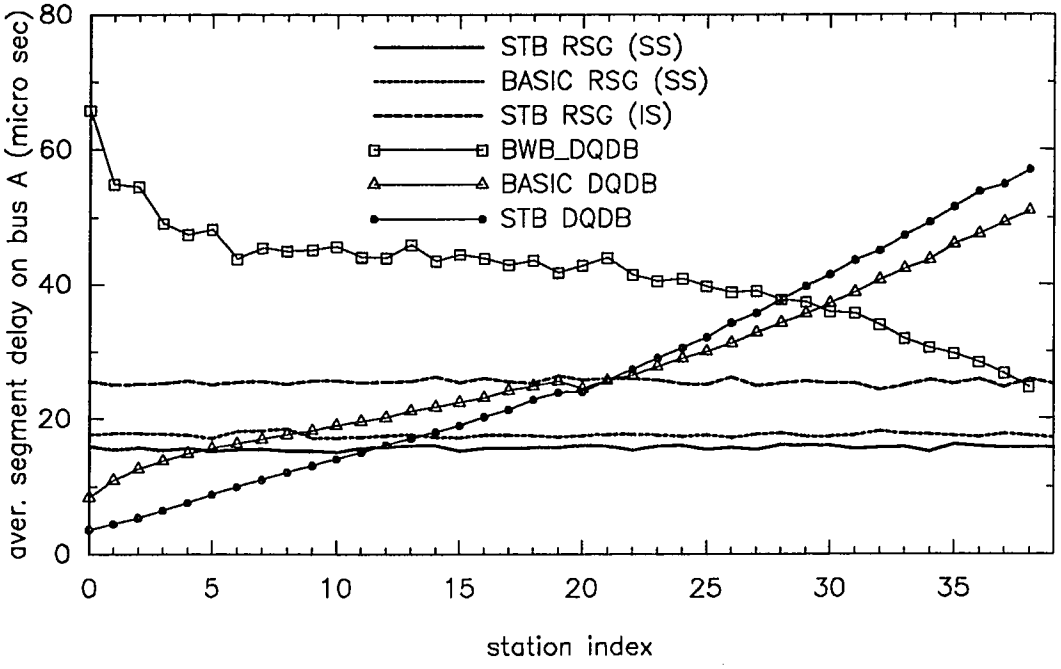

Fig.3.5:Average segment delay on bus A. Comparison of Standby RSG(SS,IS), Basic RSG(SS), BWB_DQDB, Basic DQDB and Standby DQDB. Offered load 0.9.

We point out that the behavior of BWB_DQDB, shown in Figs. 3.4 and 3.5, is due to the way we have implemented this mechanism. We remind the reader that in [7] two implementations of BWB_DQDB have been proposed. The first uses one RQ_CTR and one CD_CTR at each station. The second uses only one RQ_CTR and gives higher priority to the requests from the downstream stations by allowing a station to transmit only when its RQ_CTR is 0. Although both implementations evenly distribute the available bandwidth among the competing stations under overload conditions, they affect differently the segment delay of the various stations under lower data loads. Throughout

this chapter we have considered the simpler of the two implementations which requires only one RQ_CTR at each station. Figs. 3.4 and 3.5 then show that the closer to SG a station is, the higher its segment delay.† We see in Fig. 3.4 that the variation of the delay encountered by the various stations in the case of Basic DQDB is smaller than the corresponding variation in the case of Standby DQDB. The reason for this difference is that in the case of Basic DQDB the stations send a request bit for every segment they transmit which increases the number of request bits on bus B and slows down the upstream stations.

In the case of BWB_DQDB, the reason that upstream stations encounter higher delays is the free slots they must allow to pass. These free slots not only directly increase the delay of the upstream stations (because of the lower bandwidth they now see), but also give the opportunity to downstream stations to send additional requests upstream (since each slot that a downstream station writes enables this station to send another request). This slows down the transmissions of upstream stations even more. We finally point out that from Fig. 3.5 one can deduce the shape of the average segment delay curves of Fig. 3.4 since the average segment delay curves for bus B will be the inverse of those in Fig. 3.5. Notice that the end stations in the case of DQDB can transmit only on one bus and, for this reason, their average delay weighted over both busses has the same low value as the delay on one bus.

In Fig. 3.6 we consider the same system of Figs. 3.4 and 3.5, and show the effect of the station location on the variance of the delay using as performance measure the coefficient of variation, defined by $\sqrt{\mathrm{Var}(W_i)}/\overline{W_i}$, where $\mathrm{Var}(W_i)$ and $\overline{W_i}$ are, respectively, the variance and mean of the delay encountered by all segments of station "i". We see that the behavior of Standby DQDB and Basic DQDB is similar, that is, the delay variation of the end stations is small and as we move towards the center of the bus the

† We have also simulated the BWB_DQDB implementation that uses one RQ_CTR and one CD_CTR at each station. We have observed that in this case the shape of the average segment delay vs station index curves strongly depends on the network size (latency).

coefficient of variation first increases and then starts to decrease; with the stations in the center of the bus having, in the case of Basic DQDB, a smaller coefficient of variation than the end stations. However, the effect of the station location is much stronger in the case of Standby DQDB. In the case of the other schemes, the effect of the station location on the delay variation is rather minor, with Standby RSG (SS) demonstrating a higher variation than Basic RSG (SS), which is consistent with the observed behavior of Standby and Basic DQDB. Furthermore, Standby RSG (IS) has higher delay variation than Standby RSG (SS). This result is expected since the delay of the segments varies with the bus of transmission. Finally BWB_DQDB, which induces the highest average delays in Fig. 3.4, demonstrates the smallest delay variation.

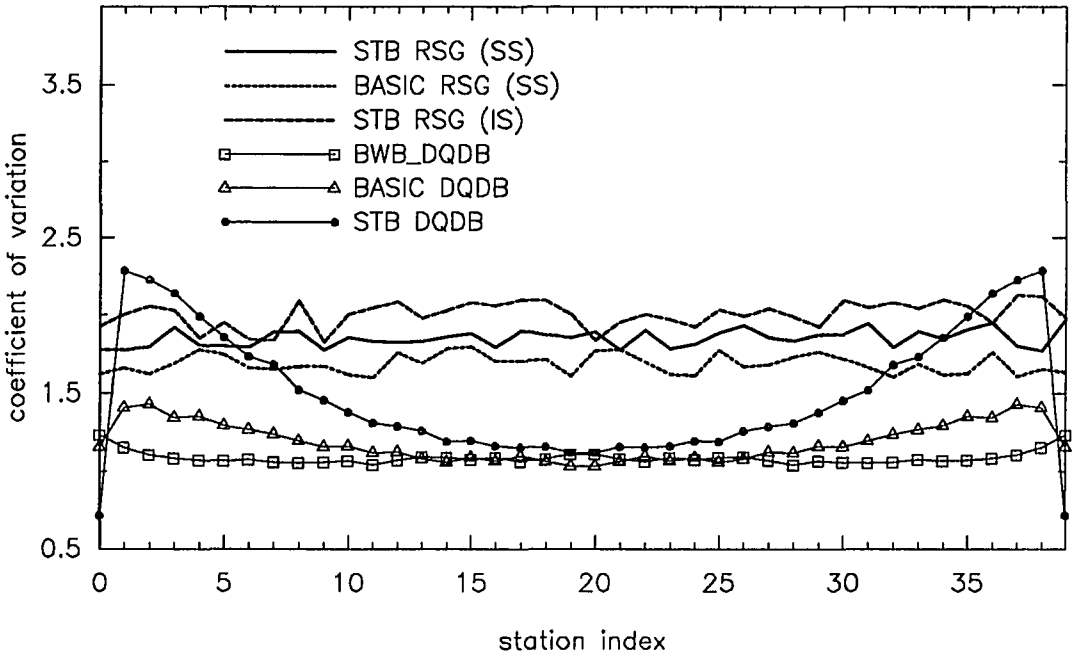

Fig.3.6:Delay variation comparison of Standby RSG(SS,IS), Basic RSG(SS), BWB_DQDB, Basic DQDB and Standby DQDB. Offered load per bus 0.9.

In Fig.3.7 the same system parameters with Fig. 3.4 are considered. However, in this case the stations transmit messages which are 20 segments long. Fig. 3.7 shows the

effect that the station location has on the average message delay of RSG, Basic DQDB and Standby DQDB. The message delay is defined as the elapsed time from the instant a message arrives at a station until its last segment has been transmitted onto the bus. Also in this case, RSG provides similar delays to all stations. Furthermore, the delay characteristics of Basic and Standby DQDB are very similar to the ones in Fig. 3.4, yet intensified.

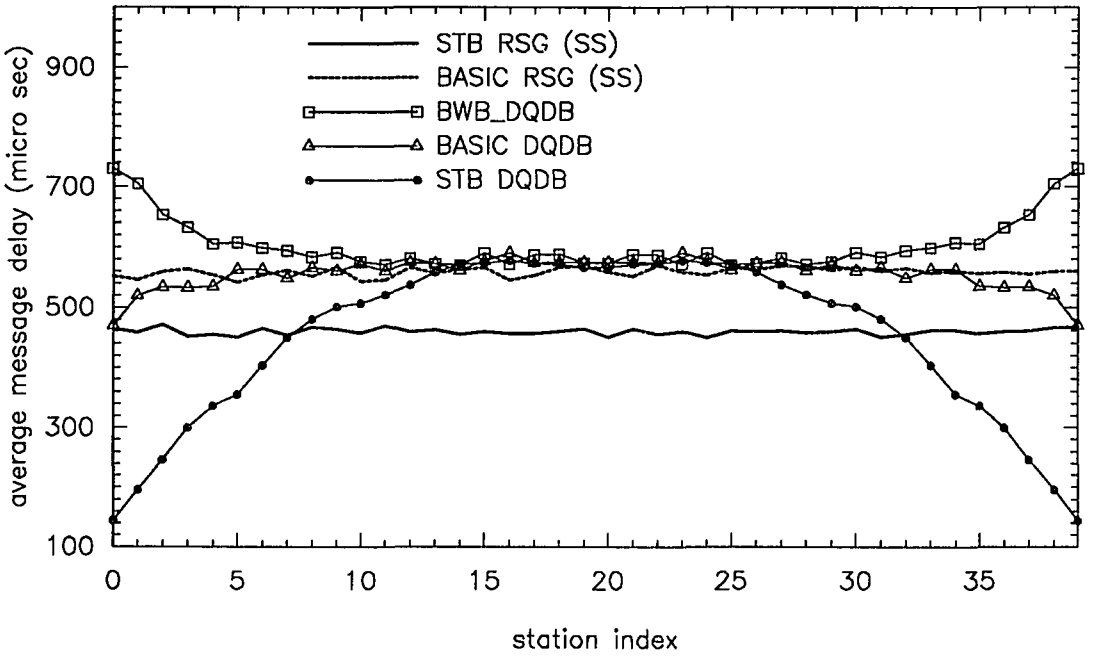

Fig.3.7:Delay comparison of Standby RSG(SS), Basic RSG(SS), BWB_DQDB, Basic DQDB and Standby DQDB. Offered load per bus 0.9. Constant message size = 20 segments.

The performance results shown in Figs. 3.4 through 3.7 clearly demonstrate that the selection of Standby versus Basic DQDB for the operation of RSG is a selection between lower average delay versus lower delay variation. Moreover, the selection of the IS_SG versus the SS_SG mechanism is a selection between simplicity of implementation and better performance. In the remaining figures of this chapter we focus on the performance of Standby RSG (SS) and refer to it simply as RSG.

In Fig. 3.8 we compare the throughput of RSG with that of the other schemes under saturation conditions. We assume that all stations have infinite queues of waiting segments so that they can write on any unreserved slot that is passing by. We then examine how the location of the stations on the busses affects their maximum throughputs by considering the total throughput over both busses. We observe that RSG is the only scheme in which all stations have the same throughput. BWB_DQDB provides the same throughput to all the stations except from the first and last one. The reason for this discrepancy is that these stations can only transmit segments on one bus, and therefore, the bandwidth they receive is half of that of the other stations. This is not the case with the RSG scheme where a station transmits segments on one bus, only while it acts as SG. Moreover, the utilization of the busses in the case of RSG exceeds one, since the isolated bus segment is being used each time the SG switches to the next station. Under the assumption that all stations have infinite number of segments for all the other stations, in the case of RSG, the utilization of the busses is 149%. † Fig. 3.8 also shows that in the case of Basic and Standby DQDB the throughput of a station decreases as it approaches the middle of the bus. This behavior is more profound in the case of Standby DQDB where most of the channel bandwidth is allocated to the two end stations on the busses. This behavior is expected since the end stations are the first ones to "see" all the idle slots. Therefore, they can transmit their segments on any non-reserved slot which is passing by. In contrast, the remaining stations see mostly busy slots, and before they can transmit they have to send a request bit, wait for this bit to travel upstream to reserve a slot, and then wait for the reserved slot to arrive. Only then they can transmit a segment and send the next request bit. For this reason the stations which are closer to the slot generator attain higher throughputs.

† During the N_{switch} slots a station is the SG, 320 slots are utilized from the main network. During the switch of the SG, the old "SG" transmits another 80 slots into bus A of the isolated bus segment in between the "old" and the "new" SG. Also, the "new" SG transmits another 78 slots into bus B of the isolated bus segment during the switch of the SG. Thus, 320 slots are generated and 478 segments are transmitted during the sojourn time of the SG.

It has been shown that in the case of Basic DQDB the initial conditions affect the bandwidth distribution among the various stations [10]. In the case of Fig. 3.8, as well as for all other figures in this chapter which assume saturated stations, all saturated stations are activated immediately, when the simulation starts. This means that in the case of Basic DQDB, initially, and despite the fact that all stations see idle slots and transmit their segments, they also send request bits. Thus, both busses are filled with request bits which drastically reduce the throughput of the end stations. Since for any segment a station transmits it can insert another request bit, the number of request bits does not decrease. Hence, the advantage of the end stations (because they see the idle slots first) is waived by the large number of request bits they receive. Notice also that an end station is the only one whose request bits will not produce any reserved slots and for this reason has a disadvantage over the second station on the bus which will receive more bandwidth. The combination of these reasons results in the capacity curve of Fig. 3.8.

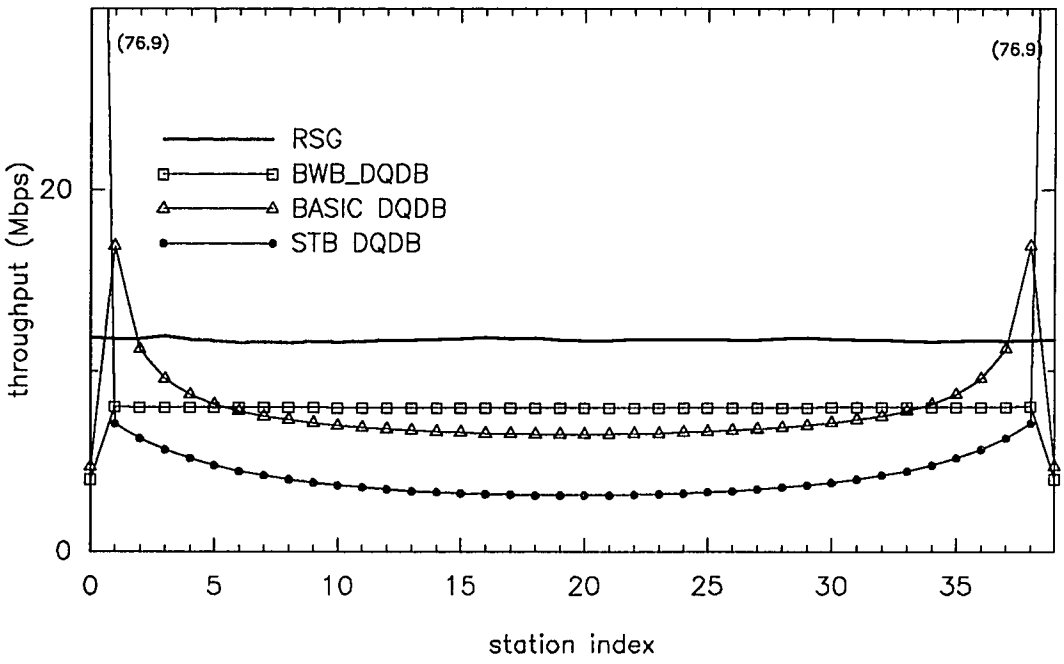

Fig.3.8: Effect of station location on throughput. Saturated stations. Comparison of RSG, BWB_DQDB, Basic DQDB and Standby DQDB.

In Fig. 3.9 asymmetric load conditions are considered. Three stations, i.e., "12", "25", and "38", are saturated and can write on any unreserved slot they see. The total offered load by the remaining 37 stations is 0.6 per bus. It is interesting to see the amount of bandwidth that the various stations receive in this case. Fig. 3.9 shows the total throughput of the stations over both busses. We observe that all schemes provide the requested throughput to the lightly loaded stations regardless of their position on the bus.

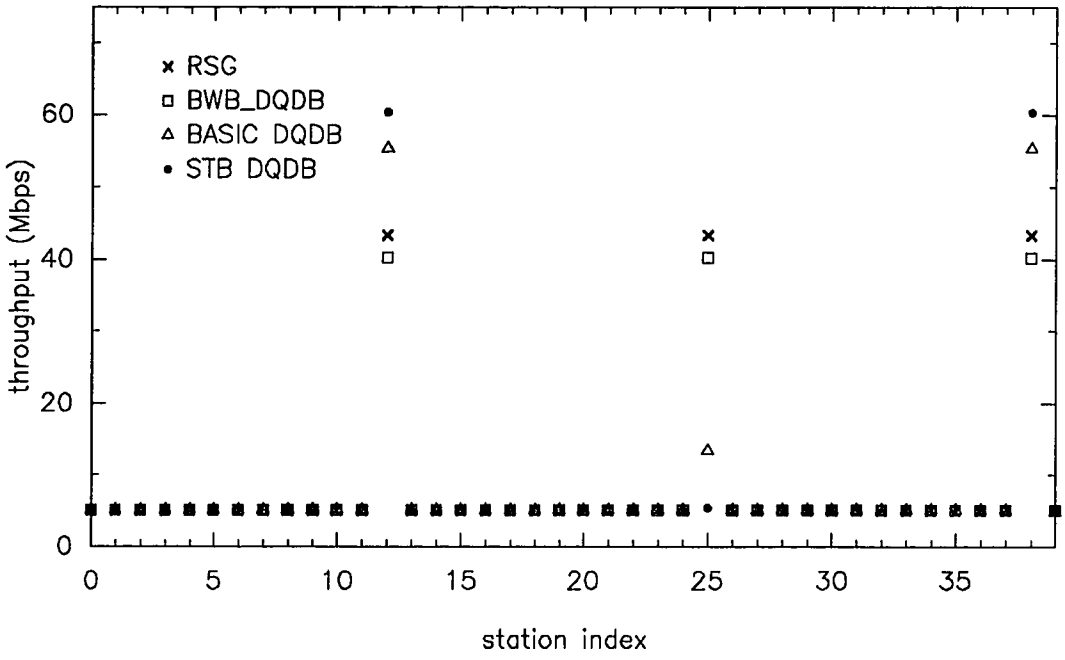

Fig.3.9:Effect of station location on throughput. Three saturated stations. The offered load of non-saturated stations is 0.6 per bus. Comparison of RSG, BWB_DQDB, Basic DQDB and Standby DQDB.

Furthermore, RSG and BWB_DQDB provide the same throughput to the overloaded stations; 43.4 Mbps with RSG, and 40.3 Mbps with BWB_DQDB. The throughput of the saturated stations in the case of BWB_DQDB is lower due to the free slots that the stations intentionally allow to pass by. In the case of Basic and Standby DQDB the throughputs of stations "12" and "38" are similar, 55 Mbps under Basic DQDB and 60 Mbps under Standby DQDB, and significantly higher than the throughput of station "25". In fact, in the case of Standby DQDB the throughput of the saturated station "25" is

almost equal to that of the underload stations. Fig. 3.9 is thus in agreement with Fig. 3.8 which provides the first indication that under overload conditions Basic and Standby DQDB favor the end stations, with Standby DQDB providing the end stations with most of the bandwidth.

Fig. 3.10 considers asymmetrically located stations on the busses; a repeated sequence of 7 busy and 3 idle stations. All busy stations can become slot generators. In Fig. 3.10 we assume that a total offered load of 0.9 per bus is evenly distributed among all (busy) stations, and show the effect of station location on delay for both RSG and BWB_DQDB. We show the average delay on bus A, as well as the average delay over both busses. We see that in the case of RSG, the average delay of the segments transmitted on bus A is similar to the average delay of all segments transmitted over both busses, and is not affected by the station location on the bus. However, in the case of BWB_DQDB, the two delays are significantly different. If we consider the average segment delay on bus A, we see that the first station in each sequence of 7 busy stations encounters significantly higher delays.

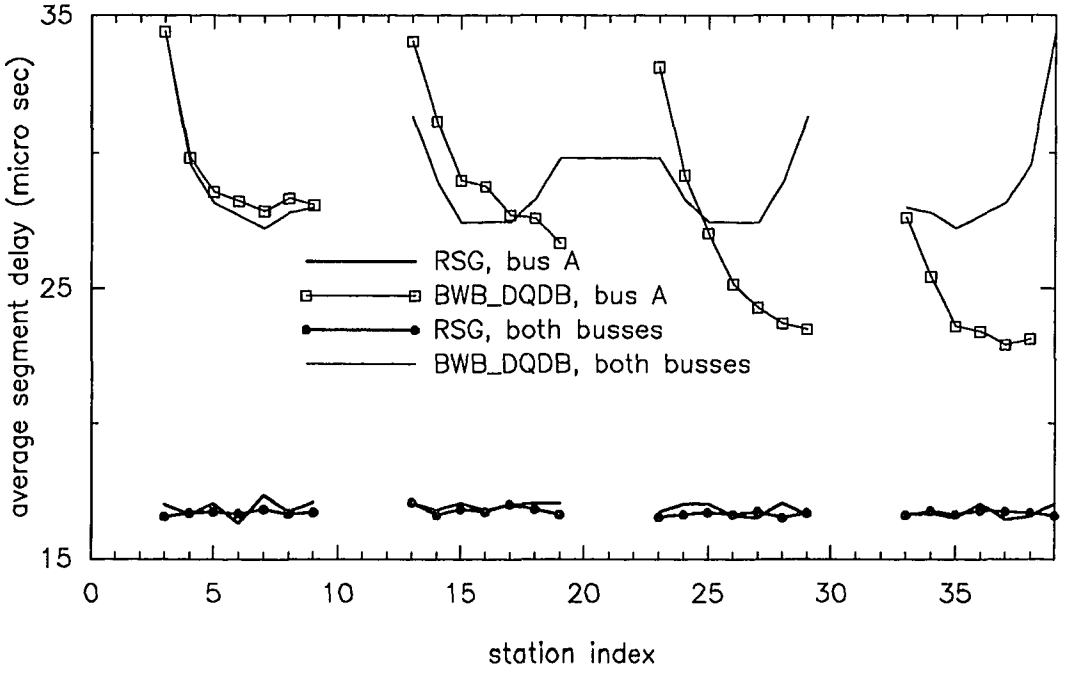

Fig.3.10 Asymmetrically located stations under linear load. Sequences of 7 busy, 3 idle stations. Delay comparison of RSG and BWB_DQDB. Offered load per bus 0.9.

Furthermore, as the distance of the stations from the slot generator increases, the delays of the stations in each group decreases. This behavior of BWB_DQDB is consistent with that of Fig. 3.5. Considering the segment delay over both busses, the end stations in each group encounter the higher delays, with the two end stations in each bus encountering the highest delays; a behavior consistent with that of Fig. 3.4. Finally, the previous system under saturation conditions has also been considered. We have found that both schemes provide all active stations with similar bandwidths, regardless of their locations on the busses.

In all figures, but Fig. 3.7, we have computed the throughput and the average segment delay for each user after a total number of 2×10^6 segments has been transmitted on both busses. In Fig.3.7, where messages of 20 segments are transmitted, the simulations were run for 10×10^6 segment transmissions. These figures have shown that in the case of RSG, a small variation of the average segment delay can be observed.

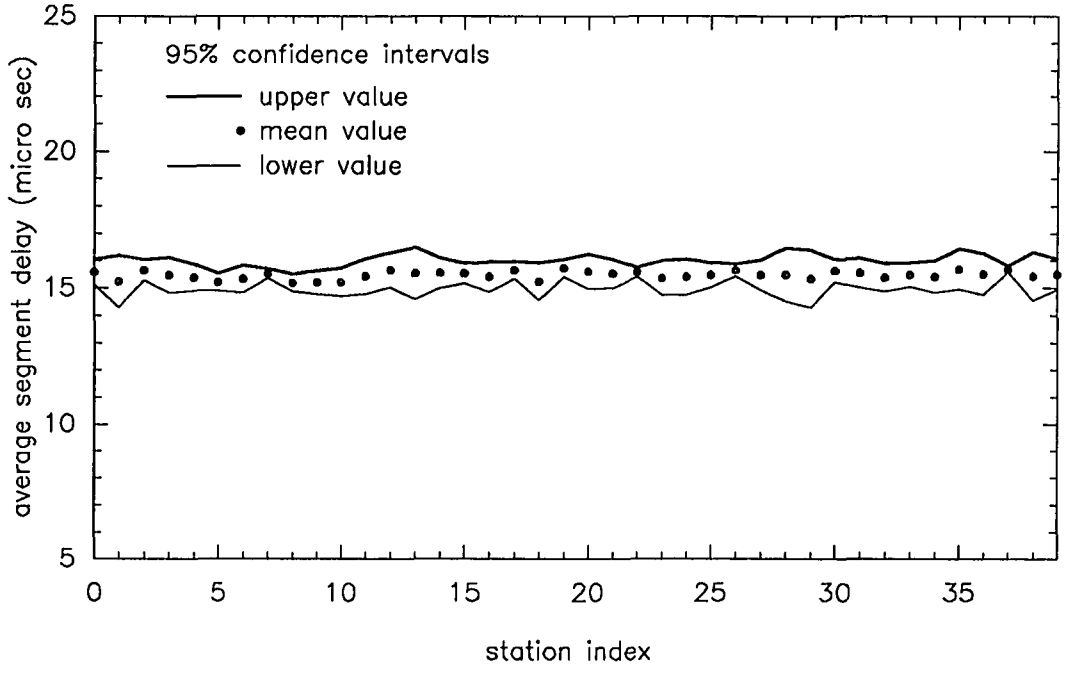

Fig.3.11:Effect of station location on average segment delay.
Offered load per bus 0.9.

In order to determine if there are systematic reasons for this delay variation, besides the statistical fluctuations due to a slow convergence to a steady state, we have computed, and plotted in Fig.3.11, the 95% confidence intervals for the segment delays of the various stations. We have used four independent runs at a total offered load of 0.9 per bus, i.e., the system parameters of Fig. 3.4. We see that there is no indication of favoritism for any of the stations on the busses. At lower loads, the confidence intervals become more narrow and the observed delay variation becomes less than one slot.

In all the performance results that we have considered until this point, we have focused our attention on the long term performance characteristics of RSG. That is, we have shown throughputs and average delays over very long time intervals, during which the responsibility of generating slots has rotated many times around the ring. We have demonstrated that RSG introduces fairness due to the cyclic rotation of the station locations. Consequently, no station is favored over a time interval of $T = N*N_{switch}$ slots, i.e. a full rotation of the RG. For N=40 and N_{switch}=160 slots, T=17.5 msec. In order to provide a better insight into the operation of RSG(SS) we show in Figs. 3.12 and 3.13 its transient behavior.

In Fig. 3.12 we have used the same parameters as in Fig. 3.4., and plotted the average access delay of station "0" on bus A and B, when SG is station "0", "1", "2", ..., "39". The access delay of a segment is defined as the delay of the segment from the instant it becomes first in the transmission queue until its first bit is transmitted onto the bus. As it is expected, the delay decreases as the station approaches the SG. Recall here that when station "0" is the SG (and after it has inserted the MSG bit) transmits also on bus A; the SS_SG mechanism utilizes the isolated bus segment. We have not plotted this access delay, since station "0" is the only one transmitting on that bus segment and thus it is not delayed. The same stands for the transmissions of station "0" on bus B , when SG is station "39". Finally, we mention that the shape of the curves remains very similar when the Basic RSG or the IS_SG mechanism is used.

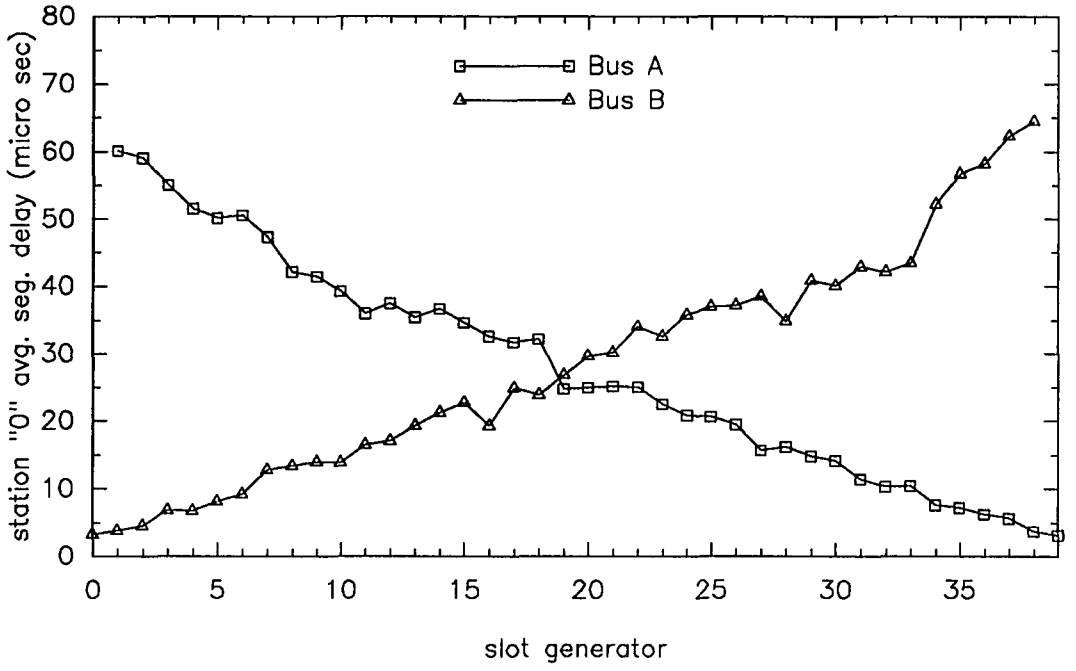

Fig. 3.12: Effect of the SG location on station "0" average segment delay. Offered load per bus 0.9.

Fig. 3.13 shows the transient behavior of station "0", in the case of saturated conditions. Here, we plot the station "0" throughput, when SG is station "0", "1", "2", ..., "39". We see that the performance of the station improves as it approaches the SG. Interestingly, on bus B the performance of the station slightly improves also when the station is very far from the SG. This discrepancy between the two busses is due to the opposite rotation of the SG on each bus. On bus A, the SG rotates on the same direction as the information flows. On the contrary, on bus B, the SG rotates on the opposite direction. Finally, similarly to Fig. 3.12, we have not plotted the throughput of the station when it uses the isolated bus segment. If the transmissions on the isolated part of the network are taken into account, the throughput of station "0" on the isolated bus segment will be 77.76 Mbps on and 75.81 Mbps when SG is station "0" and "39" respectively.

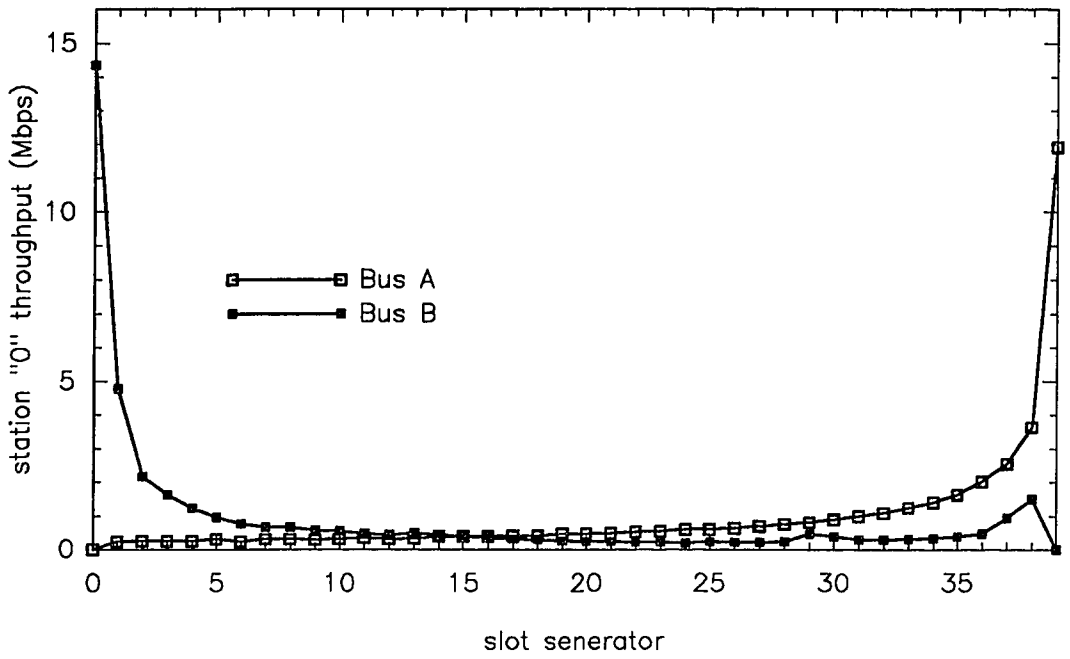

Fig.3.13: Effect of the SG location on station "0" throughput. Saturated conditions.

3.4 The RSG Scheme with Multiple Priorities

3.4.1 Protocol Description

As we have already mentioned in chapter 2 the DQDB MAC mechanism can be extended to support multiple priority classes of traffic. This is achieved by using a separate RQ_CTR and CD_CTR for each priority class at each station, and a separate request bit per class in the *Access Control Field* (ACF) of each slot. The priority service is implemented by having the RQ_CTR at a particular priority level counting requests only at the same or higher priority levels. Thus, RQ_CTR keeps track of all queued segments in downstream stations of the same or higher priority. The CD_CTR at a particular priority level is decremented for every idle slot seen on bus A, and is incremented for every

RB=1 seen on bus B of higher priority level. In this way higher priority segments have access to the medium ahead of already queued lower priority segments.

The access mechanism described above, although it provides true priority service when segments from the same station are considered, it cannot guarantee overall lower delays for the high priority messages. Depending on the location of the stations, lower priority traffic may get more bandwidth and encounter lower delays than higher priority traffic. Even the BWB mechanism cannot guarantee the same bandwidth to all high priority users; it can only guarantee that under overload conditions the high priority users will take as much bandwidth as the lowest priority users. For this reason, three priority mechanisms, based on extensions of BWB_DQDB, have been proposed and investigated in [35]. The first mechanism guarantees a minimum amount of bandwidth per station, with the highest priority class being capable to acquire all this bandwidth. Lower priority classes may receive some bandwidth only after the higher priority classes have satisfied their bandwidth requirements. The second mechanism can guarantee a minimum throughput to every priority class inside a station, regardless of the number of priority classes that are present at the station. Finally, the third priority mechanism enables higher priority classes to modify the rate at which they allow free slots to pass by, and in this way acquire more bandwidth.

In this section we investigate the performance of RSG when it uses the priority mechanism that has been proposed for DQDB (and in particular, the Standby version with Simultaneous Switching), which requires a separate RQ_CTR, CD_CTR, and request bit for each priority class in the system. Furthermore, we compare RSG with the proposed priority mechanism of Basic DQDB, as well as with the first of the previously mentioned priority mechanisms of BWB_DQDB (which has the simplest implementation). † According to this mechanism, different priority classes may use different values

† A thorough performance investigation of all three proposed priority mechanisms is provided in the next chapter, where the BWB_DQDB is compared to the NSW_BWB scheme.

of M, that is M_i for class i.‡ Each station always transmits its highest priority segments first in a FIFO order. Furthermore, every time a segment of priority i is transmitted, the station increases the value of a *Bandwidth Balancing Counter* (BWB_CTR) by $1/M_i$. When BWB_CTR exceeds one, the station will increase its RQ_CTR by one and decrease its BWB_CTR by one. It is shown in [35] that if $M_{h,j}$ is the value of M of the highest priority class at station "j", and all priority classes are overloaded, then the guaranteed throughput to station "j" is given by $T_j = M_{h,j} / (1 + \sum_{i=1}^{N} M_{h,i})$, where N is the number of stations in the system.

3.4.2 Performance and Fairness of the RSG Scheme with Multiple Priorities

We consider the same network of section 3.3, i.e. an 155.52 Mbps channel, 40 stations, 53 bytes slots, and 2 slot times inter-station distance. We assume that each station supports all classes of traffic. We first investigate the case of two priority classes. In this case we have selected $M_1 = 8$ and $M_2 = 4$ for BWB_DQDB. Again, for BWB_DQDB, we have used the version that does not require any CD_CTR. We point out that in the case of underload conditions for a given class, we assume that each user of this class transmits to any other user of the same class with the same probability, i.e. we assume a *linear* load on each bus.

In Fig. 3.14 we show the effect of station locations on delay of both high and low priority segments. The total offered load by all stations is 0.95 per bus and is evenly distributed among the two classes (0.475 each). We observe that with RSG, users at the same priority level encounter similar delays, regardless of their position on the bus. Moreover, the average segment delay of the high priority users is significantly lower than the delay of the low priority users. In the case of Basic DQDB the delay of all users increases as their location approaches the middle of the busses. However, the delay vari-

‡ We use the convention that smaller values of the index i correspond to higher priority levels, i.e. i=1 indicates a higher priority than i=2.

ation for the high priority users is not significant. In the case of BWB_DQDB the end users for both priority classes encounter significantly higher delays. The high priority users closer to the ends of the busses encounter higher delays than the low priority users located in the middle of the busses. Furthermore, the average segment delay of the high priority users is significantly higher than the corresponding delays in the case of RSG and Basic DQDB. Finally, the average segment delays of the low priority users in the case of RSG are lower than the smallest average delays of the low priority users in the case of BWB_DQDB (encountered by the low priority users in the middle of the bus).

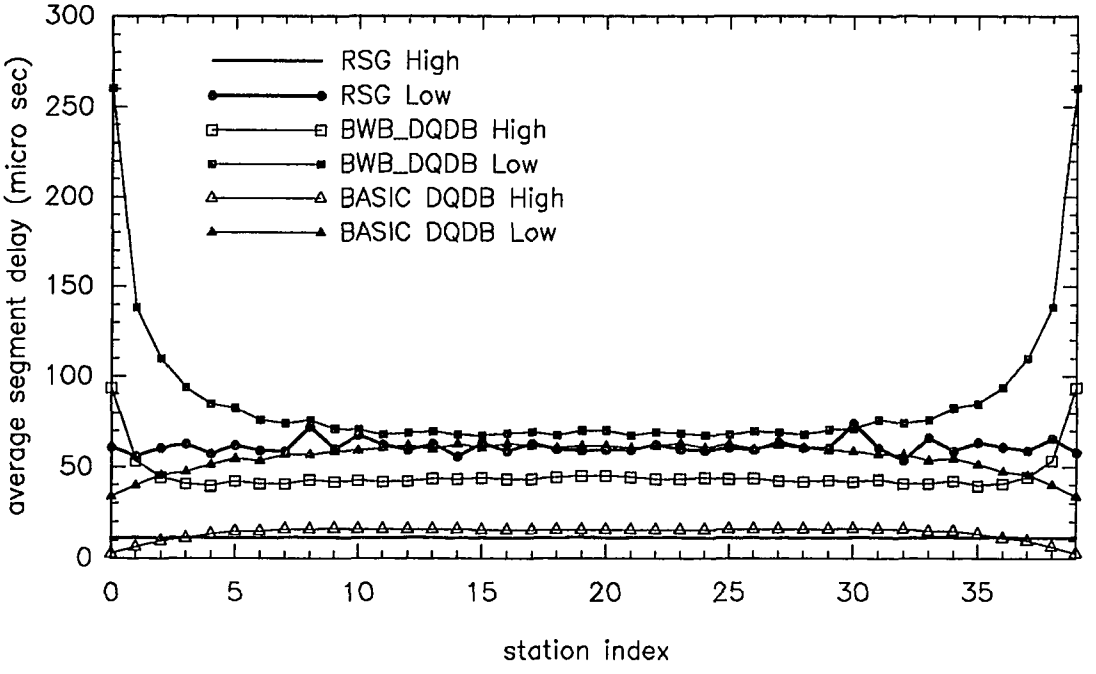

Fig.3.14: Two priority classes of traffic. Delay comparison of RSG, BWB_DQDB and Basic DQDB. Total offered high and low priority load 0.95 per bus. Traffic mix: 50% high, 50% low.

In Fig. 3.15 we assume that the lower priority users are saturated. The total offered load by the high priority users is 0.6 per bus. It is interesting to see in this case the effect of the saturated low priority users on the high priority users. Fig. 3.15 shows that all three schemes provide the requested throughputs to high priority users, independently of

their position on the busses. Furthermore, RSG and BWB_DQDB evenly distribute the remaining bandwidth among the low priority users, with the exception of the low priority users at the two end stations in the case of BWB_DQDB. The reason for this discrepancy is that the end users can transmit only on one bus, while at the same time, they have to compete with high priority users who at the end stations have the highest load, since they transmit segments to 39 other users on this bus. As a result, the total amount of bandwidth they receive is significantly lower. The behavior of the low priority users in the case of Basic DQDB is similar to the one observed in Fig. 3.8, where we investigated the behavior of Basic DQDB under one traffic class and overload conditions. That is, the end users have small throughputs because they can transmit on one bus only, stations "2" and "38" acquire most of the bandwidth, and the throughputs of the other stations decrease as we approach the center of the busses.

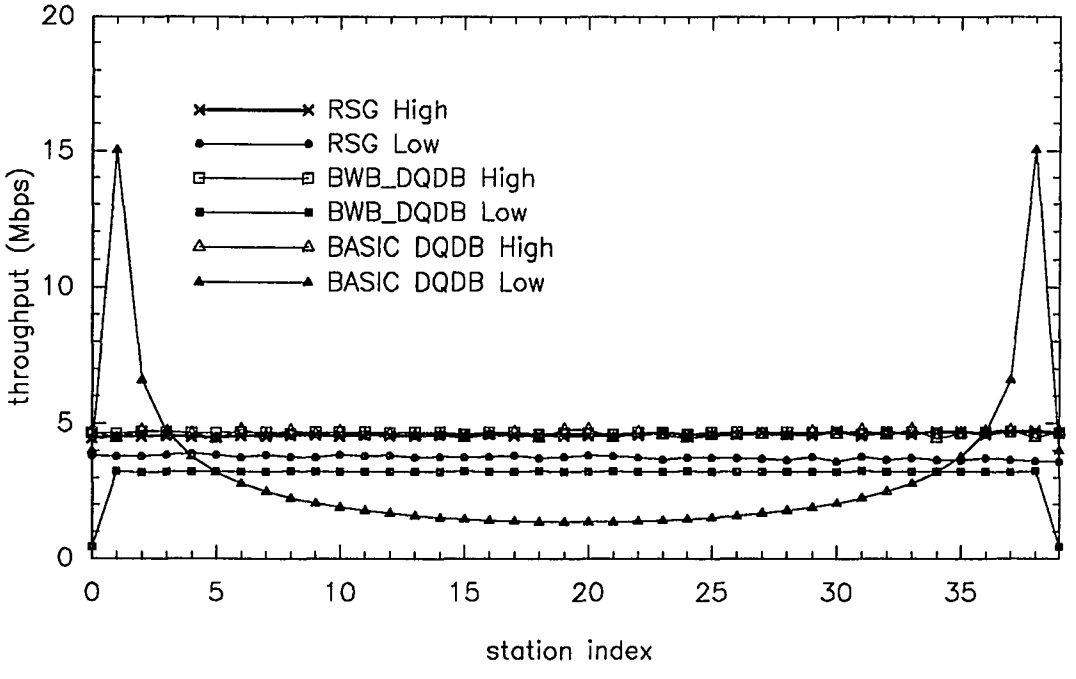

Fig.3.15:Two priority classes of traffic. Effect of station location on throughput. Saturated low priority queues. Total offered high priority load 0.6 per bus.

Because of the very asymmetric throughput distribution in the case of Basic DQDB, in the remaining figures of this section, we only compare the performance of RSG with that of BWB_DQDB. In Figs. 3.16 and 3.17 we show the effect of the station location on the average segment delay under an offered load of 0.95 per bus, and two different traffic mixes. In Fig. 3.16 the traffic mix is 20% high priority and 80% low priority while in Fig. 3.17 the traffic mix is 80% high priority and 20% low priority. These figures clearly show, again, that BWB_DQDB severely penalizes the end users of both priority classes. In contrast, RSG provides similar delays to the users of the same priority class. Furthermore, the average segment delay of the high priority users is significantly lower in the case of RSG. Finally, when the traffic mix is 20% high priority and 80% low priority, the average segment delay of the low priority users in the case of RSG is similar to the average segment delay of the high priority users in the case of BWB_DQDB.

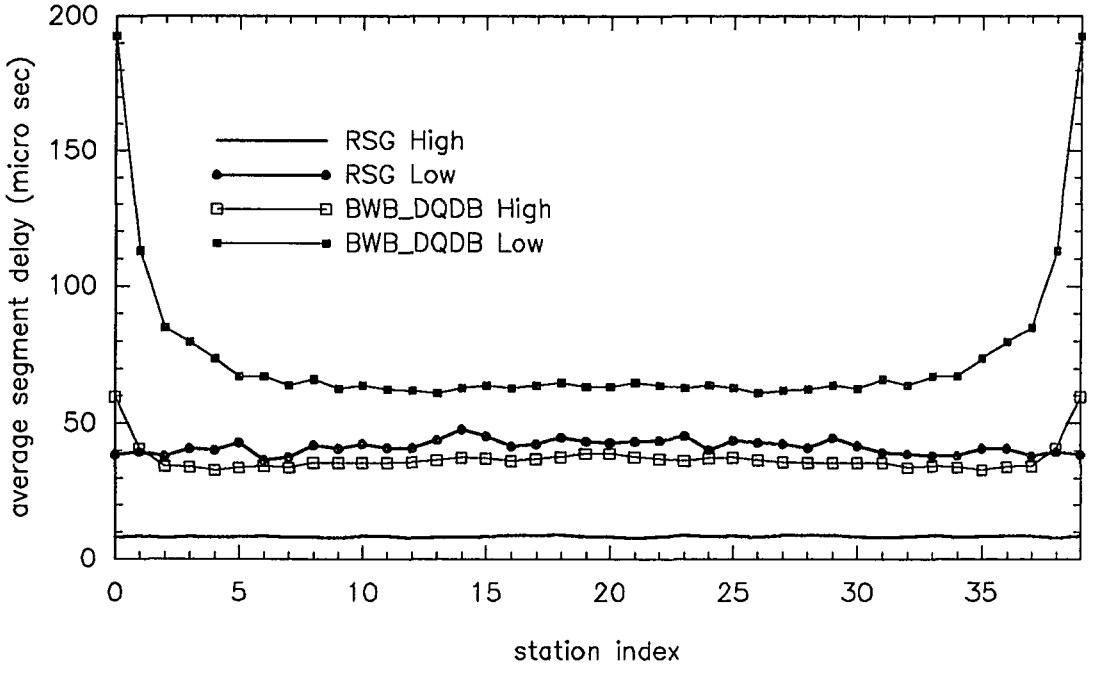

Fig.3.16:Two priority classes of traffic. Delay comparison of RSG and BWB_DQDB. Offered high and low priority load 0.95 per bus. Traffic mix: 20% high, 80% low.

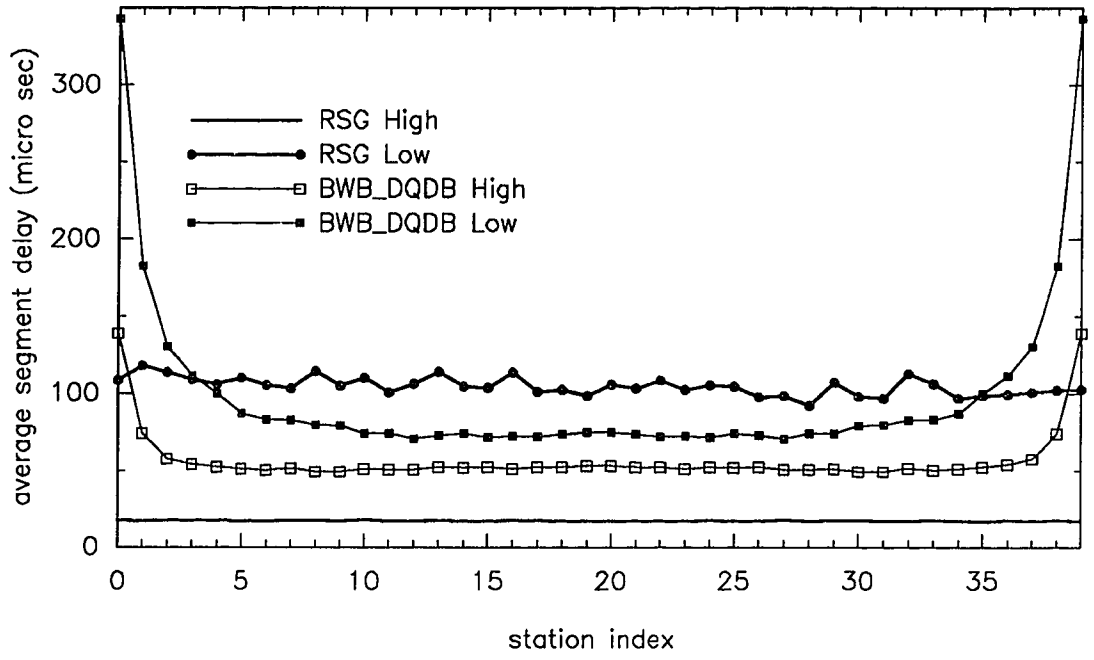

Fig.3.17:Two priority classes of traffic. Delay comparison of RSG and BWB_DQDB. Offered high and low priority load 0.95 per bus. Traffic mix: 80% high, 20% low.

In Fig. 3.18 we show the effect of the station location on the throughput when the high priority users at stations "12", "25", and "38" are saturated. The offered load by all low and remaining high priority users is 0.6 per bus, evenly distributed over the two classes; i.e. 0.3 each. We see that with both schemes the saturated high priority users acquire most of the bandwidth. Furthermore, the overloaded users in the case of RSG receive more bandwidth than the overloaded users under BWB DQDB, a behavior which is similar to the one observed in Fig. 3.9 for a single traffic class. The reason is again the bandwidth wasted by BWB. This behavior is mainly observed when the number of overloaded users is small. This is because the bandwidth loss is due to the idle slots that the last of the overloaded users allows to pass by. Therefore, if the number of saturated users is small the amount of bandwidth that each one of them will receive will be significant and the last of them will allow a significant number of idle slots to pass by. Because

in both Fig. 3.9 and Fig. 3.18 the load of the lightly loaded stations is linear and the last of the saturated users is at station "38", none of the slots that user "38" allows to pass will be used by station 39 since this station is last on bus A. Fig. 3.18 also shows, as expected, that the low priority users that share the same stations with the saturated high priority users do not receive any bandwidth. The remaining low and high priority users receive approximately equal bandwidths, which are independent of the station locations on the busses.

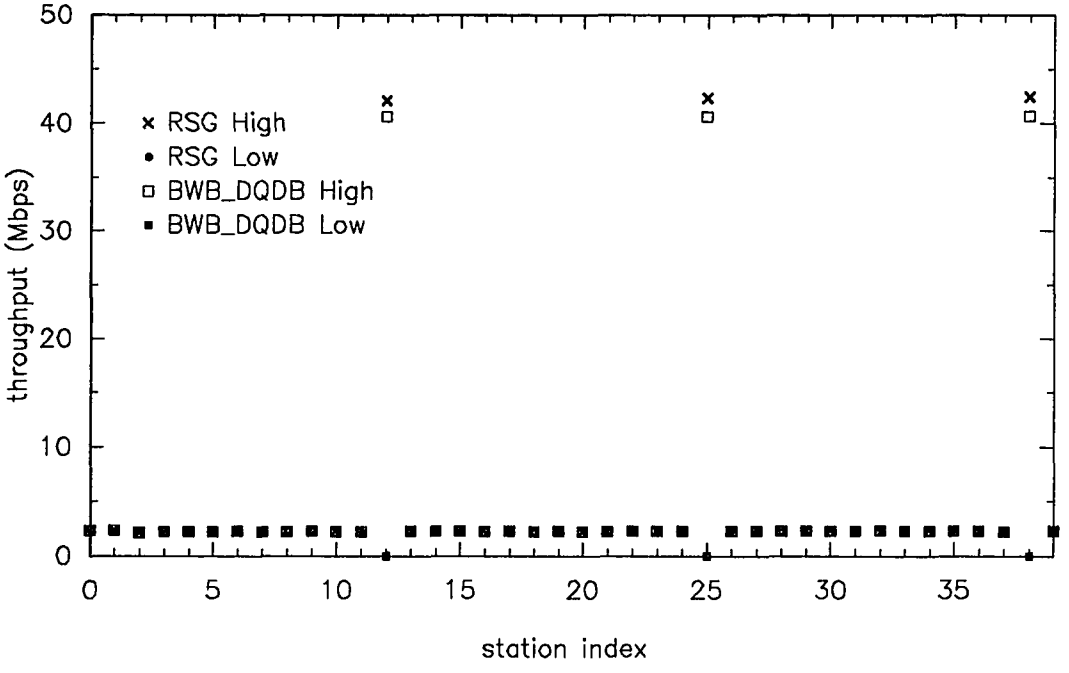

Fig.3.18:Two priority classes of traffic. Effect of station location on throughput. Asymmetric load. Three high priority queues are saturated. The total offered load of the remaining high and low priority users is 0.6 per bus. Traffic mix: 50% high, 50% low.

The rest of this section deals with the performance of RSG under three priority classes of users in the system. In the case of BWB_DQDB we have chosen $M_1=8$, $M_2=4$, and $M_3=2$. In Fig. 3.19 we show the effect of the station location on the delay of each class when a 0.95 load per bus is evenly distributed among all three priority classes. We see that RSG provides similar delays to the users of the same priority class, regadless

of their position on the bus. In contrast, BWB_DQDB severely penalizes the end users. Besides, its high priority users at the end stations encounter higher delays than the low priority users located in the middle of the bus. We can also see that with RSG the high and medium priority users encounter similar delays, which are much lower than the corresponding delays encountered by the high priority users in the case of BWB_DQDB. Finally, the average delay of the low priority users in the case of RSG is similar to the minimum average delay of the medium priority users in the case of BWB_DQDB (which is encountered by the users in the middle of the bus).

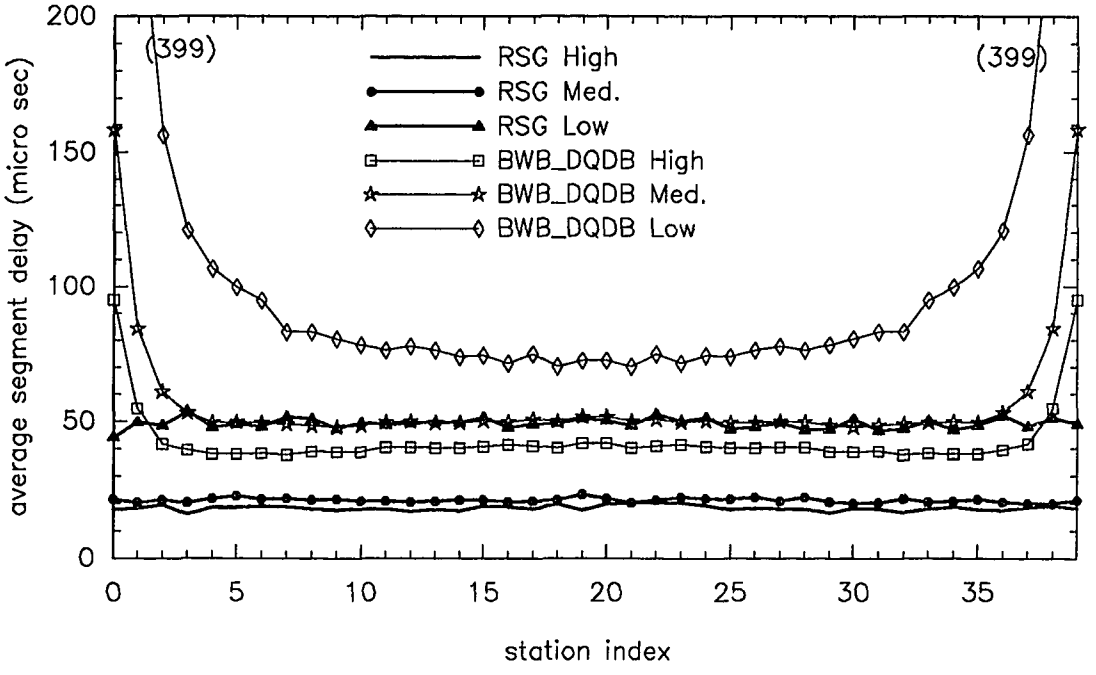

Fig.3.19:Three priority classes of traffic. Delay comparison of RSG and BWB_DQDB. An offered load of 0.95 per bus is evenly distributed among the three priority classes.

In Fig. 3.20 we assume that the low priority users at all stations are saturated and investigate their effect on both the high and medium priority users. Again, 0.6 load per bus is evenly distributed among the remaining high and medium priority users. We see that both schemes provide the same throughput to high and medium priority users, which

is independent of their location on the bus. In the case of low priority users, RSG provides similar throughputs to all of them whereas the end stations in the case of BWB_DQDB receive less bandwidth (because they can transmit only on one bus). Furthermore, the behavior of the two schemes, as shown in Fig. 3.20, is consistent with that of Fig. 3.15, where only two priority classes are considered. The reason low priority users in Fig. 3.20 seem to acquire more bandwidth than high priority users, whereas in Fig. 3.15 the inverse is true, is that the 0.6 data load per bus is now divided over both high and medium priority users.

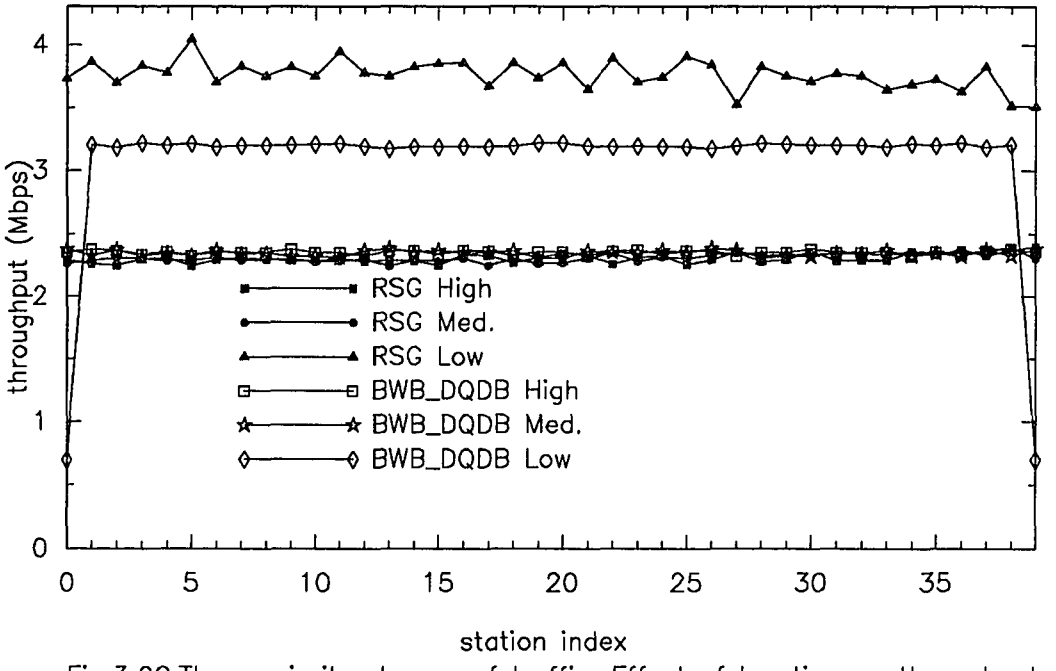

Fig.3.20:Three priority classes of traffic. Effect of location on throughput. Low priority queues are saturated. An offered load of 0.6 per bus is evenly distributed among the high and medium priority classes.

Finally, in Fig. 3.21 we show the bandwidth allocation in the case where the low priority users at stations "12", "25", and "38" are saturated. The offered load by the non-saturated stations is 0.6 per bus , evenly distributed among the high, medium, and low priority classes. We see that for both schemes most of the bandwidth is taken by the

saturated low priority users. The remaining users receive the same bandwidth regardless of their priority or position on the bus. A similar behavior is observed, and for this reason is not shown, when the medium or the high priority users (instead of the low priority users) are saturated at the above three stations. The only difference is the value of throughput of the various priority users at the overloaded stations "12", "25", and "38". If the medium priority users are overloaded, then the low priority users will not receive any bandwidth. If the high priority users are overloaded, then both the medium and low priority users will not receive any bandwidth.

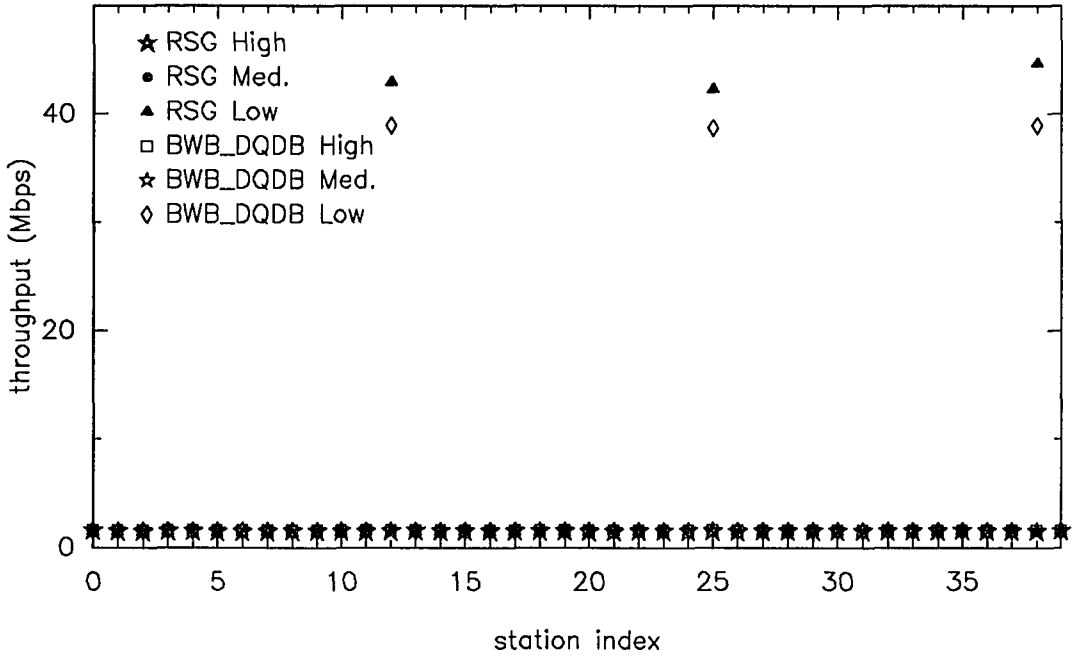

Fig.3.21:Three priority classes of traffic. Effect of location on throughput. Asymmetric load. Three low priority queues are saturated. The total offered load of the non-saturated users is 0.6 per bus. Traffic mix: 1/3 high, 1/3 medium, 1/3 low.

3.5 Conclusions

In this chapter we have introduced the Rotating Slot Generator (RSG) scheme for dual bus architectures, a Medium Access Control protocol appropriate for networks with high bandwidth-latency product. This scheme uses the looped bus architecture of the

Distributed Queue Dual Bus (DQDB) Metropolitan Area Network (MAN) in which slot generator capabilities for both busses are incorporated within every station. With RSG the responsibility for generating slots rotates around the loop, as stations take turns being slot generators. Thus, there are no favorable locations on the busses since station positions relative to the slot generator change periodically.

We have considered two variations of the RSG MAC mechanism, the Standby and Basic RSG, and two switching mechanisms, the IS_SG and SS_SG. We have investigated the performance of RSG under various types of loading, as well as, in the presence of a single and multiple priority classes of traffic. We have shown that Standby RSG provides lower average delay, but higher variance, than Basic RSG. Furthermore, the SS_SG mechanism, can provide higher throughputs and lower delays than the IS_SG mechanism, at the cost of simplicity. We have also compared the RSG performance with Basic DQDB, Standby DQDB and BWB DQDB. We have found that RSG is fair not only in terms of throughput under overload conditions, but also in terms of average segment delay in underload conditions, even under very high system utilizations.

In the presence of multiple priority classes of traffic, RSG continues to be fair in the sense that users of the same priority class receive the same bandwidth and encounter similar delays regardless of their position on the bus. Moreover, the average segment delays of the higher priority users are significantly smaller than those of the lower priority users. On the other hand, our comparative investigation of the other variations of DQDB has shown that the Basic and Standby DQDB can show dramatic unfairness in terms of both throughput and delay. BWB_DQDB is fair in terms of throughput under overload conditions, however, it induces significantly higher delays in underload conditions without reducing the strong effect of the station location on its delay. Furthermore, depending on the relative positions of the stations on the busses, lower priority users may encounter smaller average segment delays than higher priority users.

Finally, it is important to mention that RSG is fair in the "long term", that is, if a full rotation of the SG is considered. We have also investigated the transient behavior of the RSG scheme, during the rotation of the SG. In this case, RSG behaves similarly to the DQDB scheme. For small values of N_{switch}, the time it takes for the SG to complete a full rotation is short enough to make RSG efficient for most of the real time applications. However, future networks will support services with very bursty loads and strict delay requirements. In this respect, RSG, as well as any other topological solution to the DQDB fairness problem, may not be an appropriate MAC scheme. For such applications, we require MAC mechanisms that can "react" fast to different load configurations and reach a fair state within a few slots. In the next chapter we introduce the NSW_BWB, a scheme that achieves this behavior, without wasting any channel slots.

CHAPTER 4

THE NO SLOT WASTING BANDWIDTH BALANCING MECHANISM

4.1 Introduction

In this chapter we introduce a new bandwidth balancing mechanism for DQDB, the No Slot Wasting BWB (NSW_BWB) mechanism. The advantage of the proposed mechanism is that it exhibits a similar behavior with the current BWB mechanism of DQDB without, at the same time, wasting any channel slots. This enables it to converge faster to the steady state where fair bandwidth allocation is achieved. We investigate the throughput and delay performance of the new mechanism and provide analytical models which describe its behavior both under overload and underload conditions. We also introduce a variation of NSW_BWB, the Immediate TAR Use NSW BWB (ITU_NSW) mechanism, which can improve significantly the overall performance of the system, especially under the presence of multiple priority classes of traffic. We compare NSW_BWB and ITU_NSW and examine their ability to provide priorities similar to the ones that have been proposed for BWB_DQDB. Furthermore, we compare their performance with that of BWB_DQDB. Some of the research efforts, presented in this chapter, have also been presented in [72,73,74].

The organization of the rest of the chapter is as follows. In sections 4.2 and 4.3 we introduce the NSW_BWB and ITU_NSW mechanisms respectively. In section 4.4. we formally demonstrate the fairness of the two schemes. In section 4.5 we investigate the performance of NSW_BWB and ITU_NSW under one traffic class. In section 4.6 we examine their ability to provide arbitrary bandwidth distribution. In section 4.7 we introduce a queueing analytic model that can describe the behavior of the new mechanisms and investigate its accuracy under various offered loads and network configurations. In section 4.8 we examine the ability of NSW_BWB and ITU_NSW to provide similar priority scheduling algorithms with the ones that have been proposed for BWB_DQDB. Finally, in section 4.9 we present the conclusions.

4.1 NSW_BWB Mechanism

BWB_DQDB [15] achieves fairness by allowing each station to receive a multiple M of the unused channel bandwidth. The stations create this idle bandwidth by artificially incrementing their RQ_CTR by 1 and letting an empty slot pass by every time they transmit M segments. This slot can be written by the first active downstream station with CD_CTR=0. This station has then the opportunity of sending an additional request upstream, if it has another segment queued for transmission, decreasing in this way even more the transmission rate of the upstream stations. The basic problem with BWB_DQDB is that the free slot, that upstream stations allow to pass, may be wasted. In fact, a fraction of the slots is indeed wasted; the smaller the value of M, the higher the bandwidth loss. For this reason a value of M equal to 8 or 9 is proposed which keeps the worst case bandwidth wastage low. This worst case appears when there is only one active station on the bus and is equal to 1/(1+M). For the above values of M, i.e. 8 and 9, the bandwidth loss is 11.1% and 10.0%, respectively. However, the analysis in [35] has shown that as the value of M decreases it takes less time for the system to reach the steady state. Therefore, there is trade-off between channel utilization and convergence speed.

The objective of the NSW_BWB mechanism is to enable stations to know whether a free slot will be written by a downstream station before they allow it to pass. If this is possible, no slots will be wasted and the throughput of the various stations will be higher. Furthermore, the stations can use a smaller value of M and decrease the required time to reach the steady state. NSW_BWB can inform the stations beforehand about the future use of a slot, by introducing the Transmit Additional Request (TAR) bit in the ACF of each slot. According to NSW_BWB, whenever a station (say "j") transmits its M_{th} segment instead of increasing RQ_CTR by one, it sets TAR=1 in the written slot. The first active downstream station (say "i"), with available segments for which *requests* have not been sent, will erase the TAR=1 bit and transmit an *extra request* upstream. This *request*

will be seen by station "j" which will increase its RQ_CTR by one. We see that an *extra request* will be sent upstream, only if a downstream station has an available segment. Thus, the idle slot that will not be written by upstream station "j", will be certainly written by downstream station "i". In this way NSW_BWB does not waste any slots.

We point out that although NSW_BWB has similarities with BWB_DQDB, it is a quite different mechanism; this will become more evident when we examine the case of lightly loaded stations and provide its detailed description. For instance, an obvious difference from BWB_DQDB is that the *extra request* that will be sent by downstream station "i" will not be seen only by station "j", but also by its upstream stations which will increase their request counters. We compensate the upstream to "j" stations by not allowing station "j" to send a *request* for the next waiting segment in its queue when it has set TAR=1 in the last slot it has written. This next segment will be transmitted when station "j" sees an idle slot that has not been reserved by the downstream stations, or when station "j" sees a TAR=1 bit on the forward channel and transmits an *extra request* on the reverse channel.

In the case of NSW_BWB a station does not send a *request* only for the first segment in its queue, as in DQDB. Therefore, it must know for which of the queued segments a *request* has been sent on the reverse bus. A pointer may be used to show to the last of the "requested service" segments. It is also evident that the stations that can erase TAR=1 bits (i.e. the stations with queued segments for which *requests* have not been sent) do not know beforehand which of the passing slots have TAR=1. However, these stations can set TAR=0 to every passing slot before they examine it. In this way, the stations will not have to delay every passing slot in order to read it and then take an action. If TAR was 1, before the station reset it to 0, the station would send an *extra request* on the reverse bus. In the sequel we show analytically the convergence of NSW_BWB in the case of two active and overloaded stations in the system.

4.2.1 Convergence Analysis under two Overloaded Stations

We consider that stations "1" and "2" are the only active and overloaded stations in the system; they try to transmit segments on any free slot that passes by. Their distance, measured in time the signal takes to travel from "1" to "2", is D_{12} slots. Fig.4.1 depicts

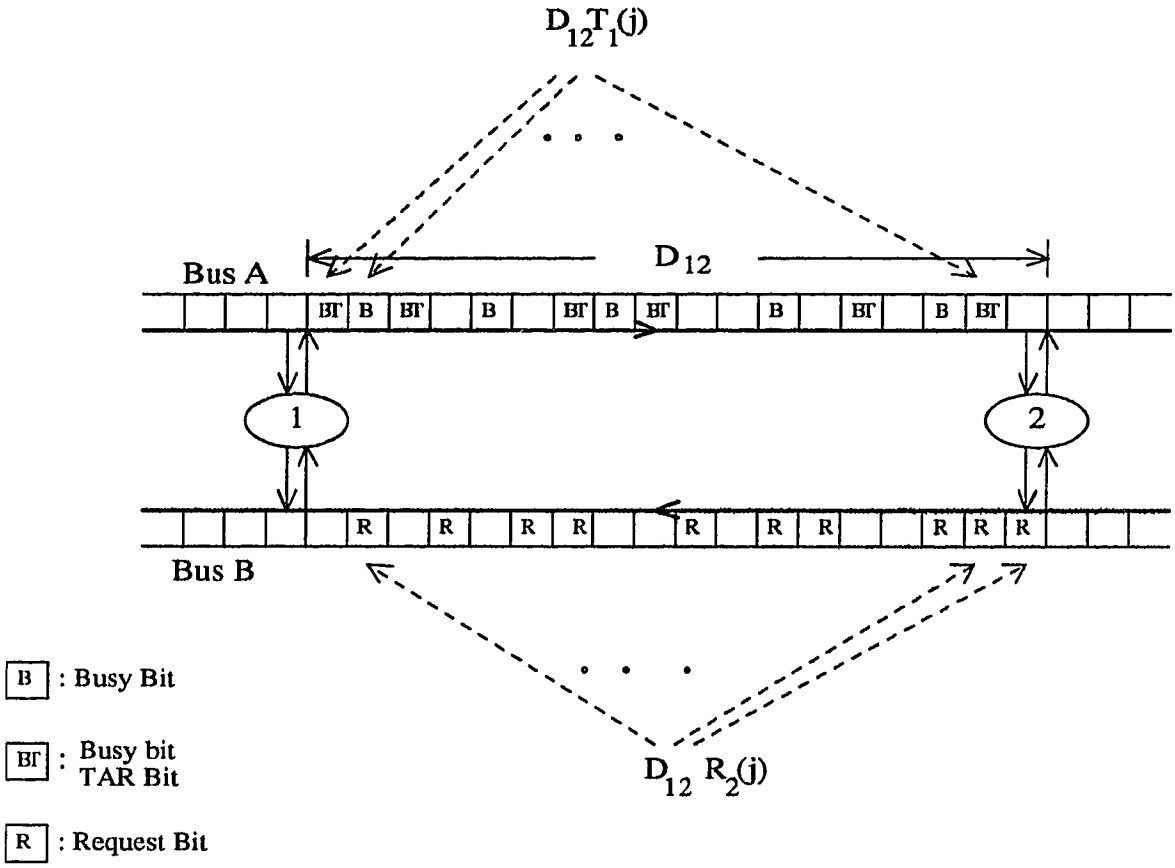

Fig.4.1: System snapshot at time $(j+1)D_{12}$, M=2.

this configuration. Furthermore, we consider the DQDB MAC mechanism that requires one only RQ_CTR inside each station. It is evident that the RQ_CTRs of both stations will eventually go to 0 and will never, thereafter, become greater than one. Let $T_1(j)$ and $T_2(j)$ be the throughputs of stations "1" and "2", respectively, during the time interval $[j\,D_{12}, (j+1)D_{12}]$, where j=0,1,2,..; we define throughput to be the fraction of channel bandwidth acquired by each station. Let $R_2(j)$ be the percentage of slots during $[j\,D_{12}, (j+1)D_{12}]$ that carry *requests* from station "2". If $T_1(j-2)$ is the throughput of station "1" during the interval $[(j-2)D_{12}, (j-1)D_{12}]$, then station "2" will see

$T_1(j-2)D_{12}/M$ TAR=1 bits and transmit $(1-T_1(j-2))D_{12}$ segments during the interval $[(j-1)D_{12}, jD_{12}]$. Since station "2" will send a *request* for each TAR=1 bit it sees, and **will not send** a *request* bit for every M_{th} segment it transmits, the number of *requests* $R_2(j-1)D_{12}$ it will transmit during the same $[(j-1)D_{12}, jD_{12}]$ interval will be:

$$R_2(j-1)D_{12} = \frac{T_1(j-2)D_{12}}{M} + (1-T_1(j-2))D_{12}\left[1-\frac{1}{M}\right]$$
$$= \left[1-\frac{1}{M}\right]D_{12} - \left[1-\frac{2}{M}\right]T_1(j-2)D_{12} \qquad (4.2.1)$$

and the throughput $T_1(j)$ of station "1" during the interval $[jD_{12}, (j+1)D_{12}]$ will be:

$$T_1(j) = 1-R_2(j-1) = \frac{1}{M} + \left[1-\frac{2}{M}\right]T_1(j-2) \qquad (4.2.2)$$

It is also evident that:

$$T_2(j) = 1-T_1(j-1) \qquad (4.2.3)$$

We can now derive the z-transforms of the above functions and find the expressions for $T_1(j)$ and $T_2(j)$. In fact we have done so and we have found that both expressions consist of the constant 1/2 and terms of the form $(1-2/M)^{j/2}$; which show that the throughputs converge to 1/2, regardless of the value of M. However, our mechanism does not waste any bandwidth and for this reason we can select whatever value of M we desire. If we choose M=2, then equation (2) shows that $T_1(j)=1/2$. That is, regardless of initial conditions, the mechanism converges to the steady state where each station receives half of the channel bandwidth within a $2D_{12}$ interval. In comparison the current BWB_DQDB mechanism with value of $M=9$, which in the case of two overloaded stations wastes 5.26% of the channel bandwidth, requires $22D_{12}$ slots to bring the throughputs to 90% of the way towards the steady state [35]. On the other hand the maximum convergence speed of BWB_DQDB is achieved when the value of $M=1$. In this case $1/(1+2M)=1/3$ of the bandwidth is wasted when there are two active overloaded stations. However, even in this case, over an interval of $2D_{12}$ slots, the stations'

throughputs have moved only 75% of the way towards the steady state. Furthermore, if only one station remains active the bandwidth wastage will rise to 50%.

The question that naturally arises is what happens when a station sets TAR=1 to every segment that transmits. Again, each station acquires 50% of the channel bandwidth. Nevertheless, the sequence of its segment transmissions is strongly affected by the initial conditions. For instance, if at t=0 all D_{12} slots between stations "1" and "2" are written by the active and overloaded station "1" and at this instant station "2" becomes active (i.e. at t=0 there are no *requests* in transit), then the two stations will alternately transmit $2D_{12}$ segments on the forward bus. By selecting M=2, the written slots by the two stations are more evenly distributed in time.

4.2.2 Many Active Stations

In this section we use simulation to investigate the behavior of NSW_BWB in the case of many active stations. We consider a high capacity network of 155.52 Mbps, a slot size of 53 bytes, and a signal propagation delay of 5 $\mu sec/Km$. Our simulation results show that in the case of overloaded stations, regardless of the number of active stations or initial conditions, NSW_BWB will not waste any bandwidth and provide all stations with the same throughput. We now look at the case where some of the active stations are overloaded and some are underloaded. We first consider three active stations. The inter-station distances are $D_{12}=38$ slots and $D_{23}=40$ slots. In Table 4.1 we show the throughputs of the three stations when two of them have a load of .8 segments/slot (each) and the third has a load of .25 segments/slot. We consider all three cases of loading, i.e. when the lightly loaded station is the first, the second, and the last station on the bus. In addition (for comparison), we have included the corresponding throughputs in the case of BWB_DQDB. Table 4.1 shows that in all three cases of loading, NSW_BWB does not waste bandwidth and provides the requested throughput to the lightly loaded station. Furthermore, it evenly distributes the remaining bandwidth among the heavily loaded stations when the lightly loaded station is the first or the last station on the bus. However,

when this station is in the middle, the first of the heavily loaded stations acquires more bandwidth. In the sequel we explain the reason for this behavior.

Table 4.1: Throughput performance under asymmetric load.

Offered Load			Throughput BWB_DQDB M=8			Throughput NSW_BWB M=2		
stat. 1	stat. 2	stat. 3	stat. 1	stat. 2	stat. 3	stat. 1	stat. 2	stat. 3
0.25	0.80	0.80	0.250	0.353	0.353	0.250	0.375	0.375
0.80	0.25	0.80	0.353	0.250	0.353	0.400	0.250	0.350
0.80	0.80	0.25	0.353	0.353	0.250	0.375	0.375	0.250

In the case of NSW_BWB each station transmits two types of *requests* to the upstream stations: *regular* and *extra requests*. *Regular* is the *request* that a station sends when a segment becomes first in its transmission queue. *Extra* is the *request* that a station sends when it sees a TAR=1 bit. If all stations are overloaded, then each one of them will erase all TAR=1 bits that will see on the forward channel. Furthermore, the number of TAR=1 bits that each station will erase will be equal to the number of TAR=1 bits that will insert. This is evident since if a station inserts more TAR=1 bits than it erases, its cut back on *regular requests* due to the transmission of TAR=1 bits (for each TAR=1 bit sent downstream a *regular request* will not be sent upstream) will be higher than the *extra requests* it will insert. Therefore, its throughput and number of TAR=1 bits that will send will decrease. If the station erases more TAR=1 bits than it inserts, the number of *extra requests* that will send will be larger than the number of *regular requests* that will cut back and its throughput and number of TAR=1 bits that will send will increase. These two actions will bring the number of TAR=1 bits that each station of the system transmits to the same level. Since the number of slots that each station writes is the multiple M of

the number of TAR=1 bits it transmits and all stations send the same number of TAR=1 bits, the throughputs of the various stations will be the same.

Consider now the case where there is a lightly loaded station between two overloaded stations, the value of M is four, and four segments arrive at the empty queue of the lightly loaded station. This station will initially send a *regular request* for the first segment. Let us now assume that three TAR=1 bits appear on the forward channel. The lightly loaded station will erase all three of them and send three *extra requests* upstream. However, when the reserved slots arrive and the lightly loaded station transmits its segments, it will not be able to send any *regular request* upstream since there is no other waiting segment in its queue. In contrast, if the station is overloaded, there will always be available segments for which *requests* have not been sent. Furthermore, although it has erased four TAR=1 bits, it will send only one downstream, because, according to the protocol, a station must send one TAR=1 bit every M=4 transmitted segments. Consequently, the last overloaded station will see a lower number of TAR=1 bits than the first overloaded station has sent, and its throughput will be smaller. It is evident that in all other topologies, i.e. when the lightly loaded station is the first or the last on the bus, the two overloaded stations will see, erase, and send the same number of TAR=1 bits and for this reason their throughputs will be the same.

In order to deal with the uneven bandwidth distribution, in the case of lightly loaded stations, two additional counters, the UNRG_CTR and RG_CTR, have been introduced in each station. These counters ensure that the rate at which a station uses TAR=1 bits does not exceed the rate at which transmits TAR=1 bits, i.e. one every M segments.

UNRG_CTR keeps track of the number of segments, in the station's queue, which may allow the station to use a TAR=1 bit. We call these segments **unregistered.** RG_CTR holds the number of the remaining segments in the queue. We call these segments **registered.** It is evident that the sum of RG_CTR and UNRG_CTR provides the total number of segments in the station's queue. The operation of the two counters is as

follows. Every time a new segment arrives at the station, UNRG_CTR increases by 1; this is an unregistered segment. When a segment becomes the first in the queue a *request* is sent upstream, UNRG_CTR decreases by 1, and RG_CTR increases by 1; the segment has become registered. Consider now that a TAR=1 bit is seen on the forward channel. We want to ensure that the station will erase this TAR=1 bit only if it can also send a TAR=1 bit on the forward channel. Since each station transmits one TAR=1 bit every M transmitted segments, the station will check its UNRG_CTR. If UNRG_CTR $\geq$ M, the station will erase the TAR=1 bit, send an *extra request* upstream, decrease UNRG_CTR by M, and increase RG_CTR by M. UNRG_CTR is decreased by M and the first M unregistered segments become registered because their presence cannot be used again by the station to erase another TAR=1 bit. Otherwise, the station would erase more TAR=1 bits than it can send. This is also why the station should look only at the content of its UNRG_CTR to decide whether it should erase a TAR=1 bit. It is evident from our previous discussion that the registered segments precede the unregistered segments in the station's queue.

In addition to UNRG_CTR and RG_CTR, each station must have a Bandwidth Balancing Counter (BWB_CTR) to indicate when a TAR=1 bit must be sent onto the channel. BWB_CTR increases by one for every segment transmitted by the station. If it becomes equal to M, the station will set TAR=1 in the written slot and reset BWB_CTR to 0. At the same time, the station will set a flag (TAR_flag) to 1 to indicate that no *regular request* can be sent for the next segment. We will use the term flag_segment for this segment. A *regular request* should not be sent for the flag_segment in order to compensate the upstream stations for the *extra request* they have seen; that was sent by the downstream station when it erased the TAR=1 bit on the forward channel. In the sequel we show how the transmission of the flag_segment is guaranteed.

In the case of light traffic conditions the most probable way the flag_segment can be transmitted is through an empty slot that has not been reserved by any of the down-

stream stations, i.e the RQ_CTR of the station is 0. We now examine the case of heavy traffic conditions where stations can transmit only on slots they have previously reserved. We first consider the case of registered segments. We can classify the registered segments into type "A" and "B". Type "A" is a registered segment that reaches the head of the station's queue as an unregistered segment, sends a *regular request* upstream, and becomes registered. Type "B" is a registered segment that belongs to a group of M segments that become registered when the station erases a TAR=1 bit on the forward channel and sends an *extra request* on the reverse channel. We can now show that all registered segments are guaranteed transmission.

The transmission of a type "A" segment is guaranteed by its own regular *request.* This request will reserve a slot that can be written only by this segment since it is the first in the queue. Therefore, it is also guaranteed that a type "A" segment will not use a slot reserved by a type "B" segment. Since the type "B" segments use all the slots they reserve, their transmission is also guaranteed. The reason is that for each group of M type "B" segments, an *extra request* is sent beforehand and M-1 *regular requests* during the transmission of the M segments of this group; only M-1 *regular requests* are sent because when BWB_CTR becomes M, no *request* will be sent for the next segment. Thus, the total number of *requests* each group of M type "B" segments sends is also M and their transmission is guaranteed.† It is now evident that since the transmission of all registered segments is guaranteed, so is the transmission of the registered flag_segments.

We now consider the case of the unregistered flag_segments. That is, an unregistered segment becomes first in the queue when the value of TAR_flag is 1 and hence no *request* can be sent on the reverse channel. One way this case may appear is when

---

† Since the *extra request* is sent as soon as a group of M segments becomes registered, it is common for a slot that has been reserved by an *extra request* sent by a type "B" group, say $GRPB_2$, to be written by a segment that belongs to a previous type "B" group, say $GRPB_1$. However, since each group of M segments will send M *requests* (one extra and M-1 regular), $GRPB_2$ will not have any problem; since its *extra request,* that was used by $GRPB_1$, will be returned to $GRPB_2$ as a *regular request.*

individual segments arrive at a station and are transmitted one by one by the station. If the station is initially idle and all of its counters, as well as its TAR_flag, are 0, then when the M_{th} segment is transmitted BWB_CTR will become M, a TAR=1 bit will be sent downstream, TAR_flag will become 1, and BWB_CTR will be reset to 0. Consequently, the $(M+1)_{th}$ segment that will arrive at this station will be a flag_segment. Since this segment cannot sent a *request* it will remain unregistered. One way this segment can be transmitted is to wait for the arrival of additional segments. When UNRG_CTR becomes equal to or greater than M it is certain that the flag_segment will be transmitted. The reason is that one upstream station will eventually send a TAR=1 bit that will be erased by our station. An *extra request* will then be sent upstream and the first M segments will be converted into registered segments‡ whose transmission, as we have seen, is guaranteed. Nevertheless, if the particular station is lightly loaded, waiting for UNRG_CTR to become M may delay significantly the transmission of the segments. Furthermore, if the unregistered flag_segment is the last segment of the station, there is a possibility of deadlock. For this reason we allow a station that has a flag_segment and whose RG_CTR is 0, to erase a TAR=1 bit even if its UNRG_CTR is less than M. In this case the station will set the TAR=1 bit to 0, send an *extra request* upstream, increase RG_CTR by UNRG_CTR, and reset both UNRG_CTR and TAR_flag to 0. We point out that the erasure of the TAR=1 bit in this case will not enable the station to erase more TAR=1 bits than it will insert, because the presence of the flag_segment indicates that a TAR=1 bit has already been sent by the station. We now describe the complete NSW_BWB operation.

---

‡ If in the operation of the system RG_CTRs are only used by the stations and our station is the first among the active ones it is again guaranteed that the flag_segment will be transmitted. The reason is that eventually one downstream station will send a TAR=1 bit downstream which means it will not send a *request* upstream. Therefore, our station will see an unreserved slot and will be able to transmit its flag_segment.

4.2.3 NSW_BWB Operation

The operation of UNRG_CTR, RG_CTR, BWB_CTR and TAR_flag inside a station is based on the following four events: segment arrival, segment becomes first in queue, segment transmission, and TAR=1 is seen on forward channel. In the sequel we describe the reaction of the station to each one of these events.

a) Segment arrival: UNRG_CTR increases by one. If a long message arrives at the station UNRG_CTR will increase by the number of segments in the message.

b) Segment becomes first in queue: If TAR_flag=0 and RG_CTR>0 (i.e. a registered non flag_segment), the station will simply send a *regular request* on the reverse bus. If TAR_flag=0 and RG_CTR=0 (i.e. a unregistered non flag_segment), the station will send a *regular request,* increase RG_CTR by one, and decrease UNRG_CTR by one. If TAR_flag=1 (i.e. a flag_segment), the station will not take any action.

c) Segment transmission: If TAR_flag=1 and RG_CTR=0 (i.e. an unregistered flag_segment which writes an idle unreserved slot), UNRG_CTR will decrease by one. In any other case (i.e. a registered segment) RG_CTR will decrease by one. Notice that TAR_flag and RG_CTR cannot be both 0 because had the TAR_flag been 0 (i.e. the next segment a non flag_segment), the arrived segment would have sent a *regular request* upstream and become a registered segment (i.e. RG_CTR>0). In all the above cases BWB_CTR will increase by one and TAR_flag will be set to 0. If by increasing BWB_CTR its value becomes equal to M, the TAR bit will be set to 1 in the written slot, BWB_CTR will be reset to 0, and TAR_flag will be set to 1; to indicate that the next segment is a flag_segment.

d) TAR=1 is seen on forward channel: If UNRG_CTR $\geq$ M, then the TAR=1 bit will be set to 0, an *extra request* will be sent upstream, UNRG_CTR will decrease by M, RG_CTR will increase by M, and TAR_flag will be set to 0 (if it is one). The TAR=1 bit will be set to 0 and an *extra request* will be sent upstream also in the

case where 0< UNRG_CTR < M, RG_CTR=0, and TAR_flag=1. In the last case RG_CTR will be set to UNRG_CTR, and UNRG_CTR and TAR_flag will be set to 0. In all other cases the station will allow the TAR=1 bit to pass by.

We finally mention that the operation of RQ_CTR and CD_CTR is as in DQDB. We have used the complete NSW_BWB mechanism in the case of Table 4.1 and we have found that the throughputs of the overloaded stations are similar regardless of the location of the lightly loaded station on the bus. In Table 4.2 we show the offered loads and corresponding throughputs in the case of five active stations and under various load configurations. The distance between neighbor stations is 16 slots. It is evident that NSW_BWB can guarantee the requested bandwidth to lightly loaded stations and evenly distribute the remaining bandwidth among overloaded stations. We point out here that in the rest of the chapter NSW_BWB refers to the complete mechanism that requires one UNRG_CTR, RG_CTR, BWB_CTR and TAR_flag inside each station.

According to the NSW_BWB operation, a station can use a TAR=1 bit and send an *extra request* upstream, only if it has M unregistered segments in its queue; or in the case where the first segment in its queue was a flag_segment. This behavior discriminates against lightly loaded stations whose queue sizes are (initially) small and therefore they cannot benefit from the TAR=1 bits they see on the forward channel. The queue sizes of these stations must be built up first before they can start gaining from the NSW_BWB operation. In essence, NSW_BWB penalizes the lightly loaded stations initially in order to favor them later. This behavior is a direct consequence of the bandwidth balancing requirement which imposes that each station must not erase more TAR=1 bits that it can transmit. NSW_BWB meets this requirement by allowing a station to erase a TAR=1 bit and send an *extra request* only when it is certain that will send it back, i.e. it has at least M unregistered segments in its queue. This behavior is intensified if large values of M are considered. Although it is desirable to use small values for M, there are cases where the use of higher values for M is unavoidable. For instance, we will see in section 4.6 that

if we assign different values of M to the various stations, the bandwidth they will receive will be proportional to these values. Thus, if it is desirable to allocate more bandwidth to a certain station, we have to assign to it a higher value of M. Then, the current NSW_BWB mechanism will reduce the ability of this station to take advantage of the TAR=1 bits it sees on the channel. For this reason, in the next section we present a variation of NSW_BWB which preserves the nice, fair properties of the current NSW_BWB mechanism, and at the same time enables the lightly loaded stations to take advantage of every TAR=1 bit seen on the channel.

4.3 ITU_NSW Mechanism

The Immediate TAR Use No Slot Wasting (ITU_NSW) bandwidth balancing mechanism allows a station to send an *extra request,* whenever it sees a TAR=1 bit, provided that there is at least one segment for which a request has not been sent. This mechanism requires four counters, the RG_CTR, UNRG_CTR, BWB_CTR as well as the Debit TAR Counter (DBTAR_CTR) which counts the number of TAR=1 bits the station must send onto the channel, i.e. owes to the downstream stations. Moreover, it requires one register, the Number of available TAR bits register (*NTAR_R*) which counts the number of TAR=1 bits that the station will send on the channel if it transmits all its queued segments. The meaning and operation of RG_CTR and UNRG_CTR is slightly different in the case of ITU_NSW. RG_CTR counts the number of segments at the station's transmission queue for which a *request, regular or extra,* has already been sent upstream. UNRG_CTR counts the rest of the segments. Certainly, as in the case of NSW_BWB, the sum of RG_CTR and UNRG_CTR is always equal to the number of queued segments. The operation of BWB_CTR is identical to the one of NSW_BWB. The utility of the DBTAR_CTR and the *NTAR_R* will become evident in the sequel.

As we have already mentioned, in the case of ITU_NSW a station does not have to wait to accumulate M unregistered segments in its queue before it can erase a TAR=1 bit

and send an *extra request* upstream. It can use it immediately as long as UNRG_CTR > 0, i.e. it has at least one segment for which a request has not been sent. However, contrary to NSW_BWB, not all TAR=1 bits that enable a station to send an *extra request* are reset to 0. The values of *NTAR_R* and DBTAR_CTR determine when a TAR=1 bit should be erased. *NTAR_R* provides the number of TAR_segments in the station's queue. TAR_segment is a segment whose transmission will make BWB_CTR equal to M and therefore result to the transmission of a TAR=1 bit downstream. For instance, if M=2, RG_CTR=1, UNRG_CTR=5, and BWB_CTR=0, then $NTAR\_R = 3$ since during the transmission of the six segments BWB_CTR will become equal to M (=2) three times and hence three TAR=1 bits will be sent downstream. In general,

$$NTAR\_R = \lfloor (BWB\_CTR + RG\_CTR + UNRG\_CTR) / M \rfloor$$

Since, the value of *NTAR_R* provides the number of TAR=1 bits that a station can transmit downstream it also provides the number of TAR=1 bits that this station can erase. In order for a station to know whether it should erase a passing TAR=1 bit, it must know how many TAR=1 bits has already erased. This is exactly the information provided by the Debit TAR Counter (DBTAR_CTR) which indicates the number of TAR bits that a station owes to the downstream stations (because it has erased them). Every time a station, with UNRG_CTR > 0, sees a TAR=1 bit on the forward channel it will send an *extra request* upstream, increase RG_CTR by one, and decrease UNRG_CTR by one. In addition, if DBTAR_CTR < *NTAR_R*, the station will erase the TAR=1 bit and increase DBTAR_CTR by 1. Finally, whenever a station sends a TAR=1 bit and its DBTAR_CTR is greater than 0 the station will decrease the value of DBTAR_CTR by one.

The operation of *NTAR_R* and DBTAR_CTR guarantees that the station does not insert less TAR=1 bits than it erases. However, as it will be shown in section 4.4 where the formal proof of the fairness of ITU_NSW is presented, the correct operation requires for the mechanism to guarantee that if a station has used at least as many TAR=1 bits as it is to insert, then it must erase the same number of TAR=1 bits. Otherwise, an over-

loaded downstream station maybe favored over an upstream overloaded station. In order to provide a better insight into the operation of ITU_NSW, we present a scenario in which a station erases fewer TAR=1 bits than it inserts, although it uses as many as it inserts. Let us consider that M=4, RG_CTR=1, UNRG_CTR=1 and BWB_CTR=0. In this case the station cannot erase any TAR=1 bit, since there is no TAR_segment present in its queue, i.e. *NTAR_R*=0. Assume now that the station observes a TAR=1 on the forward channel. Then, it sends an *extra request,* increases RG_CTR by one and decreases UNRG_CTR by one. Later on, two additional segments arrive at the station's queue. The values of both RG_CTR and UNRG_CTR become now equal to 2. The station can now transmit all its four queued segments, without the need of observing another TAR=1 bit; during the transmission of the first two segments, it will insert two more *regular requests,* and in this way reserve all the slots required to transmit the four segments. The last segment that will be transmitted is a TAR_segment and its TAR bit will be set to 1. We see in this example that the station has used one TAR=1 bit, has send one TAR=1 bit but has not erased any TAR=1 bit. In general, this scenario occurs when a station sends an *extra request,* which is used for the transmission of a TAR_segment, before this TAR_segment has arrived at the station. However, notice that whenever this happens, then at the instant the TAR_segment is transmitted RG_CTR is greater than 0 (an *extra request* has been inserted) and DBTAR_CTR is 0 (the TAR_segment was not present when the *extra request* was inserted). Thus, the correct operation requires that in the case where *RG_CTR*>0 and *DBTAR_CTR*=0, the TAR bit should not be set to 1 on a transmitted segment, although the value of BWB_CTR has become equal to M. We now present the complete ITU_NSW mechanism by describing the station reaction to the various events:

a) Segment arrival: The station increases its UNRG_CTR by one; if a long message arrives, the station will increase the value of UNRG_CTR by the number of segments in the message. Then, the station will update the value of *NTAR_R*.

b) Segment becomes first in queue: If BWB_CTR is less than M-1 (i.e. the segment is not a TAR_segment) and UNRG_CTR > 0, (i.e. there is at least one segment for which a request has not been sent), the station will send a *request* upstream, increase RG_CTR by one, and decrease UNRG_CTR by one.

c) Segment transmission: If RG_CTR is greater than 0, it will decrease by one. Otherwise, UNRG_CTR will decrease by one. In both cases BWB_CTR will increase by one. If by increasing BWB_CTR its value becomes equal to M, BWB_CTR will be reset to 0 and DBTAR_CTR will decrease by one (if it is greater than 0). If DBTAR_CTR was greater than 0 or both DBTAR_CTR and RG_CTR were equal to 0 the station will set the TAR bit to 1. Otherwise, i.e. RG_CTR > 0, DBTAR_CTR=0, the station will not set the TAR bit to 1. Finally the value of *NTAR_R* will be updated.

d) TAR=1 is seen on the forward channel: If UNRG_CTR is greater than 0, the station will send a request upstream, increase RG_CTR by one, and decrease UNRG_CTR by one. Furthermore, if *NTAR_R* is greater than DBTAR_CTR the station will reset the TAR bit to 0 and increase DBTAR_CTR by one.

Our simulation of the ITU_NSW mechanism has shown that it can distribute the channel bandwidth in the same way as NSW_BWB, that is, it can provide the requested bandwidth to the lightly loaded stations and evenly distribute the remaining bandwidth among the overloaded ones. Therefore, the performance results, shown in Table 4.2, also apply to ITU_NSW. In the next section we provide a formal proof of the scheme's fairness properties, regardless of the loading conditions.

Table 4.2: Throughput performance under asymmetric load.

Offered Load					Throughput NSW_BWB, M=2				
stat. 1	stat. 2	stat. 3	stat.4	stat.5	stat. 1	stat. 2	stat. 3	stat.4	stat.5
0.20	0.75	0.20	0.75	0.20	0.20	0.20	0.20	0.20	0.20
0.75	0.75	0.20	0.20	0.75	0.20	0.20	0.20	0.20	0.20
0.75	0.75	0.20	0.20	0.20	0.20	0.20	0.20	0.20	0.20
0.75	0.30	0.15	0.10	0.75	0.25	0.25	0.15	0.10	0.25

4.4 Fairness of the ITU_NSW Scheme.

In this section we formally prove that the ITU_NSW mechanism is fair. It is fair in the sense that under any arbitrary offered load configuration it will provide the requested bandwidth to the lightly loaded stations and, at the same time, evenly distribute the remaining bandwidth among the overloaded ones. First, we define the following terms. We say that an empty slot passing in front of a station is an *unreserved* slot, if and only if the station's RQ_CTR and CD_CTR are both zero, i.e. if the station has the perception that no other station intends to use the empty slot. We say that a TAR=1 bit is *effectively erased* if and only if it is reset to zero or results into transmitting a TAR_segment without setting its TAR bit; when the TAR_segment is transmitted DBTAR_CTR is 0 and RG_CTR is greater than 0. The last case may occur when a TAR_segment is transmitted in an empty slot that has been reserved by an extra request before this TAR_segment had joined the station's queue. Then, we effectively erase the TAR=1 segment that caused this transmission by not inserting a TAR=1 bit on the transmitted TAR_segment. Also, we call a station *saturated,* if and only if, it effectively erases all passing TAR=1 bits and is able to transmit on every unreserved slot. Otherwise, we call the station *non-saturated.*

It is evident that if the average segment arrival rate at a station is lower than its average segment transmission rate, then the rate at which the station observes TAR=1 bits on the forward channel is lower than the rate the value of *NTAR_R* increases. Thus, the station will eventually become saturated (use all unreserved slots and erase all passing TAR=1 bits). Finally, we say that the network is *saturated* if and only if there exists at least one saturated station.

Lemma 1: Given that the most upstream station, i.e. station "1", is saturated, then the fraction E_i of slots carrying TAR=1 bits that station "i" effectively erases is given by E_i=min(x_i / M , t_i), where t_i is the fraction of slots carrying a TAR=1 bit that station "i" observes and x_i is its throughput.

Proof: Station "1" does not erase any TAR=1 bits. So E_1=0 and the lemma holds; since t_1=0. Let us now consider a very long time interval T and a downstream station "i" (either saturated or not). There is at least one saturated station, station "1", located upstream to station "i". In order for station "i" to be able to transmit x_iT segments over a period T it must send at least x_iT requests on the reverse channel to reserve the x_iT slots for its transmissions. Is is evident from the operation described in the previous section that station "i" cannot insert more than x_iT requests, either regular or extra; a station may insert a request only if its UNRG_CTR is greater than 0. Consequently, the total number of requests that station "i" will insert will be equal to x_iT. Let RR_i and ER_i denote, respectively, the number of regular and extra requests that station "i" inserts during the period T. Then we have that RR_i+ER_i should be equal to x_iT. It is also obvious from the operation of the ITU_NSW that the number of regular requests RR_i station "i" inserts is less than or equal to $(M-1)x_iT/M$. From this inequality we derive that: $ER_i \geq \frac{x_iT}{M}$. Therefore, station "i" sends at least x_iT/M extra requests and thus it has to observe at least x_iT/M TAR=1 bits (i.e. $t_i \geq x_i/M$), in order to transmit the x_iT segments. In the sequel we will show that station "i" will effectively erase exactly x_iT/M TAR=1 bits.

Let us consider the time instance station "i" transmits a TAR_segment. We may distinguish two cases according to the value of DBTAR_CTR:

(i) DBTAR_CTR is greater than 0.

(ii) DBTAR_CTR is equal to 0.

In both cases, according to the operation of ITU_NSW, RG_CTR should be greater than 0, since station "i" can transmit segments only in slots that have already been reserved; there is at least one saturated station located upstream to station"i". Case (i) implies that the station has already erased a TAR=1 bit. In the second case the station has made a request for the current TAR_segment. That request was made through a TAR=1 bit which was not reset to 0; DBTAR = 0. This scenario could occur if and only if at the time the TAR=1 bit was observed, there was at least one unregistered segment and the TAR_segment was not present at the station's transmission queue. This TAR_segment is now transmitted without setting the TAR bit to 1. This is equivalent to setting TAR=1 and erasing the TAR bit that allowed the station to insert the extra request. Thus, in all cases for every TAR_segment station "i" transmits it effectively erases a TAR=1 bit. The station will transmit $x_i T/M$ TAR_segments and consequenlty it will erase the same number of TAR=1 bits. □

Corollary 1.1: Given that station "1" is saturated, all downstream stations observe the same fraction x_1/M of the slots carrying TAR=1 bit on the forward channel.

Proof: Station "1" will set $x_1 T/M$ TAR=1 bits. Since each station erases as many TAR=1 bits as it inserts, all stations will observe the TAR=1 bits which were inserted by the most upstream station. □

Corollary 1.2: Given that the most upstream saturated station is station "j", a station "i", located downstream to station "j" (i > j), sees x_j/M TAR=1 bits.

Proof: Station "j" is saturated and so it will effectively erase all the TAR=1 bits that

observes. Consequently, all stations located downstream to "j" see x_j/M TAR=1 bits. □

Theorem (ITU_NSW fairness): Regardless of the offered load configuration, there always exists a certain value x such that all stations with offered load greater than or equal to x have the same throughput x (saturated stations) and the rest of the stations have throughput equal to their offered load (non-saturated stations).

Proof: In the case of non-saturated stations all of them receive the required bandwidth (otherwise al least one of them would have been saturated) and the theorem holds with value of x the throughput of the station with the highest offered load. Let us now consider the case where the first overloaded station is station "1". Corollary 1.1 shows that all stations (either saturated or non-saturated) will see the same fraction of slots carrying TAR=1 bits (=x_1/M). We will first show that all stations that have an offered load greater than x_1 will receive the same bandwidth x_1. Let us assume that this is not true and that there is a station "j" which has throughput greater than x_1. If $x_j > x_1$, then also $x_j/M > x_1/M$. Since station "1" is saturated, station "j" can write only on reserved slots, i.e. its throughput x_j must be equal to the fraction of slots on which it sends requests. However, the maximum fraction of slots on which station "j" can send requests is $x_1/M + x_j(M-1)/M < x_j$. Therefore, x_j cannot be greater than x_1. It is also not possible for a station "j" with offered load greater than x_1 to have throughput less than x_1. If this was true the station would have received its bandwidth through the x_j(M-1)/M regular requests and the x_j/M extra requests it inserts. However, it observes a fraction of $x_1/M > x_j/M$ of slots carrying a TAR=1 bit which implies that the station does not use all the TAR=1 bits it observes, although its UNRG_CTR is greater than 0. Thus, we have shown that if station "1" is saturated all stations with offered load greater than x_1 have the same throughput x_1. Finally, every underloaded station "j" with offered load less than x_1 receives its bandwidth since the fraction of slots carrying TAR=1 bits it observes (x_1/M) is larger that its own requirement x_j/M for inserting extra requests.

Let us now consider the case where the first saturated station on the bus is station "j". This station writes on every passing unreserved slot it sees on the forward channel and erases every TAR=1 bit it observes. If its throughput is x_j , then it will insert x_j/M TAR=1 bits. According to our previous discussion the downstream to "j" stations with offered load less than x_j receive the required throughput and the rest of the downstream stations have throughput x_j. Furthermore, since station "j" is the first saturated station, its upstream stations also receive bandwidth equal to their offered load. It remains to show that there is no station "i" upstream to "j" with throughput greater than x_j. It is evident from the ITU_NSW operation that station "i" cannot erase more TAR=1 bits than it will insert. This is because the condition *NTAR_R* > DBTAR_CTR allows a station to erase a TAR=1 bit if and only if the station has a TAR_segment queued for transmission. Let us denote again with t_j the fraction of slots carrying TAR=1 bits that station "j" sees on the forward bus. It is evident that t_j is greater than or equal to max(x_i/M, i=1,2,...,j-1). Since station "j" erases all TAR=1 bits it observes on the forward channel, its throughput, x_j, should be greater than or equal to Mt_j, i.e. $x_j \geq Mx_i/M = x_i$. □

4.5 NSW_BWB and ITU_NSW Performance under one Traffic Class

In this section we use simulation to investigate the performance of NSW_BWB and ITU_NSW. Furthermore, we compare the two versions of NSW with BWB_DQDB. As in the cases of Tables 4.1 and 4.2, we consider a high capacity network of 155.52 Mbps, a slot size of 53 bytes, and a signal propagation delay of 5 $\mu sec/Km$.

The behavior of the overloaded stations in NSW_BWB and ITU_NSW is identical. Furthermore, both schemes distribute the available bandwidth among the users exactly in the same way. Therefore, in cases where the throughput performance is investigated we use the same curve to describe both schemes and refer to them with the generic term NSW. In Fig.4.2 we show the convergence speed of NSW when there are only two stations present on the bus and at a distance of D_{12}=78 slots; corresponding to a cable

length of 42.53 Km. The horizontal axis represents time, measured in slots, with the ticks appearing in multiples of the end-to-end propagation delay. Initially, station "1" is only active and overloaded and acquires all channel bandwidth. We can clearly see the amount of bandwidth that is wasted in the case of BWB_DQDB, which is significant when M=2. At time t=312, station "2" becomes active and tries to acquire all the bandwidth. Fig.4.2 shows what the analysis had predicted, i.e. the convergence speed of NSW is significantly higher than that of BWB_DQDB; even when the value of M is 2, the convergence speed of BWB_DQDB is much lower. We can also see that NSW does not waste any bandwidth whereas the bandwidth loss of BWB_DQDB, especially when M=2, is high. We point out that in Fig.4.2, as well as in the subsequent figures, each point shows the bandwidth that a station has received since the previous measurement (point). Therefore, a throughput of .5 at t=624 means that the two stations have reached steady state and have started receiving the same bandwidth at time t=624-2*78=468. This is consistent with equation (4.2.2) which indicates that if M is equal to 2, the system will reach steady state within one round-trip propagation delay.

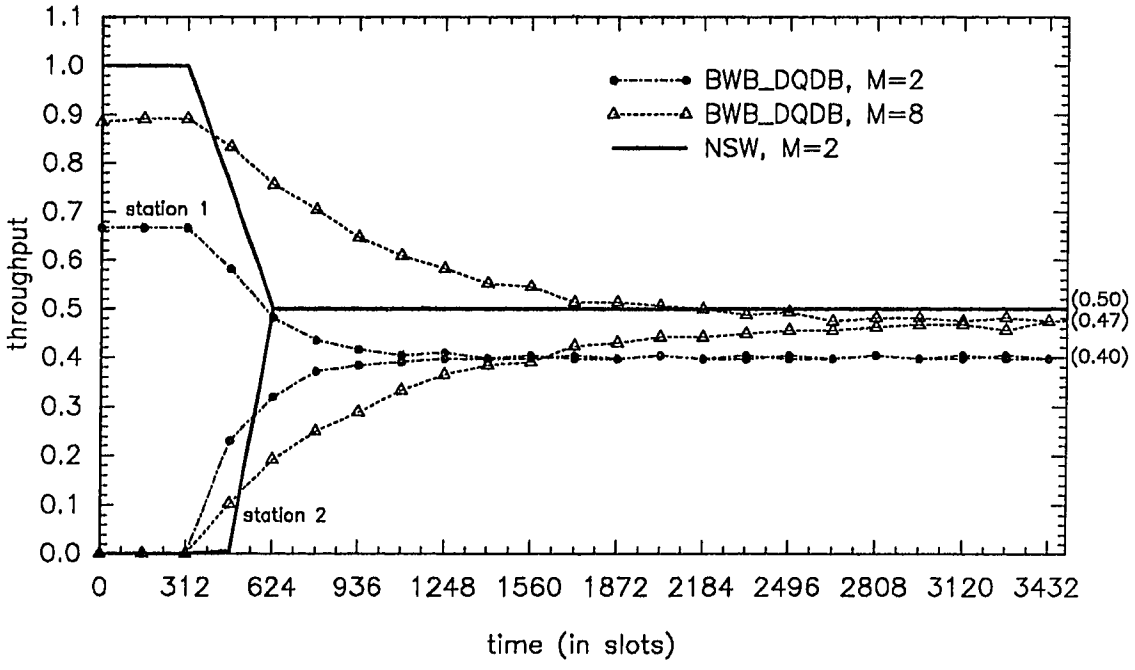

Fig.4.2:Throughput performance. Comparison of BWB_DQDB and NSW. D_{12}=78 slots.

In Fig.4.3 we compare the convergence speed of the two schemes in the case of three stations. The distances between neighbor stations are $D_{12}=38$ slots and $D_{23}=40$ slots. Initially, station "1" is the only active and overloaded station on the bus. Again, we can see the significant bandwidth loss in the case of BWB_DQDB. At t=468, station "2" becomes active and tries to acquire all the bandwidth. We see that NSW converges very fast to the steady state; it is evident that this is not the case for BWB_DQDB. At t=3588, station "3" becomes active and tries to acquire all the bandwidth. NSW, once more, arrives much earlier at the steady state providing stations with slightly higher throughputs.

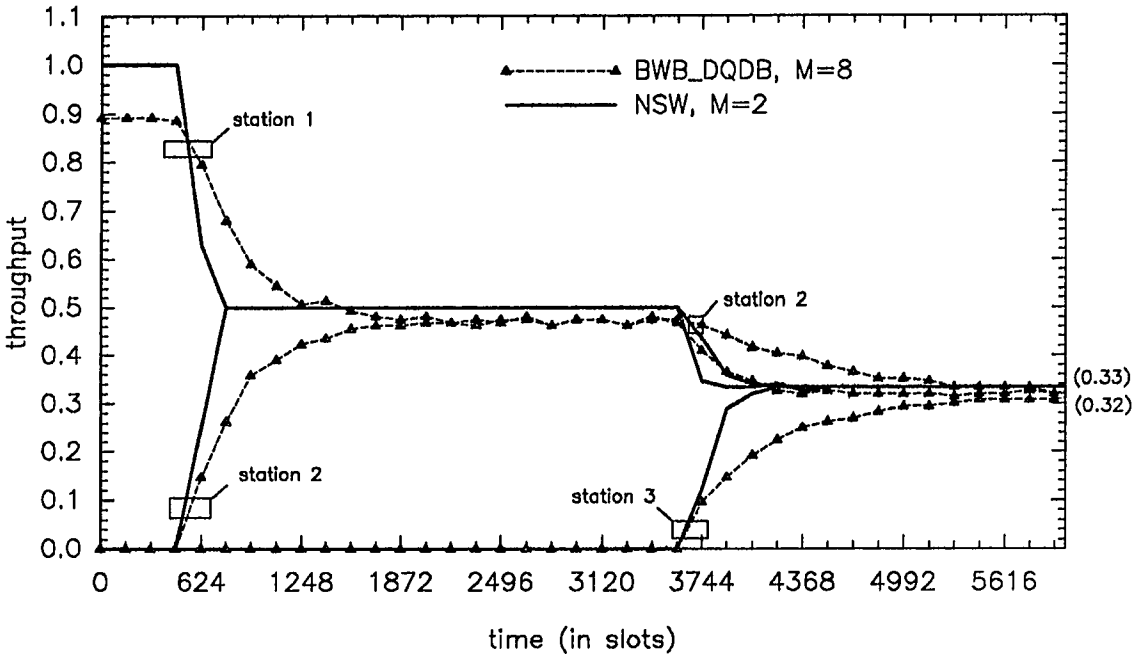

Fig.4.3:Throughput performance. Comparison of BWB_DQDB and NSW.
$D_{12}=38$ slots. $D_{23}=40$ slots.

In Fig.4.4 we consider the three stations network of Fig.4.3 but with the stations now becoming active in the order "1" first, "3" second, and "2" third. The much faster convergence of NSW, when the second station (station "3") becomes active, is again evident. However, we can observe a significant difference in the behavior of the two schemes when station "2" becomes active, at time t=3588, and tries to acquire all the bandwidth. In the case of BWB_DQDB the throughputs of stations "1" and "3" gradually

decrease, while at the same time, the throughput of station "2" increases. However, in the case of NSW, the throughput of station "3" temporarily increases whereas the throughput of station "1" decreases. The reason is the following. When station "3" becomes active, the RQ_CTR of station "2" will initially increase since "2" will see the *extra requests* from "3" on the reverse channel and all slots written by "1" on the forward channel. When "1" sees the *request* of "3" and allows free slots to go by, the rate of idle slots on the forward bus that "2" will see will be the same with the rate of *requests* on the reverse

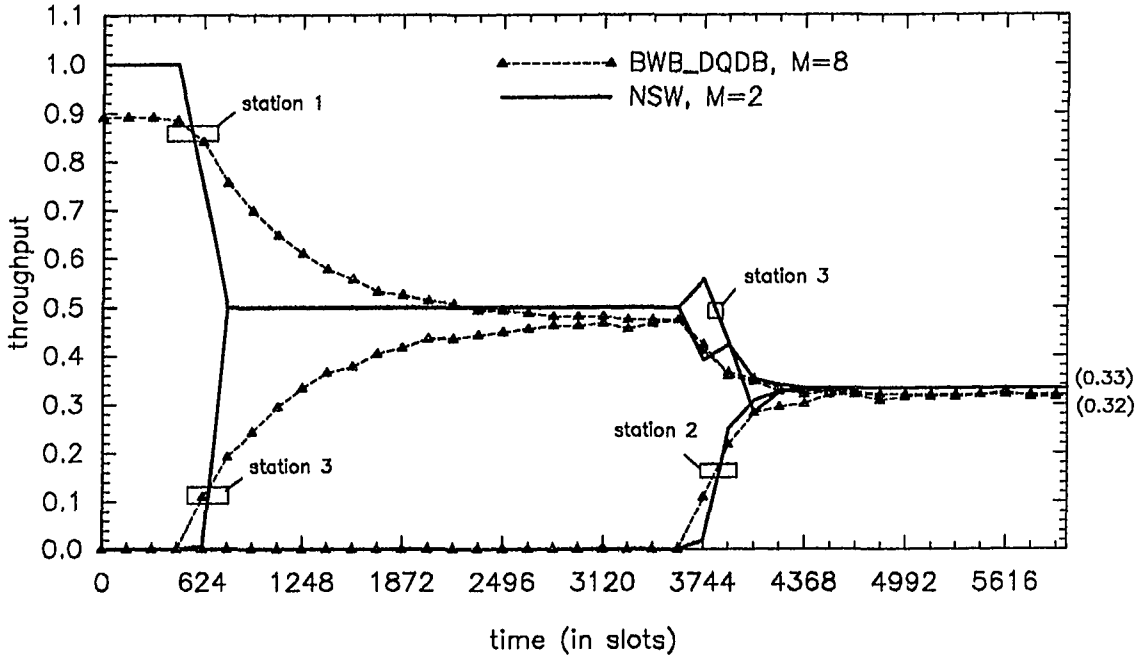

Fig.4.4:Throughput performance. Comparison of BWB_DQDB and NSW. D_{12}=38 slots. D_{23}=40 slots.

bus that "3" will send. Hence, the RQ_CTR of station "2" will not decrease. When station "2" becomes active, it will start sending *requests*. However, "2" will not be able to write on any idle slot until its RQ_CTR becomes 0. Thus, all idle slots, allowed by station "1" to travel downstream, will be written by station "3"; whose throughput, for this reason, will temporarily increase. We point out that this behavior will not be observed if the three stations become active in the order "3" first, "1" second and "2" last. Although, again, station "2" becomes active last, its RQ_CTR will not increase, when station "1"

becomes active, but oscillate between 1 and 0; since the rates of idle slots on the forward bus and *requests* on the reverse bus, that "2" sees, will be the same. Therefore when "2" becomes active, its RQ_CTR will be 0 or 1 and it can start writing immediately on the idle slots that will pass on the forward bus. Consequently, the throughput of station "3" will not increase. This last behavior has also been verified by simulation results.

In the remaining figures of this section we carry out a delay comparison of NSW_BWB, ITU_NSW and BWB_DQDB. We consider, for all schemes, the MAC mechanism implementation that requires one CD_CTR and one RQ_CTR inside each station. However, for completeness, in some cases we include the version of ITU_NSW which uses only the RQ_CTR. We call this scheme RQ_ITU_NSW. We consider a dual bus network consisted of 20 stations and two cases of network size with inter-station distances 2 and 10 slots, respectively; corresponding to a total cable length of 20.72 and 103.6 Km, respectively. We compare the delays encountered by the different stations considering their transmissions on the forward bus. We define as average message delay, the average elapsed time from the instant a message arrives at station "i" until the last segment of this message is about to start its transmission onto the medium. We have assumed Poisson arrivals for the messages and that each station transmits to any other station with the same probability. This means that the load of the stations linearly decreases as we move towards the end of the bus; it is obvious that station "19" does not transmit any message on the forward bus and that its data load is 0.

In Fig.4.5 we consider independent segment transmissions, i.e. each message consists of a single segment. The forward bus utilization is .9. Fig.4.5 shows that NSW_BWB and ITU_NSW have almost identical delay characteristics. It also indicates that the absence of the CD_CTR the delay variation among the stations can be significantly reduced. Furthermore, it shows that BWB_DQDB has a smaller delay variation than either NSW_BWB or ITU_BWB, but larger than that of RQ_ITU_NSW. This is due to that BWB_DQDB allows more free slots to go downstream and the remote from

the slot generator stations see earlier transmission opportunities. In contrast, in NSW, a station must ask for a free slot by sending a *request*. For this reason NSW, in the case of single segment transmissions, is not as responsive as BWB_DQDB. The penalty that BWB_DQDB has to pay is the bandwidth that wastes which results in much higher average (over all stations) delays

In Fig.4.6 we compare the behavior of the schemes when messages that consist of 20 segments are transmitted by the stations and the forward bus utilization is .9. Fig.4.6 shows that under long messages the effectiveness of NSW_BWB becomes evident. NSW not only decreases the delay variation among the stations but also provides significantly lower delays than BWB_DQDB. We also see that, contrary to what may have been expected from Fig.4.5, RQ_ITU_NSW is not the most fair scheme. Actually, in this case the absence of the CD_CTR favors the downstream stations over the upstream ones. It is now ITU_NSW that provides the smaller delay variation among the stations.

In Fig.4.7 we consider the transmission of very long messages, consisted of 100 segments. Again. the forward bus utilization is 0.9. Fig.4.7 shows that in all cases the upstream stations are penalized. Furthermore, BWB_DQDB has the highest message delays. The performance characteristics of NSW_BWB and ITU_NSW are very similar. Finally, the delay variation in the case of RQ_ITU_NSW is slightly higher than in NSW_BWB or ITU_NSW.

In Fig 4.8 we consider the same system parameters as in Fig 4.6, but we now increase the interstation distance to 10 slots. This extends our network to 103.6 Km. Fig 4.8 shows that as the network becomes longer the more the upstream stations are favored. This behavior is due to the longer distance that the requests, inserted by the downstream stations, have to travel in order to inform the upstream stations about the presence of segments ready for transmission. As expected, ITU_NSW and RQ_ITU_NSW are the schemes with the best delay characteristics under this network configuration.

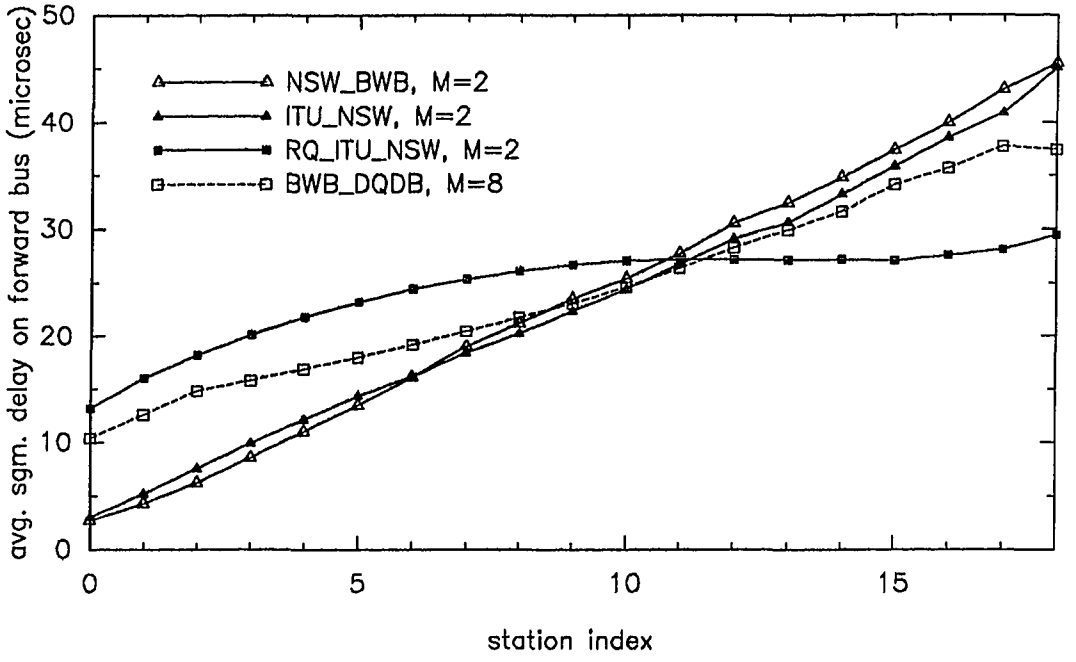

Fig.4.5:Delay comparison of NSW_BWB, ITU_NSW, RQ_ITU_NSW and BWB_DQDB. Bus utilization 0.9.

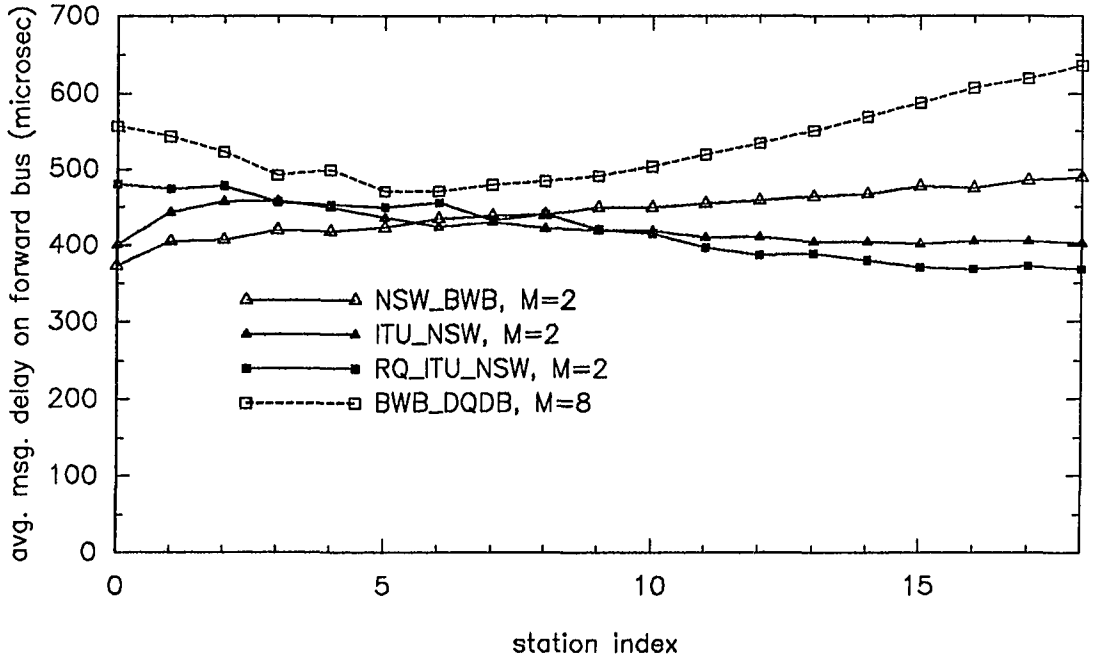

Fig.4.6:Delay comparison of NSW_BWB, ITU_NSW, RQ_ITU_NSW and BWB_DQDB. Bus utilization 0.9. Constant message size of 20 segments.

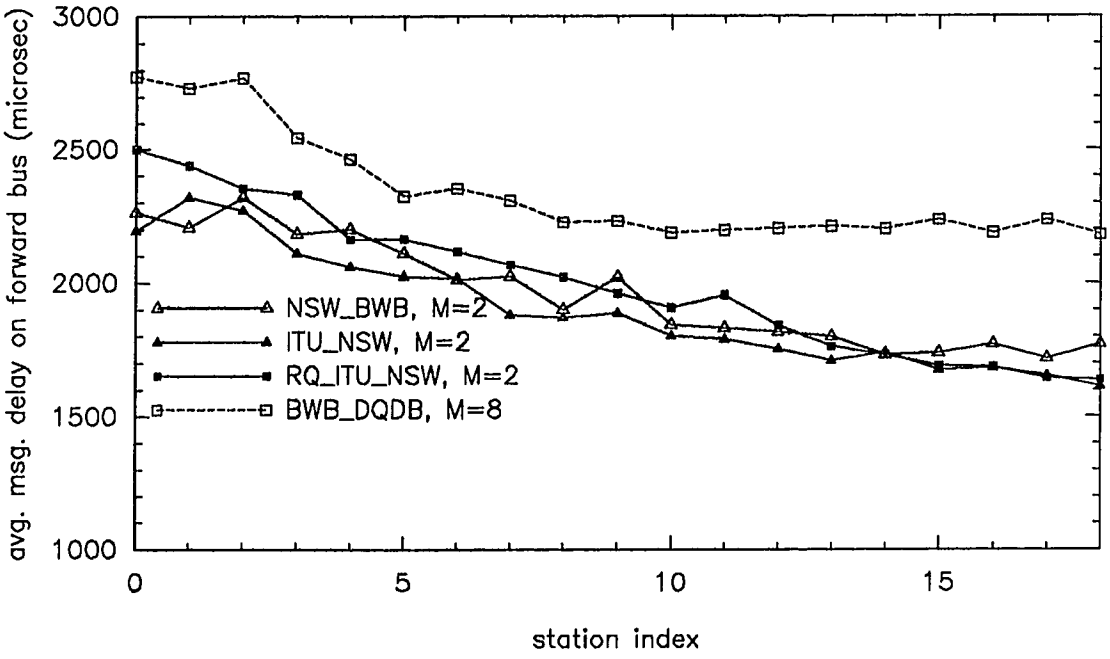

Fig.4.7:Delay comparison of NSW_BWB, ITU_NSW, RQ_ITU_NSW and BWB_DQDB. Bus utilization 0.9. Constant message size of 100 segments.

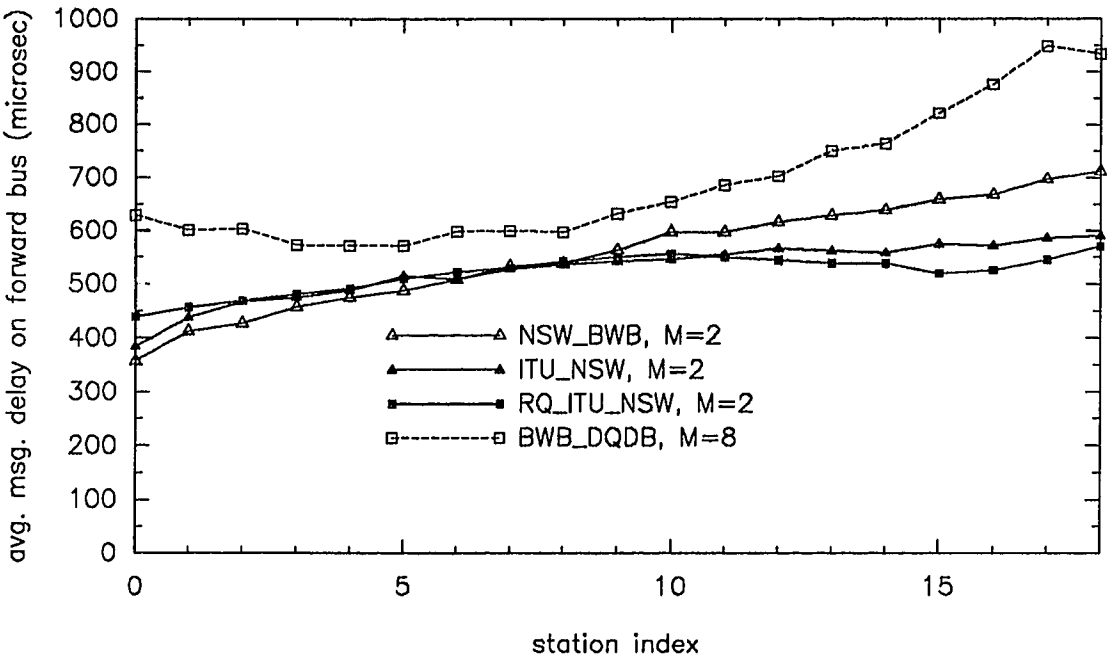

Fig.4.8:Delay comparison of NSW_BWB, ITU_NSW, RQ_ITU_NSW and BWB_DQDB. Bus utilization 0.9. Constant message size of 20 segments. Long Network.

4.6 Arbitrary Bandwidth Distribution

NSW_BWB and ITU_NSW have also the capability to distribute the channel bandwidth among stations in any arbitrary way. In this case we have to assign different values of M to the various stations. If M_i is the value of M given to station "i" and all N stations of the system are overloaded, then the throughput T_i of station "i" will be given by:

$$T_i = \frac{M_i}{\sum_{j=1}^{N} M_j} \tag{4.6.1}$$

Equation (4.6.1) holds due to the same reason that makes the even bandwidth allocation possible. That is, in the case of overloaded stations each station will erase all TAR=1 bits that observes and will send an equal number of TAR=1 bits. Since the number of segments that each station "i" transmits is a multiple M_i of the number of TAR=1 bits that it sends and all stations transmit the same number of TAR=1 bits, their throughputs will be proportional to their values of M.

We can also compute the throughputs of the various stations when some of them are lightly loaded and others are overloaded. In this case the lightly loaded stations will receive all the requested bandwidth. The remaining bandwidth will be distributed among the overloaded stations in a way which is proportional to their values of M. However, it is not immediately obvious by the offered load r_i and value of M_i which of the stations are underloaded and which are overloaded. A recursion must be used to identify the underloaded stations. Let $N_{un}(k)$ be the set of stations that are found to be underloaded during the k_{th} step of the recursion. Let $N_{al,un}(k)$ be the set of all stations that have been found to be underloaded up to, and including, the k_{th} step of the recursion, i.e. $N_{al,un}(k) = \bigcup_{l=1}^{k} N_{un}(l)$. Let $N_{al,ov}(k)$ be the compliment set of $N_{al,un}(k)$. Then the stations of set $N_{un}(k)$ are the stations that belong to set $N_{al,ov}(k-1)$ and their offered load satisfies the following inequality:

$$r_i \le \left[1 - \sum_{j \in N_{al,un}(k-1)} r_j \right] \frac{M_i}{\sum_{l \in N_{al,ov}(k-1)} M_l} \tag{4.6.2}$$

Let n^* be the maximum step of the recursion for which $N_{un}(n^*)$ is not the null set, i.e. n^* is the step of the recursion in which all the underloaded stations have been identified. Then the throughput T_i of station "i" will be:

$$T_i = \begin{cases} r_i & \text{if } i \in N_{al,un}(n^*) \\ (1 - \sum_{j \in N_{al,un}(n^*)} r_j) \dfrac{M_i}{\sum_{l \in N_{al,ov}(n^*)} M_l} & \text{if } i \in N_{al,ov}(n^*) \end{cases} \tag{4.6.3}$$

In Table 4.3 we show the bandwidth distribution in the case of a network consisted of five active stations, with inter-station distance of 16 slots, and with the following values of M : $M_1 = M_4 = 3$, $M_2 = M_5 = 4$, $M_3 = 6$. The corresponding values of throughput in the case of overload conditions, provided by (4), will be $T_1 = .15 = T_4$, $T_2 = .2 = T_5$, and $T_3 = .3$. We first look at the case where all five stations are overloaded with the same offered load of .8 segm./slot. We can easily see that the throughputs of the stations computed from the simulation coincide with the ones derived from equation (4.6.1). Then we look at three different cases of loading in which stations can have higher, lower or equal loads with the ones guaranteed by (4.6.1). We see that the stations with load lower than or equal to the one guaranteed by (4.6.1) receive all the requested bandwidth. The remaining bandwidth is distributed among the rest of the stations according to equation (4.6.3).

In Fig.4.9 we consider a network of three stations which have been assigned the following values of M: $M_1 = 2$, $M_2 = 3$, and $M_3 = 5$. Due to the bandwidth wastage, the corresponding values of M in the case of BWB_DQDB are $M_1 = 6$, $M_2 = 9$ and $M_3 = 15$. Initially, station "2" is active and overloaded and receives in the case of NSW the entire bandwidth. In the case of BWB_DQDB its throughput is .9. At t=156, station "1" becomes active. We see that NSW converges faster to steady state providing stations "1" and "2" with throughputs .4 and .6, respectively; due to the bandwidth loss BWB_DQDB provides slightly lower throughputs. Finally, at t=2964, station "3" becomes active.

Again, NSW_BWB converges faster to steady state providing a throughput of T_1=.2, T_2=.3, and T_3=.5 segm/slot to the stations.

Table 4.3: Bandwidth distribution under different values of M.

Offered Load					Throughput NSW				
stat. 1	stat. 2	stat. 3	stat.4	stat.5	stat. 1	stat. 2	stat. 3	stat.4	stat.5
M=3	M=4	M=6	M=3	M=4	M=3	M=4	M=6	M=3	M=4
0.80	0.80	0.80	0.80	0.80	0.15	0.20	0.30	0.15	0.20
0.10	0.20	0.20	0.10	0.80	0.10	0.20	0.20	0.10	0.40
0.20	0.20	0.20	0.20	0.20	0.20	0.20	0.20	0.20	0.20
0.20	0.20	0.20	0.20	0.3	0.18	0.20	0.20	0.18	0.24

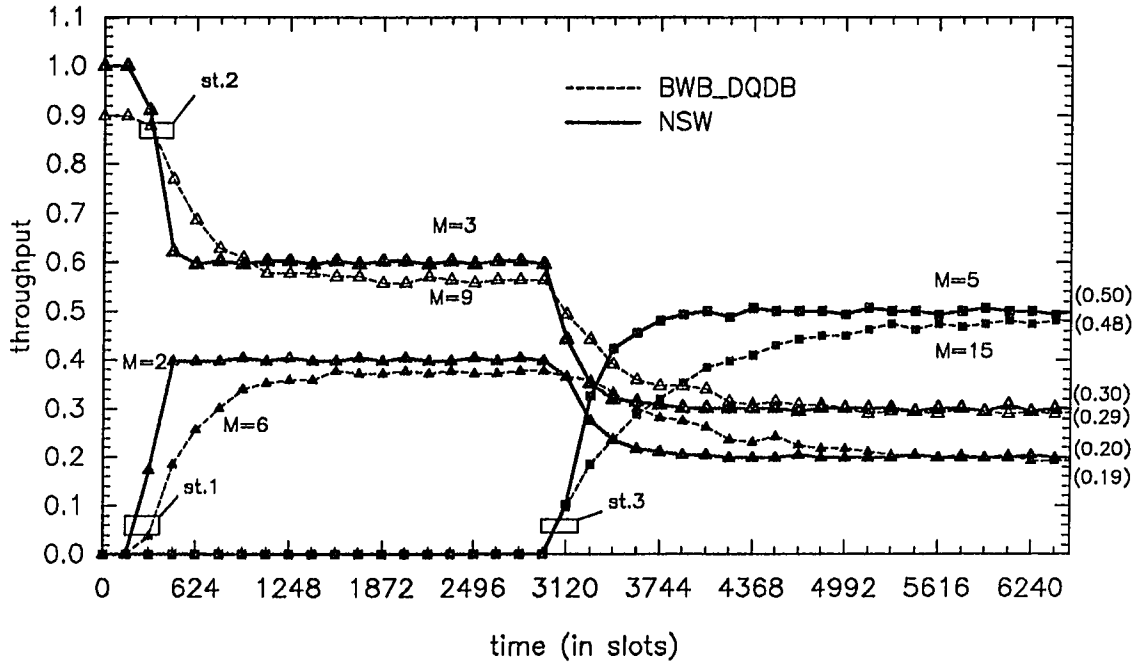

Fig.4.9:Throughput performance. Comparison of BWB_DQDB and NSW. D_{12}=38 slots. D_{23}=40 slots.

4.7 A Queueing Model for the NSW scheme.

In this section we provide a queueing analytic model which is capable of describing the behavior of NSW. We consider the version of NSW that uses only the RQ_CTR. Furthermore, we assume that independent segments arrive at the stations according to a Poisson distribution. We model each station "i" as a multiqueue single server queueing system and define the service policy which is followed as well as the arrival process to each one of the queues. In fact, it is the nature of the arrival process to each queue that encaptures the interprocess dependencies among different stations, the effect of the presence of the TAR bit and the effect of the request mechanism on the reverse bus. Throughout this section we assume that our network has N stations, indexed from 1 to N, and that all stations transmit on the forward bus. Furthermore, we assume that the interstation distance is d slots, with $d \geq 1$.

4.7.1 Model Description.

In this section we discuss the queueing model which is used to describe the behavior of the NSW scheme. Each station "i" is described by a separate three-queue single server queueing model, as it is shown in Fig. 4.10. The **B-Queue** models the passing slots that have already been written by an upstream station, i.e. the busy slots traveling on the forward bus. The **R-Queue** models the incoming requests from the downstream stations, i.e requests traveling on the reverse bus. Each incoming request increases the station's RQ_CTR by one which is then reduced by one for every empty slot seen on the forward bus. Since we assume that there is no CD_CTR inside the stations the incoming requests should be served before the local segments. Furthermore, the **S-Queue** models the traffic which is generated at the station, i.e. the local segments queued for transmission. Finally, the service time of all customers of all classes is deterministic and equal to the duration of one slot (1 time unit).

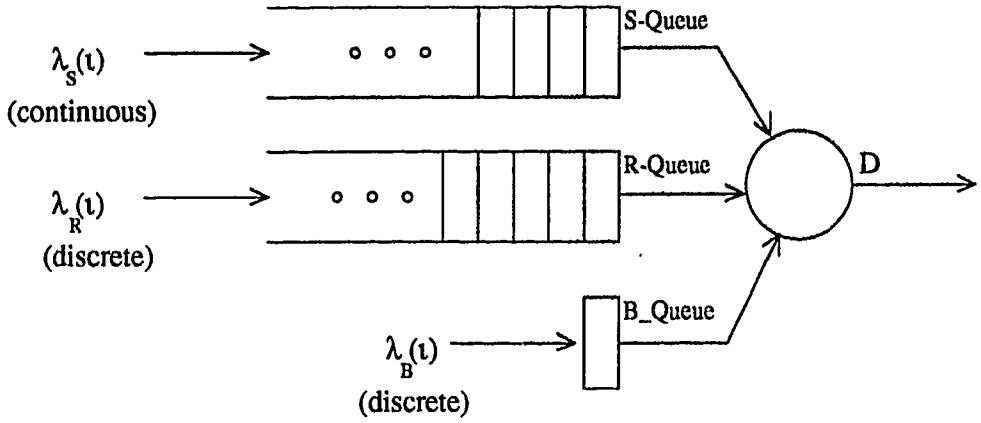

Fig.4.10 : The queueing model for a single station.

We have assumed that the station observes the beginning of a slot on the reverse bus just before it observes the beginning of a slot on the forward bus. Under this assumption it is impossible for an R type customer to wait for service while an S type customer is being serviced. This is because if an empty slot and a RB=1 arrive at the same time and there is at least one S type customer waiting for service, the empty slot will be used to serve the request. Therefore, R type customers (requests) have absolute priority over S type customers (local segments). It is also evident from the operation of the NSW that B type customers (busy slots) have absolute priority over the other two (requests, local segments) customer classes. B and R type customers arrive at discrete time instants, i.e. at the slot boundaries and they can start their service at the next slot. The mean arrival rate of class B, R and S type of customers is $\lambda_B(i)$, $\lambda_R(i)$ and $\lambda_S(i)$ respectively. Certainly, $\lambda_B(1)=\lambda_R(N)=0$. B and R type customer arrival processes are modeled as a first order two-state Markov chain. We have chosen this model because it can efficiently describe the variance of the arrival processes. If the arrivals of the B and R type customers were modeled as Bernoulli processes then we would not have been able to capture the effect of the various correlations into our model. This point will become more evident in the sequel. The two-state Markov chain $\{ X_k \}$, shown in Fig.4.11 provides a first-order approximation of the actual pattern of B and R type customer arrivals. $X_k=1$ indicates the

arrival of a customer during the k_{th} slot. X_k=0 indicates that no customer arrives during the k_{th} slot. A separate two-state Markov chain is required for the description of each of the arrival processes for the B and R type of customers. The mean arrival rate, $\lambda_B(i)$ or $\lambda_R(i)$, is equal to the steady state probability that the chain is at state "1". Furthermore, local segments (i.e. S-customers) arrive according to the Poisson distribution with mean $\lambda_S(i)$.

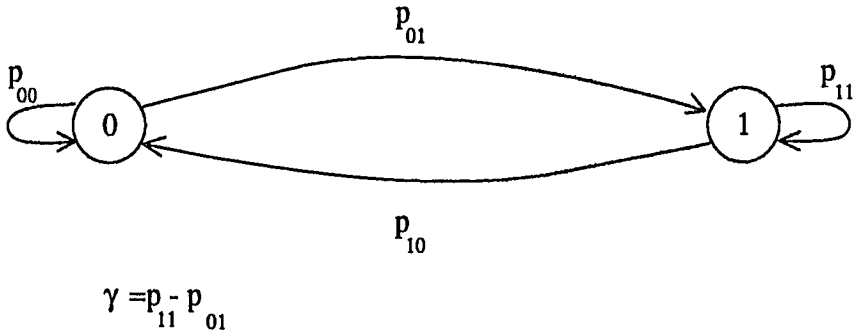

Fig.4.11 : The two-state Markov Chain describing the arrival process.

Let $D_j(i)$ denote the average delay of customers of priority "j" at station "i", where "j" can be either B, R or S. It is evident that $D_B(i)$=1. Let also $D_X^{FIFO}(i)$ denote the average delay that a customer experiences in an equivalent queuing system, in which the service discipline is FIFO and the arrival process is the superposition of the arrival processes of customers that belong into set X. Since the B and R type customers have preemptive priority over the S type of customers, the presence of the S type customers does not affect their delays. Furthermore, our system is work conserving and thus, the following equation holds:

$$\frac{\lambda_B}{\lambda_{B,R}} D_B + \frac{\lambda_R}{\lambda_{B,R}} D_R = D_{\{B,R\}}^{FIFO} \tag{4.7.1}$$

We also have that:

$$\frac{1}{\lambda}\left[\lambda_B D_B + \lambda_R D_R + \lambda_S D_S\right] = D_{\{B,R,S\}}^{FIFO} \tag{4.7.2}$$

where $\lambda_{B,R}=\lambda_B+\lambda_R$ and $\lambda=\lambda_B+\lambda_R+\lambda_S$. Equations (4.7.1) and (4.7.2) hold for every station and so we have dropped the index "i" for the involved symbols. We now use a result

from [75] where a FIFO discipline queuing model has been studied. The arrival process of that system is the outcome of the superposition of M arrival processes, where each arrival process has been modeled as a first order two-state Markov process, identical to the ones we have used to describe the arrivals of B and R type customers in our queuing model. It has been shown in [75] that the average delay D^{FIFO} experienced by the customers is given by:

$$D^{FIFO}=\frac{\sum_{n=1}^{M}\sum_{m>n}^{M}\left[1+\frac{\gamma_n}{1-\gamma_n}+\frac{\gamma_m}{1-\gamma_m}\right]}{\sum_{n=1}^{N}\lambda_n\left(1-\sum_{i=1}^{N}\lambda_k\right)} \tag{4.7.3}$$

where $\gamma_n=p_{11}-p_{10}$ is the variability of the n_{th} arrival process; in the case of Bernoulli arrivals $\gamma_n=0$. D^{FIFO} is the average delay in the case where all the arrival processes are discrete-time processes and arrivals occur at the slot boundaries. However, in our queueing model local segments arrive according to the Poisson distribution. Since their service can start only at the beginning of the next slot we may write the following equation:

$$D^{FIFO}_{(B,R,S)}=D^{FIFO}+\left[\frac{\lambda_S}{2(1-\lambda_S)}+0.5\right]\frac{\lambda_S}{\lambda} \tag{4.7.4}$$

From equations (4.7.1)-(4.7.4) we can calculate D_R and D_S, provided we can describe the arrival process for each class of customers. That is, we have to define the transition probabilities for each of one of the two-state Markov chains that describe the arrival process of the B and R customers.

4.7.2 The R-Queue Arrival process.

In order to define the Markov chain that describes the R-Queue arrival process we have to find the average arrival rate of R type customers at station "i", $\lambda_R(i)$, as well as the variability $\gamma_R(i)$ of their arrivals. Notice that for the accurate representation of the behavior of the system this arrival process must take into consideration the exact operation of the protocol as much as possible. For instance, in the case of the NSW protocol,

each station may not insert a request for every arriving segment. This occurs when a flag_segment (TAR_segment) is transmitted and RG_CTR=0, i.e. the flag_segment (TAR segment) is transmitted into an unreserved slot. The model for the arrival process must take into account such an event. Furthermore, simulation results have shown that in the case of independent segment transmissions the probability of more than one segments waiting at the stations queue is very small. Therefore, in our analysis we have assumed that a segment can send a request after it has become first in queue † . In the sequel we first derive analytic estimates for the average arrival rate $\lambda_R(i)$ and then for the variability $\gamma_R(i)$. We define the following notation which we use throughout this subsection.

$e(i)$: percentage of slots that carry TAR=1 bits which are erased by station "i".

$r(i)$: percentage of slots that carry requests inserted by station "i".

$t(i)$: percentage of slots seen by station "i" and carrying TAR=1 bits.

$t_0(i)$: percentage of slots seen by station "i" which are busy and have TAR=0.

$I(i)$: percentage of slots seen by station "i", which are *unreserved.* A slots seen by station "i" is *unreserved* if it is empty and station's "i" RQ_CTR is equal to 0, i.e. a slot that can be written by station "i".

$B_f(i)$: a summation operator equal to $\sum_{j=i+1}^{N} f(j)$.

$F_f(i)$: a summation operator equal to $\sum_{j=1}^{i-1} f(j)$.

It is evident from the above definitions that $I(i)=(1-F_{\lambda_s}(i)-\lambda_R(i))$.

Each station inserts a TAR=1 bit every M segments it transmits. Station "i" will observe all the TAR=1 bits that the upstream stations insert and are not erased by another upstream station. Thus, we have that:

---

† Under this assumption NSW_BWB and ITU_NSW become identical with respect to the mechanism for inserting requests on the reverse bus.

$$t_0(i)=F_{\lambda_s}(i)-t(i)=F_{\lambda_s}(i)-\frac{F_{\lambda_s}(i)}{M}+F_e(i) \qquad (4.7.5)$$

Let $p_k(i)$ denote the probability that station "i" transmits its local segment in the k_{th} slot from the time the segment arrived at the station's S-Queue without inserting a request for the transmitted segment. Since we have assumed that only the first local segment can send a request upstream, a segment can be transmitted without inserting a request on the reverse bus if and only if it is a flag_segment (TAR_segment), the k_{th} slot is an *unreserved* slot and no TAR=1 bit is seen during the preceding k-1 slots. We may approximate the probability that no unreserved or busy slot with TAR=1 is seen during the first k-1 slots by $(t_0(i)+\lambda_R(i))^{(k-1)}$. We can also use the following estimate for the probability that the k_{th} slot is an *unreserved* slot: $I(i)=(1-F_{\lambda_s}(i)-\lambda_R(i))$ Then, $p_i(k)$ will be given by :

$$p_k(i)=(t_0(i)+\lambda_R(i))^{(k-1)}\,(1-F_{\lambda_s}(i)-\lambda_R(i)) \qquad (4.7.6)$$

The probability $p_{nr}(i)$ that a segment of station "i" is transmitted without inserting a request given that the segment is a flag_segment (TAR_segment) is equal to $\sum_{k=1}^{\infty} p_k(i)$. It is evident that:

$$r(i)=\left[\frac{M-1}{M}+(1-p_{nr}(i))\frac{1}{M}\right]\lambda_S(i) \qquad (4.7.7)$$

Then, by replacing the expression of $p_{nr}(i)$ (using (4.7.6)) into (4.7.7), the following expression for $p_{nr}(i)$ is derived:

$$r(i)=\left[\frac{M-1}{M}+\left(1-\frac{1}{1-t_0(i)-\lambda_R(i)}(1-F_{\lambda_s}(i)-\lambda_R(i))\right)\frac{1}{M}\right]\lambda_S(i) \qquad (4.7.8)$$

where $\lambda_R(i)$ can be expressed as $B_r(i)$. In fact, equation (4.7.8) is a non-linear system of N-1 equations (for i=2 up to N) with $r(i)$ and $e(i)$ unknown. We have assumed that only the first queued segment may insert a request. Is is easy to see that, under this assumption, the percentage of slots on which station "i" inserts regular requests will be

$[(M-1)/M]\lambda_S(i)$. Furthermore, the station will erase exactly one TAR=1 bit for every extra request it inserts. Then, under our assumption, the following equation can be written:

$$r(i)=\frac{M-1}{M}\lambda_S(i)+e(i) \tag{4.7.9}$$

Equation (4.7.9) provides the additional equations needed for the solution of the non-linear system (4.7.8). Then, the mean arrival rate for station's "i" R-Queue will be simply equal to $B_r(i)$.

The above approximate analysis provides estimates for the average arrival rate of requests that station "i" sees on the reverse bus. We also need to calculate the variability $\gamma_R(i)$ of the R type customer arrival process. We may assume that each station generates requests according to a Poisson distribution with mean $r(i)$. A station that wants to insert a request on the reverse bus, will set to 1 the first RB=0 it observes. This means, that a station can insert a request on the reverse bus only at the end of a train of RB=1 which has been produced by the downstream stations. Let us consider now the RBs seen by station "i". Let L(i) be a random variable that is equal to the length of a run of RB=1 followed by an RB=0 which is observed by station "i"; according to our definition of L(i) when k RB=0 have been seen on the reverse bus, then k distinct runs have been observed. It has been shown in [76] that:

$$E[L(i)]=\frac{1}{1-\lambda_R(i)} \quad \text{and} \quad E[L^2(i)]=\frac{1}{(1-\lambda_R(i))^3} \tag{4.7.10}$$

It is also evident from the two-state Markov process of Fig.4.11 that:

$$Pr\{L(i)=l\}=\begin{cases} p_{00}, & l=1 \\ p_{01}p_{11}^{(l-2)}p_{10}, & l>1 \end{cases} \tag{4.7.11}$$

Then, after some calculations the following expressions for $L(i)$ can be derived:

$$E[L(i)]=\frac{1-\gamma_R(i)}{1-p_{11}}=\frac{1}{1-\lambda_R(i)} \tag{4.7.12}$$

$$\gamma_R=\frac{Var(L(i))-\dfrac{\lambda_R(i)}{(1-\lambda_R(i))^2}}{Var(L(i))+\dfrac{\lambda_R(i)}{(1-\lambda_R(i))^2}}=\frac{\lambda_R(i)}{2-\lambda_R(i)}, \quad Var(L(i))=E[L(i)^2]-E[L(i)]^2$$

Given the values for $\gamma_R(i)$ and p_{11}, from equations (4.7.12), we can derive the following expressions for the the transition probabilities of the two-state Markov of Fig.4.11:

$$\begin{aligned} p_{01}&=\frac{2\lambda_R(i)(1-\lambda_R(i))}{2-\lambda_R(i)}, \quad p_{00}=1-p_{01} \\ p_{11}&=\frac{3\lambda_R(i)-2\lambda_R^2(i)}{2-\lambda_R(i)}, \quad p_{10}=1-p_{11} \end{aligned} \tag{4.7.13}$$

4.7.3 The B-Queue Arrival process.

In this subsection we will derive estimates for the transition probabilities of the two-state Markov chain which describes the arrivals of the B type customers. The distribution of the arrival process of the busy slots (B-customers) is drastically affected by the network size, the presence of the request bits on the reverse channel and the presence of the TAR bit on the forward channel. It seems extremely difficult to provide an accurate description of the arrival pattern of the BB=1 bits at station "i", because of all the complicated interdependencies among the different processes. In our analysis we consider the busy slots arrival process at station "i" when this station is "active". We say that station "i" is "active" if and only if at least one of the R or S queues is not empty. Considering the arrival process of busy bits at the time intervals during which station "i" is active can better encapute the effect of the reservation mechanism for the NSW scheme. For example, let us consider station "2" and let us assume that is located just after station "1", i.e. their distance is 0. Then if station "2" inserts a request for every segment it transmits, it effectively does not see any BB=1 on the forward bus, i.e $\lambda_B(2)=0$. This can be the case if we consider that arrival process of busy bits only when station "2" is "active". How-

ever, if we consider this process at every time instant, then we will derive that $\lambda_B(2)=\lambda_S(1)$.

Classification of S and R Type Customers.

We classify the S and R customers into three different categories according to the time instants they insert their requests. **Type-a** customers, are those which as soon as they arrive they request a slot from the upstream stations. All the R-customers and the local segments for which a regular request is inserted are type-a customers. **Type-b** customers are those that eventually insert a request before they receive service; but not at the time they arrive. These are all the S-customers that insert an extra request because a TAR=1 bit was seen on the forward bus during their waiting time. Finally, the rest of the S-customers are **type-c** customers. These are merely the flag_segments (TAR_segments) that are transmitted before a TAR=1 bit is seen on the forward bus. In the remaining of this section unindexed variables will refer to station "i". In the cases that more than one stations are involved, the appropriate index is used.

Let Z be a random variable that describes the length of a run of busy slots, followed by an empty slot that is seen by station "i" since the time instant an R or S type customer has arrived. For instance, if a local segment arrives at time T and the first empty slot arrives at station "i" at time $T+z$, then $Z=z+1$. Let p_{cl} denote the probability that the next R or S type customer to be served is of type cl, $cl \in \{a,b,c\}$. Let also Z_{cl} be a conditional random variable which is equal to Z, given that the next R or S type customer, scheduled for transmission is of type cl. We will first estimate $p_{cl}, E[Z_{cl}]$ and $E[Z_{cl}^2]$. Then it is evident that $E[Z]$ and $E[Z^2]$ will be given by:

$$\begin{aligned} E[Z] &= p_a E[Z_a]+p_b E[Z_b]+p_c E[Z_c] \\ E[Z^2] &= p_a E[Z_a^2]+p_b E[Z_b^2]+p_c E[Z_c^2] \end{aligned} \qquad (4.7.14)$$

Calculation of p_{cl}.

The definition of type-a customers includes all S type customers which insert a reg-

ular request and all R type customers. Thus the probability, p_a, that a customer is type-a can be estimated as:

$$p_a=\frac{\frac{(M-1)}{M}\lambda_S(i)+\lambda_R(i)}{\lambda_S(i)+\lambda_R(i)} \tag{4.7.15}$$

A customer is of type-b if its is not type-a and inserts a request (extra) before its transmission. In subsection 4.7.2 we have already calculated the probability $p_{nr}(i)$ that a local segment is transmitted without inserting a request. So we have that:

$$p_b=(1-p_{nr}(i))(1-p_a)=(1-\frac{1}{1-t_0(i)-\lambda_R(i)}(1-F_{\lambda_S}(i)-\lambda_R(i))(1-p_a) \tag{4.7.16}$$

Finally, the probability of type-c customer is given by:

$$p_c=p_{nr}(i)(1-p_a)=1-p_a-p_b \tag{4.7.17}$$

Calculation of $E[Z_a]$ and $E[Z_a^2]$.

Let us assume, momentarily, that station "i" and all stations located downstream to it, are not inserting any requests. It is evident that the order in which stations "1" to "i-1" access the forward bus does not affect the distribution of the busy slots seen by station "i". Thus we may assume that station "1" can access the bus first, station "2" can access the slots that station "1" allows pass empty and so on. We have also assumed that arrive at each station according to the Poisson distribution with mean $\lambda_S(i)$. We now model the arrival process of BB=1 seen by station "i" in this case as a two-state Markov process. By following a similar procedure to the one that led to equations (4.7.13) we may derive the follwing estimates for the transition probabilities:

$$\begin{aligned} p_{01}&=\frac{2F_{\lambda_S}(i)(1-F_{\lambda_S}(i))}{2-F_{\lambda_S}(i)} \qquad p_{00}=1-p_{01} \\ p_{11}&=\frac{3F_{\lambda_S}(i)-2F^2{}_{\lambda_S}(i)}{2-F_{\lambda_S}(i)} \qquad p_{10}=1-p_{11} \end{aligned} \tag{4.7.18}$$

Let us now attempt to take into consideration the requests inserted by station "i" and all the stations located downstream to station "i". We may assume that the sequence of busy slots seen by station "i" evolves according to the two state Markov chain which we

defined above, in equation (4.7.18), and is interrupted by the requests that are inserted by stations "i" through "N". Let X(i) be a random variable that describes the length of a run of busy slots, including the first empty slot, that station "i" sees from an arbitrary chosen time instant (and until the first empty slot). X(i) depends on whether the slot that passed the station just before the commence of the observation period was empty, i.e. X(i) depends on the current state of the Markov chain. Let now, $X_e(i)$ be the length of a run X(i) given that the preceding slot was empty and $X_b(i)$ the length of a run X(i) given that the preceding slot was busy. Then we may approximate the PMF of $X_e(i)$ as follows:

$$Pr\{X_e(i)=j\}=\begin{cases} p_{empty}, & j=1 \\ p_{busy}p_{01}p_{11}^{(j-2)}(1-\lambda_R(i-1))^{(j-1)}, & j>1 \end{cases} \tag{4.7.19}$$

where p_{empty} and p_{busy} is the probability that the next slot is empty, given that the current slot is empty or busy respectively. These two probabilities can be approximated by:

$$\begin{aligned} p_{empty}&=p_{00}+\lambda_R(i-1)-p_{00}\lambda_R(i-1) \\ p_{busy}&=p_{10}+\lambda_R(i-1)-p_{10}\lambda_R(i-1) \end{aligned} \tag{4.7.20}$$

Similarly with equation (4.7.19), we may write for X_b the following:

$$Pr\{X_b(i)=j\}=\begin{cases} p_{busy}, & j=1 \\ p_{busy}p_{11}^{(j-1)}(1-\lambda_R(i-1))^{(j-1)}, & j>1 \end{cases} \tag{4.7.21}$$

In the sequel, we calculate estimates for $E[Z_a]$ and $E[Z_a^2]$, assuming that the stations are located d slots apart. Fig.4.12 shows the network topology in this case. A type-a customer will insert its request as soon as it arrives at station "i". It is easy to see from Fig.4.12 that this request may affect the sequence of busy bits that station "i" sees only after $2d$ slots. Furthermore, given that the first $2d$ slots are busy, then the arrival process of the busy slots at station "i" after the $2d$ slots have passed is identical to the arrival process of busy slots at station "i-1". The probability, $p_{/e}$ that the first $2d$ slots seen by station "i" are busy given that the preceding slot was empty is equal to $1-\sum_{j=1}^{2d} Pr\{X_e(i)=j\}$. The same probability, given that the preceding slot was busy $p_{/b}$ is equal to

$1-\sum_{j=1}^{2d} Pr\{X_b(i)=j\}$.

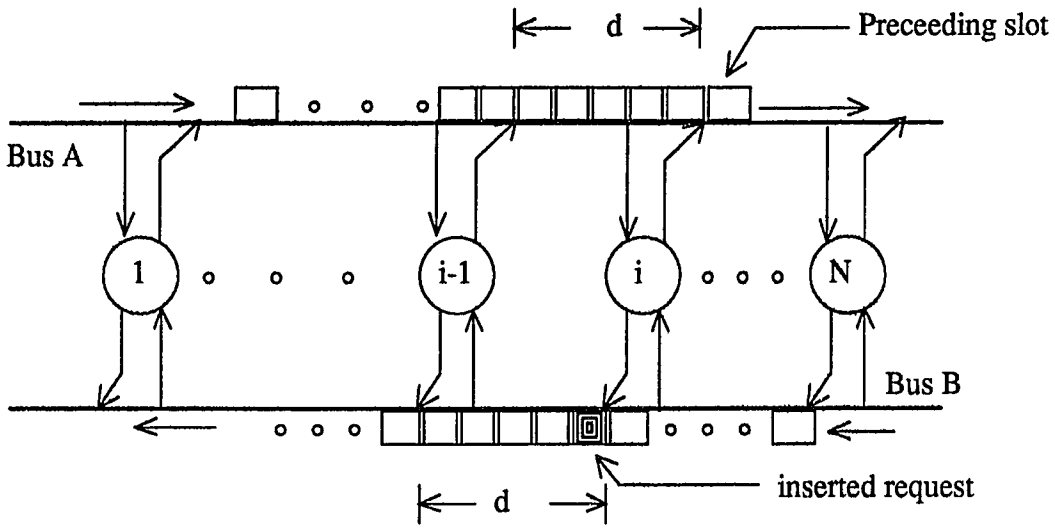

Fig.4.12: Network configuration just after the type "a" customer has arrived.

Taking into account the recursive nature of the pattern of the busy slots we can write the following equations:

$$E[Z_a(i)/empty]=\sum_{j=1}^{2d} jPr\{X_e(i)=j\}+\sum_{j=2d+1}^{\infty} jp_{/e}Pr\{Z_a(i-1)/busy=j-2d\}$$
$$=\sum_{j=1}^{2d} jPr\{X_e(i)=j\}+p_{/e}(E[Z_a(i-1)/busy]+2d) \qquad (4.7.22)$$
$$E[Z_a(i)/busy]=\sum_{j=1}^{2d} jPr\{X_b(i)=j\}+\sum_{j=2d+1}^{\infty} jp_{/b}Pr\{Z_a(i-1)/busy=j-2d\}$$
$$=\sum_{j=1}^{2d} jPr\{X_b(i)=j\}+p_{/b}(E[Z_a(i-1)/busy]+2d)$$

In a similar way we can derive recursive expressions for $E[Z_a^2(i)/empty]$ and $E[Z_a^2(i)/busy]$:

$$E[Z_a^2/empty]=\sum_{j=1}^{2d} j^2Pr\{X_e(i)=j\}+ \qquad (4.7.23)$$
$$p_{/e}(E[Z_a^2(i-1)/busy]+4dE[Z_a(i-1)/busy]+(2d)^2)$$
$$E[Z_a^2/busy]=\sum_{j=1}^{2d} j^2Pr\{X_b(i)=j\}+$$
$$p_{/b}(E[Z_a^2(i-1)/busy]+4dE[Z_a(i-1)/busy]+(2d)^2)$$

We have calculated the conditional first and second moments for Z_a. If p_0 is the probability that the slot on the forward bus is empty at the time a type-a customer arrives, than the unconditional first and second moments are given by †:

$$E[Z_a]=p_0E[Z_a/empty]+(1-p_0)E[Z_a/busy] \qquad (4.7.24)$$
$$E[Z_a^2]=p_0E[Z_a^2/empty]+(1-p_0)E[Z_a^2/busy]$$

In the sequel, we carry out an approximate calculation for $E[Z_b]$ and $E[Z_b^2]$.

Calculation of $E[Z_b]$ and $E[Z_b^2]$.

Type-b customers are flag_segments (TAR_segments) which insert a request before their transmission. Let Y be a random variable that describes the elapsed time from the instant a segment arrives at the station, until it sees a TAR=1 bit and inserts an extra request. After the extra request has been inserted we may assume that the customer behaves like a type-a customer. Furthermore, we assume that the time interval until a TAR=1 bit is seen and the time interval from the instant an extra request is inserted until the instant the first empty slot is seen, are independent. Then we may write the following expressions for $E[Z_b]$ and $E[Z_b^2]$:

$$E[Z_b]=E[Z_a/busy]+E[Y]$$
$$E[Z_b^2]=E[Z_a^2/busy]-E[Z_a/busy]^2+E[Y^2]-E[Y]^2+E[Z_b]^2 \qquad (4.7.25)$$

Following similar steps to the ones of section 4.7.2 we may write the following equations for Y:

$$E[Y]=\frac{1-t_0(i)-\lambda_R(i)}{t(i)}\sum_{j=1}^{\infty}jt(i)(t_0(i)+\lambda_R(i))^{(j-1)}$$
$$E[Y^2]=\frac{1-t_0(i)-\lambda_R(i)}{t(i)}\sum_{j=1}^{\infty}j^2t(i)(t_0(i)+\lambda_R(i))^{(j-1)} \qquad (4.7.26)$$

From equations (4.7.25) and (4.7.26), $E[Z_b]$ and $E[Z_b^2]$ can be calculated.

Calculation of $E[Z_c]$ and $E[Z_c^2]$.

Type-c customers are the flag_segments (TAR_segments), for which no extra request

† The calculation of p_0 is carried out at the end of this section.

will be sent on the reverse channel. This means that an unreserved slot is seen before a TAR=1 bit is observed. As in section 4.7.2 we can approximate the probability that Z_c is equal to j slots by:

$$Pr\{Z_c=j\}=(1-F_{\lambda_s}(i)-\lambda_R(i))(t_0(i)+\lambda_R(i))^{(i-1)} \tag{4.7.27}$$

and from equation (4.7.27) compute $E[Z_c]$ and $E[Z_c^2]$.

Given the value for p_0, i.e. the probability that a type "a" customer arrives when the passing slot on bus A is empty we can calculate $E[Z]$ and $E[Z^2]$. The final step is to describe a two-state Markov chain that can provide the same mean and variance of consecutive busy slots with the random variable Z. In order to find the transition probabilities of this Markov chain consider the time interval starting from a randomly selected moment until the first empty slot is generated. The duration of this time interval should have mean value equal to $E[Z]$ and second moment equal to $E[Z^2]$. Let A be the random variable that describes the length of this time interval. Let also $Pr\{A=j/k\}$ by the probability that $A=j$, given that at the comence of this time interval the Markov chain was at state k , k=0,1. Then after some calculations we can derive the following expressions:

$$\begin{aligned} E[A/0]&=\frac{p_{10}+p_{01}}{p_{10}} \quad \text{and} \quad E[A/1]=\frac{1}{p_{10}} \\ E[A^2/0]&=1-p_{01}+p_{01}\frac{2+p_{10}+p_{10}}{p_{10}^2} \quad \text{and} \quad E[A^2/1]=\frac{2-p_{10}}{p_{01}^2} \end{aligned} \tag{4.7.28}$$

Let $\pi_0=p_{01}/(p_{01}+p_{10})$ and $\pi_1=p_{10}/(p_{01}+p_{10})$ be the steady state probabilities of the Markov chain under consideration. Then we have that:

$$\begin{aligned} E[Z]&=\pi_0 E[A/0]+\pi_1 E[A/1] \\ E[Z^2]&=\pi_0 E[A^2/0]+\pi_1 E[A^2/1] \end{aligned} \tag{4.7.30}$$

From equations (4.7.30) we can calculate the transition probabilities for the two state Markov chain that describes the arrival process of the busy slots at station "i". Then mean arrival rate $\lambda_B(i)$ of B type customers generated by this Markov chain is equal to $\frac{p_{01}}{p_{01}+p_{10}}$. These transition probabilities can be calculated provided that we have

estimated the value of the probability p_0 in equation (4.7.24). Recall that p_0 is the probability that when a type "a" customer arrives at a station, the current slot on the forward bus is busy. We have assumed that p_0 should be equal to $F_{\lambda_S}(i)$, unless this assumption makes the mean arrival rate of B customers, $\lambda_B(i)$, greater than $F_{\lambda_S}(i)$. In this case p_0 has the value that forces $\lambda_B(i)$ to become equal to $F_{\lambda_S}(i)$

4.7.4 Model Accuracy

In this subsection we investigate the accuracy of the queuing model under various offered loads, network sizes and values of M. In all figures we assume linear load, i.e. each station transmits to any other station with the same probability. In all cases we compare the analytically derived delays with the corresponding delays produced from simulations of RQ_ITU_NSW. We point out here that under independent segment transmission NSW_BWB and ITU_NSW have almost identical delay performance (see Fig.4.5).

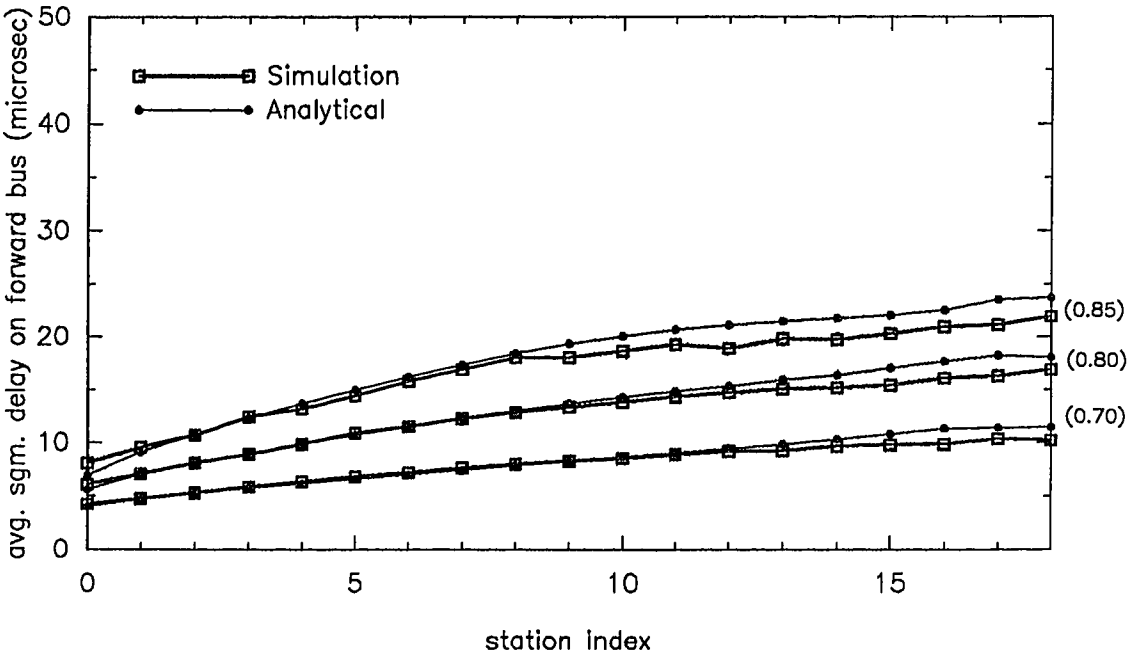

Fig.4.13: Analysis and simulation comparison. M=2. Interstation distance of 2 slots.

In Fig.4.13 we have assumed a dual bus network consisted of 20 stations which are located 2 slots apart; the end to end propagation delay is 38 slots. Furthermore, the value

of M for all stations is 2. We compare simulation and analytical results for three different bus utilizations,i.e. 0.70, 0.80 and 0.85. We see that analysis and simulation provide almost identical results, with the exemption of the most downstream stations in the case of 0.85 load. However, even in this case, the maximum difference between analysis and simulation does not exceed 1.5 slots. The difference is mainly due to the approximations that have been made on the arrival pattern of the busy slots. In that case we have assumed that a local segment cannot insert a request until it becomes first in queue. However, in ITU_NSW it is possible for a segment to insert a request before it becomes first in queue. The higher the utilization, the more frequently this event may occur.

In Fig 4.14 we use the system parameters of Fig.4.13 with the exception of the interstation distance which is 1 slot, i.e. we consider a shorter network. We see that in this case analytic results are almost identical to the ones derived from simulations, in all cases and for all stations. The difference in the downstream station delays observed in Fig.4.13 does not exist any more because the network is shorter and so the variability of the arriving at the station busy slots is considerably lower.

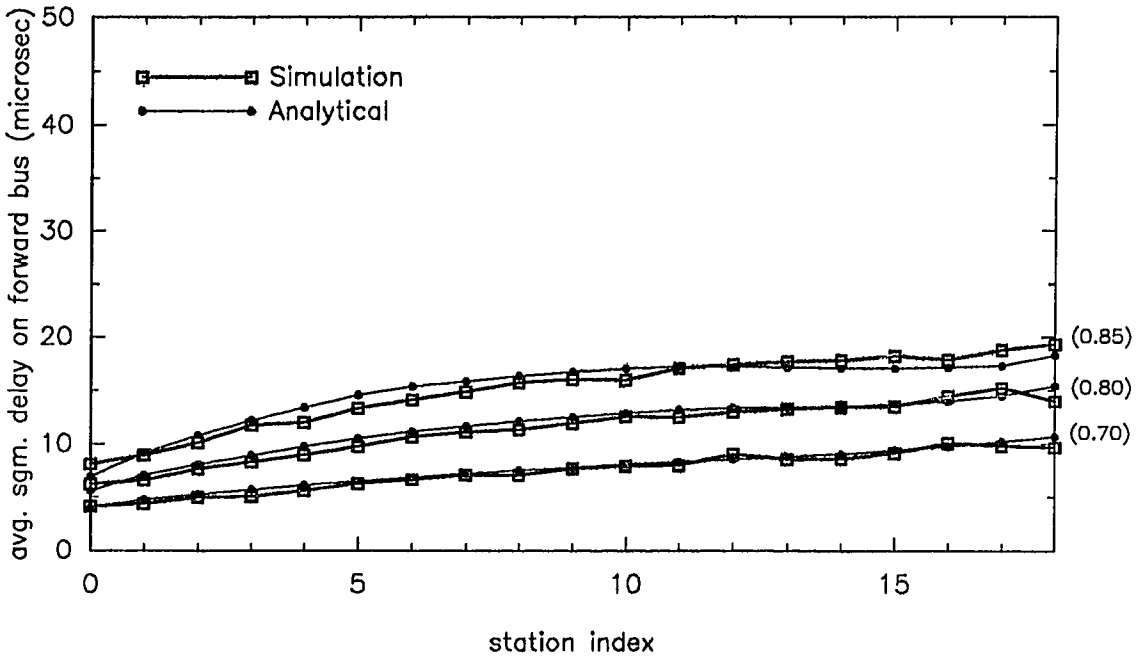

Fig.4.14: Analysis and simulation comparison. M=2. Interstation distance of 1 slot.

The next two figures, 4.15 and 4.16, are the corresponding to Figs.4.13 and 4.14 respectively, with M=8. These figures show that when the value of M increases the analytical results for the delay of the upstream stations are higher than the simulation ones. The reason for this behavior is that the analytical model slightly overestimates the number and the variability of requests that are seen by the stations. Nevertheless, the maximum difference between the analytical and the simulation results is less than 2 slots.

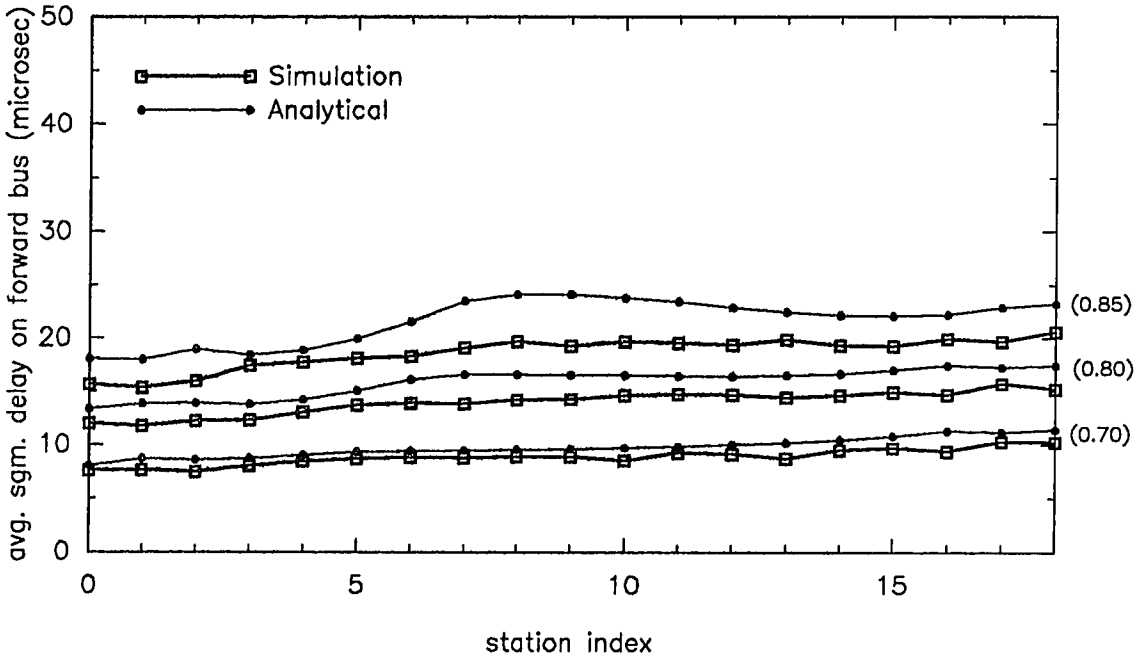

Fig.4.15: Analysis and simulation comparison. M=8. Interstation distance of 2 slots.

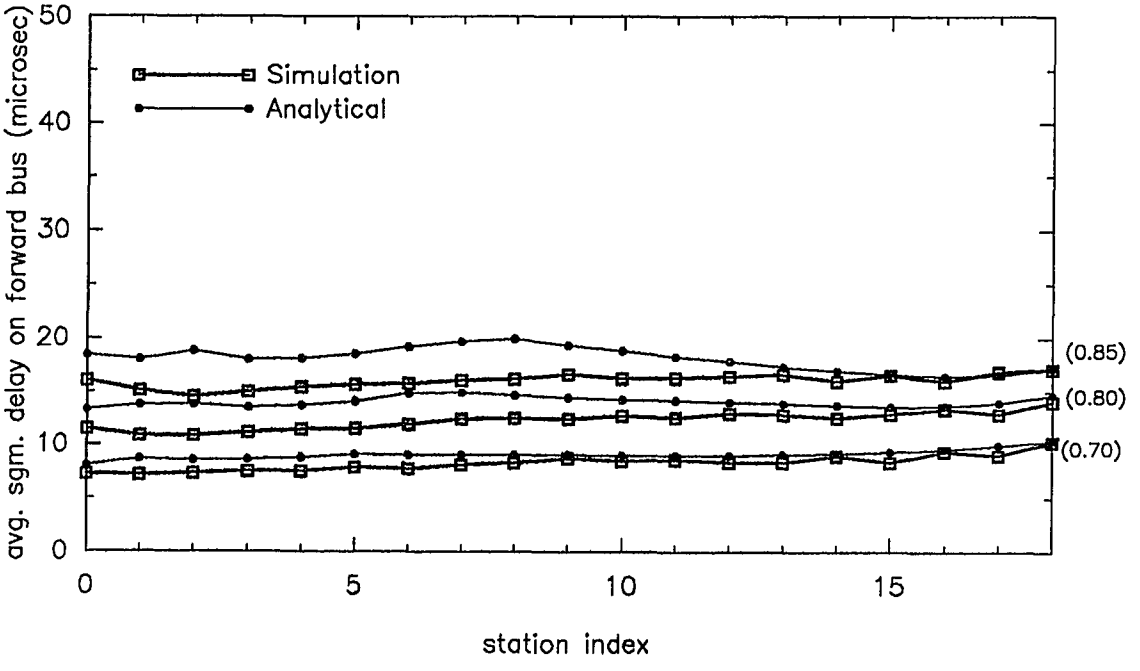

Fig.4.16: Analysis and simulation comparison. M=8. Interstation distance of 1 slot.

4.8 Priority Mechanisms

In this section we investigate the ability of NSW_BWB and ITU_NSW to provide similar priority services with the ones provided by the priority mechanisms introduced and investigated in [35] for BWB_DQDB. We consider that each station may serve different traffic classes with each class having its own queue of segments.

4.8.1 Bandwidth Balancing over Priority Classes

The objective of this mechanism is to guarantee a certain value of throughput to each priority class, regardless of the number of classes at each station. It is therefore similar with the objective of BWB_DQDB priority mechanism introduced in section 6 of [35]. We use a different value of M for each traffic class, M_i for class "i", and consider the station to behave as if it consisted of separate sub-stations with one traffic class per sub-station. The ordering of these sub-stations, inside the station, can be either from the highest priority class to the lowest priority class or vice versa. This means that if class "i" sees the slots on the forward bus before class "j", then class "i" will see the slots on the reverse bus after class "j"; in this case the *requests* that are inserted by class "j" will also be seen by class "i". This scheme requires each slot to have only one *busy, request,* and TAR bit, regardless of the number of priority classes in the system. The operation of each class (sub-station) is identical to the operation of a station in the case of a system under one traffic class described in sections 4.2 and 4.3 for NSW_BWB and ITU_NSW respectively. Consequently, each class "i" should maintain its separate counters, flags or registers.

4.8.2 Bandwidth Balancing over Stations

The objective of this mechanism is to guarantee a minimum throughput to each station. The value of this throughput is given, in the case of overloaded traffic classes, by equation (4.6.1) considering only the highest priority class at each station. The highest priority class can acquire all the bandwidth allocated to a station. A lower priority class

may receive some bandwidth only after all higher priority classes have satisfied their bandwidth requirements. Therefore, the objective of this mechanism is similar to the one of section 5 in [35].

This mechanism can be easily implemented in the case of ITU_NSW. It requires one UNRG_CTR, RG_CTR, DBTAR_CTR and $NTAR\_R$, and a separate BWB_CTR_i for each class "i". Each time a class "i" segment is transmitted BWB_CTR_i is increased by 1 and its value against M_i is checked. Furthermore, one additional counter for each class "i", the LQS_CTR_i, is needed that records the number of class "i" segments that are present in the station's local queue. It is increased by 1 for every arrival of a class "i" segment and decreased by one for every transmission of a class "i" segment. The station operation is identical to the one described in section 4.3. Only the calculation of the value for $NTAR\_R$ has to be slightly modified. Now, the number of TAR_segments is given by:

$$NTAR\_R = \sum_i \left\lfloor (BWB\_CTR_i + LQS\_CTR_i)/M_i \right\rfloor$$

The implementation of the BWB over stations is rather complex in the case of NSW_BWB. The simplicity of this mechanism in the case of BWB_DQDB is due to that every time a segment is transmitted a *request* is sent upstream. However, in the case of NSW_BWB, in order to compensate the upstream stations for the *extra request* they see, the stations do not send a *regular request* for the flag_segments. It is then possible for a low priority class to send an *extra request* and reserve a slot, which is then written by a higher priority segment. Although in most of the cases this may not create any problem, in special cases of loading it may create a deadlock for the lower priority classes. For this reason each class "i" inside a station must have its own RG_CTR_i, $UNRG\_CTR_i$, BWB_CTR_i, and TAR_flag_i. However, as in the case of BWB_DQDB, only one *busy* and *request* bit are required per slot, and only one RQ_CTR and CD_CTR are required per station.

The reaction of each priority class at a station to the events **"segment arrival"**,

"segment transmission", "segment becomes first in queue" and **"TAR=1 is seen on forward bus"** is the same with the one described in section 4.2. For instance, if a segment of priority "i" is transmitted, only BWB_CTR_i will increase by one. However, if a TAR=1 bit is seen on the forward channel, the highest priority classes of the station will have the right first to erase the TAR bit and send an *extra request* upstream. That is, a TAR=1 bit may be erased by a lower priority class only if it cannot be erased by a higher priority class. We finally point out that each station, regardless of the number of priority classes that supports, has only one transmission register. Then, by **"segment first in queue"** we mean that a segment has entered the station's (single) transmission register. A non preemptive priority is given to the higher priority classes. In the sequel, we describe the algorithm that decides which segment will become **first in queue.**

If the RG_CTRs of all classes are 0, then the first non flag_segment, in the highest non-empty priority queue inside the station, will enter the transmission register and become the **first in queue.** At the same time the segment will become registered, a *regular request* will be sent upstream, and the values of the corresponding RG_CTR and UNRG_CTR will increase and decrease, respectively, by one. Otherwise, the highest priority segment, for which either RG_CTR is greater than 0 or both RG_CTR and TAR_flag are 0, will become **first in queue.** Although this algorithm provides absolute priority to the highest priority segments of a station in overload conditions, this is not always the case in underload conditions. For instance if TAR_flag_i =1 and a lower priority class "j" (i<j) has its RG_CTR_j >0, then the station will transmit lower priority segments although it may have a high priority segment waiting. Certainly, this will happen until a TAR=1 bit is seen on the forward channel, or the low priority class "j" transmits all of its registered segments. We can decrease the waiting time of priority "i" segment in the following way. If the value of RG_CTR_k of a lower priority class "k" is greater than M_k, then the station can transfer the *extra request,* that class "k" has sent, to the higher priority class "i" and allow class "i" to write on the idle slot that was reserved by the

extra request. Since the TAR=1 bit that created the *extra request* was actually used by class "k" and is now considered to have been used by class "i", the following actions should be taken. Class "i" must set TAR_flag_i to 0, increase its RG_CTR_i by $\min(UNRG\_CTR_i, M_i)$, and decrease its $UNRG\_CTR_i$ by $\min(UNRG\_CTR_i, M_i)$. Class "k" must increase its $UNRG\_CTR_k$ by M_k and decrease its RG_CTR_k by M_k.

4.8.3 Adaptive Bandwidth Balancing over Priority Classes

The objective of this scheme is similar to the objective of the priority mechanism introduced in section 7 of [35], where, in addition to a separate *request* bit, the slot also carries a separate *busy* bit for each priority class. In this way each station knows the priority of the segment carried by the slot. Each class "i" inside a station behaves as a separate substation with its own CD_CTR_i, RQ_CTR_i and BWB parameter M_i. In [35], the RQ_CTR_i of class "i' inside a station counts only the *requests* of same or higher priority which are seen on the reverse channel. Moreover, higher priority classes receive more bandwidth by considering the slots written by lower priority segments as part of the unreserved idle slots they are obliged (by BWB) to pass to the downstream stations. For instance, if the BWB_CTR_i of class "i" at a station becomes equal to M_i and a busy slot with a lower priority segment is seen on the forward channel, class "i" will consider the written slot as equivalent to the unreserved free slot it was obliged to "send" to the downstream stations. Hence, it can reset its BWB_CTR_i to 0 and continue writing on idle slots without actually allowing a "real" idle slot to go downstream. The above operation makes the bandwidth received by higher priority classes to be independent of the lower priority classes, i.e. the bandwidth which is available for balancing among the users of class "i" is the bandwidth left unused by the higher priority classes.

In [35] the above mechanism is implemented by using one gate per class that separates the authorized from the unauthorized segments of the class inside a station. Authorized are the segments of a class that can be transmitted consecutively without allowing an unreserved idle slot to go by. Unauthorized are the remaining segments of

the class. Let AU_i and UN_i be the number of authorized and unauthorized, respectively, segments of class "i" at a station. Every time a busy slot written by a lower priority segment is seen on the forward channel, the gate opens and the number of the authorized segments increases by $INCR_i = MIN\ (M_i, UN_i)$; class "i" can authorize at most as many segments as they were present in its queue. We also point out that $MIN\ (M_i, UN_i)$ segments are authorized when an empty slot is seen on the forward channel, RQ_CTR_i is 0 (i.e. the idle slot is an unreserved slot), and $AU_i = 0$. Notice that class "i" is not allowed to write on this slot.

The corresponding priority mechanism in the case of NSW_BWB requires also by the slot to carry a separate *request* and *busy* bit for each priority class. Again, each class inside a station behaves as a separate substation with its own RQ_CTR_i, CD_CTR_i, RG_CTR_i, $UNRG\_CTR_i$, BWB_CTR_i, and TAR_flag_i. Nevertheless, there is only one TAR bit per slot. We saw that in the case of multiple priority classes, the priority mechanism in [35] enables a higher priority class "i" that sees a lower priority segment on the forward bus to increase its number AU_i of authorized segments by $INCR_i = MIN\ (M_i, UN_i)$. This is equivalent to saying that class "i" will not "send" an unreserved idle slot to the downstream stations at the time this slot was due but will postpone it by the time required to transmit the $INCR_i$ additional segments. However, in the case of NSW_BWB, unreserved idle slots are not allowed to pass to the downstream stations. Instead, upstream stations write on these slots setting the TAR bit to 1. It is now evident that in order for NSW_BWB to behave similarly with the previous BWB_DQDB priority mechanism, every time a lower priority segment is seen on the forward channel, class "i" must postpone sending a TAR=1 bit by the same amount of time.

Class "i" can implement this in a straightforward way by using one additional parameter $M_{i,tr}$. $M_{i,tr}$ describes the number of additional segments that class "i" should transmit before it can send a TAR=1 bit onto the channel. $M_{i,tr}$ is initialized with 0. BWB_CTR_i increases by 1 for every segment that class "i" transmits, and TAR=1 bits are

erased according to the values of M_i and $UNRG\_CTR_i$. Consider now that a busy slot with a lower priority segment is seen on the channel. If $UNRG\_CTR_i = 0$, class "i" will not take any action. If $UNRG\_CTR_i > 0$, then the values of both RG_CTR_i and $M_{i,tr}$ will increase by $INCR_i = MIN\,(M_i, UNRG\_CTR_i)$ and the value of $UNRG\_CTR_i$ will decrease by $INCR_i$; notice the similarity between AU_i and RG_CTR_i as well as between UN_i and $UNRG\_CTR_i$. The objective of class "i" is to delay sending a TAR=1 bit by $INCR_i$ additional segments. A straightforward way of accomplishing that is by freezing the operation of BWB_CTR_i for its next $M_{i,tr}$ transmissions. That is, whenever class "i" transmits a segment and $M_{i,tr}$ is greater than 0, BWB_CTR_i does not increase by one. However, both RG_CTR_i and $M_{i,tr}$ decrease by one. Notice that class "i" must first check its $M_{i,tr}$ to decide whether it should increase BWB_CTR_i and then decrease $M_{i,tr}$ by one. By freezing the operation of BWB_CTR_i, as long as $M_{i,tr} > 0$, the transmission of the $INCR_i$ registered segments becomes transparent to the operation which is related to the erasure and transmission of TAR=1 bits.†

If RG_CTR_i is greater than 0 at the instant the $INCR_i$ segments become registered, then class "i" will have at least one outstanding‡ *request* and the transmission of the $INCR_i$ segments will be guaranteed. This request will be returned to the other registered segments of class "i" when the last of the $INCR_i$ segments is transmitted. However, if $RG\_CTR_i = 0$ then the first of the unregistered segments is a flag_segment and class "i" does not have an outstanding request. Therefore, if a lower priority busy slot is seen on the forward channel and $RG\_CTR_i = 0$, then $M_{i,tr}$ and RG_CTR_i should increase by $INCR_i$, $UNRG\_CTR_i$ should decrease by $INCR_i$, and a *regular request* **should be sent** upstream.

---

† We point out that the segments of class "i" are transmitted in the order of their arrival. Therefore, the $INCR_i$ registered segments will be transmitted after the registered segments that were (possibly) present in the queue when the $INCR_i$ segments became registered. This will not affect the operation since BWB_CTR_i counts transmitted segments only and is not interested in how the segments became registered.

‡ We call a request sent by class "i" "outstanding" when the corresponding slot that has been reserved has not yet arrived at class "i".

In order to clarify the operation of the adaptive priority mechanism we provide now an illustrative example. Consider the case where RG_CTR_i=0, $UNRG\_CTR_i$ =0, BWB_CTR_i =0, $M_{i,tr}$ =0, M_i =2, TAR_flag_i =0, and the queue of class "i" is empty. Let us assume that a priority "i" segment arrives at the station and $UNRG\_CTR_i$ becomes 1. A *regular request* is now sent upstream, RG_CTR_i increases to 1, and $UNRG\_CTR_i$ decreases to 0. Let us assume that before the first segment is transmitted, 13 additional priority "i" segments arrive at the station. At this instant $UNRG\_CTR_i$ =13, RG_CTR_i =1, BWB_CTR_i =0, $M_{i,tr}$ =0, and M_i =2. Consider now that 4 TAR=1 bits are seen on the forward channel. Class "i" will erase all four of them, send 4 *extra requests* upstream, increase RG_CTR_i by 8, and decrease $UNRG\_CTR_i$ by 8. The values of the counters will now be RG_CTR_i =9, $UNRG\_CTR_i$ =5, and BWB_CTR_i=0; and class "i" will have 5 outstanding *requests*. After a while, and before the transmission of any segment by class "i", two lower priority segments are seen on the forward channel. The values of $M_{i,tr}$, RG_CTR_i and $UNRG\_CTR_i$ will first become 2, 11 and 3, respectively, and then 4, 13 and 1. If another lower priority segment is seen by class "i", the value of $M_{i,tr}$ will increase only by $INCR_i = MIN\ (M_i, UNRG\_CTR_i) = MIN\ (2, 1) = 1$, i.e. we will now have $M_{i,tr}$ =5, RG_CTR_i =14, and $UNRG\_CTR_i$ =0. We see that the queue of class "i" has 14 registered segments. One *regular request* has been sent for the first one of them and four *extra requests* for the next 8. The last 5 registered segments are due to the lower priority slots seen on the forward channel. Since the value of $M_{i,tr}$ is 5, BWB_CTR_i will not change during the transmission of the first 5 segments. That means that no TAR=1 bit will be sent by class "i" and that *regular requests* will be sent upstream for all 5 segments. Therefore, after the transmission of the 5 segments, the values of RG_CTR_i and BWB_CTR_i will be 9 and 0, respectively, and class "i" will have 5 outstanding *requests*. That is, class "i" will have the same number of registered segments (9), outstanding *requests* (5), and value of BWB_CTR_i (0), that had before the arrival of the two lower *busy* slots on the forward channel. Since M_i=2, class "i" will transmit 4 TAR=1 bits. We

see that the only effect of the two *busy* slots is to delay the sequence of events, that will enable the transmission of the 4 TAR=1 bits, by the time needed to transmit the 5 additional segments.

Despite the similarities between BWB_DQDB and NSW_BWB the two mechanisms are quite different and, for this reason, the correct operation of NSW_BWB requires that: a) The RQ_CTR_i of class "i" at each station must count both the lower and higher priority *requests* seen on the reverse channel; in the case of BWB_DQDB, high priority classes ignore the lower priority *requests*. b) The value of $M_{i,tr}$ must increase by M_i when both *busy* and *request* bits of lower priority are observed by class "i"; in the case of BWB_DQDB, the lower priority *requests* on the reverse channel do not affect the opening of the gate of class "i". The above modifications enable NSW_BWB to behave in a similar way with BWB_DQDB. More importantly, they ensure that the steady state throughput of each class is independent of its location on the bus. Consider for instance the case where there are only one high and one low priority users in the system and they are overloaded. Assume that the high priority user is downstream. Then the throughput of the upstream low priority user will be equal to the rate at which the high priority user will not send *requests* on the reverse channel. Assume now that the high priority user is upstream. Then the rate at which the low priority user will transmit will be equal to the rate at which it will send *requests* on the reverse channel. It is evident that if the high priority user does not take into account these *requests*, the low priority user will not be able to transmit any segment. In this case the throughputs of the two users will be different than before, and therefore their location on the bus will affect the bandwidth they can acquire.

In the previous example when the high priority user (say of class "i") is downstream, it will increase $M_{i,tr}$ by M_i every time it sees the *busy* bit of a slot that has been written by the upstream lower priority user (say of class "j"). Since the same action should be taken by the high priority user when it is upstream, i.e. increase its $M_{i,tr}$ by M_i

for every lower priority segment transmitted onto the channel, this user must also increase its $M_{i,tr}$ by M_i whenever it sees a lower priority *request* on the reverse channel. This justifies the above difference "b)" of NSW_BWB from BWB_DQDB.

The complete adaptive BWB over priority classes mechanism is now as follows. Each class "i", inside a station, behaves as a separate substation with its own counters, TAR_flag_i, M_i, and $M_{i,tr}$. A separate *busy* and *request* bit is used in the Access Control Field of the slot for each priority class. RQ_CTR_i counts both higher and lower priority *requests*. We now describe the reaction of substation (class) "i" to the various events.

a) Segment arrival: Class "i" behaves as in section 4.2.

b) Segment becomes first in queue: If $M_{i,tr}>0$, class "i" will send a *regular request* upstream. Otherwise ($M_{i,tr}=0$), class "i" will behave as in section 4.2.

c) Segment transmission: If $M_{i,tr}>0$, then both $M_{i,tr}$ and RG_CTR_i will decrease by one. Otherwise ($M_{i,tr}=0$), class "i" will behave as in section 4.2.

d) TAR=1 is seen on forward channel: Class "i" behaves as in section 4.2.

e) A busy bit or a request bit of lower priority is seen by class "i": $M_{i,tr}$ and RG_CTR_i increase by $INCR_i = MIN(M_i, UNRG\_CTR_i)$ and $UNRG\_CTR_i$ decreases by $INCR_i$. Furthermore, if $INCR_i>0$ and RG_CTR_i was 0 before its increase by $INCRi$, class "i" will send a *regular request* upstream.

Similar to the NSW_BWB is the implementation of the adaptive over classes priority mechanism in the case of ITU_NSW. A separate busy bit and request bit per priority class is required in the ACF of every slot. Each class "i", inside a station, behaves as a separate substation with its own counters and registers. RQ_CTR_i counts both higher and lower priority requests. Furthermore, each substation should maintain an extra counter, the Cancelled TAR_segments Counter ($CTAR\_CTR_i$). $CTAR\_CTR_i$ indicates the number of TAR_segments that must be transmitted as regular ones, i.e without setting the TAR bit to 1, due to the presence of busy or request bits of lower priority. Notice that this

counter is not required for the lowest priority. Each time the station sees a busy or request bit of lower priority it should cancel the next TAR_segment scheduled for transmission by increasing the value of $CTAR\_CTR_i$ by one. $CTAR\_CTR_i$ can be increased only when there is at least one TAR_segment which has not yet been condidered, i.e. $NTAR\_R_i$ is greater than $DBTAR\_CTR_i$. Certainly, the value of $NTAR\_R_i$ should take into account the cancelled TAR_segments. Thus, we have that:

$$NTAR\_R_i = \left\lfloor (BWB\_CTR_i + RG\_CTR_i + UNRG\_CTR_i)/M_i \right\rfloor - CTAR\_CTR_i$$

If $CTAR\_CTR_i$ is greater than 0 when a TAR_segment becomes first in queue, a *regular request* will be sent upstream. If $CTAR\_CTR_i$ is greater than 0 when a TAR_segment is transmitted, the TAR_segment will be transmitted as a regular one and CTAR_CTR will decrease by 1.

The substation reaction to the various events is as described in section 4.3 with the following additional operations for the events **b)** and **c):**

b) Segment becomes first in queue: If both $UNRG\_CTR_i$ and $CTAR\_CTR_i$ are greater than 0 and BWB_CTR_i is equal to M_i-1 , a request will be sent on the reverse bus.

c) Segment transmission: If by increasing BWB_CTR_i it becomes equal to M_i and $CTAR\_CTR_i$ is greater than 0, then $CTAR\_CTR_i$ will decrease by 1 and the TAR bit will not be set to 1. Otherwise, $DBTAR\_CTR_i$ will decrease by 1; if it is greater than 0.

Furthermore, the detailed station reaction when a busy or request bit of lower priority is seen on the forward bus is as follows:

e) A lower priority busy or request bit is seen on the channel: If $NTAR\_R_i$ is greater than $DBTAR\_CTR_i$ then $CTAR\_CTR_i$ will increase by 1. Furthermore, if BWB_CTR_i is equal to M_i-1, $DBTAR\_CTR_i$ is equal to 0 and $CTAR\_CTR_i$ was equal to 0, a *request* will be sent on the reverse bus.

Similarly with section 4.8.2, the adaptive over classes priority mechanisms for NSW_BWB and ITU_NSW can be modified to provide the corresponding over stations adaptive mechanisms. For NSW_BWB the same algorithm, with the one in section 4.8.2 can be used to determine which segment should become every time first in queue. In the case of ITU_NSW the use of a separate for each priority LQS_CTR is sufficient for this purpose.

4.8.4 Throughput Analysis in Overload Conditions

In this section we derive analytic estimates of the throughputs of the various stations in the presence of the previous priority mechanisms and under overload conditions. Since, in this case, the behavior of NSW_BWB and ITU_NSW is identical our throughput analysis applies to both mechanisms.

We have arleady seen in the case of the non-adaptive bandwidth balancing mechanisms, equation (4.6.1) can provide the throughput of the various classes when all stations are overloaded. In the particular case of the non-adaptive BWB over stations mechanism, equation (4.6.1) provides the throughput of the highest priority class in each station where as the throughputs of all other classes are 0. We now concentrate on the adaptive mechanism. In this case, the throughput of each class "i" depends on the value $M_{i,tr}$ ($CTAR\_CTR_i$), which is not constant, and therefore equation (4.6.1) cannot be used. In the sequel we present an analysis which can derive very good estimates of the various station throughputs.

In the case of the adaprive BWB over classes, each priority class behaves as a separate station (substation). Since every class can see the total load on the forward bus, through the *busy* and *request* bits of the slots, classes of the same priority will receive the same bandwidth. Let T_i be the throughput of class "i" at a station, N_i the number of stations that transmit class "i" segments, M_i the value of M for class "i", and "n" the number of priority classes in the system; where class "i" is of higher priority than class "j" when

i<j. Since all traffic classes are overloaded, each class at every station will erase all TAR=1 bits that will see. Furthermore, at equilibrium, the number of TAR=1 bits that each class at a station will erase will be equal to the number of TAR=1 bits that will transmit. Therefore, under overload conditions, all priority classes, regardless of station, will see the same number of TAR=1 bits. Consider now a very large time interval T_{ime}. During T_{ime} the lowest priority class at each station will transmit $T_n\, T_{ime}$ segments and send $T_n\, T_{ime} / M_n$ TAR=1 bits; since "n" is the lowest priority class, BWB_CTR_n will never freeze its operation. During the same interval the second lowest priority class at each station will send $T_{n-1}\, T_{ime} / M_{n-1} - N_n\, T_n\, T_{ime}$ TAR=1 bits. The reason is that for each *busy* or *request* bit of priority "n" that priority class "n-1" sees, it will increase its $M_{n-1,tr}$ by M_{n-1}; which is equivalent to cutting back one TAR=1 bit. Following the same approach and keeping in mind that all stations see the same number of TAR=1 bits, the following equation can be derived:

$$\frac{T_n}{M_n} = \frac{T_{n-1}}{M_{n-1}} - N_n\, T_n = \ldots = \frac{T_1}{M_1} - N_2 T_2 - \ldots - N_{n-1} T_{n-1} - N_n\, T_n \qquad (4.8.1)$$

From equation (4.8.1) the throughput T_i of class "i" at each of the N_i stations can be expressed as:

$$T_i = T_n \prod_{j=i}^{n-1} M_j \left[\frac{1}{M_{j+1}} + N_{j+1} \right] \qquad (4.8.2)$$

NSW_BWB does not waste any bandwidth. Therefore, $\sum_{i=1}^{n} N_i\, T_i = 1$ and the following expression for T_i can be derived:

$$T_i = \frac{\prod_{j=i}^{n-1} M_j \left[\frac{1}{M_{j+1}} + N_{j+1} \right]}{N_n + \sum_{k=1}^{n-1} N_k \prod_{j=k}^{n-1} M_j \left[\frac{1}{M_{j+1}} + N_{j+1} \right]} \qquad (4.8.3)$$

In the case of the adaptive BWB over stations mechanism, the highest priority class at each station will acquire all the bandwidth allocated to the station. Its throughput T_i

will then be given by equation (4.8.3) with N_i describing now the number of stations whose class "i" has the highest priority.

4.8.5 Performance of NSW_BWB and ITU_BWB Priority Mechanisms

In this section we investigate and compare the throughput and delay performance of the previous mechanisms. In Figs. 4.17 through 4.22 we examine the convergence of these mechanisms towards the steady state. We consider three stations with inter-station distances D_{12}=38 and D_{23}=40 slots. Station "1" has low priority segments, station "2" has high and medium priority segments, and station "3" has high priority segments. We use the notation "st.i,j" to indicate priority class "j" (j=H,M,L for high, medium and low, respectively) inside station "i" (i=1,2,3). In Fig.4.17 through Fig.4.20 the values $M_h = 6$, $M_m = 4$, and $M_l = 2$ have been assigned to high, medium and low priority classes, respectively. In Fig.4.21 and Fig.4.22 all three classes use the same value of M, i.e. $M_h = M_m = M_l = 4$. Since, the behavior of NSW_BWB and ITU_NSW is identical in overload conditions, in the figures that we consider the throughput performance characteristics we do not distinguish the two schemes and use the generic term NSW to refer to both of them.

In Fig.4.17 we examine the convergence of BWB over priority classes mechanism described in section 4.8.1. The low priority class at station "1" becomes active first and acquires, in the case of NSW, the entire bandwidth. The corresponding throughput, in the case of BWB_DQDB, is .66. Then the high priority class at station "3" becomes active and tries to acquire all the bandwidth. We see that the throughput of station "3" increases, the throughput of station "1" decreases, and after a while both settle to their steady state values which are, for high and low priority respectively, $T_h = .75$ and $T_l = .25$, in the case of NSW, and $T_h = .66$ and $T_l = .22$, in the case of BWB_DQDB; we see again that NSW converges faster. At t=2340, the medium priority class at station "2" becomes active. As a consequence, the throughput of the medium priority class increases and the throughputs of both high and low priority classes decrease and settle to their new steady state values,

which are $T_h = .5$, $T_m = .33$, and $T_l = .17$, in the case of NSW, and $T_h = .46$, $T_m = .31$ and $T_l = .15$, in the case of BWB_DQDB. Finally, at t=4212, the high priority class at station "2" becomes active. We see that the throughputs of all other classes start to decrease and after a while the system reaches a steady state where the high priority classes inside stations "2" and "3" receive the same bandwidth. The throughputs of the high, medium and low priority class are $T_h = .33$, $T_m = .22$ and $T_l = .11$, respectively, in the case of NSW and $T_h = .31$, $T_m = .21$ and $T_l = .10$ in the case of BWB.

In Fig.4.18 we examine the convergence of BWB over stations priority mechanism described in section 4.8.2. The different priority classes become active in the same order as in Fig.4.17. The main difference between Fig.4.18 and Fig.4.17 is that in Fig.4.18 when the high priority class at station "2" becomes active, it shuts off completely the medium priority class inside the same station. In this case the throughputs of the high and low priority classes will be $T_h = .43$ and $T_l = .14$, under NSW, and $T_h = .40$ and $T_l = .13$, under BWB_DQDB.

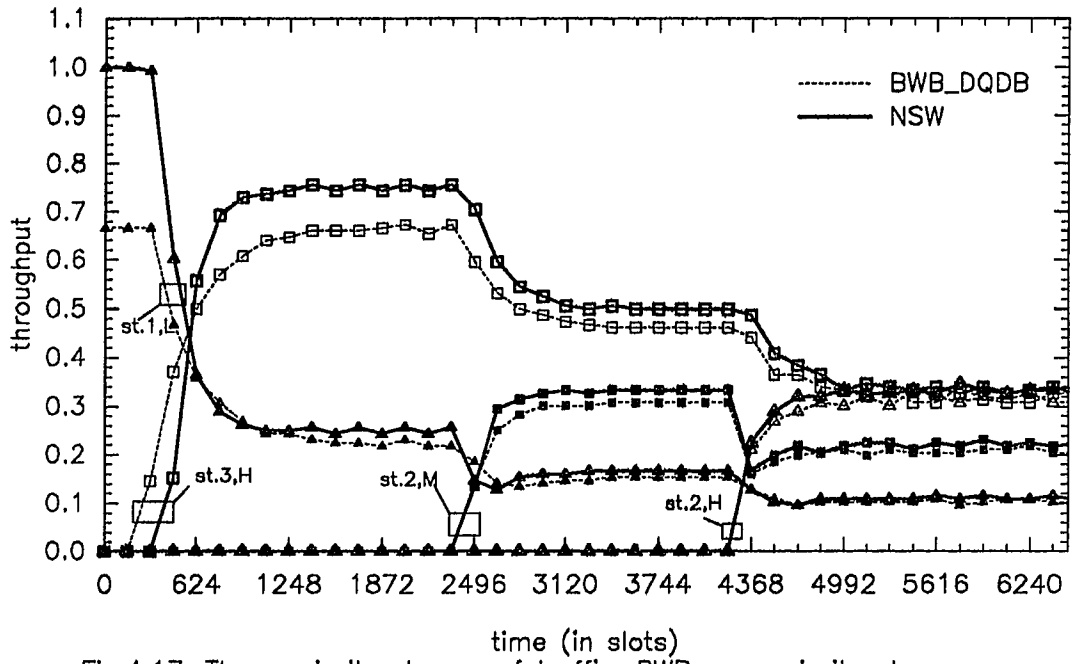

Fig.4.17: Three priority classes of traffic. BWB over priority classes. Comparison of BWB_DQDB and NSW. M_h=6, M_m=4, M_l=2. D_{12} =38 slots. D_{23}=40 slots.

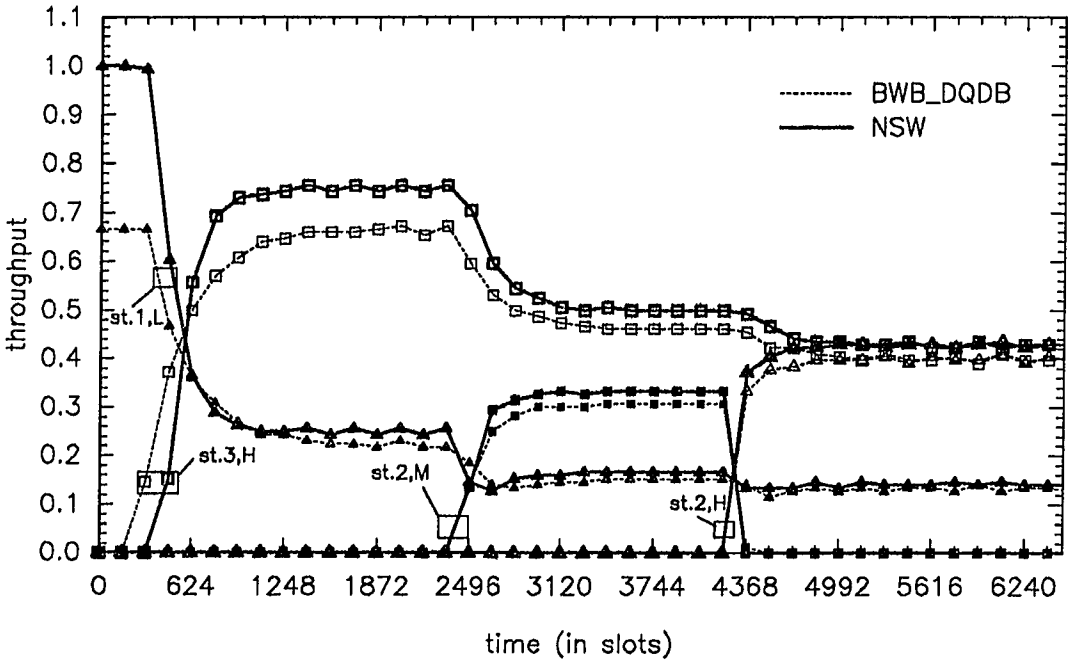

Fig.4.18: Three priority classes of traffic. BWB over stations. Comparison of BWB_DQDB and NSW. M_h=6, M_m=4, M_l=2. D_{12} =38 slots. D_{23}=40 slots.

In Fig.4.19 we examine the convergence of the adaptive BWB over priority classes mechanism. The different priority classes become active in the same order as in Fig.4.17 and Fig.4.18. Initially, the low priority class at station "1" receives (again) all the bandwidth in the case of NSW; and .66 of the bandwidth in the case of BWB_DQDB. At t=312 the high priority class inside station "3" becomes active. We see that its throughput temporarily becomes one, and then settles to its steady state value. T_h and T_l are .9 and .1, under NSW, and .86 and .095, under BWB_DQDB. The reason for this behavior is the following. Initially, the high priority class at station "3" will see a large number of low priority *busy* bits. Consequently, its $M_{i,tr}$ value in the case of NSW, and the number of authorized segments in the case of BWB_DQDB, will temporarily become very large. As a result it will not set any TAR=1 bit for a while and will acquire the entire bandwidth. At t=4992, when the high priority class at station "3" has reached its steady state, the medium priority class at station "2" becomes active. We see that the throughput of the high priority class very slightly is affected and essentially the medium priority class takes its bandwidth from the low priority class; the values of T_h, T_m, and T_l are .86, .12 and .02, under NSW, and .86, .11 and .02, under BWB_DQDB. Finally, the high priority class at station "2" becomes active. The throughputs of all other priority classes start to decrease and after a while the system reaches its steady state where NSW and BWB_DQDB provide almost identical throughputs to the same priority classes. Their values are $T_h = .46$, $T_m = .06$, and $T_l = .01$; in fact the throughputs provided by NSW are slightly higher in the third decimal digit.

We see in Fig.4.19 that the high priority class at station "3", due to the very large number of low priority *busy* bits that initially sees, increases significantly the value of $M_{h,tr}$, for NSW_BWB (and $CTAR\_CTR_h$ for ITU_NSW), temporarily captures all the bandwidth, and delays its convergence to the steady state. If our objective is to provide higher priority classes with as much bandwidth as possible, then such a behavior may be desirable. However, if we are interested in a faster convergence to the steady state, then

we could eliminate the observed throughput overshoot in the following way. We can introduce in the case of NSW_BWB an upper threshold $M_{i,thr}$ on the value of $M_{i,tr}$; which counts the number of segments that have become registered due to lower priority *busy* or *request* bits. That is, if the value of $M_{i,tr}$ becomes equal to or greater than $M_{i,thr}$, then class "i" will not increase the value of $M_{i,tr}$ every time it sees a *busy* or *request* bit of lower priority. In the same way, for ITU_NSW, we can introduce a threshold $C_{i,trh}$ on the value of $CTAR\_CTR_i$ which counts the number of TAR_segments that are cancelled. In order for the two mechanisms to be identical we should have that $M_{i,thr}=M_i C_{i,thr}$. It is evident that thresholds are needed only for the high and medium priority classes.

In the case of BWB_DQDB, the value of AU_i does not say how the class "i" segments have become authorized. Hence, if we would like to apply on BWB_DQDB an upper threshold operation similar to the one of NSW_BWB, we also need (for BWB_DQDB) a parameter $M_{i,tr}$ to provide the number of class "i" segments which became authorized due to lower priority busy slots. The operation of BWB_DQDB must now be enhanced as follows. Whenever a lower priority busy bit is seen on the forward channel the gate can open only if $M_{i,tr} \leq M_{i,thr}$. In this case, $INCR_i = MIN\ (M_i, UN_i)$ additional class "i" segments become authorized and $M_{i,tr}$ increases by $INCR_i$. When an authorized segment is transmitted AU_i should decrease by one. Furthermore, if $M_{i,tr}$ is greater than 0, $M_{i,tr}$ should also decrease by one.

In Fig.4.20 we show the throughput performance of the system of Fig.4.19 when the value of the upper threshold is 312 for NSW_BWB and 52, 78 for the high, medium priority classes, respectively, in the case of ITU_NSW. We have also considered the same threshold (312) for the case of BWB_DQDB. We see that the overshoot observed in Fig.4.19 does not exist anymore and both NSW and BWB_DQDB converge faster to steady state.

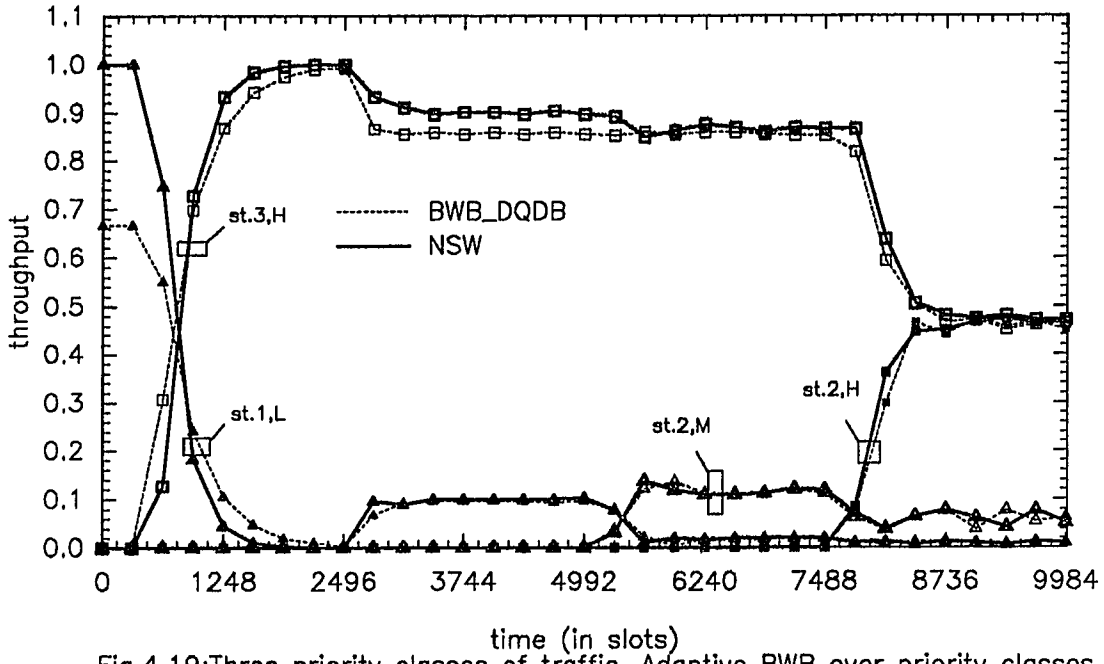

Fig.4.19:Three priority classes of traffic. Adaptive BWB over priority classes. Comparison of BWB_DQDB and NSW. M_h=6, M_m=4, M_l=2. D_{12}=38 slots. D_{23}=40 slots.

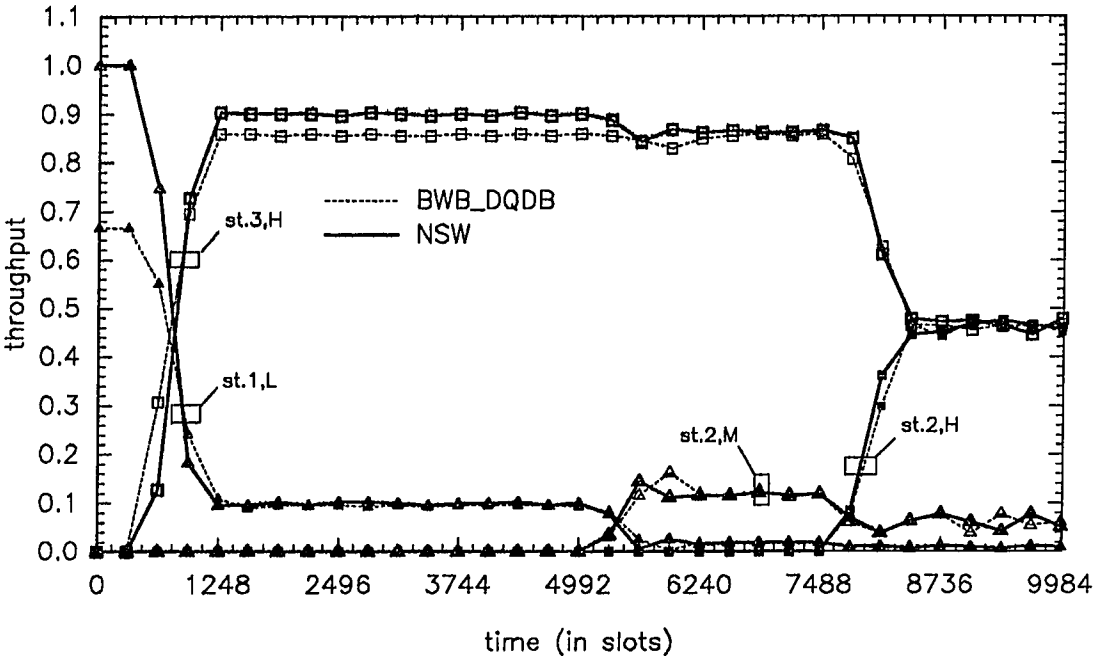

Fig.4.20:Three priority classes of traffic. Adaptive BWB over priority classes with upper thresholds. Comparison of BWB_DQDB and NSW. M_h=6, M_m=4, M_l=2. D_{12}=38 slots. D_{23}=40 slots.

In Fig.4.21 we show that one should be very careful with the selection of upper thresholds because in some cases they may have a negative effect on convergence. We consider the same system of Fig.4.20, where the convergence of the adaptive BWB over priority classes mechanism has been examined, but with $M_h = M_m = M_l = 4$. The same upper threshold has been considered for the high and medium priority classes, i.e., 312 for NSW_BWB and 78 for ITU_NSW.

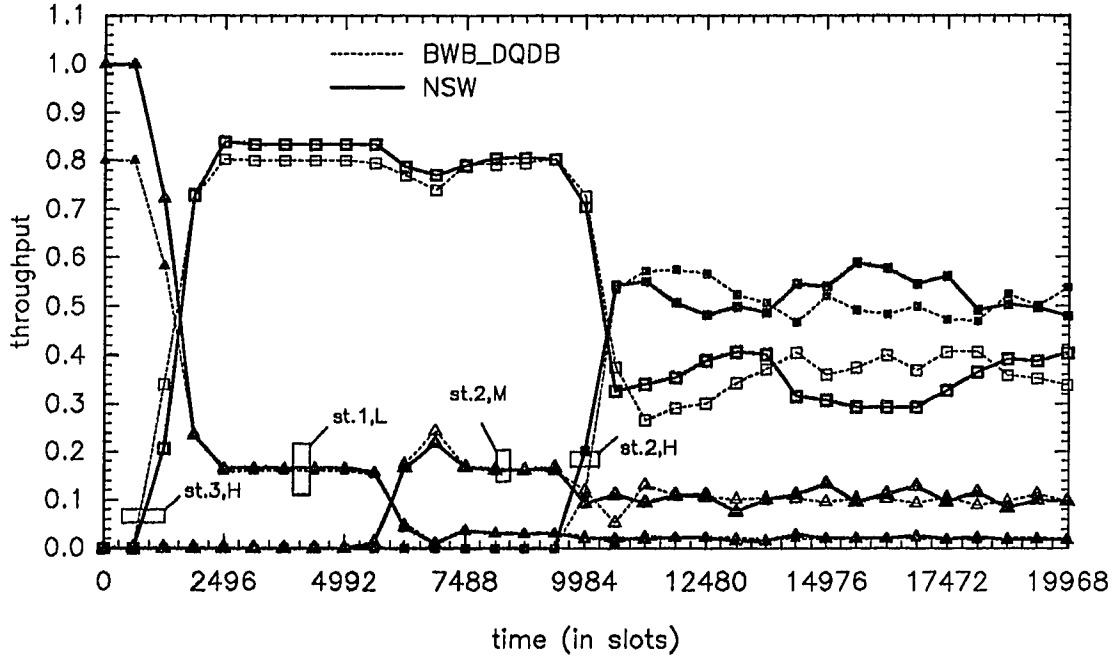

Fig.4.21:Three priority classes of traffic. Adaptive BWB over priority classes with upper thresholds. Comparison of BWB_DQDB and NSW. M_h=4, M_m=4, M_l=4. D_{12}=38 slots. D_{23}=40 slots.

As it has been expected there is no throughput overshoot after the activation of the high priority class inside station "3", at t=624. However, the system very slowly converges to the steady state after the activation of the high priority class inside station "2", at t=9360. In fact the system has not reached steady state at t=19,968. We could speed up the convergence of the system by selecting a higher value for the upper thresholds, i.e. 624. But then a throughput overshoot will start to appear after the activation of the high priority class inside station "3"; simulation results have shown that this overshoot is not as high as it would be if no upper thresholds were present. This behavior motivated us to introduce the idea of adaptive upper thresholds. That is, the value of $M_{i,thr}$ for priority class

"i" does not remain constant but it may change. Since a throughput overshoot may appear when a priority class becomes active, we propose that the value of $M_{i,thr}$ ($C_{i,thr}$) must start with a small initial value $M_{i,init}$ ($C_{i,init}$). That is, when the queue of class "i" becomes empty $M_{i,thr}$ ($C_{i,thr}$) should be reset to $M_{i,init}$ ($C_{i,init}$). When messages start arriving at this queue, class "i" must increase $M_{i,thr}$ ($C_{i,thr}$) by $M_{i,step}$ ($C_{i,step}$) once every round trip propagation delay and until its queue becomes empty. At this instant it will be reset reset to the initial value.

In Fig.4.22 we consider the system of Fig.4.21 and investigate the behavior of BWB_DQDB and NSW_BWB when adaptive upper thresholds are used with $M_{i,init}=76$ ($C_{i,init}=19$) and $M_{i,step}=156$ ($C_{i,step}=39$). We see that the throughput overshoot has

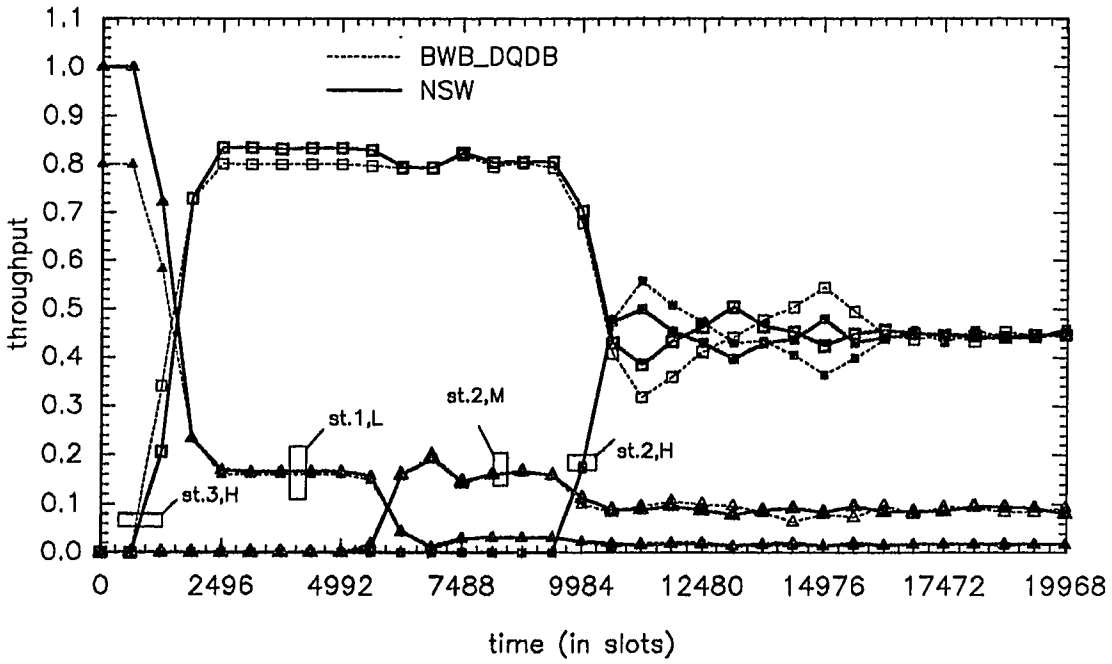

Fig.4.22:Three priority classes of traffic. Adaptive BWB over priority classes with adaptive upper thresholds. Comparison of BWB_DQDB and NSW. M_H=4, M_M=4, M_L=4. D_{12}=38 slots. D_{23}=40 slots.

been eliminated and that the system converges much faster to steady state. In fact the system provides similar throughputs to the high priority classes in stations "2" and "3" much faster than a first glance may indicate. The reason is that the throughput curves of the two stations are intersected. Again, NSW demonstrates a better behavior since the difference

in the values of throughput between "2" and "3" is smaller and their throughput curves are intersected more times. We finally point out that the observed oscillation of the throughput curves is not due to the presence of upper thresholds. Our simulation results have shown that if no upper thresholds are used, then identical will be the behavior of the system after the activation of the high priority class inside station "2", at t=9360. The only difference will be a throughput overshoot that will be observed after the activation of the high priority class inside station "3", at t=624.

We conclude the discussion on the performance results under overload conditions by mentioning that the steady state values of throughput that have been computed by our simulation results, coincide with the analytic estimates derived from equation (4.8.3).

In the remaining figures of this section we carry out a delay comparison of NSW_BWB, ITU_NSW and BWB_DQDB. We consider the same network as in Fig.4.5, i.e. a network consisted of 20 stations with inter-station distance equal to 2 slots. We also assume *linear* loading with a total offered load equal to 0.9. This load is evenly distributed among the three priority classes, i.e. 0.3 per class. The stations transmit messages of constant length equal to 20 segments.

In Figs. 4.23 and 4.24 we compare the average message delay performance of NSW_BWB, ITU_NSW and BWB_DQDB in the case of the non-adaptive BWB over classes mechanism, with M_h=6, M_m=4 and M_l=2 for all schemes. In Fig. 4.23 we compare the average message delays of ITU_NSW and NSW_BWB and in Fig. 4.24 those of ITU_NSW and BWB_DQDB. These figures show that ITU_NSW has the smallest delay variation among the various users of the same priority class. Furthermore, in the case of ITU_NSW the delay of the low priority users is much higher than the delay of the high and medium priority users. However, medium and high priority users experience similar delays. The reason for this behavior is that the values of M_h and M_m do not distinguish drastically the performance of the two priority classes under underload conditions; although under overload conditions high priority users can acquire one and a half more

bandwidth than medium priority users. In the case of NSW_BWB the message delay increases as the station index increases. Moreover, as we approach the end of the bus the delay of the higher priority users becomes worse than the delay of the lower priority users. The reason for this behavior is that in the case of NSW_BWB the higher the value of the BWB parameter, the more unregistered segments the station needs in order to erase a TAR=1 bit and send an *extra request*. Thus, low priority users insert more *extra requests* than medium and high priority users and their average delay is significantly lower. Finally, Fig.4.24 shows that in the case of BWB_DQDB the stations encounter higher average delays, due to the slots that are being wasted.

In Figs. 4.25 and 4.26 we consider the same schemes as in Figs. 4.23 and 4.24 respectively, and compare their delay performance in the case of the adaptive BWB over classes priority mechanism, with $M_h=M_m=M_l=2$ for NSW and $M_h=M_m=M_l=8$ for BWB_DQDB. Also in this case, ITU_NSW has the best performance out of all three schemes. The unfairness of NSW_BWB observed in Fig.4.23 is now itensified since the value of the BWB parameter increases every time a lower priority transmission is observed by the station. Finally, the delay variation of BWB_DQDB is also increased in this case because of the increase of the BWB parameter.

Figs. 4.23 through 4.26 clearly show that although the priority mechanisms for ITU_NSW and NSW_BWB distribute the available bandwidth identicaly, they have a quite different performance when underload conditions are considered. ITU_NSW outperforms both NSW_BWB and BWB_DQDB schemes. However, in all three mechanisms certain lower priority users have better performance characteristics than other higher priority users; depending on their location on the bus.

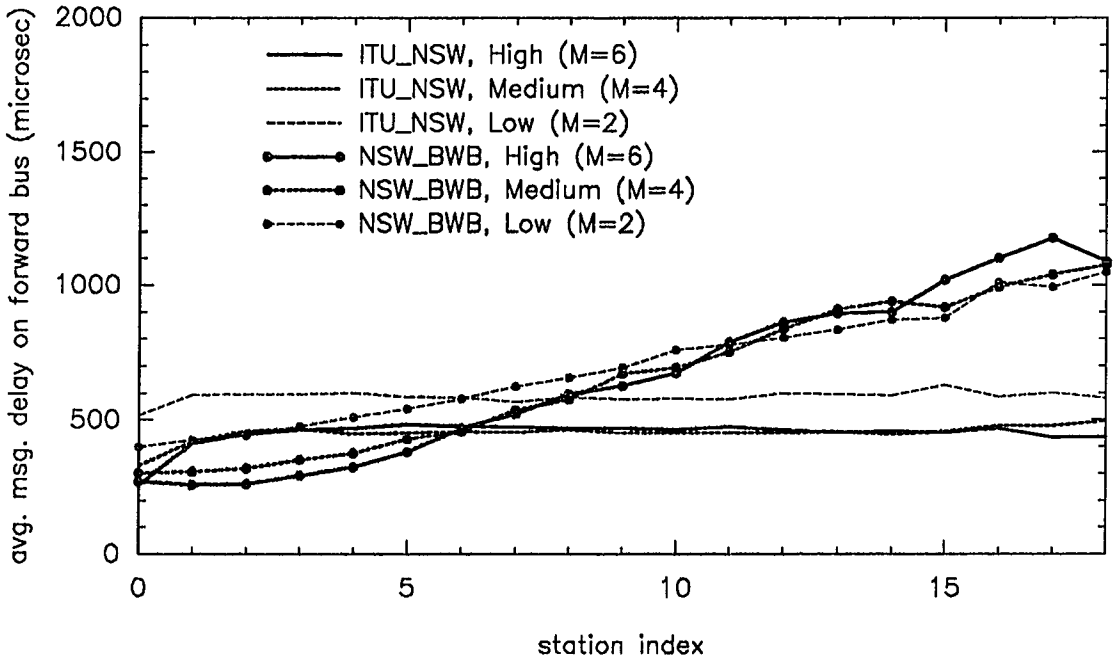

Fig.4.23:Delay comparison of NSW_BWB and ITU_NSW. Offered load per class 0.3. Total bus utilization 0.9. Constant message size of 20 segments. BWB over classes.

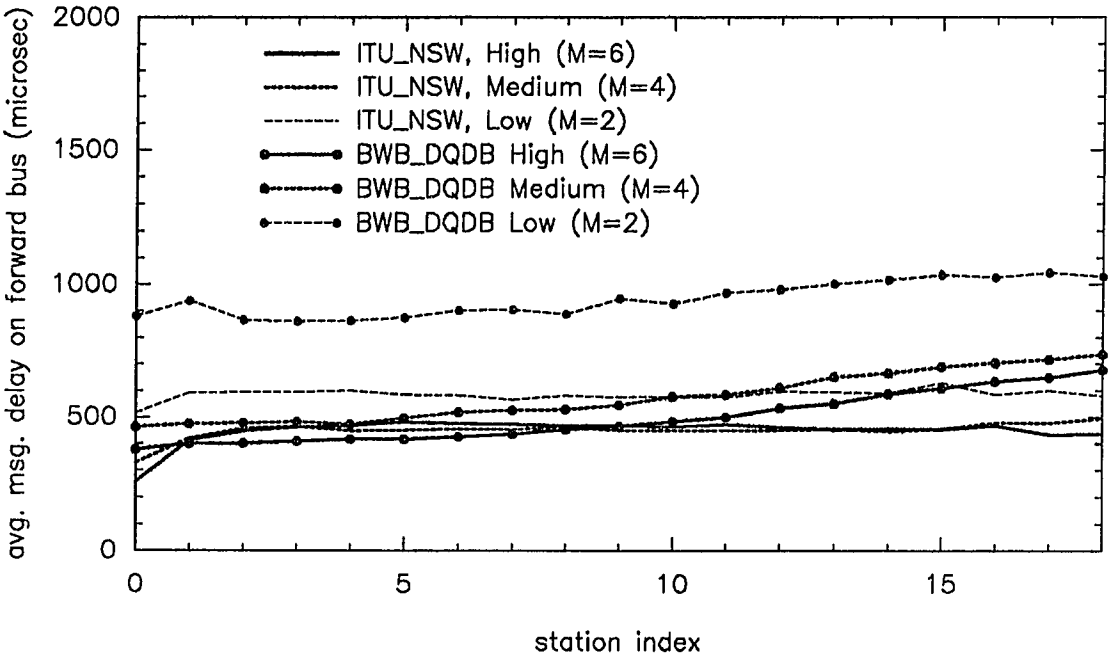

Fig.4.24:Delay comparison of ITU_NSW and BWB_DQDB. Offered load per class 0.3. Total bus utilization 0.9. Constant message size of 20 segments. BWB over classes.

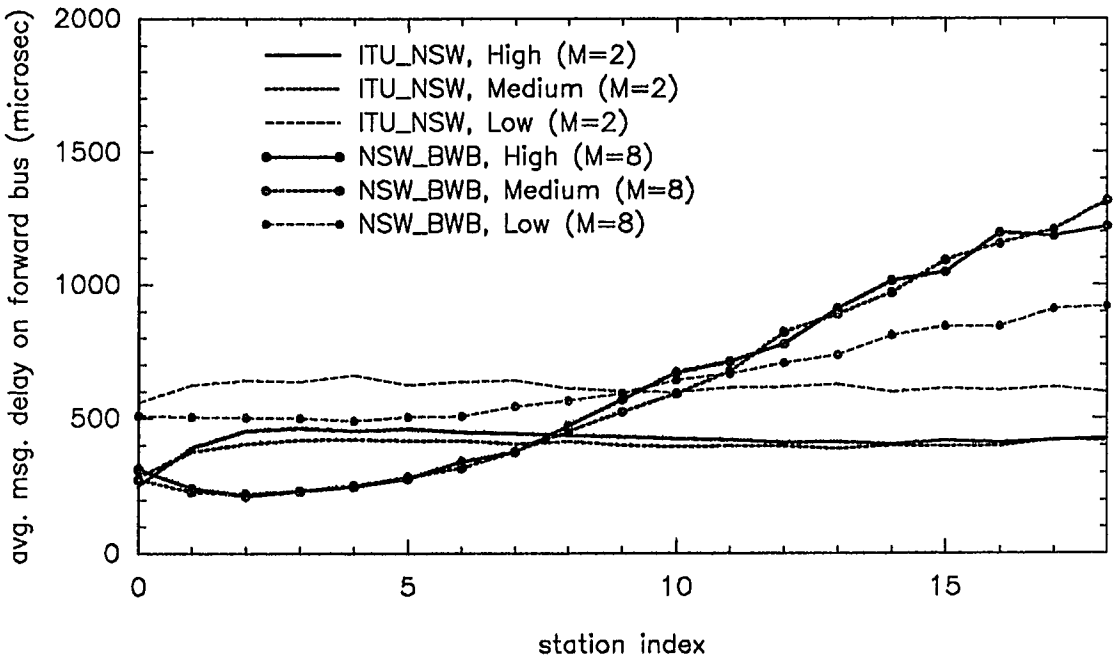

Fig.4.25:Delay comparison of NSW_BWB and ITU_NSW. Offered load per class 0.3. Total bus utilization 0.9. Constant message size of 20 segments. Adaptive BWB over classes.

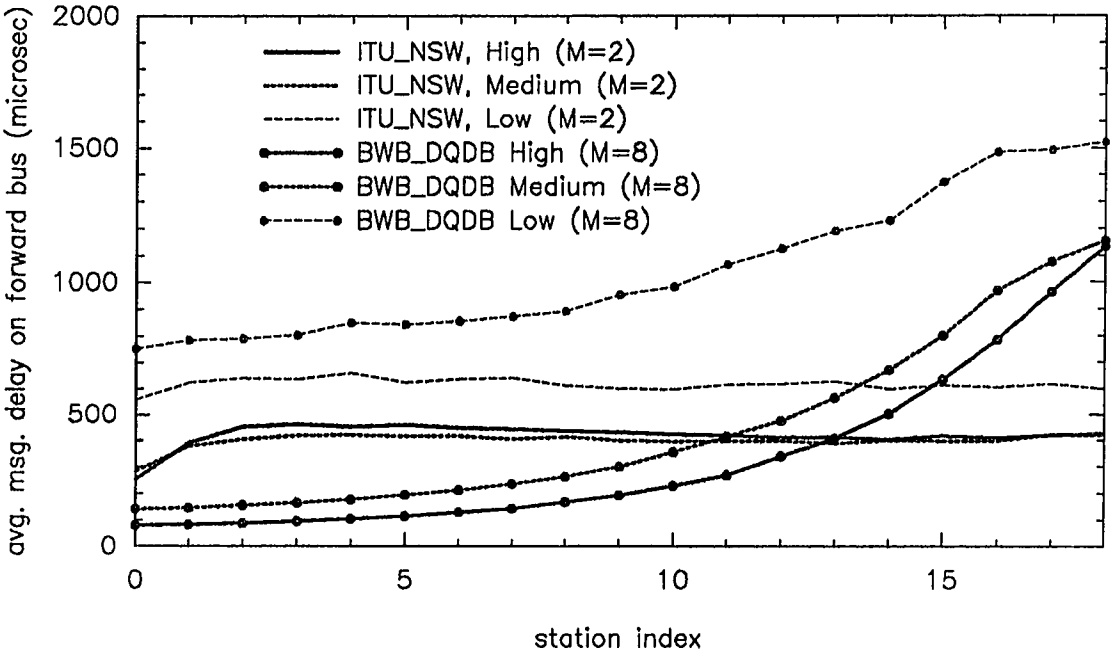

Fig.4.26:Delay comparison of ITU_NSW and BWB_DQDB. Offered load per class 0.3. Total bus utilization 0.9. Constant message size of 20 segments. Adaptive BWB over classes.

4.9 Conclusions

In this chapter we have introduced two variations of a new BWB mechanism for dual bus architectures, the NSW_BWB and ITU_NSW. Their operation requires the use of one additional control bit in the ACF of the slot and achieves bandwidth balancing by enabling downstream stations to send additional *requests*, instead of requiring upstream stations to allow free slots to pass by. In this way no slots are wasted, smaller values of the parameter M can be used, and the system can converge faster to the steady state than the current BWB mechanism of DQDB. We have investigated the performance of NSW_BWB and ITU_NSW under one traffic class and have verified their faster convergence, fair bandwidth allocation, and ability to distribute the channel bandwidth in any arbitrary way among the network users. We have also examined their delay behavior and we have found that both NSW_BWB and ITU_NSW outperform BWB_DQDB. Moreover, ITU_NSW can provide lower delays to the downstream stations by allowing them to insert their requests on the reverse bus much earlier than in the case of NSW_BWB. We have also provided a queueing analytic model that can encapture the behavior of NSW_BWB or ITU_NSW under independent segment transmission.

We have also examined the ability of the introduced schemes to support multi-priority traffic. We have found that NSW_BWB and ITU_NSW can meet the objectives of the BWB_DQDB priority mechanisms proposed and investigated in [35]. That is, they can guarantee a minimum bandwidth per station or priority class, and can enable higher priority classes to receive more bandwidth by virtually modifying the rate at which they can send TAR=1 bits. In this last case we have observed that higher priority classes may temporarily shut off lower priority classes completely and delay the convergence of the system to steady state. We have therefore investigated ways that can eliminate the observed throughput overshoot and decrease the time required by the system to reach steady state. Finally, we have investigated the delay performance of the various priority schemes and we have shown that ITU_NSW is the most effective, both under the adap-

tive and the non-adaptive bandwidth balancing priority mechanisms, providing the smallest delay variation among the users of the same priority.

CHAPTER 5

AN EFFECTIVE PRIORITY MECHANISM FOR THE SUPPORT OF TIME CRITICAL TRAFFIC

5.1 Introduction

In chapter 4 we have investigated the performance of various BWB priority mechanisms. We have shown that the non-adaptive BWB over classes priority mechanism can provide arbitrary bandwidth distribution among the different priority classes by simply selecting the appropriate value of M^p for each class "p". The adaptive BWB over classes mechanism enables higher priority users to dynamically decrease the rate at which they send TAR=1 bits and acquire in this way the most of the channel bandwidth. However in the case of underload conditions, the station location continues to affect the delays of the various priority users, especially in the case of NSW_BWB. Furthermore, both the adaptive and non-adaptive priority mechanisms in order to provide higher priority classes with more bandwidth, in essence increase their values of M and thus significantly reduce their convergence speed towards the steady state. Thus, although these priority mechanisms are very effective in distributing the channel bandwidth in overload conditions, they cannot react very fast to changes of the offered load. That is, a temporary overload of a low priority class, may significantly affect the delays of a high priority class. In the case that the high priority traffic is generated by a real time application it is possible that the corresponding packets may not satisfy their delay requirements. For this reason, in this chapter we present a new priority mechanism which can provide almost absolute priority to higher priority users within an end_to_end propagation delay time interval. As a result, overloads of lower priority traffic do not affect the performance of higher priority traffic. We apply the new mechanism on both ITU_NSW and NSW_BWB (P_ITU_NSW and P_NSW_BWB) and investigate its effect on their delay performance. Furthermore, we show how it can be applied also in the case of BWB_DQDB (P_BWB_DQDB). Finally,

we investigate its capacity to support voice and video traffic and compare its performance with the adaptive BWB over classes priority scheme.

The organization of the rest of chapter 5 is as follows. In section 5.2 we introduce the new priority mechanism and show how it can be applied in the case of ITU_NSW and NSW_BWB. In section 5.3 we show its implementation in the case of BWB_DQDB. In section 5.4 we use simulation to investigate the performance of P_ITU_NSW, P_NSW_BWB and P_BWB_DQDB as well as the adaptive over classes BWB priority mechanism for DQDB. In section 5.5 we examine the capacity of P_ITU_NSW to support voice and video traffic. In section 5.6 we derive analytic estimates for the maximum number of voice and video sources that can be supported by the P_ITU_NSW network. In section 5.7 we present our conclusions.

5.2 The P_ITU_NSW and P_NSW_BWB Priority Mechanisms

The objective of the proposed here priority mechanisms is to enable higher priority classes to acquire the requested bandwidth as fast as possible by "shutting off" the access of the lower priority classes. The key idea relies on the interpretation given to the request bits in the ACF of the slots. In the DQDB standard, as well as, the overwhelming majority of the proposed priority schemes [7,35,39], a priority "i" request bit serves as a request for an empty slot from the upstream users of equal or lower priority. That is, if a user of priority "j" sees a request of priority higher than or equal to "j" on the reverse bus it must allow an extra free slot to pass by. In the case of the proposed mechanism request bits of a certain priority are count only by the users of the same priority; not by the lower priority users. In order for a priority "i" user to request a free slot from the upstream stations, it has to insert a request bit for every priority "j" less than or equal to "i". The advantage of this approach is that it allows a station to send lower priority requests earlier and have faster access to the bus. According to the proposed scheme, every time a message of priority "i", consisted of k segments arrives at a station, the station inserts k

requests for each priority lower than "i". In this way, the lower priority classes will allow k additional slots to pass by and the presence of lower priority traffic will have a minimal effect on the delay of the priority "i" message. Furthermore, the priority "i" requests are inserted according to the NSW protocol used, i.e ITU_NSW or NSW_BWB. In this way the available bandwidth to each priority classes can be evenly distributed among its users. Since priorities are introduced by sending different requests for each priority class, we can use M=2 for all classes and achieve the fastest possible throughput convergence among the users of the same priority. In the sequel, we describe in detail the P_ITU_NSW mechanism.

Each priority class "i" inside a station behaves as a separate substation with its own RQ_CTR_i, BWB parameter M^i, BWB_CTR_i, RG_CTR_i, $UNRG\_CTR_i$ and $DBTAR\_CTR_i$. Furthermore, each slot carries a separate request bit, RB_i, and TAR bit, TAR_i for each priority class. Each substation takes into account only the request bits of the same priority, i.e. the substation of priority "i" increments its RQ_CTR_i by one for every RB_i =1 seen on the reverse bus, and reacts only to the TAR_i=1 bits. The substation reaction to the events "segment arrival", "segment becomes first in queue" , "segment transmission" and "TAR=1 is seen on the forward bus" remains identical with the station reaction to these events described in section 4.3. In addition, when a segment of priority "i" arrives at a substation, a request for each priority less than "i" will be sent upstream. In the case of a message arrival, the substation sends a request for each segment of the message.

P_ITU_NSW aims to preempt the transmission of lower priority traffic whenever there is higher priority present onto the network. For this reason, the P_ITU_NSW does not use any CD_CTRs. This means that a substation of priority "i" may transmit its segment if and only if its RQ_CTR_i is 0. Thus, incoming requests of priority "i" have preemptive priority over the local segments. A stream of request bits of priority "i", seen by a substation of the same priority "i", will suspend the transmission of the locally

queued segments at least for a number of slots equal to the length of the request stream. Notice, however, that the absence of the CD_CTR cannot lead a substation to starvation.

Similar to P_ITU_NSW is the implementation of the new priority mechanism in the case of NSW_BWB. That is, a separate TAR=1 bit for each priority class is required in the ACF of the slot, each class inside a station has its own counters, there is no CD_CTR and each class counts only the requests of its own priority. The reaction of each user to the various events is the one described in section 4.2. In addition when a segment of priority "i" arrives at the station, a request for each priority less than "i" will be sent upstream.

The 802.6 IEEE standard supports three priority classes and provides two unused bits for future specification. If P_ITU_NSW is to support three priority classes of traffic, it will require three TAR bits in the ACF of the slot, i.e. one more than what the standard provides. We now describe an equivalent implementation for the new priority mechanism that can use the current slot format of the 802.6 network. We implement the three TAR bit action by using only two bits in the following way. The bit combinations 11, 10 and 01 correspond to a high, medium and low TAR=1 bit respectively. The 00 bit combination represents the case where no TAR=1 bit is carried by the slot. The problem, when using two bits to implement the operation of the three TAR bits, is that the station must first read the two bits and then decide whether to erase them. Otherwise, a lower priority class may reset a TAR=1 bit of a higher priority class. However, if we would like to be in consistency with the standard, the station should be able to modify the TAR=1 bits on the fly, i.e. before it has read them. In this case we can use the following approach. We allow class "i" to reset the two bits before it has read them. If the two bits were 00, the class will not take any action. If the two bits were carrying a TAR=1 bit of priority "i" an extra request is send upstream. If the two bits were carrying a TAR=1 bit of some other priority, class "i" would have to set it back in the next slot. Notice that in this last case every time class "i" inserts a TAR=1 bit on the next slot, this slot maybe

carrying some other priority TAR=1 bit, which class "i" will then have to insert in the subsequent slot. This procedure is repeated until a 00 combination or a TAR=1 of priority "i" is observed. In the last case, the class "i" will send an extra request upstream.

5.3 The P_BWB_DQDB Priority mechanism

In this section we show how the new priority mechanism can be applied in the case of BWB_DQDB. In P_ITU_NSW (or P_NSW_BWB) a user of priority "i" transmits m segments in m slots. Consequently, it is sufficient for this user to request m slots from the upstream users of lower priority. However, this is not the case for P_BWB_DQDB. In order for the "i" priority user to transmit m segments, it requires $m+m/M^i$ slots; m are written by the station and m/M^i slots are forced to pass by empty. Thus, it is not sufficient for the "i" priority user to send only m lower priority requests upstream. On the other hand, if it sends $m+m/M^i$ requests, it may send significantly more requests. For instance, consider three active users. The most upstream is of low priority and the other two of high priority. Assume that $M^h=2$ and that each high priority user has 10,000 segments queued for transmission. Then, the two high priority users may transmit their segments in 25,000 slots; 20% of the bandwidth is wasted. If each station inserted $m+m/M^i$ requests of lower priority, 5,000 extra slots would have been wasted. For this reason, we introduce in the ACF of the slot a separate bit for each priority class "i", except the lowest one , the Extra Request Bit (ERB_i). This bit is used to carry the extra requests that higher priority users have to send upstream.

We now describe the complete P_BWB_DQDB scheme. Each priority class "i" inside a station is a separate substation with its own RQ_CTR_i. Furthermore, each slot carries a separate request bit for each priority class and the Extra Request Bit (ERB_i) for each priority "i", but the lowest one. Each substation increases its RQ_CTR_i for every RB_i =1 or ERB_j =1 seen on the reverse channel, where "j" is a priority higher than "i". Also, each substation "i" artificially increases its RQ_CTR_i by 1 for every M^i segments

it transmits. Upon the arrival of a segment of priority "i" to the substation a request is sent upstream for each priority less than "i". When a segment becomes first in queue the substation inserts a request of the same priority "i" on the reverse channel. Furthermore, higher priorities request from the upstream lower priorities the idle bandwidth required for their operation. This is accomplished through the ERB_i bit and an extra counter ER_CTR_i kept inside each substation of priority "i". ER_CTR_i increases by one every time an ERB_i is seen on the reverse bus. Every time RQ_CTR_i is artificially increased by 1 and ER_CTR_i =0 the substation sets the next ERB_i to one. However, if ER_CTR_i is greater than 0, the substation will not send an extra request upstream (i.e. set ERB_i to one), but it will decrease ER_CTR_i by one. Finally, whenever the substation's queue becomes empty, ER_CTR_i is reset to 0.

In the case of P_BWB_DQDB higher priorities converge faster to the steady state where fairness is achieved than in the case of the adaptive BWB over classes mechanism. This is because high priority users do not have to increase their BWB parameter in order to receive more bandwidth. Nevertheless, P_BWB_DQDB does not allow lower priority classes to use the idle bandwidth of higher priority classes, unless no higher priority user is located downstream. For example, if we consider two high and two low priority substations, in alternating sequence (H-L-H-L), with $M^l=M^h = 8$, then each high priority user will receive 8/17 of the channel capacity and the most downstream low priority user will receive the remaining 1/17 of the channel capacity. This minor throughput unfairness of the lower priority classes is not important, when the performance superiority of the P_BWB_DQDB against the adaptive over classes BWB_DQDB is considered.

In the next section we investigate the delay characteristics of the P_ITU_NSW and P_BWB_DQDB priority mechanisms and compare them with the adaptive BWB over classes scheme.

5.4 Delay Performance

In this section we carry out a delay comparison of P_ITU_NSW, adaptive over classes BWB_DQDB (from now on called BWB_DQDB), and P_BWB_DQDB. We have assumed a network consisted of 20 stations, with 2 slots interstation distance, which can support three priority classes of traffic. We compare the delays encountered by the stations under different message sizes and load configurations. Since, our main motivation for introducing the new priority mechanism was to provide higher priorities not only with low average delay but also with small delay variation we consider in most of the cases the 95th percentiles of the delay, i.e. the minimum value for which 95% of the transmitted messages have message delay less than this value.

In Fig.5.1 we compare the performance of P_ITU_NSW, BWB_DQDB and P_BWB_DQDB when the total bus utilization is 0.9; offered load per priority class 0.3. We have considered message transmissions with constant message size equal to 20 segments. Fig.5.1 shows that the P_ITU_NSW priority scheme is definitely superior than the other two mechanisms. The average message delay experienced by the stations in the case of P_ITU_NSW is significantly lower than that of BWB_DQDB or P_BWB_DQDB. What is more interesting is that the delay variation among the stations in the case of P_ITU_NSW is very small. BWB_DQDB demonstrates the greater delay variation among different users of the same priority level. The great potential of the proposed priority mechanism is clearly shown by the performance of P_BWB_DQDB, which is a substantial improvement over BWB_DQDB.

One performance metric which cannot be shown when average delays are drawn is the degree of deviation from the average value of the message delay. For this reason, we have plotted the 95th percentiles of the stations' delay. Figs. 5.2-5.4 show the average segment delay and the 95th percentiles for P_ITU_NSW, BWB_DQDB and P_BWB_DQDB respectively, for the load configuration of Fig.5.1. Again it is easy to see that P_ITU_NSW is a better scheme than both BWB_DQDB and P_BWB_DQDB. It is

also interesting to mention here that in the case of P_ITU_NSW, the 95th percentiles of all stations of higher priority are less than the average message delays of lower priority. This means that at least 95% of the messages of a higher priority experience lower delays than the average delay of the messages of a lower priority. However, this is not the case

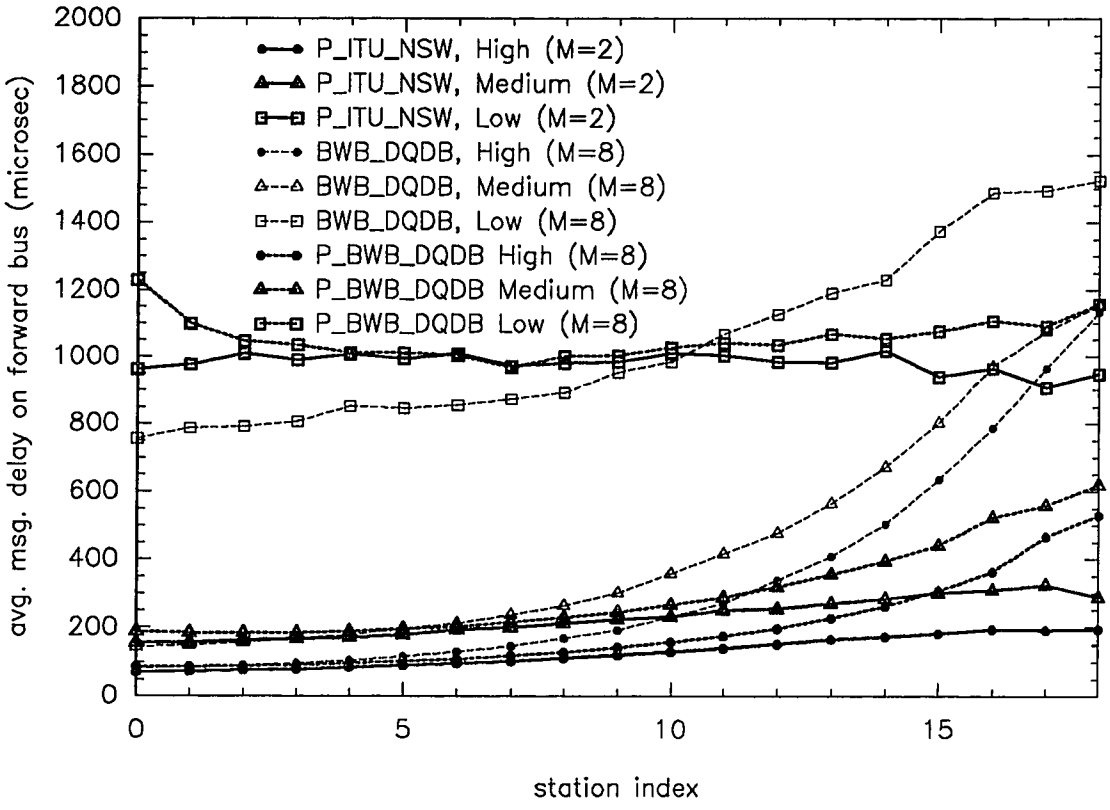

Fig.5.1: Delay comparison of P_ITU_NSW, BWB_DQDB and P_BWB_DQDB. Offered load per class 0.3. Total bus utilization 0.9. Constatnt message size of 20 segments.

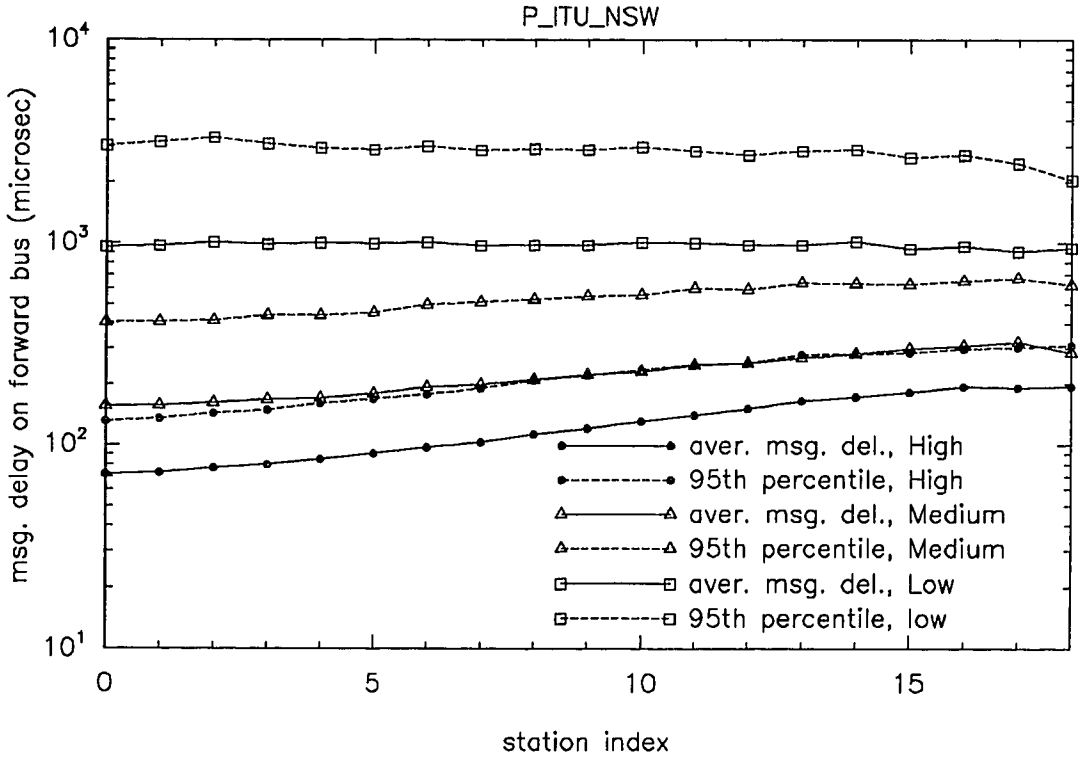

Fig.5.2: Average message delay and 95th percentiles for P_ITU_NSW.
Offered load per class 0.3. Total bus utilization 0.9.
Constant message size of 20 segments.

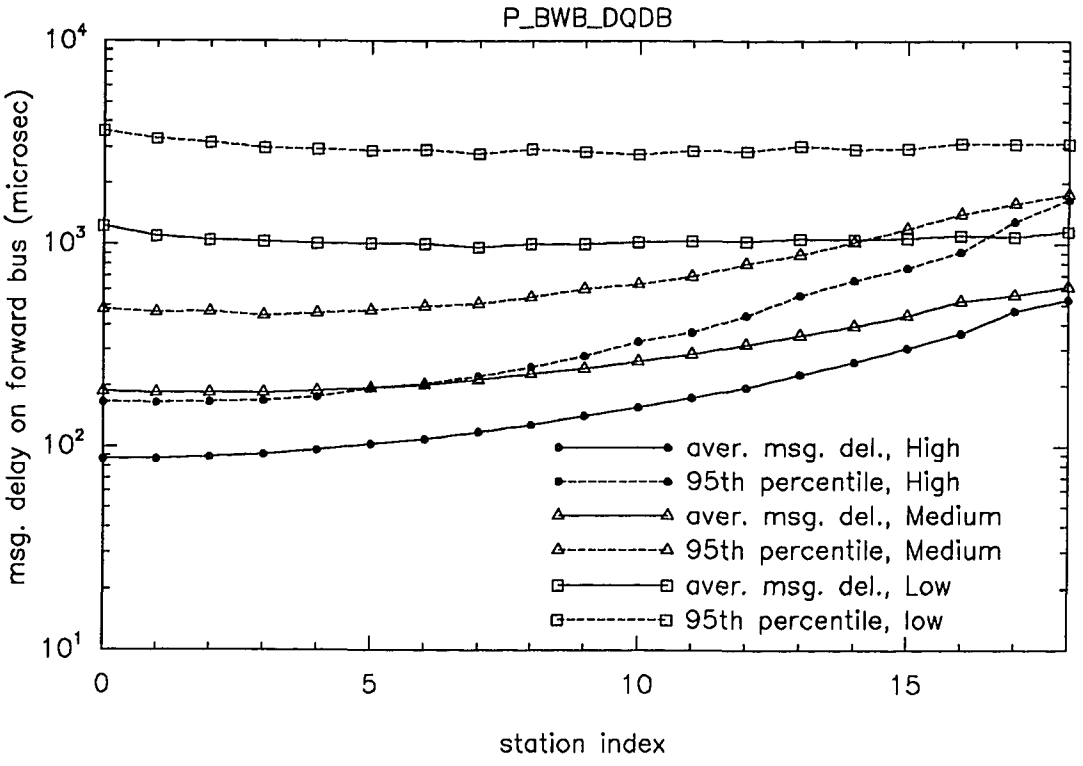

Fig.5.3: Average message delay and 95th percentiles for P_BWB_DQDB. Offered load per class 0.3. Total bus utilization 0.9. Constant message size of 20 segments.

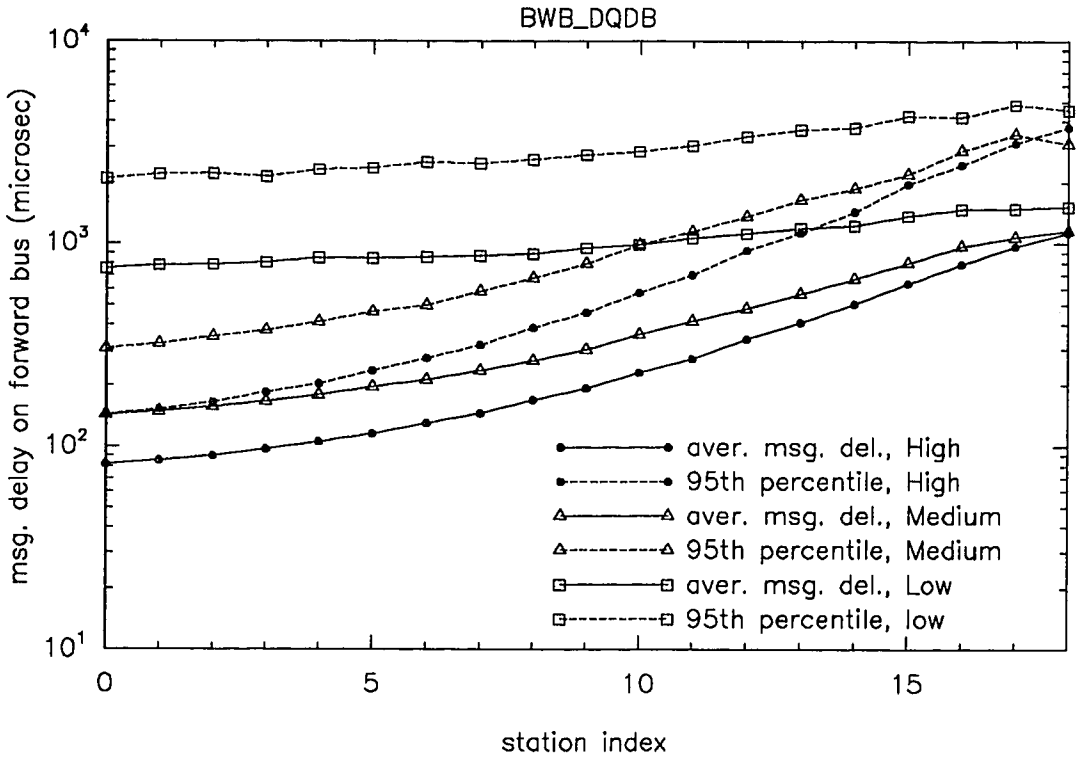

Fig.5.4: Average message delay and 95th percentiles for BWB_DQDB.
Offered load per class 0.3. Total bus utilization 0.9.
Constant message size of 20 segments.

with the other two schemes. We have also examined the performance of the NSW_BWB scheme under the absolute priority mechanism and have found that its delay characteristics are very similar with those of P_ITU_NSW. However, in the case of P_ITU_NSW the delay variation among the same priority users is slightly lower than in the case of P_NSW_BWB. For this reason we focus only on the preformance of P_ITU_NSW. Furthermore, we have also investigated the performance of P_ITU_NSW in the case where two bits are used to implement the action of the three TAR bits and have found that both implementations have identical delay characteristics.

In Figs. 5.5 through 5.8 we have considered a different load configuration. We have assumed that all high and medium priority users have an aggregate load of 0.6, i.e. 0.3 per class (linearly distributed among the stations), and that there is only one active low priority user with offered load 0.3. We have positioned the low priority user at the very beginning of the bus, inside station "0", since we are interested in showing how the three priority schemes behave under unfavorable load configurations. Finally, we have considered constant message size of 20 segments. In Fig. 5.5 we compare the average delay characteristics of P_ITU_NSW, BWB_DQDB and P_BWB_DQDB. In Figs. 5.6-5.8 we show the average and the 95th percentiles of the delay for the three priority schemes. The main conclusion from this set of figures is that even under unfavorable load configurations for the high and medium priority users P_ITU_NSW demonstrates an excellent behavior providing the lowest delays and minimizing both the delay variation and the effect of the station location on the performance. Also in the case of P_BWB_DQDB the effect of the station location on the performance is not significant. Its higher delays than those of P_ITU_NSW are due to the slots that this mechanism wastes.

In the last set of figures (5.9 through 5.12) we compare the average and 95th percentiles of the delay of the three schemes under the previous configuration. The only difference in this case is that we consider the transmission of long messages, consisted of 100 segments. P_ITU_NSW demonstrates almost a perfect behavior with both the aver-

age and 95th percentiles of the delay characteristics being straight lines. In contrast, BWB_DQDB exhibits the highest delay variation. We finally point out that in all figures 5.6 through 5.12 lower delays are encountered by both the medium and high priority users inside station "0". This is because they are located inside the same station with the active low priority user, and thus they can request immediately their slots by the low priority user, whenever they have a message queued for transmission.

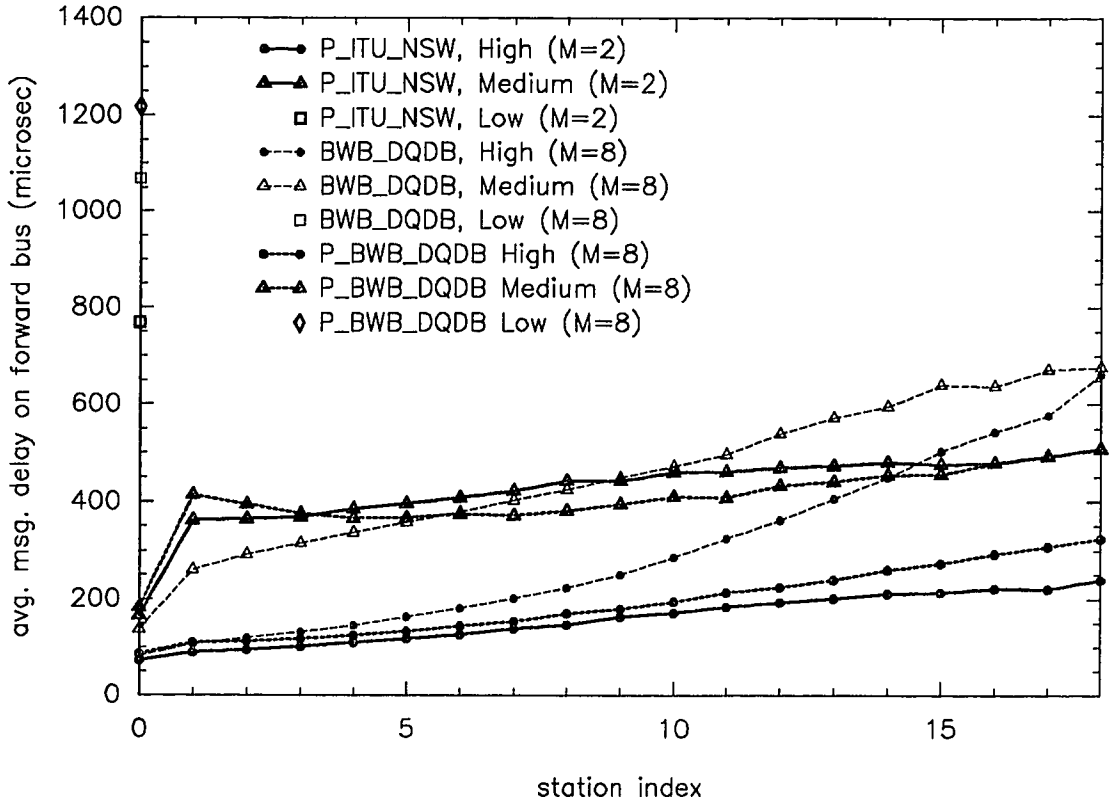

Fig.5.5: Delay comparison of P_ITU_NSW, BWB_DQDB and P_BWB_DQDB. Offered load for high and medium priority classes 0.3. Low priority only in station "0" with load 0.3. Total offered load 0.9. Constant message size of 20 segments.

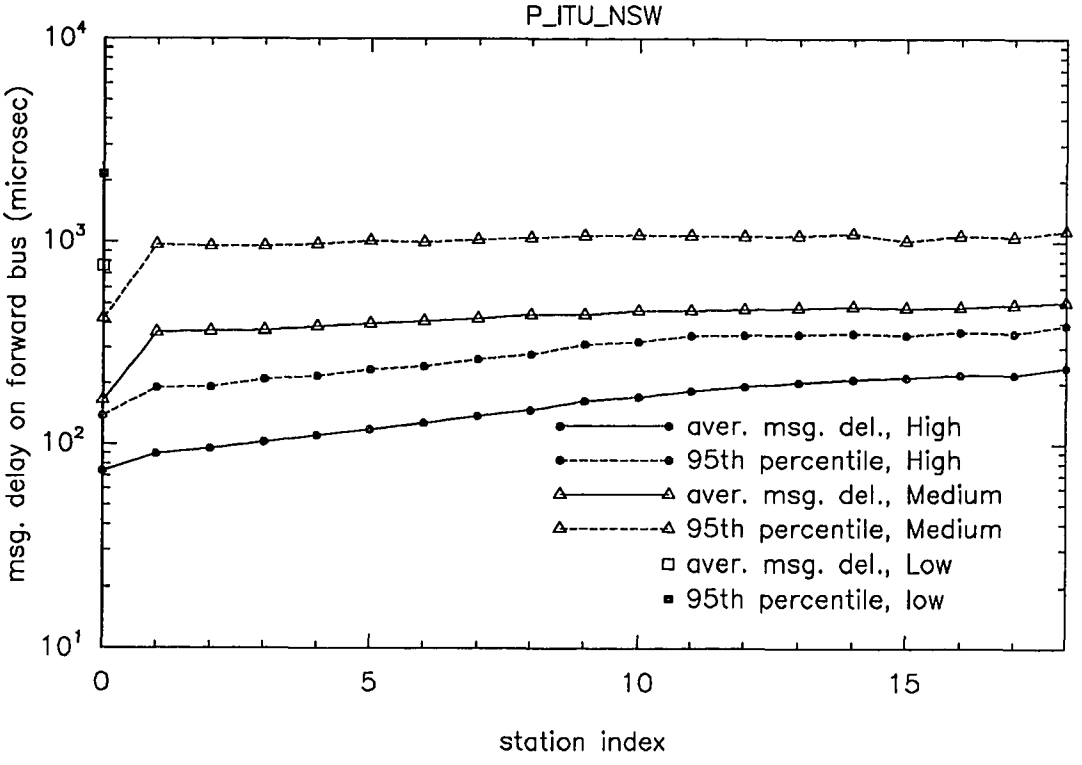

Fig.5.6: Average message delay and 95th percentiles for P_ITU_NSW. Offered load for high and medium priority classes 0.3. Low priority only in station "0" with load 0.3. Total offered load 0.9. Constant message size of 20 segments.

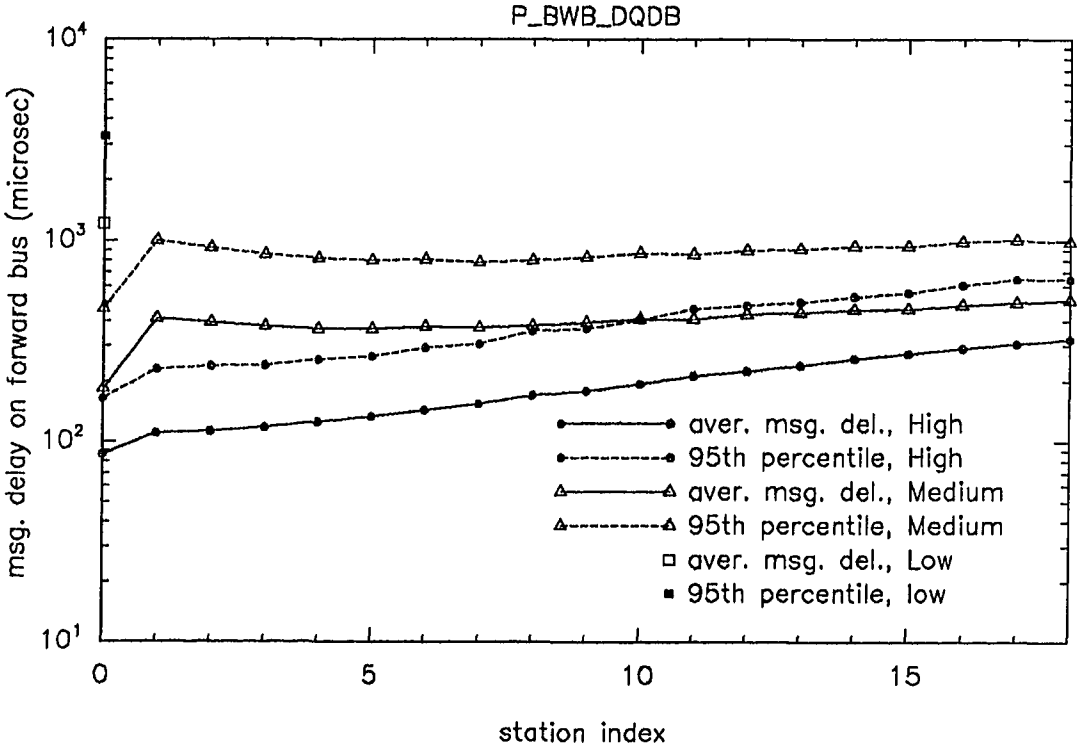

Fig.5.7: Average message delay and 95th percentiles for P_BWB_DQDB. Offered load for high and medium priority classes 0.3. Low priority only in station "0" with load 0.3. Total offered load 0.9. Constant message size of 20 segments.

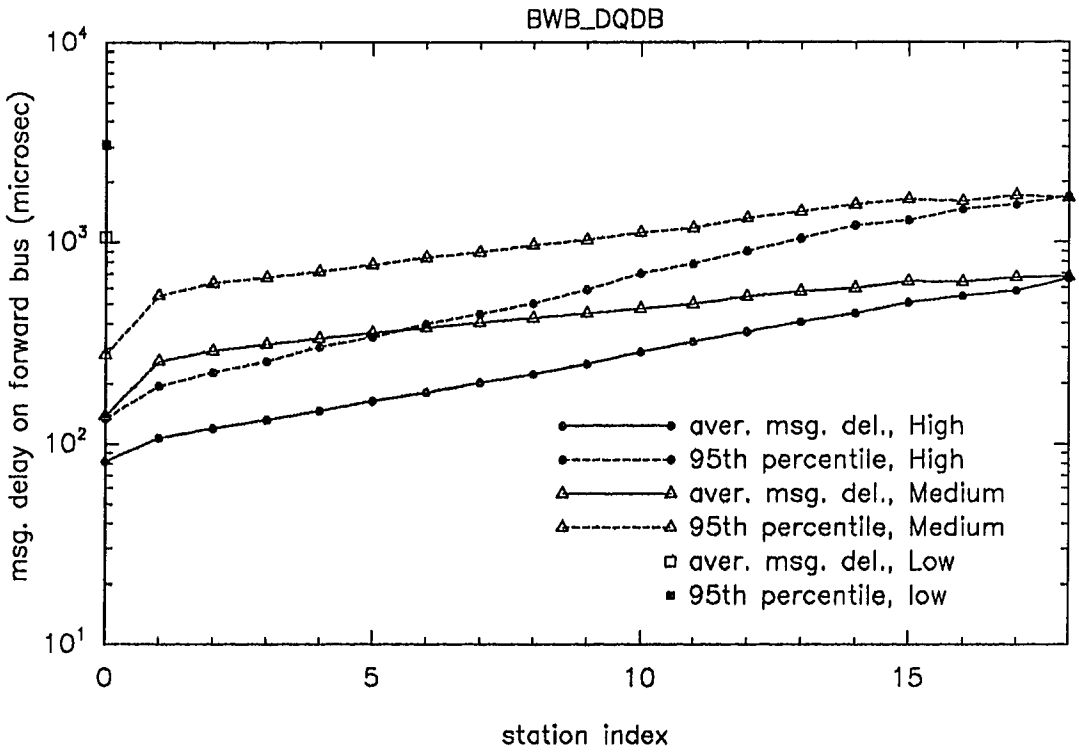

Fig.5.8: Average message delay and 95th percentiles for BWB_DQDB. Offered load for high and medium priority classes 0.3. Low priority only in station "0" with load 0.3. Total offered load 0.9. Constant message size of 20 segments.

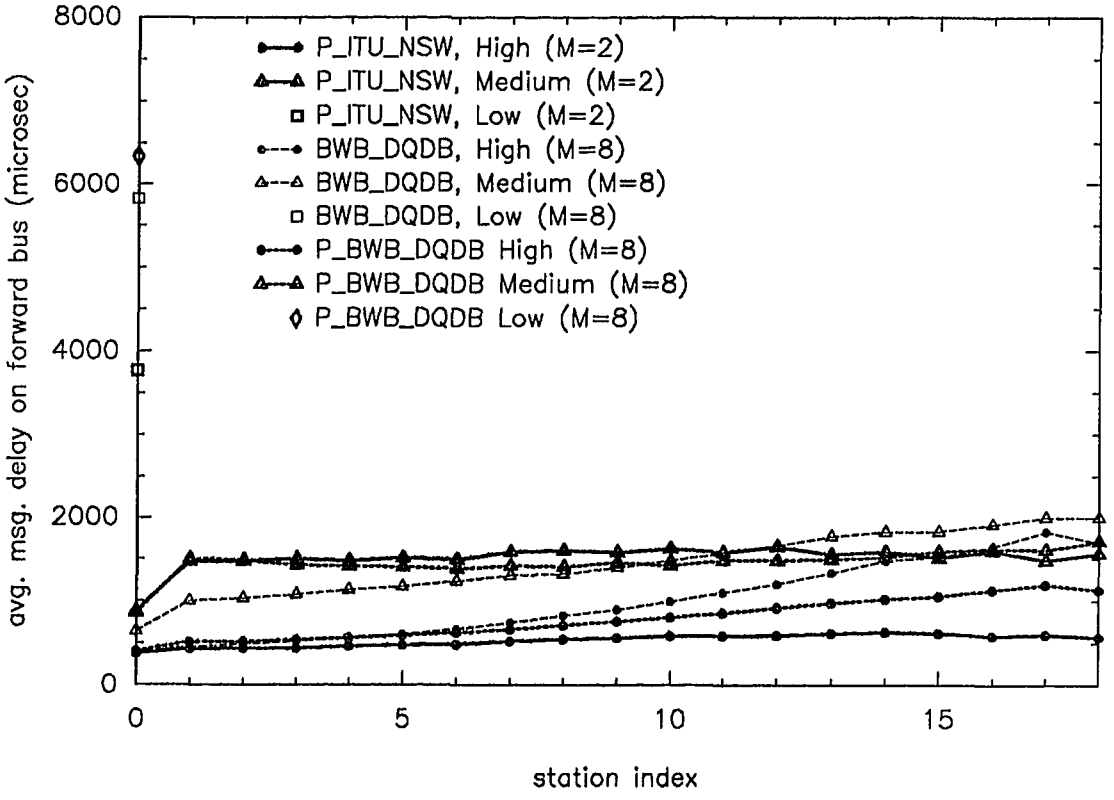

Fig.5.9: Delay comparison of P_ITU_NSW, BWB_DQDB and P_BWB_DQDB. Offered load for high and medium priority classes 0.3. Low priority only in station "0" with load 0.3. Total offered load 0.9. Constant message size of 100 segments.

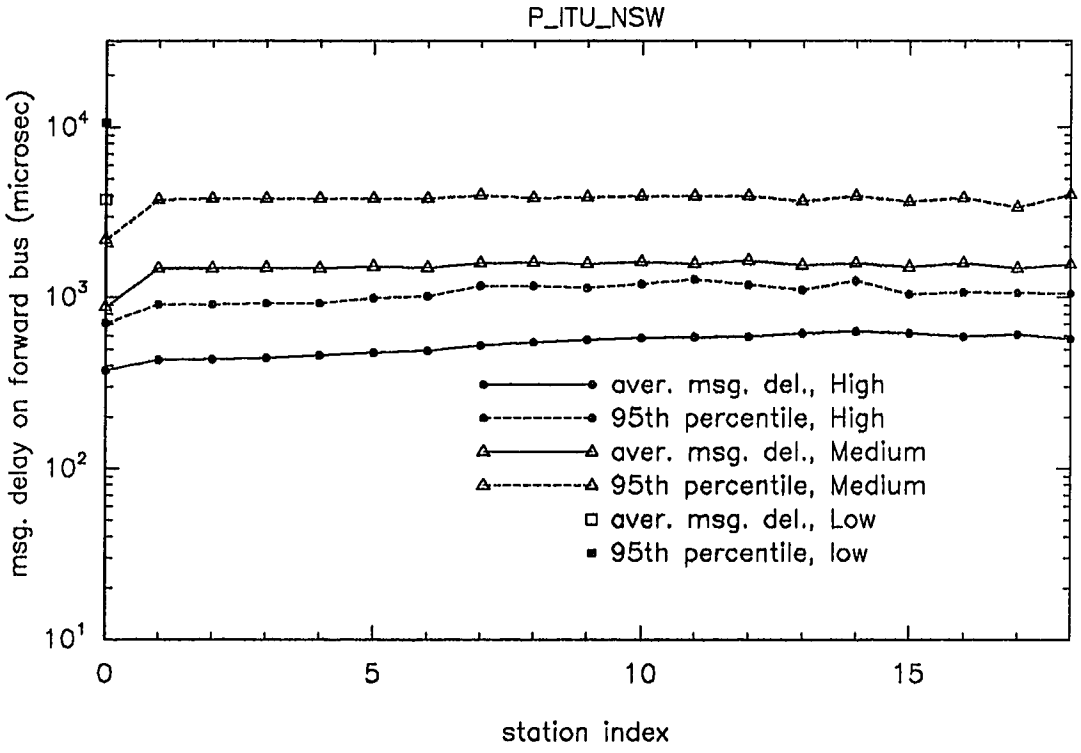

Fig.5.10:Average message delay and 95th percentiles for P_ITU_NSW. Offered load for high and medium priority classes 0.3. Low priority only in station "0" with load 0.3. Total offered load 0.9. Constant message size of 100 segments.

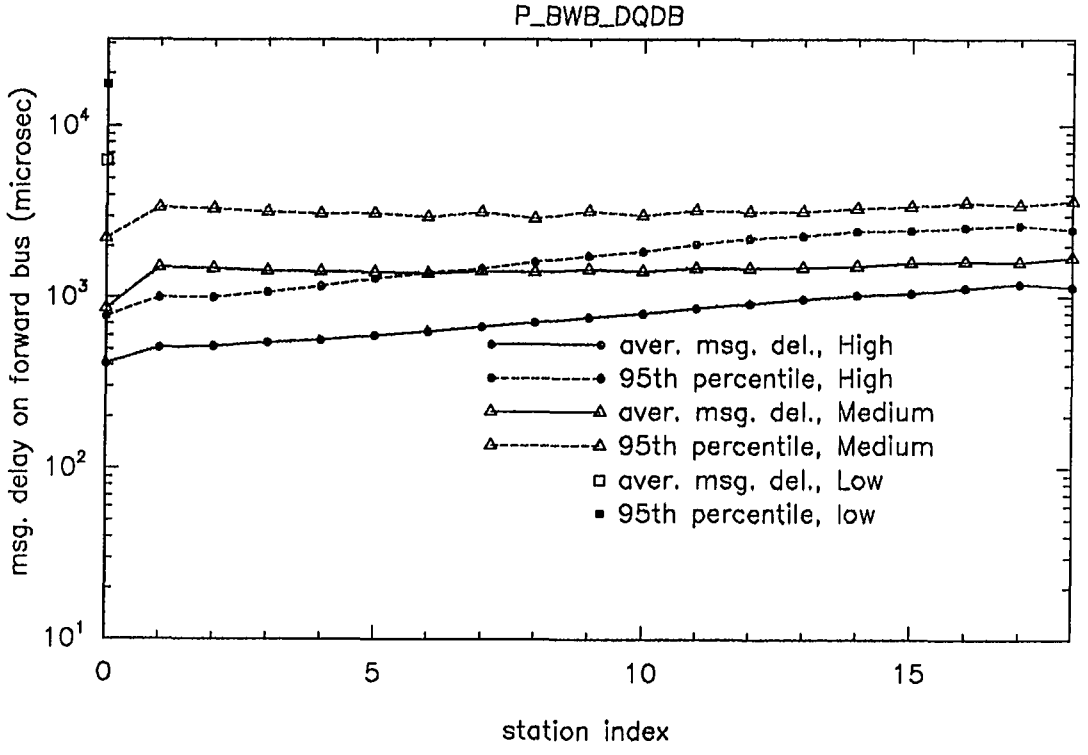

Fig.5.11:Average message delay and 95th percentiles for P_BWB_DQDB. Offered load for high and medium priority classes 0.3. Low priority only in station "0" with load 0.3. Total offered load 0.9. Constant message size of 100 segments.

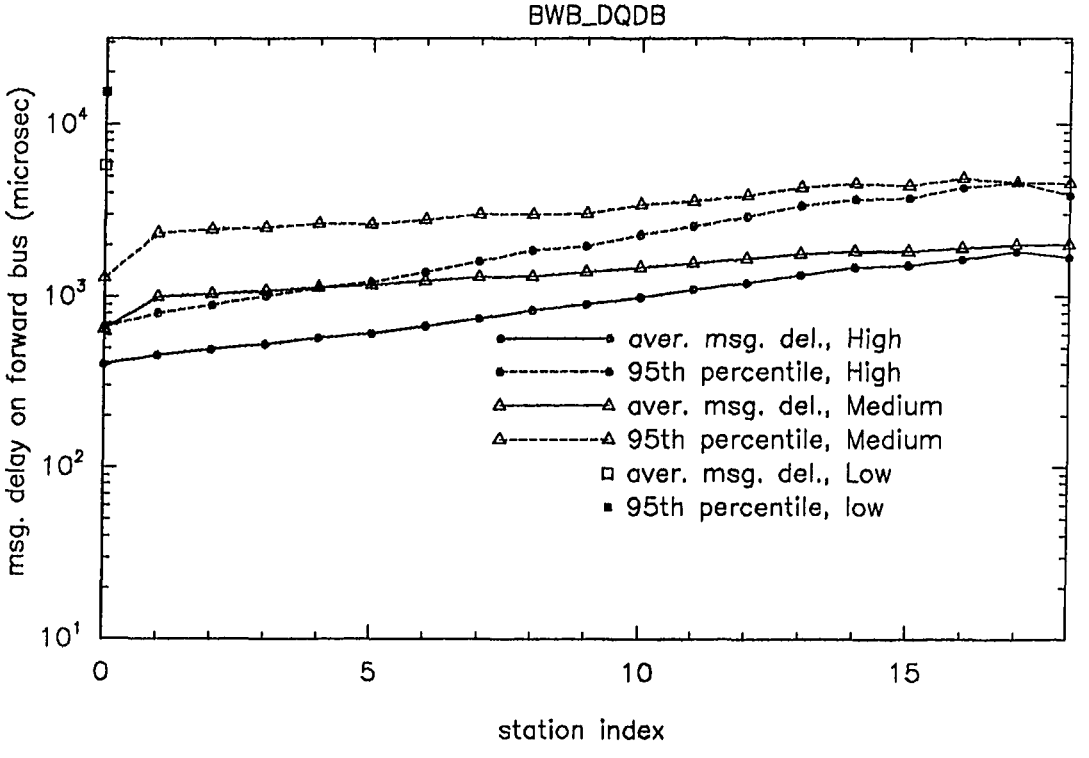

Fig.5.12:Average message delay and 95th percentiles for BWB_DQDB. Offered load for high and medium priority classes 0.3. Low priority only in station "0" with load 0.3. Total offered load 0.9. Constant message size of 100 segments.

5.5 Voice/Video Performance

In this section we investigate the capacity of the different priority mechanisms to support time critical traffic, that is, voice and video. We first describe the traffic load models that we have adopted in order to emulate the generation of voice and video traffic. Then, we compare the performance of P_ITU_NSW, BWB_DQDB and P_BWB_DQDB under the presence of voice for various load configurations. We demonstrate the effectiveness of P_ITU_NSW and show that it can effectively support significantly more voice sources than the other two schemes. Finally, we carry out a thorough performance investigation of the P_ITU_NSW under the presence of both voice and video sources.

Voice Process Model

A great amount of research activity has been devoted in modeling a speech signal pattern [77,78,79] generated by a voice source. A typical speech pattern consists of alternating talk spurt and silence periods. We have assumed that talk spurts and a silence periods are exponentially distributed with mean 1.5 sec and 2.25 sec respectively. Moreover, only the talk spurts are packetized and transmitted through the network. We have also assumed that the packetization interval P_v is constant. If R is the encoding rate of the voice coder and d is the data field of the slot, then we have that P_v = d/R. For the DQDB network the data field of the packet is 44 bytes (=352 bits). In the case that PCM is used, then R=64Kbps and the packetization interval P_v is 5.5 msec. Finally, if a station supports more than one voice sources, then they are multiplexed and transmitted in first-in-first-out order. The delay constraint that the voice packets should meet in our system is the following. Each voice packet must be transmitted before the next one from the same voice source is generated, i.e. within 5.5 msec. Otherwise, the old packet is overwritten by the newly generated one and considered blocked or clipped. A very important measure of the system's performance is the percentage of clipped packets (% of clipping). Due to the redundancy of the voice signal a 0.5% to 1.5% of voice packet loss does not cause any noticable distortion of the voice signal at the destination.

Video Process Model

Video signals, like voice signals, carry a considerable amount of redundant information. For this reason, the investigation of coding algorithms which can reduce the amount of video bits that must be transmitted over the channel is a very interesting and active research area. Here, we have assumed that the interframe coding scheme which has been modeled in [80] is used for the compression of the video signal. Also in this case, we have considered that the different video sources which are generated at the same station are multiplexed and transmitted at a first-in-first-out order. Each video source generates 30 frames per second and the video frame size is 500x500 pixels. The average number of bits per pixel for the n_{th} frame is modeled as a first-order autoregressive Markov Process described by the relation:

$$\lambda(n) = \alpha\,\lambda(n-1) + \beta\, w(n) \tag{5.5.1}$$

where $w(n)$ is a sequence of independent Gaussian random variables with mean w and variance 1. The steady state distribution of $\lambda(n)$ is Gaussian with mean $E[\lambda]$ and variance σ^2 given by:

$$E[\lambda]=\frac{\beta}{1-\alpha}w \quad \text{and} \quad \sigma^2=\frac{\beta^2}{1-\alpha^2} \tag{5.5.2}$$

In order to match the measured data used in [80], $\alpha = 0.8781$, $\beta = 0.1108$ and w=0.572. We mention here that if the number of bits that are generated by a frame is less than zero, they are clipped to zero. Furthermore, when the number of bits that are generated by a video frame is greater that d (=352), the frame is broken up into multiple segments.

Performance Comparison

In Fig. 5.13 we compare the performance of P_ITU_NSW, P_BWB_DQDB and BWB_DQDB under voice transmission. We have assumed the same network configuration as in the previous figures, i.e. a network consisted of 20 stations with interstation distance of 2 slots, and the same values for the M parameter, i.e. M=2 for P_ITU_NSW

and M=8 for P_BWB_DQDB and BWB_DQDB. Furthermore, we have considered two classes of traffic, data and voice, with voice placed at higher priority than data. Finally, all data users are overloaded and 245 voice sources are present into each station but the last one.† This load configuration produces an aggregate load of voice segments equal to 0.92. In Fig 5.13 we have plotted the average segment delay of the video segments. We see that P_ITU_NSW and P_BWB_DQDB have significantly lower average segment delays as well as delay variation than BWB_DQDB. In contrast, in the case of

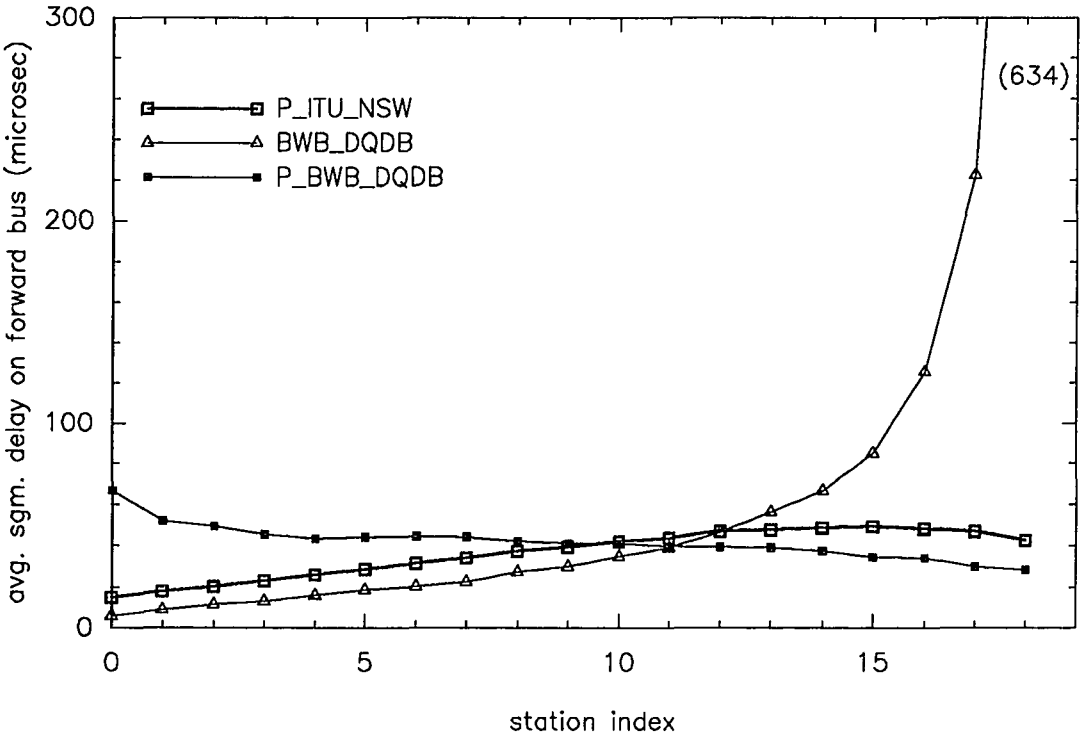

Fig.5.13: Delay comparison of P_ITU_NSW, BWB_DQDB and P_BWB_DQDB. 245 voice sources per station. Overloaded data users.

BWB_DQDB, as we move towards the end of the bus, the average delay increases drastically. Thus, in the case of BWB_DQDB the voice sources located at the downstream stations have a greater probability of being clipped. In fact our simulations have shown that for this load configuration no voice segment is clipped for both P_ITU_NSW and

† The reason for considering overloaded data users is because we are interested to investigate the voice performance under pessimistic conditions.

P_BWB_DQDB. However, in the case of BWB_DQDB a considerable percentage of voice segments, generated into the last four stations are clipped. The maximum percentage of clipping occurs into the last station, where 0.4% of the generated voice segments are clipped. The reason that BWB_DQDB penalizes the downstream stations is that each station can request a slot to transmit a voice segment only after the voice segment has become first in queue. Then, this segment has to wait until the request travels all the way to the first upstream station in order to notify all data users about its presence. In the case of P_ITU_NSW and P_BWB_DQDB each voice segment requests a free slot from the data users as soon as it is generated.

We have also considered the same system and voice load configuration but in the absence of data users and have found that P_ITU_NSW and P_BWB_DQDB are almost insensitive to the presence of data load. On the other hand, the presence of data load in the case of BWB_DQDB drastically deteriorates the performance of BWB_DQDB. In the case of BWB_DQDB and in the absence of data load no voice segment is clipped. Notice here that when only one traffic class is present onto the network, P_BWB_DQDB and BWB_DQDB become identical schemes.

In Table 5.1 we have considered the same network configuration and type of load as before. However, in this case each station carries 260 voice sources which produce a 0.98 offered load. In Table 5.1 we show, for each of the three schemes, the percentage of video segments that is clipped, the maximum percentage of voice segments that is clipped in a particular station, the average and the maximum voice segment delay. In the case that at least one voice segment is clipped we show the maximum delay that a voice segment would have, if that voice segment was not clipped. In order to show the effect of the presence of data users on the performance characteristics we have included in Table 5.1 two cases. That is, we have considered both the cases where all data users are overloaded and no data user is present. Also in this case we see that the absolute priority mechanism is fairly insensitive to the presence of lower priority users, even under such

high voice loads (0.98). P_BWB_DQDB in this case has a significant higher clipping rate than P_ITU_BWB. The main reasons for this behavior of P_BWB_DQDB is the bandwidth that wastes due to its BWB mechanism and its slow reaction to changes of load due to the large value of M (=8). Our simulation results have shown that in the case of overloaded data users, in order for BWB_DQDB and P_BWB_DQDB to provide a level of clipping similar to the one of P_ITU_NSW the number of voice sources they can support should be reduced to 245 and 252 respectively. Fig 5.13 and Table 5.1 clearly show that BWB_DQDB does not have the ability to support time critical traffic. This is mainly because its performance strongly depends on the presence of lower priority traffic. In the sequel, we only compare the P_ITU_NSW and P_BWB_DQDB schemes.

Priority Scheme		% of clipping	Max. % of clip. per station	Average delay in msec	Max. delay in msec
P_ITU_NSW	no data	0.02	0.09 / st.4	0.119	6.199
	overl. data	0.05	0.14 / st.11	0.193	6.336
BWB_DQDB	no data	0.30	0.72 / st.15	0.350	6.714
	overl. data	1.00	18.65/ st. 18	0.284	245.237
P_BWB_DQDB	no data	0.30	0.72 / st.15	0.350	6.714
	overl. data	0.33	0.68 / st.15	0.504	6.895

Table 5.1 : Voice performance comparison of P_ITU_NSW, BWB_DQDB and P_BWB_DQDB. 260 voice sources per station.

Until now we have considered the case where all voice sources are evenly distributed among the stations. It is interesting to see how the performance of P_ITU_NSW or P_BWB_DQDB is affected by the distribution of the voice sources among the stations. In Table 5.2 we have considered two different cases of loading. In the first case all voice sources are placed into the last ten stations, i.e. stations "10 to "19", with each station supporting 494 voices. In the second one, stations "0" through "9" carry 494 voice

sources each. The total number of voice sources, in each case is 4940, the same with that of Table 5.1, which generates a total offered load of 0.98. For both load configurations we have assumed that all the data users are saturated. Table 5.2. shows that the performance of the voice users is not affected by their distribution among the stations. The overall clipping percentage in the case of P_BWB_DQDB is much higher than that of P_ITU_NSW. The reason again is the fairly large value of the parameter M and the amount of bandwidth that is being wasted. The less the number of station that carry the voice sources the greater the wastage of bandwidth. For instance, when all 4940 voice sources are carried by a single station, P_BWB_DQDB cannot support them, since this load exceeds the maximum throughput of the station (0.88). We have also consider the performance of BWB_DQDB under the same load configuration. We have found that its performance deteriorates when the voices are placed towards the end of the bus. In the case that the voices are evenly placed into the last ten stations the maximum number of voices BWB_DQDB can support is approximately 430 voices per station.

Priority Scheme		% of clipping	Max. % of clip. per station	Averahe delay in msec	Max. delay in msec
P_ITU_NSW	st. 0-9	0.03	0.28 / st. 1	0.076	6.376
	st.9-18	0.04	0.16 / st. 11	0.141	6.165
P_BWB_DQDB	st. 0-9	0.21	0.71 / st. 6	0.360	6.784
	st.9-18	0.20	0.63 st. 15	0.381	6.397

Table 5.2: Voice performance comparison of P_ITU_NSW and P_BWB_DQDB. 494 voice sources per active station. Overloaded data traffic.

In the sequel, we focus only on the performance of P_ITU_NSW under the presence of both voice and video traffic. Voice and video users are considered to be of the same priority, which is higher than that of the data users. Our performance measures for video are the average and maximum video frame delay as well as the percentage of frames

which have not completed their transmission by the time the next frame form the same source has been generated. First, we consider only video transmissions. In order to show the effect of the presence of data on the performance we consider both cases, i.e. when there are no data, Table 5.3, and when the data users are overloaded, Table 5.4. We have assumed that each active station carries 2 video sources. In tables 5.3 and 5.4 we have considered three different cases, when 13, 14 or 15 stations are active. These tables show that regardless of the presence of data users P_ITU_NSW can support approximately 27 video sources. This corresponds to an offered load of 0.82. In all remaining tables we have considered only the case of overloaded data users.

Number of Video Sources	% of buffered frames	Average delay in msec	Maximum delay in msec	Offered Load
26 (2 videos/st)	0.00	2.999	23.321	0.79
28 (2 videos/st)	0.02	4.045	45.510	0.85
30 (2 videos/st)	0.15	5.868	183.522	0.91

Table 5.3: Performance of video in the absence of data traffic.

Number of Video Sources	% of buffered frames	Average delay in msec	Maximum delay in msec	Offered Load
26 (2 videos/st)	0.00	2.774	19.321	0.79
28 (2 videos/st)	0.10	5.847	79.466	0.85
30 (2 videos/st)	0.35	7.190	213.917	0.91

Table 5.4: Performance of video under overloaded data traffic.

Table 5.5. shows the performance of P_ITU_NSW when both voice and video are present. Here, all voice sources and video channels are evenly distributed among the stations. We have considered 1 video source per station, i.e. 19 videos in total, which produce an offered load of 0.57. Table 5.5 shows how the performance of the system is affected as the number of voice sources increases from 67 per station to 105 per station. We see that in all cases no voice segment is clipped. However, as the number of voice sources increases the performance of video deteriorates.

Number of voice sources	Number of video sources	% of clipping/ buffering		Average delay in msec		Maximum delay in msec		Offered load	
		voice	video	voice	video	voice	video	voice	video
1273 (67 voices/st)	19 (1 video/st)	0.00	0.00	0.08	4.59	2.36	23.32	0.25	0.57
1425 (75 voices/st)	19 (1 video/st)	0.00	0.00	0.09	4.70	2.64	28.78	0.28	0.57
1710 (90 voices/st)	19 (1 video/st)	0.00	2.14	0.11	6.36	3.10	175.72	0.34	0.57
1900 (100 voices/st)	19 (1 video/st)	0.00	16.08	0.13	23.65	3.78	368.32	0.38	0.57
1995 (105 voices/st)	19 (1 video/st)	0.00	18.62	0.15	34.19	4.10	715.03	0.40	0.57

Table 5.5: Symmetric case. Performance of voice and video under overloaded data traffic.

In the video performance results, considered up to this point, all video frames, whether late or not, are transmitted. It is interesting to see how the performance is affected when late video frames are clipped. In Table 5.6 we show the performance characteristics of the network for loading configurations that we have already examined in the case that late video segments are clipped. We see that in all cases the percentage of video frames that have not been transmitted before the next one has been generated is now significantly less. For example, in the case that 105 voices and 1 video are carried per station, this rate is reduced from 18.62% to 4.73%. Table 5.6 also shows the

characteristics of the distribution of the length of the video frame which is dropped. For instance, when 100 voices and 1 video are carried per station, on the average 75.3 segments are dropped every time a late frame is clipped.

Number of voice sources	Number of video sources	% of clipping		Average delay in msec		Offered load		Video frame clipping distribution	
		voice	video	voice	video	voice	video	Mean	Coef. var.
0	30 (3 videos/st)	*	0.19	*	3.775	*	0.91	123.1	0.69
1710 (90 voices/st)	19 (1 video/st)	0.00	0.72	0.06	4.40	0.34	0.57	79.1	0.93
1900 (100 voices/st)	19 (1 video/st)	0.00	2.91	0.12	10.99	0.38	0.57	75.3	0.93
1995 (105 voices/st)	19 (1 video/st)	0.00	4.73	0.12	11.23	0.40	0.57	78.1	0.99

Table 5.6:Performance of voice and video under overloaded data traffic.

Late video packets are clipped.

In Tables 5.7 and 5.8 we investigate the performance of P_ITU_NSW under asymmetric load configurations. In Table 5.7 we have placed all video sources into the last 10 stations. The voices are evenly distributed among all stations. The first three row entries of Table 5.6 show the performance of P_ITU_NSW when the offered load of the voice sources is greater than the video load. In this case, P_ITU_NSW can support approximately 170 voices per station, i.e. 3230 in total. The last two entries show the performance of P_ITU_NSW when most of the offered load is produced by the video sources. In this case, we see that no voice segment is clipped, even at very extreme offered loads, that is, total offered voice and video load 0.99. However the corresponding video performance is unacceptable.

Number of voice sources	Number of video sources	% of clipping/ buffering		Average delay in msec		Maximum delay in msec		Offered load	
		voice	video	voice	video	voice	video	voice	video
3230 (170 voices/st)	10 (1 video/st) (stations 9-18)	0.01	0.81	0.152	5.991	6.036	128.960	0.64	0.30
3344 (176 voices/st)	10 (1 video/st) (stations 9-18)	0.12	6.01	0.261	10.293	6.927	436.284	0.66	0.30
3382 (178 voices/st)	10 (1 video/st) (stations 9-18)	0.25	9.92	0.402	13.742	7.566	542.496	0.67	0.30
285 (15 voices/st)	30 (3 videos/st) (stations 9-18)	0.00	13.61	0.055	21.113	4.415	437.191	0.07	0.91
437 (23 voices/st)	30 (3 videos/st) (stations 9-18)	0.00	59.23	0.057	139.345	5.210	937.063	0.08	0.91

Table 5.7 :Asymmetric case. Voice and video performance under overloaded data traffic.

Finally, in Table 5.8 we examine the effect of the station location on the performance of P_ITU_NSW. We have considered that all voice sources are carried by two stations and that only one station transmits video segments. We have placed the active voice sources at the first two stations of the network, i.e. stations "0" and "1" or at the very end of the bus, i.e. stations "17" and "18". We have located the video user at the opposite end of the voice users; when stations "0" and "1" transmit voice, station "18" is the video user and when stations "17" and "18" are the voice users, station "0" transmits all the video segments. Table 5.8 shows that the station location has no effect on the performance of P_ITU_NSW in all cases.

In this section we have shown that P_ITU_NSW is an ideal mechanism for supporting time critical traffic. We have demonstrated that its performance is not affected by the presence of lower priority traffic, the distribution of the voice and video sources, nor the station location. These properties enables as to analytically estimate the maximum number of voice and video sources that P_ITU_NSW can support, such that the percen-

tage of voice segments that are clipped or video segments that are transmitted late is negligible.

Number of voice sources	Number of video sources	% of clipping/ buffering		Average delay in msec		Maximum delay in msec		Offered load	
		voice	video	voice	video	voice	video	voice	video
350 (175 voices/st) (stations:0,1)	30 (30 videos/st) (station : 18)	0.00	53.00	0.003	62.096	0.028	223.615	0.07	0.91
350 (175 voices/st) (stations: 17,18)	30 (30 videos/st) (station : 0)	0.00	51.59	0.185	55.548	0.994	178.184	0.07	0.91
600 (300 voices/st) (stations:0,1)	28 (28 videos/st) (station : 18)	0.00	34.98	0.004	29.388	0.042	136.306	0.12	0.85
600 (300 voices/st) (stations: 17,18)	28 (28 videos/st) (station : 0)	0.00	33.25	0.192	28.345	0.872	135.097	0.12	0.85
900 (450 voices/st) (stations:0,1)	26 (26 videos/st) (station : 18)	0.00	22.33	0.005	21.371	0.057	121.675	0.18	0.79
900 (450 voices/st) (stations: 17,18)	26 (26 videos/st) (station : 0)	0.00	29.56	0.150	26.336	0.768	123.432	0.18	0.79
600 (300 voices/st) (stations:0,1)	24 (24 videos/st) (station : 18)	0.00	0.00	0.003	3.121	0.035	16.732	0.12	0.72
600 (300 voices/st) (stations: 17,18)	24 (24 videos/st) (station : 0)	0.00	0.00	0.120	4.161	0.884	19.419	0.12	0.72

Table 5.8:Effect of station location on the performance of voice and video. Overloaded data traffic.

5.6 The voice/video Operational Region

In this section present a simple, approximate analytic method to estimate the number of voice sources, N_{voice}, and video sources, N_{video}, that the P_ITU_NSW priority mechanism can support such that the previous voice packet or the video frame is transmitted before the next voice packet or video frame respectively, with probability 0.999 Let P_{voice} denote the probability that a voice packet is transmitted before the arrival of the

next voice packet from the same voice source. Let also P_{video} denote the probability that the transmission of a video frame is completed before the generation of the next video frame from the same video source. We want to estimate the region, for N_{voice} and N_{video}, in which the following inequalities hold:

$$P_{voice}(N_{voice}, N_{video}) \geq 0.999 \tag{5.6.1}$$

$$P_{video}(N_{voice}, N_{video}) \geq 0.999$$

(a) Calculation of P_{video}

We assume that N_{voice} voice sources are present in the system and provide an approximate method to calculate P_{video} (N_{voice}, N_{video}). According to the video process model we have considered, each video source generates a new frame every 1/30 sec. Consequently, a frame will be transmitted before the next one is generated if it can be transmitted within 1/30 sec. Let as now consider the slots generated during a time frame of duration 1/30 sec. Since voice and video are place in higher priority than data, data users do not use any of these slots, provided of course that there is enough load generated from the voice and video sources. This assumption relies on the fact that P_ITU_NSW behaves very similarly to an absolute priority mechanism, and for this reason the presence of lower priority users does not affect the performance of higher priority ones. On the other hand, voice and video are placed at the same priority level. Consequently, voice and video users will share all the slots that are generated within the time frame under consideration. Let T_{video} be the number of slots that the N_{video} video sources require for transmitting their frames during the interval of 1/30 sec. The number of segments each video frame consists of follows the Gaussian distribution with mean $\mu = 250{,}000/352 E[\lambda] = 369.1$ segments and variance $Var = (250{,}000/352)^2\sigma^2 = 164^2$. During the time interval of 1/30 sec each video source generates exactly one frame. Thus, T_{video} follows the Gaussian distribution with mean $N_{video}\mu$ and variance $N_{video}Var$. Let $T_{avail,vid}$ be the number of slots that are available for video transmission. Each voice source, when in talkspurt, con-

sumes 64 Kbps and the available bandwidth is 129.11 Mbps. † So, all voice sources in total, consume $x=(N_{voice}*0.4*0.064)/129.11$ portion of the available bandwidth. Consequently, $T_{avail,vid}=(33{,}333/2.726)*(1-x)$. In order for the T_{video} segments to be transmitted before the next frame from the same voice source is generated we should have that $\Pr\{T_{video} \leq T_{avail,vid}\} \geq 0.999$. After some basic calculations the above inequality becomes:

$$\frac{T_{avail,vid}-369.1N_{video}}{164\sqrt{N_{video}}} \geq 3.09 \qquad (5.6.2)$$

Inequality (5.6.2) estimates the region for N_{video} such that given that N_{voice} sources are present and no voice packet is clipped, each video frame is transmitted before the next one is generated with probability greater or equal to 0.999.

(b) Calculation of P_{voice}

For the calculation of $P_{voice}(N_{voice},N_{video})$ we follow a similar approach with for the calculation of $P_{video}(N_{voice},N_{video})$. We assume that there are N_{video} video sources present and that all generated video frames are being transmitted on time. We now consider a time frame equal to voice packetization interval P_v=5.5 msec. Since voice and video traffic are placed at the same priority level, we may assume that voice sources are able to use the portion of bandwidth that video traffic does not utilize. Let $T_{avail,voic}$ be the number of slots that voice traffic can use for transmitting its segments during the time frame of the 5.5 msec. For each voice source being in talkspurt, a packet is generated every 5.5 msec. It is evident that in order to avoid any possible clipping the number of voices being in talkspurt should be less or equal to $T_{avail,voic}$. Equivalently, we may say that the probability to have any clipping during the time period of 5.5 msec is equal to the probability to have more than $T_{avail,voic}$ voices in the talkspurt phase. According to the voice process model we have adopted, a certain voice is in talkspurt with probability p=0.4. The probability to have k ($k \leq N_{voice}$) voice sources in talkspurt is then given by:

† Here, we have assumed that $0.4*N_{voice}$ voice sources are active during the 1/30 sec time frame.

$$\binom{N_{voice}}{k} p^k (1-p)^{(N_{voice}-k)} \tag{5.6.3}$$

Now the following relation is evident:

$$P_{voice}(N_{voice}, N_{video}) = 1 - \sum_{k=T_{avail,voic}+1}^{N_{voice}} \binom{N_{voice}}{k} p^k (1-p)^{(N_{voice}-k)} \geq 0.999 \tag{5.6.4}$$

In order to be able to estimate the number of voice sources N_{voice} that the P_ITU_NSW can support, we have to calculate the number of slots that are available for voice transmission, i.e. $T_{avail,voic}$. Video frames are generated in a uniformly distributed manner. Since N_{video} video frames are generated in a period of 1/30 sec, the expected number of video frames that are generated during the 5.5 msec is equal to $x = \frac{5.5\ 30}{1{,}000}$. Since, the size, and therefore the transmission time, of each video frame follows the Gaussian distribution, we can use the following estimate for the aggregate transmission time T_{video} from the x video sources during the interval T:

$$Pr\{T_{video} = t_{vid}\} = \frac{1}{\sqrt{2\pi x}\ 164} \times \exp\left\{ \frac{-1}{2x\,164^2} \times (t_{vid} - 369.1x)^2 \right\} \tag{5.6.5}$$

Then, for the available time for voice transmissions during the interval T we have the following:

$$Pr\{T_{avail,voic} = t_{voic}\} = Pr\{T_{video} = 5{,}500/2.726 - t_{voic}\} \tag{5.6.6}$$

From (5.6.4), (5.6.5) and (5.6.6) the region for N_{voice} can be calculated.

In order to investigate the accuracy of this method, in Fig 5.14 we compare the analytical results to our simulation. In this figure we have plotted, with continuous line, the boundaries of inequalities (5.6.2) and (5.6.4) for all pairs of N_{voice} and N_{video}. The asterisks in Fig 5.14 show the cases for which simulation results have been derived. Each asterisk is followed by a pair of "y/n", which indicate whether any voice segment was clipped or any video frame was buffered. For instance, at point (30,300) an "*(y,n)" appears. This means that a simulation was run with offered load of 30 video channels and 300 voices, more than 0.1% video frames were transmitted after the next one from the

same video source was generated and no voice was clipped. The same result was also predicted by the analytical method. Fig. 5.14 shows that the estimated region by our method is in a very tight agreement with our simulation results. This accuracy, does not depend on the distribution of the voice and video sources among the stations nor on the presence of data traffic.

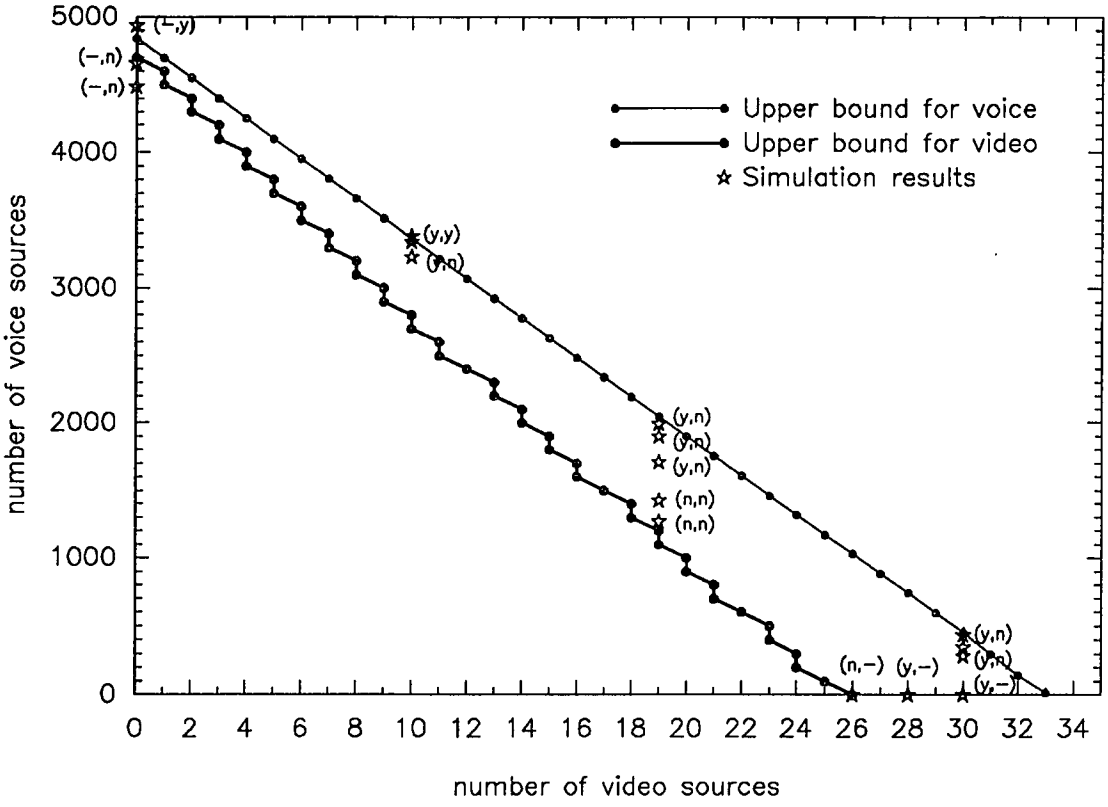

Fig.5.14: Non−clipping region for voice and video traffic. Comparison of analytic and simulation results for various load configurations.

5.7 Conclusions

In this chapter we have introduced a novel, very effective priority mechanism for ITU_NSW and NSW_BWB, which can guarantee almost immediate access to higher priority traffic. We have also implemented this mechanism in the case of BWB_DQDB. We have compared the performance of these mechanisms and have shown that

P_ITU_NSW is a substantially better mechanism than the existing mechanisms which have been proposed for the DQDB network. P_ITU_NSW can provide high priority users with the entire channel bandwidth, without wasting any channel slots. We have also shown that the performance of higher priority classes in the case of P_ITU_NSW is insensitive to the presence of lower priority traffic. Furthermore, the delay variation among users of the same priority in the case of underload conditions is minimal.

We have also examined the ability of P_ITU_NSW to support time critical traffic, that is, voice and video and have found that it can far exceed our expectations. Our simulation results have indicated that the performance of the various voice or video sources is not affected by their distribution among the stations, nor the presence of data traffic. Based on these appealing performance characteristics, we have analytically calculated the number of voice sources and video channels P_ITU_NSW can support. The derived results by our analytic method are in very good agreement with the simulation. Overall, P_ITU_NSW is the only proposed priority scheme in the area of MANs that can fairly distribute the available bandwidth, provide higher priority users with better performance characteristics and minimal delay variation in all cases, with out at the same time, wasting any channel slots.

CHAPTER 6

ERASURE NODES EFFECT ON PERFORMANCE

6.1 Introduction

The objective of slot reuse is to improve the performance of the system by effectively increasing the channel capacity. Slot reuse is achieved by introducing special nodes, named erasure nodes, that can release the slots which have been read by their destinations. In this way the same slot traveling on the bus may carry segments from more that one stations and the total throughput of the system may significantly increase. Slot reuse is implemented in the following way. Each erasure node has a buffer that can store an entire slot plus the Access Control Field (ACF) of the next slot. In addition to busy and request bits, the ACF of each slot contains one more bit, the Previous Segment Read (PSR) bit. A station that reads a slot, sets PSR=1 in the ACF of the next slot. That is, a PSR=1 in one slot indicates that the previous slot has been read by its destination. Therefore, when the read slot arrives at an erasure node it is released. It is now evident why the erasure node must have a buffer size of one slot plus the ACF of a slot, since the information of whether a slot has been read by a station is carried in the PSR bit of the next slot.

In this chapter we investigate the performance of NSW_BWB and ITU_NSW in the presence of erasure nodes and under one or multiple priority classes of traffic. In both cases, we examine the effect of the erasure node locations on the throughputs of the various stations and compare the performance of NSW_BWB and ITU_NSW with the corresponding performance of BWB_DQDB. Our simulation results reveal some very interesting properties for NSW_BWB and ITU_NSW which enable us to derive analytic estimates of its throughput performance in the general case of arbitrary number of stations and arbitrary location of erasure nodes. Results of this research effort have also been presented in [81].

The organization of the rest of chapter 6 is as follows. In section 6.2 we investigate the performance of NSW_BWB and ITU_NSW in the presence of erasure nodes and

under one traffic class. Furthermore, we compare the two mechanisms with BWB_DQDB. In sections 6.3 and 6.4 we examine the effect that the presence of erasure nodes has on the performance of the priority mechanisms presented in chapters 4 and 5 respectively. Finally, in section 6.5 we present the conclusions.

6.2 Slot Reuse under a Single Traffic Class

In the case of slot reuse whenever a station reads a slot, it sets the PSR=1 in the next slot. The erasure node has a buffer size of one slot plus one ACF of a slot and releases a slot when the PSR bit of the next slot is equal to 1. The main question that arises in the case of slot reuse is what action the erasure nodes should take with respect to the request bits on the reverse bus. The simpler approach is to have the erasure node reset as many request bits as is the number of slots it releases. In this case the erasure node must have one additional counter, the Erased Slots Counter (ES_CTR), which must increase by one for every slot erased. If the erasure node sees a request on the reverse bus and ES_CTR>0, then it will reset the request bit and decrease ES_CTR by one. The problem of this approach is that the erasure node may reset more request bits than it should. The reason is that not all of the erased slots will be used by the downstream stations. If some erased slots are not used by any station and later some stations become active and send request bits upstream, these request bits will be reset by the erasure node (since ES_CTR>0) and the downstream stations may not receive any bandwidth.

The approach that we use here is similar to the one in [82]. That is, we assume that some of the stations are also erasure nodes and that their operation is as follows. The erasure node erases every slot which is passing by and has been read by its destination. At the same time, it increases its ES_CTR by one in the following two cases: a) either (both) the Count Down Counter (CD_CTR) or (and) Request Counter (RQ_CTR) is greater than 0, i.e. CD_CTR + RQ_CTR>0, b) CD_CTR + RQ_CTR = 0 but the erasure node writes on the erased slot one of its own segments for which a request has been sent

upstream; notice that in the case of NSW_BWB and ITU_NSW there is the possibility of transmitting a segment without sending a corresponding request upstream. When an erasure node whose both the queue of requests for the reverse bus and its ES_CTR are greater than 0 sees a slot with the request bit equal to 0, it simply decreases the size of its request queue and the value of its ES_CTR by one. When an erasure node whose ES_CTR is greater than 0 sees a slot with the request bit set to 1, it resets the request bit to 0 and decreases ES_CTR by one. We point out that when the erasure node is a separate station, which erases slots and does not transmit any segments, the operation is identical to the one described above but with CD_CTR always 0.

In the case of NSW_BWB and ITU_NSW each slot carries a TAR bit. Therefore, another issue is what the erasure node should do with the TAR bit in the erased slots; if it is 1. We examine both cases, i.e. when the TAR bit is reset and when it is not reset. We will use the suffix TNE for the first approach (TAR bit Not Erased) and TE (TAR bit Erased) for the second one. For instance, ITU_NSW_TNE will indicate the ITU_NSW mechanism in the presence of erasure nodes which do not erase the TAR bit in the slots they reset. In the next sections we examine the performance of NSW_BWB and ITU_NSW under overload conditions. In this case, as we have already mentioned in chapter 4, the two mechanisms become identical. For this reason we do not distinguish them and we use the term NSW to refer to both of them.

In Fig.6.1 we consider a DQDB network in which only three stations ("1", "2" and "3") become active and compare the convergence of NSW and BWB_DQDB when an erasure node is located between stations "2" and "3". The channel capacity is 155.53 Mbps, the slot size 53 bytes, and the signal propagation delay 5 $\mu sec/Km$. The distance between stations "1" and "2" is $D_{12}=38$ slots (20.72 Km) and between "2" and "3" $D_{23}=40$ slots (21.81 Km). We have selected a value of $M=2$ for NSW, and $M=8$ for BWB_DQDB. The same values of M have also been selected for the two bandwidth balancing mechanisms in all other cases of this section; unless otherwise mentioned. In

Fig.6.1 50% of the slots generated by each station have destinations stations located before the erasure node. That is, the erasure node erases 50 % of the slot that are passing by. Here, and throughout the chapter, the destination of each segment, i.e. whether it will be before or after the erasure node, is randomly decided (every time) by the transmitting station. In Fig.6.1 the three stations become active in the order station "2" first, station "1" second, and station "3" last. We assume that when a station becomes active is over-loaded and tries to acquire all bandwidth.

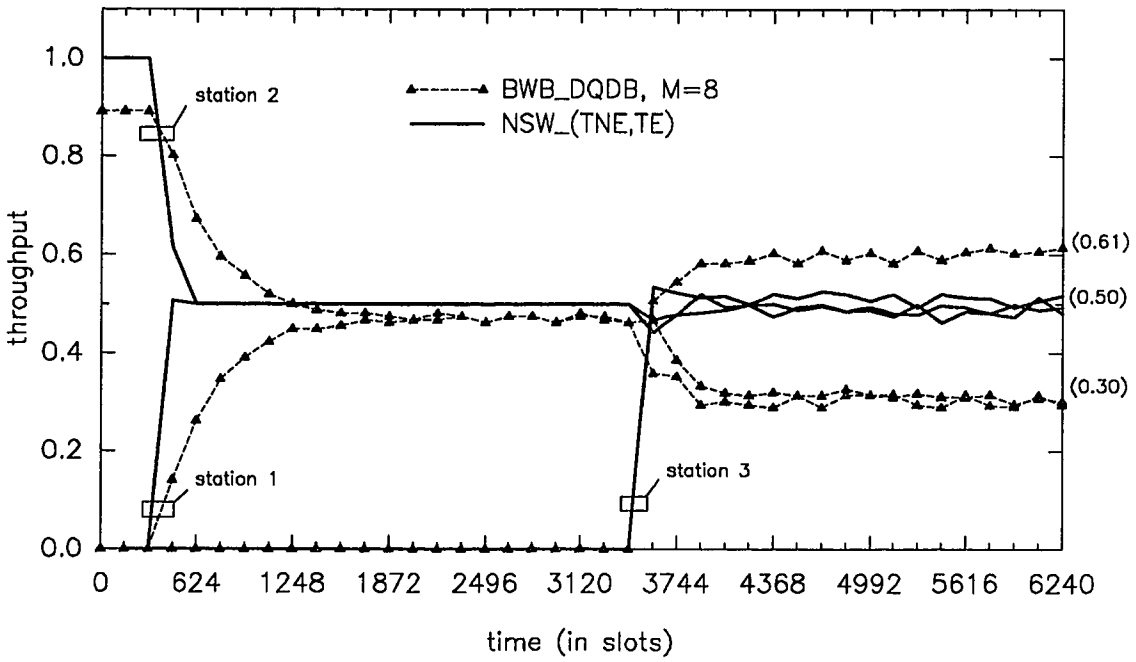

Fig.6.1: Throughput performance under slot reuse. One erasure node between stations 2 and 3. Comparison of BWB_DQDB and NSW under both TNE and TE. D_{12}=38 slots, D_{23}=40 slots. 50% of the segments transmitted by stations 1 and 2 are erased.

Our simulation results have shown that in the case of Fig.6.1 NSW_TNE and NSW_TE behave identically. For this reason we have used NSW_(TNE,TE) to indicate the corresponding throughput versus time characteristic curves. Fig.6.1 clearly shows the higher throughput provided by NSW, especially when one only station (station "2") is active in the system. Furthermore, it illustrates NSW's faster convergence towards the steady state, especially when station "1" becomes active. Finally, it shows that when sta-

tion "3" which is located behind the erasure node becomes active, the throughputs of all three stations converge to the same value, 0.50. In contrast, in the case of BWB_DQDB the steady state throughputs of the three stations are different. Stations "1" and "2" which are located on the first bus segment (before the erasure node) have a throughput of 0.30. Station "3", which is located on the second bus segment (after the erasure node), has a much higher throughput of 0.61. We see that NSW provides similar throughputs to the three stations even under slot reuse. Moreover, its total throughput of 1.50 is much higher than the 1.21 throughput of BWB_DQDB.

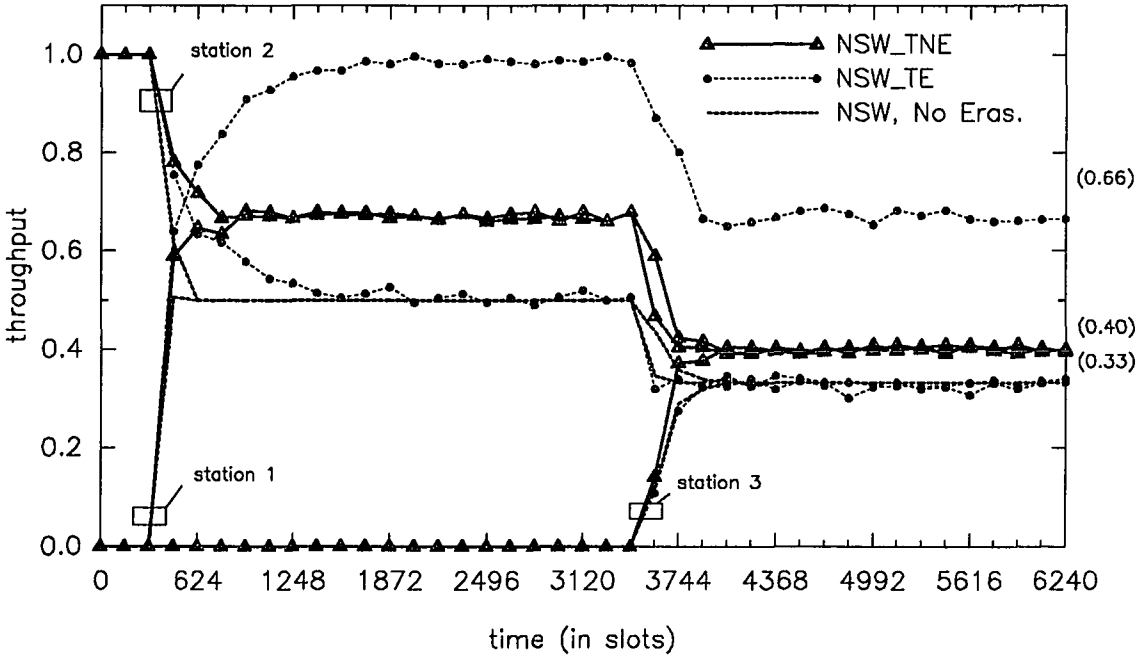

Fig.6.2: Throughput performance. Comparison of NSW with and without erasure nodes. In the case of slot reuse, one erasure node between stations 1 and 2 erases 50% of the written slots.

In Fig.6.1 NSW_TNE and NSW_TE have identical behavior. However, this is not always the case. In Fig.6.2 we consider the system of Fig.6.1 but with the erasure node located between stations "1" and "2", and with 50 % of the segments transmitted by station "1" having destination stations located before the erasure node. Then, when all three stations are active, their steady state throughputs will be similar and equal to 0.40 in the

case of NSW_TNE. On the other hand, in the case of NSW_TE the throughput of each of the stations "2" and "3" will be 0.33, whereas the throughput of station "1" will be 0.66. We see that in this case the aggregate throughput of NSW_TE, which is 1.32, is higher than the corresponding aggregate throughput of NSW_TNE, which is 1.20.

In order to get a better insight into the behavior of BWB_DQDB and NSW under the presence of erasure nodes, we have used simulation to compare their performance in the case of four overloaded stations and different locations of the erasure node. We present some of our results in Table 6.1, where we indicate the location of the erasure node by the small box with the letter E. In Table 6.1, 50% of the segments transmitted by the stations which are located before the erasure node have destination stations also located before the erasure node. In order to show the gain in throughput of the various stations which is due to the slot reuse, we have also included in the first row of Table 6.1 the steady state throughputs of the stations when there is no erasure node. The absence of the erasure node in this case is indicated by the absence of a box with a letter E. Notice that there is no difference between NSW_TNE and NSW_TE in this case, since no slot is erased.

Table 6.1 shows that the highest aggregate system throughput is provided by NSW_TE and the lowest by BWB_DQDB. Moreover, all three schemes provide the same bandwidth to the stations located on the same segment of the bus, where the different bus segments are identified by the location of the erasure node. However, NSW_TNE seems to be able to provide, in most of the cases, the same bandwidth even to stations that are located on different bus segments.

NSW_TNE tries to provide the same throughput to all stations, regardless of their location, meeting at the same time the constraint that no bus segment can have an aggregate throughput greater than one. However in some cases, due to the slots that are erased, the bandwidth which is available to the stations of a downstream bus segment may be higher. Since NSW_TNE tries to balance the throughputs of the stations in both bus seg-

ments, the best it can do is to increase the throughput of the upstream stations until their aggregate throughput becomes 1. Such a case appears in the last row of Table 6.1, where the throughput of each of the three upstream stations is 1/3 and the throughput of the fourth station is all the released bandwidth (erased slots) by the erasure node, which is 0.50.

Table 6.1: Throughput comparison with and without slot reuse. In the case of slot reuse 50% of the slots passing in front of the erasure node are erased.

BWB_DQDB M=8				NSW_TNE M=2				NSW_TE M=2			
0.24	0.24	0.24	0.24	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25
0.22	E 0.28	0.28	0.28	0.28	E 0.28	0.28	0.28	0.50	E 0.25	0.25	0.25
0.15	0.15	E 0.40	0.40	0.33	0.33	E 0.33	0.33	0.50	0.50	E 0.25	0.25
0.15	0.15	0.15	E 0.69	0.33	0.33	0.33	E 0.50	0.33	0.33	0.33	E 0.50

The ability of NSW_TNE to behave in this way is due to the TAR bits that are not erased and allow the stations behind the erasure node to have a very good estimate of the load before the erasure node. This is not the case for BWB_DQDB and NSW_TE where the stations behind the erasure node cannot tell whether an idle slot has never been written or has been written and then released. In the sequel, and because of their higher throughputs, we focus only on the performance of the two variations of NSW.

We have shown in chapter 4 that NSW can distribute the channel bandwidth in any arbitrary way among overloaded stations by simply using different values of M. Besides, if some of the stations are lightly loaded, they will always receive the requested bandwidth while the remaining bandwidth will be distributed among the overloaded stations in a way which is proportional to their values of M. Simulation results have shown that both NSW_TNE and NSW_TE can also accomplish that, in the presence of erasure

nodes, for the stations of the same bus segment. However, NSW_BWB_TNE has the ability to provide proportional bandwidths even between stations belonging to different bus segments.

In Table 6.2, we compare NSW_TNE and NSW_TE in the case of six overloaded stations, having different values of M, and under the presence of two erasure nodes; which divide the bus into three segments. We assume that 50 % and 25 % of the slots written by the stations that are located on the first bus segment have destination stations located on the first and second bus segments respectively. Furthermore, 50% of the slots written by the stations of the second bus segment have destination stations on the same (second) segment. We have included in Table 6.2, as in Table 6.1, the throughputs of the stations under no slot reuse. Table 6.2 shows that even in the case of more than one erasure nodes the throughputs of the various stations that are located on the same bus segment are proportionate to their values of M and that NSW_TNE can preserve this throughput proportionality even among the stations that are located on different bus segments. In the next two subsections we provide analytic estimates for the throughputs of the stations in the case of NSW_TNE and NSW_TE.

Table 6.2: Throughput comparison with and without slot reuse and different values of M. In the case of slot reuse 50% and 25%-50% of the busy slots seen by the first and second erasure nodes, respectively, are erased.

NSW_TNE						NSW_TE					
M=2	M=4	M=2	M=6	M=2	M=4	M=2	M=4	M=2	M=6	M=2	M=4
0.10	0.20	0.10	0.30	0.10	0.20	0.10	0.20	0.10	0.30	0.10	0.20
0.18	0.35	0.18	[E] 0.53	[E] 0.18	0.37	0.25	0.50	0.25	[E] 0.50	[E] 0.17	0.33
0.17	0.34	[E] 0.17	0.51	[E] 0.17	0.34	0.27	0.53	[E] 0.15	0.45	[E] 0.17	0.33
0.14	0.29	0.14	0.43	[E] 0.31	[E] 0.61	0.14	0.29	0.14	0.43	[E] 0.50	[E] 0.50

6.2.1 Throughput Analysis of NSW_TNE

The ability of NSW_TNE to distribute the available bandwidth among the stations in a way which is proportionate to their values of M, regardless of their location on the bus, enables us to derive analytic estimates for the throughputs.

We consider a system with N active and overloaded stations and N_E erasure nodes. Let S_r be the bus segment between erasure nodes "r" and "r+1"; S_0 is the bus segment between the slot generator and the first erasure node, and S_{N_E} is the bus segment after the last erasure node. Let N_r be the set of stations that are located on the S_r bus segment. Let M_i be the value of M for station "i", T_i its throughput, $P_{i,j}$ the fraction of slots that writes with destination station "j", and $a_{i,r}$ the fraction of slots that writes with destination a station on bus segment S_r; it is evident that $a_{i,r} = \sum_{j \in N_r} P_{i,j}$. Finally, let W_r be the total throughput of the stations on segment S_r, and E_r the number of slots erased by Erasure Node "r" (EN_r).

NSW_TNE distributes the bandwidth among the stations, on the same bus segment, proportionally to their values of M. Therefore, the following equation holds for the throughput T_i of station "i" on bus segment S_r:

$$T_i = W_r \frac{M_i}{\sum_{j \in N_r} M_j} \tag{6.2.1}$$

The fraction E_r of the slots erased by EN_r will be the fraction of slots written by upstream stations that have destinations the stations of the segment S_{r-1}. Therefore, we have:

$$E_r = \sum_{k=0}^{r-1} \sum_{i \in N_k} T_i a_{i,r-1} = \sum_{k=0}^{r-1} \sum_{i \in N_k} W_k \frac{M_i}{\sum_{j \in N_k} M_j} a_{i,r-1} = \sum_{k=0}^{r-1} W_k A_{k,r-1} \tag{6.2.2}$$

where $A_{k,r-1} = \sum_{i \in N_k} \frac{M_i a_{i,r-1}}{\sum_{j \in N_k} M_j}$ is the fraction of slots that are written by the stations of bus segment S_k and have as destinations the stations on bus segment S_{r-1}.

We can now find W_r in the following way. We subtract from the bandwidth that upstream stations allow for the stations of segment S_r, the bandwidth which is reserved by the requests of the downstream to S_r stations. Let us first consider the bandwidth W_0 of the stations located on segment S_0. The bandwidth of these stations will only be limited by the requests they see. Each station sends a request for each segment it transmits. If we now assume that under overload conditions the number of requests that each erasure node resets is equal to the number of slots it erases, the following equation will hold:

$$W_0=1-\left[\sum_{r=1}^{N_E} W_r-\sum_{r=1}^{N_E} E_r\right]=1-\sum_{r=1}^{N_E}\left[W_r-\sum_{k=0}^{r-1} W_k A_{k,r-1}\right] \tag{6.2.3}$$

We have seen that NSW_TNE has also the ability to provide throughputs which are proportionate to M_i even between stations on different bus segments; if the available bandwidth allows it. That is, $T_i/T_j = M_i/M_j$ even for "i" and "j" located on bus segments S_r and S_k with $r \neq k$. Let $B_r = \sum_{i \in N_r} M_i$. It is then evident, from (6.2.1), that:

$$\frac{W_r}{W_k} = \frac{B_r}{B_k} \tag{6.2.4}$$

and equation (6.2.3) can be written as:

$$W_0 = 1 - \max\left\{\sum_{r=1}^{N_E}\left[\frac{W_0 B_r}{B_0} - \sum_{k=0}^{r-1}\frac{W_0 B_k}{B_0} A_{k,r-1}\right], 0\right\} \tag{6.2.5}$$

We have considered the maximum in the right hand side of the above equation in order to take into account the cases where equation (6.2.4) does not hold because of the higher throughputs achieved by the downstream stations, due to the excess bandwidth generated by the erased slots. From equation (6.2.5) the following expression for W_0 is derived:

$$W_0 = \frac{1}{1+\frac{1}{B_0}\max\left\{\sum_{r=1}^{N_E}\left[B_r - \sum_{k=0}^{r-1} B_k A_{k,r-1}\right], 0\right\}} \tag{6.2.6}$$

Let now $C_{k,r}$ be the fraction of the slots written by the stations of segment S_k and having as destination stations after erasure node "r". The expression of $C_{k,r}$ is given by:

$$C_{k,r} = \sum_{i \in N_k} \frac{M_i}{\sum_{j \in N_k} M_j} \sum_{j=r}^{N_E} a_{i,j} \tag{6.2.7}$$

It is now evident that the following equation holds for the throughput W_r of the stations located on segment S_r:

$$W_r = 1 - \sum_{k=0}^{r-1} W_k \, C_{k,r} - \sum_{k=r+1}^{N_E} \left[W_k - E_k \right] \tag{6.2.8}$$

From (6.2.8), using the same procedure that lead to equation (6.2.6), the following expression for W_r can be derived:

$$W_r = \frac{1 - \sum_{k=0}^{r-1} W_k \, C_{k,r}}{1 + \frac{1}{B_r} \max\left\{ \sum_{k=r+1}^{N_E} \left[B_k - \sum_{j=0}^{k-1} B_j \, A_{j,k-1} \right], 0 \right\}} \tag{6.2.9}$$

Recursion (6.2.9) can now be used to compute estimates for all values of W_r, starting the computation from W_0 which is given by equation (6.2.6). Then using equation (6.2.1) the throughputs of all stations can be computed.

6.2.2 Throughput Analysis of NSW_TE

We can provide analytic estimates for the stations throughput also in the case of NSW_TE by making the following observation. Let us consider a station "j" which belongs into group N_r, located downstream a station "i" that belongs into group N_k. Then, if we consider the same system as in section 6.2.1, station "j" observes only an $L_{i,r} = \sum_{l=r}^{N_E} \alpha_{i,l}$ fraction of the slots written by station "i". Since now the TAR=1 bits are also erased whenever a slot is erased, station "j" sees also an $L_{i,r}$ fraction of the TAR=1 bits

set by station "i". It is now evident that if the TAR=1 bits were not erased, station "j" would have seen the same number of TAR=1 bits as in the case where station's "i" value of the bandwidth balancing parameter was $M_i^*(r)=M/L_{i,r}$. Thus, the same analytic procedure developed in the previous section 6.2.1 for NSW_TNE, can also be repeated here for NSW_TE by simply replacing equation (6.2.4) with the following:

$$\frac{Wr}{W_k} = \frac{Br}{B_k(r)} \tag{6.4.10}$$

where $B_k(r)=\sum_{i \in N_k} M_i^*(r)$. Notice that it is important in the above equation the bus segment "r" to be located downstream of bus segment "k".

In order to verify the accuracy of the analytic expressions we have used them to rederive the throughputs of the various stations in the systems considered in Tables 1 and 2. We have found that the analytically derived throughputs are in very good agreement with the corresponding throughputs derived from simulation.

6.2.3 Delay Comparison

In this section we carry out a delay comparison of BWB_DQDB, NSW_BWB_TNE, NSW_BWB_TE, and ITU_NSW_TNE. We have assumed a dual bus network consisted of 20 stations and with an inter-station distance of 2 slots. We compare the delays encountered by the various stations for their transmissions on the forward bus. Again, we define as average message delay, the average time from the instant a message arrives at a station until the instant the last segment of this message is about to start its transmission onto the medium. We assume Poisson arrivals for the messages and that each station transmits to any other station with the same probability, i.e. linear load.

In Fig.6.3 there is only one erasure node on the forward bus. Using the equations in [41] we find that station "10" should be the erasure node; notice that station "10" is the eleventh station on the forward bus. The offered load by the stations has been selected in a way that makes the forward bus utilization at station "10" equal to .85; the correspond-

ing aggregate utilization of the forward bus is 1.11. We consider the transmission of constant size messages consisted of 20 segments. Fig.2 shows that NSW_BWB_TNE and NSW_BWB_TE provide almost identical message delays.† Furthermore, the average message delay increases as we approach the erasure node. However, it remains low after the erasure node; although it shows a small increase as the station index increases. In the case of ITU_NSW_TNE the delay variation among the stations of the same bus segment is minimal with the average delay behind the erasure node being lower. We can also observe a small delay variation in the case of BWB_DQDB. Nevertheless, the average message delay is much higher on both segments of the bus.

The analysis in [41] assumes that the number of erased slots on the forward channel will be equal to the number of requests that are reset on the reverse channel. However, due to propagation delay, this is not always the case and our simulation has shown that more requests, than they should, pass on the first segment of the bus; the one before the erasure. As a result the delay of the stations in this bus segment increases. For this reason we plot in Fig.6.4 the delays of the various stations for the same offered load but with erasure node station "9". We see that in the case of NSW_BWB (both TE and TNE) the delay of station "9" (which is now the erasure node) drastically decreases, while the delay of the other stations in the first segment of the bus is only slightly affected. The cost we have to pay is the higher delays encountered by the stations which are behind the erasure node. The higher delay of these stations is due to the increased load of this bus segment because of the fewer slots that are now erased; all slots with destination "10", that were erased before, will not be erased now. In the case of BWB_DQDB, and for the same reason, the delay of all stations behind the new erasure node also increases. However, the delay of the stations in the first segment of the bus does not decrease. In fact, the delay of station "0" significantly increases. It seems that in the case of BWB_DQDB the negative

† For this reason we have not plotted the delays in the case of ITU_NSW_TE, which are almost identical with those of ITU_NSW_TNE.

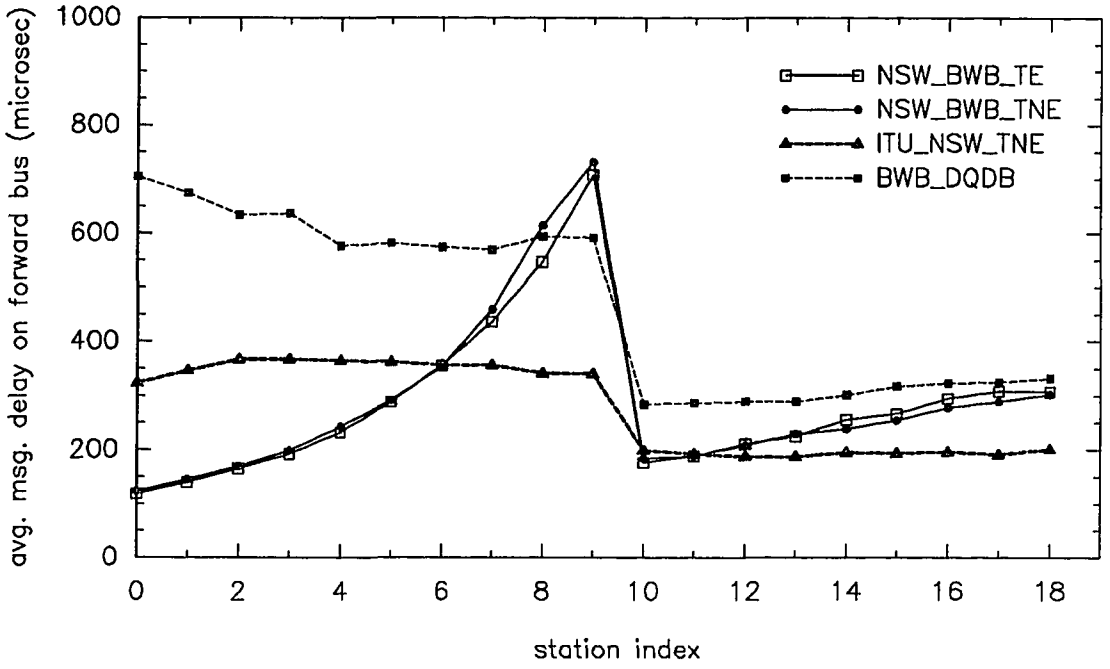

Fig.6.3:Slot reuse. Delay comparison of NSW_BWB_TE, NSW_BWB_TNE, ITU_NSW_TNE and BWB_DQDB. Bus utilization 1.11. Erasure node is station 10. Constant message size of 20 segments.

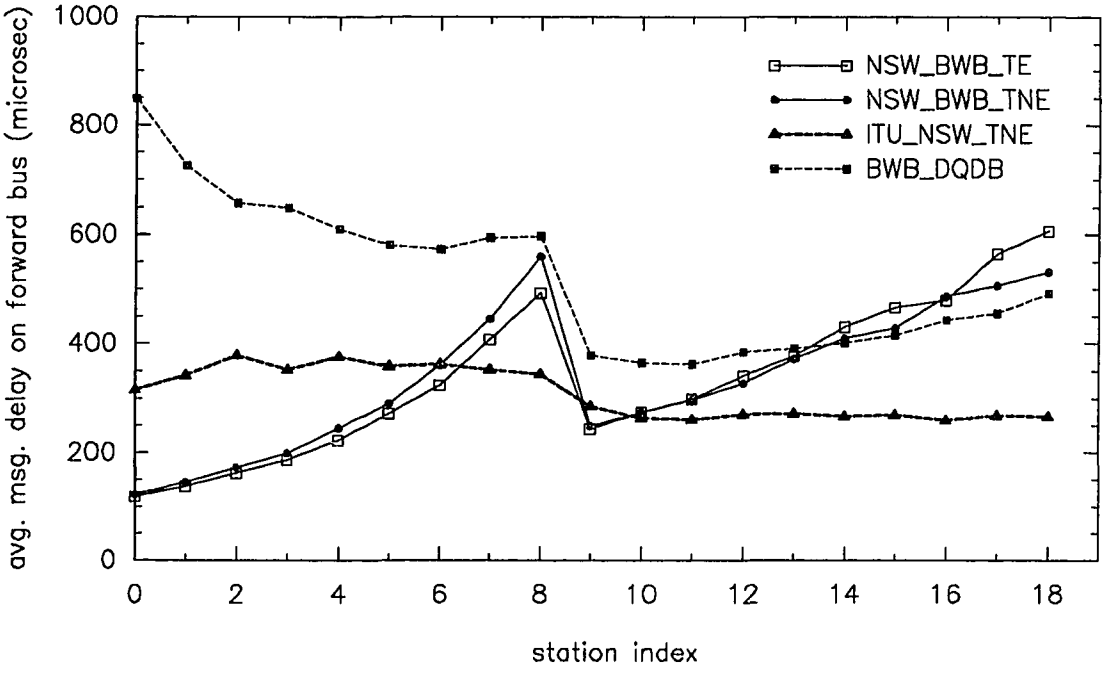

Fig.6.4:Slot reuse. Delay comparison of NSW_BWB_TE, NSW_BWB_TNE, ITU_NSW_TNE and BWB_DQDB. Bus utilization 1.11. Erasure node is station 9. Constant message size of 20 segments.

effect of the additional requests that are sent on the first bus segment, due to the higher bandwidth requirements on the second bus segment, is stronger than the positive effect due to that station "9" does not transmit anymore on the first segment of the bus. Finally, ITU_NSW_TNE demonstrates the best performance by reducing the delay variation even between stations that belong into different bus segments.

We have also investigated the performance of the previous system in the presence of two erasure nodes (stations "8" and "11"). We have found, again, that the delays provided by the TNE and TE mechanisms are almost identical. Furthermore, for each bus segment and under both schemes, the message delay increases as the station index increases and drops at the erasure node; with the lower delay in the system encountered by the stations which are located behind the second erasure node. In the case of BWB_DQDB the variation of the message delay among the stations of the same bus segment is smaller than that of NSW_BWB, but higher than that of ITU_NSW_TNE. ITU_NSW presents again the best performance. Not only it provides lower delays but also the delay variation among the stations is significantly smaller.

Finally, in Fig.6.5 we compare the delays provided by these mechanisms in a more realistic environment. We consider a system consisted of two file servers and 18 work-stations. File servers are stations "7" and "12". The file servers are also erasure nodes. We assume that file servers and work-stations generate constant size messages of 50 segments according to a Poisson distribution. The traffic consists of messages that the work-stations transmit to file servers and of messages that file servers transmit to the work-stations. We consider the following load on the forward bus. Each of the first seven work-stations generates a load of $\frac{1}{7} * \frac{3}{4} * .85$ with destination the first file server, and a load of $\frac{1}{7} * \frac{.85}{4}$ with destination the second file server; the first file server sends its responses for these work-stations on the reverse bus. In this way the total load of the forward bus at the location of the first file server is .85. The first file server generates a load of $\frac{.0.85}{4}$ with destination the stations between the two file servers, i.e. stations "8" to

"11", a load of $\frac{.85}{4}$ with destination the second file server, and a load of $\frac{.85}{4}$ with destination the stations "13" to "19". Each of the stations "8" to "11" generates a load of $\frac{1}{4} * \frac{.85}{4}$ with destination the second file server. In this way the total load of the forward bus at the location of the second file server is .85. Finally, the second file server generates a load of $.85 * \frac{3}{4}$ with destination the stations "13" to "19". The above load distribution provides an aggregate throughput of 2.125 for the forward bus.

In Fig.6.5 we plot the average message delays encountered by the file servers and work-stations considering their transmissions over both busses; the offered load on the reverse bus is the symmetric one of the offered load on the forward bus. We see that in the case of the NSW_BWB mechanisms, stations "6" and "13" encounter the largest delays, whereas in the case of BWB_DQDB the erasure nodes encounter the largest delays; due to the significant number of slots that they waste. Finally, in the case of ITU_NSW_TNE all stations have similar delays.

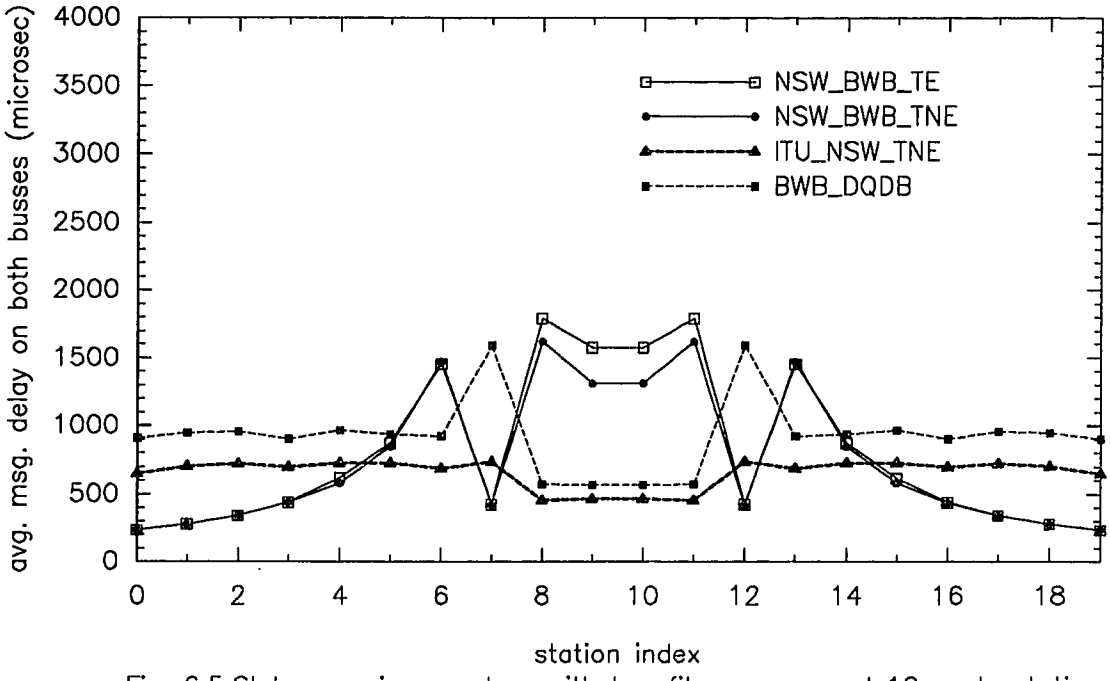

Fig. 6.5:Slot reuse in a system with two file servers and 18 work-stations. File servers are stations 7 and 12. The file servers are also erasure nodes. Total bus utilization of each bus 2.125. Constant message size of 50 segments.

6.3 Slot Reuse under Multiple Priority Classes of Traffic

In chapter 4, various BWB priority mechanisms have been investigated for NSW_BWB, ITU_NSW and BWB_DQDB. We have seen that in the case of overloaded traffic classes the throughput performance of ITU_NSW is identical to that of NSW_BWB, whereas in the case of underloaded conditions the performance of ITU_NSW is significantly better than that of NSW_BWB. Thus, in this section we mainly focus on ITU_NSW and BWB_DQDB and investigate their performance under the presence of erasure nodes and multiple priority classes of traffic. However, still when we consider overload conditions we use the generic term NSW instead of ITU_NSW to remind the reader that the presented throughput performance is the same for both ITU_NSW and NSW_BWB. We consider the BWB over traffic classes priority mechanisms (both the non-adaptive and adaptive) because they are more interesting since the throughput they can provide to each traffic class is independent of the presence of other traffic classes inside the station.

In the case of the non-adaptive BWB priority mechanisms, both NSW (i.e. ITU_NSW and NSW_BWB) and BWB_DQDB use one ES_CTR per erasure node, regardless of the number of traffic classes in the system. Its operation is identical to the one described in section 6.2. In the case of the adaptive BWB priority mechanisms, a separate request and busy bit is used for each priority class and the operation of the erasure nodes under BWB_DQDB slightly changes. In this case (i.e. BWB_DQDB), each erasure node uses a separate ES_CTR for each traffic class and its operation is similar to the one described in [82]. However, under ITU_NSW we continue to use one only ES_CTR per erasure node. The reason is that the NSW adaptive priority mechanism requires by the RQ_CTRs of the upstream users to take into account all the requests they see, regardless of their priority.

In the case of BWB over traffic classes mechanism, the performance of each class is characterized by its value of M and not the type of messages it transmits. Therefore,

Table 6.2 can be considered as describing the behavior of a system with different traffic classes and with the larger values of M assigned to the higher priority classes.

We now examine the behavior of the adaptive BWB over priority classes mechanism. In Table 6.3 we consider three stations with inter station distances $D_{12}=38$ slots and $D_{23}=40$ slots. The first station supports high priority traffic, the second station supports medium priority traffic, and the last station supports both high and low priority traffic. The values of M assigned to high, medium and low priority traffic are $M^h=6$, $M^m=4$, and $M^l=2$ respectively. In Table 6.3 we show the steady state throughputs of the various classes, under different configurations of active high, medium, and low priority users. We consider both cases, i.e. when there is no erasure node and when there is one erasure node between stations "2" and "3". We assume that in the case of erasure nodes 50 % of the slots written by each station have destination stations before the erasure node. The first column in Table 6.3 provides the identity of the active classes.

Table 6.3: Throughput comparison with and without slot reuse. Adaptive BWB over priority classes. $M^h=6$, $M^m=4$, $M^l=2$. In the case of slot reuse 50% of the slots passing in front of the erasure node are erased.

	NSW No slot reuse				NSW_TNE				NSW_TE				BWB_DQDB			
Active classes	1,H	2,M	3,H	3,L	1,H	2,M	3,H	3,L	1,H	2,M	3,H	3,L	1,H	2,M	3,H	3,L
1,H	1.00	-	-	-	1.00	-	E	-	1.00	-	E	-	0.86	-	E	-
1,H 2,M	0.88	0.12	-	-	0.88	0.12	E	-	0.88	0.12	E	-	0.86	0.11	E	-
1,H - 2,M 3,H	0.47	0.06	0.47	-	0.80	0.10	E	0.53	0.80	0.11	E	0.52	0.65	0.11	E	0.58
1,H - 2,M 3,H - 3,L	0.46	0.06	0.46	0.01	0.75	0.09	E	0.56	0.80	0.10	E	0.53	0.65	0.09	E	0.58

Table 6.3 shows that in the case of the adaptive BWB priority mechanism, the high priority class acquires most of the bandwidth under both NSW and BWB_DQDB.

Furthermore, the presence of the erasure node improves the throughput performance on both segments of the bus. Again, the total throughput provided by NSW_TE is slightly higher than the corresponding throughput provided by NSW_TNE. However, both outperform BWB_DQDB.

6.3.1 Throughput Analysis of NSW_TNE

In this section we compute estimates for the throughputs of the different priority classes, inside the various stations, under overload conditions. We have mentioned that in the case of the non-adaptive BWB over classes mechanism, the throughput performance of each priority class "p" is determined by the value of the bandwidth balancing parameter M^p. Since each priority class inside a station behaves as a separate (sub)station, the analysis of section 2.1 can be directly used to evaluate the various throughputs.

In the case of the adaptive BWB over classes priority mechanism the acquired bandwidth by a traffic class "p" does not depend solely on the value of M^p and therefore the analysis becomes more complex. The key property of NSW_TNE that enables us to compute estimates of the throughputs in this case is that each traffic class, regardless of priority and bus segment, will erase and transmit (if allowed by the available bandwidth) the same number of TAR=1 bits.

We now consider a system of N stations and N_E erasure nodes that divide the forward bus into $N_E + 1$ segments. As in section 6.2.1, we have used S_r to indicate the bus segment between erasure nodes "r" and "r+1", and N_r to indicate the set of stations located on this segment. Let P be the number of priority classes in the system, T_i^p the throughput of class "p" at station "i", and $a_{i,r}^p$ the fraction of slots that class "p" at station "i" writes with destination the stations of bus segment "r". It is evident that erasure node "r" (EN_r) erases all the written slots that have destinations the stations of segment S_{r-1}. The fraction E_r^* of all slots, regardless of priority, that are erased by EN_r is then given by:

$$E_r^* = \sum_{k=0}^{r-1} \sum_{i \in EN_k} \sum_{p=1}^{P} T_i^p \, a_{i,r-1}^p \tag{6.3.1}$$

Let R_r^p be the fraction of slots on the reverse bus that stations of segment S_r see to carry priority "p" requests which have been inserted by stations located on bus segments downstream to S_r, i.e. these requests have not been reset by erasure node EN_{r+1}. Let W_r^p be the total throughput of priority class "p" in the same segment. R_r^p can be computed if from the priority "p" requests that stations of bus segment S_{r+1} see and insert, we subtract those that are erased by EN_{r+1}. Since the priority "p" requests that stations of bus segment S_r insert are equal to the number of priority "p" slots they write, the following equation for R_r^p holds:

$$R_r^p = R_{r+1}^p + W_{r+1}^p - E_{r+1}^* \frac{R_{r+1}^p + W_{r+1}^p}{\sum_{j=1}^{P} (R_{r+1}^j + W_{r+1}^j)} \tag{6.3.2}$$

where the last term in the right hand side of the above equation indicates the fraction of priority "p" requests that are erased by erasure node "r+1".

Let U_r^p be the fraction of slots that have been written by upstream priority "p" classes and are seen by the stations of bus segment S_r. The number of these slots can be computed if from the number of priority "p" slots that the stations of bus segment S_{r-1} see and write, we subtract those that are erased by the erasure node EN_{r-1}. Therefore, the following equation holds:

$$U_r^p = U_{r-1}^p + W_{r-1}^p - \sum_{k=0}^{r-1} \sum_{i \in EN_k} T_i^p \, a_{i,r-1}^p \tag{6.3.3}$$

Let finally X_i^p be the fraction of slots for which class "p" at station "i" sets TAR=1, and T_{ime} a very large time interval. Then in the case of the non-adaptive BWB mechanism, the TAR=1 bits that class "p" at station "i" transmits are $T_i^p T_{ime} / M^p$. However, in the case of the adaptive BWB mechanism, for every request or busy bit of lower priority

that class "p" at station "i" sees, it postpones the transmission of a TAR=1 bit by the time required to transmit M^p additional segments, that is, class "p" cuts back the transmission of one TAR=1 bit. Therefore, the total number of TAR bits $X_i^p T_{ime}$ that this priority class transmits can be computed if from $T_i^p T_{ime} / M^p$ we subtract the total number of lower priority busy and request bits that this class sees. Hence, the following expression holds for X_i^p :

$$X_i^p = \frac{T_i^p}{M^p} - \sum_{j=1}^{p-1} (R_r^j + U_r^j + W_r^j) \tag{6.3.4}$$

where we have assumed that station "i" is located on bus segment S_r.

We have seen in chapter 4 that each priority class erases and transmits the same number of TAR=1 bits, and that this is the reason that enables NSW to provide throughput fairness or arbitrary bandwidth distribution. This is also true, in the presence of erasure nodes, for the priority classes in the same bus segment. However, NSW_TNE can achieve that even between different bus segments unless, due to the erased slots, the available bandwidth in downstream segments allows higher throughputs; for instance last row of Table 6.1. Therefore, in our analysis we initially assume that the same number of TAR=1 bits is erased and sent by any priority class, regardless its location on the bus. That is, the following equation holds:

$$X_i^p = X_j^k \quad p,k=1,2...,P, \; i,j=1,2,...,N \tag{6.3.5}$$

Equation (6.3.5) together with equations (6.3.1)-(6.3.4), can provide (N*P-1) independent nonlinear equations for T_i^p. Another equation can be derived from the requirement that the aggregate throughput in the last bus segment S_{N_E} must be equal to 1, that is:

$$\sum_{i=1}^{N} \sum_{p=1}^{P} T_i^p \, a_{i,N_e}^p = 1 \tag{6.3.6}$$

The above non-linear equations can then be solved using numerical methods; we have used the Newton-Raphson's method [83].

The above system of equations is based on the assumption that the same TAR=1 bits are sent and erased in each bus segment. However, as we have seen, in some cases the aggregate throughput of all priority classes located on the next downstream bus segment may be higher. Since the NSW_TNE mechanism tries to balance the throughputs in the two bus segments, the best it can do is to make the aggregate throughput in the upstream bus segment equal to 1. Then the aggregate throughput in the downstream segment is equal to the fraction of slots erased by the erasure node. Since for each erased slot a request will also be erased, no requets will pass to the upstream bus segment from the downstream stations.

It is now evident that once we have computed the values of T_i^p, from the previous equations, we have to check whether the aggregate throughput in all bus segments is less or equal to 1. If this is the case then the estimated throughputs are the correct ones. Otherwise, we find the most upstream bus segment for which the aggregate throughput is greater than one. Let S_{r_1} be this segment. We now divide the forward bus into two parts. The first part includes the bus segments S_0, S_1, ... S_{r_1}, and the second part the bus segments S_{r_1+1}, S_{r_1+2}, ... S_{N_E}. We then use the previous analytic method and derive the throughputs of all priority classes in the first part. Since S_{r_1} was the most upstream bus segment for which a higher than 1 aggregate throughput was observed, no bus segment in the first part can have an aggregate throughput greater than one and the computed throughputs will be the correct ones. We now use the same analytic method and derive the throughputs of the second part; where in the summations in equations (6.3.1) and (6.3.3) the throughputs of the various priority classes in the upstream bus segments, before S_{r_1+1}, are now known. If the derived throughput values do not make the aggregate throughput of any bus segment in the second part greater than 1, the computation will end. Otherwise we will repeat the same procedure, i.e. divide the second part into two parts and so on.

We have used the previous analytic method considering different configurations of stations and erasure nodes and we have found that it provides very good estimates for the throughputs. In the sequel we present two simple examples that clarify the above analytic method.

Example 1

We consider the network of Fig.6.5 which consists of 4 stations and one erasure node, located between stations "2" and "3", which divides the forward bus into S_0 and S_1 segments; the number of stations in the two bus segments are $N_0=2$ and $N_1=2$. Stations "1" and "3" support high priority traffic and stations "2" and "4" support medium priority traffic, with $M^h=6$ and $M^m=4$. The erasure node erases 50 % of all written slots that are passing by. Therefore, $a^h_{1,0} = 1/2 = a^h_{1,1} = a^m_{2,0} = a^m_{2,1}$ and $a^h_{3,1} = 1 = a^m_{4,1}$. Our unknowns are the throughputs T^h_1, T^m_2, T^h_3 and T^m_4 of the 4 stations. From (6.3.5) we select $X^h_1 = X^m_2$, $X^h_3 = X^m_4$ and $X^m_2 = X^m_4$ for our system of equations.

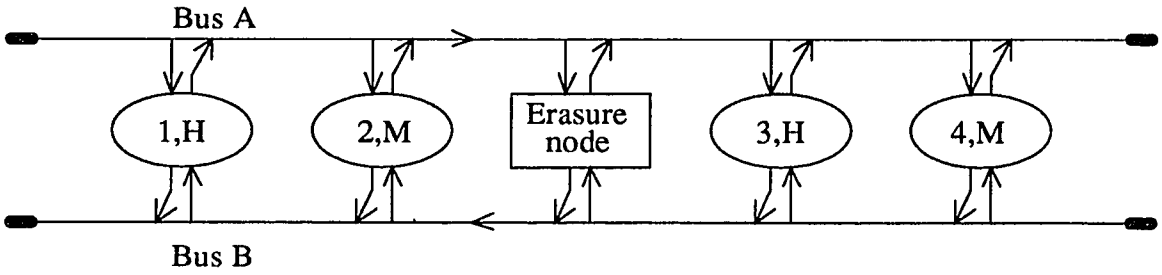

Fig.6.6: Network of Example 1. $M^h=6$, $M^m=4$. 50% of the busy slots passing in front of the erasure node are erased.

Substituting from (6.3.4) the expressions of X^p_i (where p=h,m and i=1,2,3,4) we get:

$$X^h_1 = \frac{T^h_1}{M^h} - T^m_2 - \left(T^m_4 - \frac{T^h_1 + T^m_2}{2} \frac{T^m_4}{T^h_3 + T^m_4} \right) = X^m_2 = \frac{T^m_2}{M^m} \qquad \text{(Ex.1)}$$

$$X^h_3 = \frac{T^h_3}{M^h} - T^m_4 - T^m_2 \frac{1}{2} = X^m_4 = \frac{T^m_4}{M^m} \qquad \text{(Ex.2)}$$

$$X_2^m = \frac{T_2^m}{M^m} = X_4^m = \frac{T_4^m}{M^m} \Rightarrow T_2^m = T_4^m \qquad \text{(Ex.3)}$$

The fourth equation is derived from the requirement that the total throughput in the last bus segment must be equal to 1, i.e. equation (6.3.6). The fraction of slots which are erased by the erasure node is $(T_1^h + T_2^m)/2$. Therefore, we have:

$$\frac{T_1^h + T_2^m}{2} + T_3^h + T_4^m = 1 \qquad \text{(Ex.4)}$$

We see that in the case of Example 1 all equations are linear and straightforward to solve; unfortunately in most of the cases they are nonlinear. Their solution provides the following values for the throughputs: $T_1^h = T_3^h = 42/69$ and $T_2^m = T_4^m = 4/69$. and $T_2^m = T_4^m = 4/69$. Since the aggregate throughput in each bus segment is less than one, $T_1^h + T_2^m = 46/69$ in S_0 and $T_3^h + T_4^m = 46/69$ in S_1, the computed throughputs are the correct ones, as we have also verified from simulation results.

Example2

In this example we consider a case where we have to break the forward bus into two parts. We consider the network of Fig.6.7 which consists of 4 stations and two erasure nodes that divide the forward bus into S_0, S_1 and S_2 bus segments. The number of stations in the three segments are $N_0 = 2$, $N_1 = 1$ and $N_2 = 1$. The first erasure node is located between stations "2" and "3", and the second between stations "3" and "4". Stations "1" and "2" support high priority traffic, and stations "3" and "4" medium priority traffic; with $M^h = 6$ and $M^m = 4$. Each erasure node erases 50 % of the written slots that are passing by; since the first erasure node erases 50 % of the busy slots, the second erasure node erases the 25 % of the slots that are written by stations "1" and "2". Therefore, $a_{1,0}^h = a_{1,1}^h = 1/2 = a_{2,0}^h = a_{2,1}^h = a_{3,1}^m = a_{3,2}^m$, $a_{1,2}^h = 1/4 = a_{2,2}^h$ and $a_{4,2}^h = 1$. Our unknowns are the throughputs T_1^h, T_2^h, T_3^m and T_4^m.

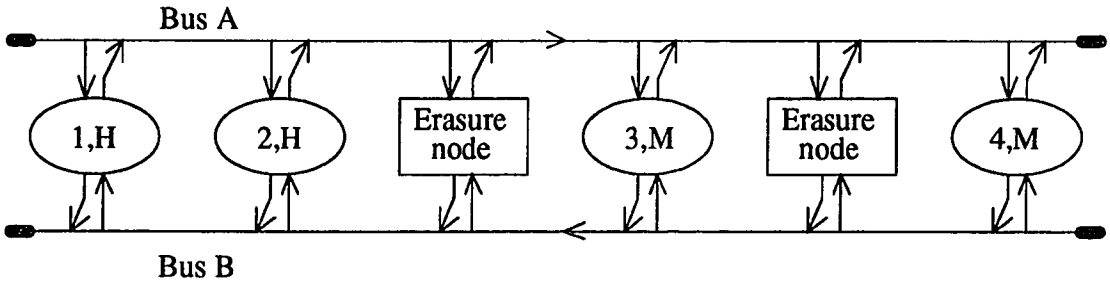

Fig.6.7: Network of Example 2. M^h=6, M^m=4. 50%, 25%-50% of the busy slots passing in front of the first and second erasure nodes, respectively, are erased.

From (6.3.5) we will use $X_1^h = X_2^h$, $X_1^h = X_3^m$ and $X_3^m = X_4^m$ for our system of equations. Substituting from (6.3.4) the expressions of X_i^p (where p=h,m and i=1,2,3,4) we get:

$$X_1^h = X_2^h \Rightarrow T_1^h = T_2^h \tag{Ex.5}$$

$$X_1^h = \frac{T_1^h}{M^h} - \left[T_3^m - \frac{T_1^h + T_2^h}{2} + T_4^m - \frac{T_1^h + T_2^h}{4} - \frac{T_3^m}{2} \right] = \frac{T_3^m}{M^m} \tag{Ex.6}$$

$$X_3^m = \frac{T_3^m}{M^m} = X_4^m = \frac{T_4^m}{M^m} \Rightarrow T_3^m = T_4^m \tag{Ex.7}$$

As in the previous example, the fourth equation is derived from the requirement that the aggregate throughput in the last bus segment must be equal to 1, from which we have:

$$\frac{T_1^h + T_2^h}{4} + \frac{T_3^m}{2} + T_4^m = 1 \tag{Ex.8}$$

The above system of linear equations can then be solved. It provides the following values for the throughputs: $T_1^h = T_2^h$ =42/81 and $T_3^m = T_4^m$ =40/81. We see that with previous values the aggregate throughput of segment S_0 becomes $T_1^h + T_2^h$ = 84/81, i.e greater than one. We therefore divide the forward bus into two parts. The first part

includes bus segment S_0 and the second part bus segments S_1 and S_2.

The solution for the first segment is very simple because equation (Ex.5) continues to hold, since stations "1" and "2" are on the same bus segment. Since $T_1^h = T_2^h$ and $T_1^h + T_1^h = 1$, the values of the throughputs are $T_1^h = T_2^h = 1/2$.

For the second part of the forward bus we can now use equations (Ex.7) and (Ex.8) which provide the following system of equations:

$$T_3^m = T_4^m \tag{Ex.9}$$

$$\frac{1}{4} + \frac{T_3^m}{2} + T_4^m = 1 \tag{Ex.10}$$

from which $T_3^m = T_4^m = 1/2$.

6.3.2 Delay Comparison

In this section we consider a dual bus network which supports three priority classes of traffic and compare the delay performance of ITU_NSW_TNE and BWB_DQDB under both priority mechanisms; the performance of ITU_NSW_TE is very similar with that of ITU_NSW_TNE and it is not shown.

In Fig.6.8 we consider the 20 stations system of Fig.6.2 that has erasure node station "9". We assume that each station supports high, medium, and low priority traffic; with $M^h = 6$, $M^m = 4$ and $M^l = 2$. The total offered load on the forward bus is 1.11 and is evenly distributed among the three classes. We assume a linear load for each class and that all classes transmit fixed size messages consisted of 20 segments. In Fig.6.7 we show the average message delay performance of ITU_NSW_TNE and BWB_DQDB in the case of the non-adaptive BWB over traffic classes priority mechanism.

We see that in the case of ITU_NSW_TNE the average message delay, in the first bus segment, first increases and then decreases as the station index increases. In the

second bus segment the station's delay is not affected by its location. The delays encountered by high and medium priority classes inside the same station are similar while delay of the low priority class is clearly higher. Although there is no significant difference in the delays of the high and medium priority classes in the case of ITU_NSW_TNE, these delays are lower than the delays encountered by the high and medium priority classes in the case of BWB_DQDB. In the case of BWB_DQDB the delays for both the high and low priority classes increase with the station index. The delay of the low priority is significantly higher due to the slots that BWB_DQDB wastes.

In Fig.6.9 we consider the system of Fig.6.8 and compare the delay performance of ITU_NSW_TNE and BWB_DQDB in the case of the adaptive BWB over traffic classes mechanism. We see, again, that in the case of ITU_NSW high and medium priority classes have similar delays whereas the delays of the low priority class is clearly higher, especially in the bus segment located before the erasure node. In the case of BWB_DQDB the delay of the medium priority class is higher than the corresponding delay of the high priority class. These delays are initially low, even lower than those for ITU_NSW, and increase with the station index. In fact, in the last station of the bus they become higher that those of ITU_NSW. Finally, the delay of the lower priority class is high on the first bus segment and significantly lower on the second one. Fig. 6.9 clearly shows that ITU_NSW provides the smallest delay variation for all three priority classes of traffic.

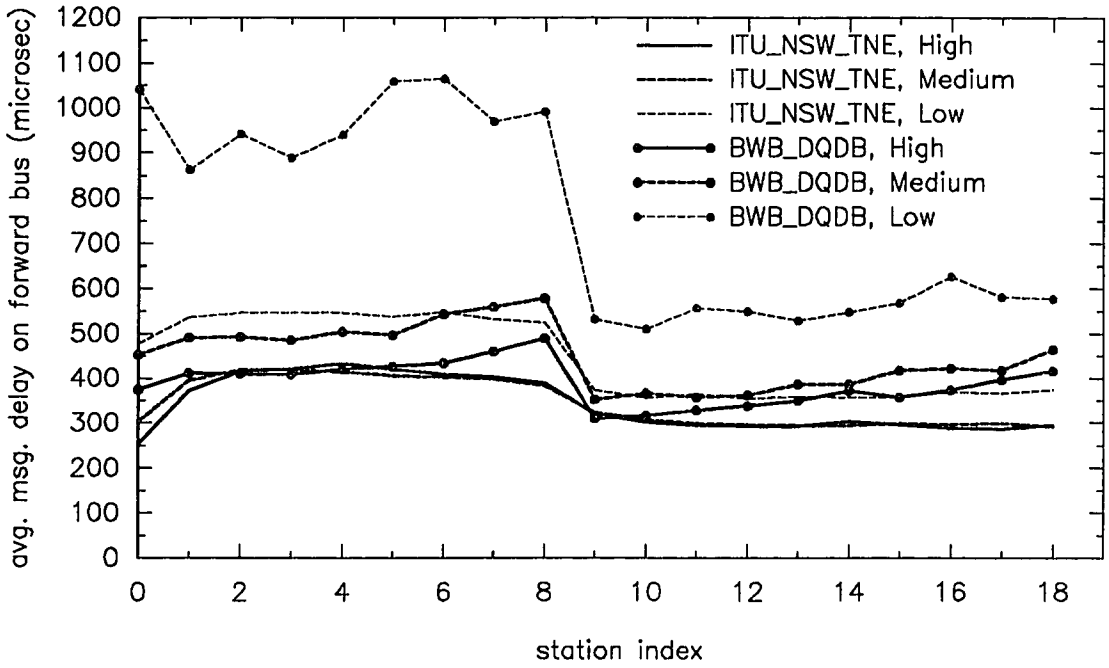

Fig.6.8: Slot reuse. Delay comparison of ITU_NSW_TNE and BWB_DQDB. BWB over priority classes. M_h=6, M_m=4, M_l =2. A total load of 1.11 is evenly distributed among the priority classes. Erasure node is station 9. Constant message size of 20 segments.

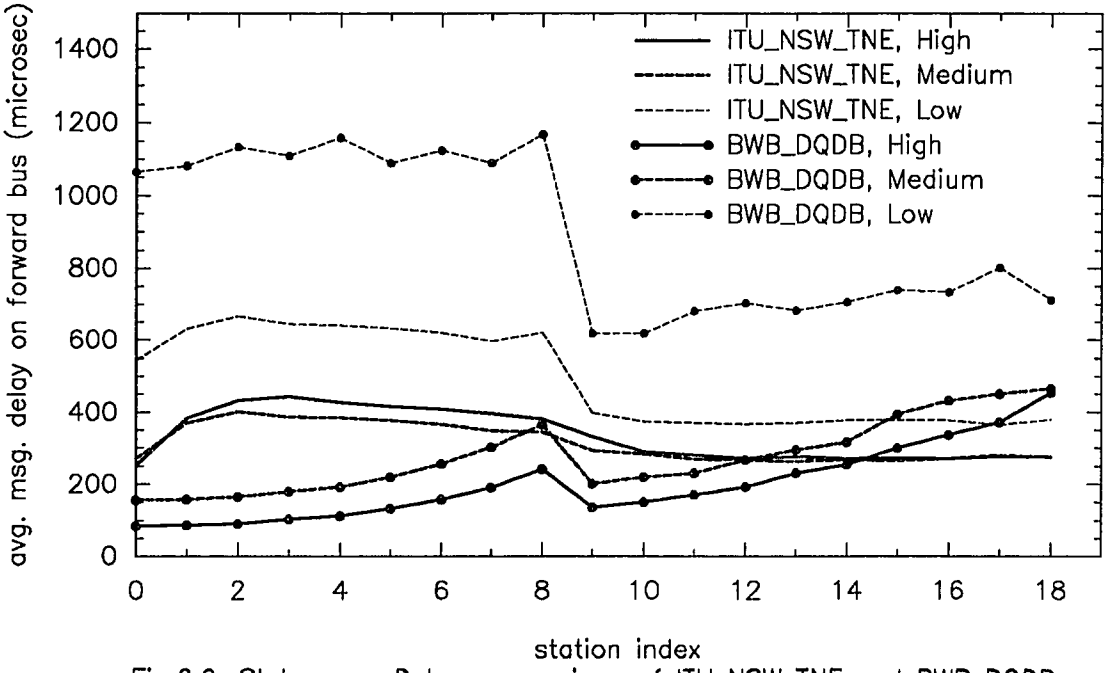

Fig.6.9: Slot reuse. Delay comparison of ITU_NSW_TNE and BWB_DQDB. Adaptive BWB over classes. M_h =6, M_m=4, M_l =2. A total load of 1.11 is evenly distributed among the priority classes. Erasure node is station 9. Constant message size of 20 segments.

6.4 Slot Reuse under the Absolute Priority Mechanism

In this section we investigate the performance of P_ITU_NSW and P_BWB_BQDB under the presence of erasure nodes. These two priority mechanisms which have been introduced in chapter 5 enable high priority users to "shut off" lower priority ones within a time interval equal to the end_to_end propagation delay.

In the case of P_ITU_NSW and P_BWB_DQDB the operation of the erasure nodes is as follows. Each priority class uses a separate ES_CTR, i.e ES_CTR_i for class "i". The erasure nodes erase every busy slot which is passing by and has been read by its destination. We now consider the conditions under which an erasure node increases the value of an ES_CTR. First we examine the case of BWB_DQDB. Let assume that the erasure node erases a slot and "k" is the highest priority class for which at least one of the following two conditions is true: a) RQ_CTR_k is greater than 0, b) there is a "k" priority segment queued for transmission at the erasure node. In this case, the erasure node will increase by 1 all the ES_CTRs of priority equal to or lower than "k". The reason is that every time a station inserts a request of priority "k" it is guaranteed that it has already sent a separate request for each priority less than "k". We now consider the case of P_ITU_NSW. The main difference here is that a traffic class inside a station (including the erasure node) may not send a request for every queued segment (because of the presence of TAR_segments). However, notice that when a segment of priority "i" arrives at a station requests of lower priority are sent on the reverse bus regardless of the status of the arriving segment, i.e whether it is a TAR_segment or not. For this reason the conditions that determine which ES_CTR will increase when the erasure node erases a slot must by slightly modified. That is, the erasure node should increase by one the values of the ES_CTRs of all priorities equal to or less than "k", where "k" is the highest priority class for which: a) $RQ\_CTR_k > 0$ or b) $RG\_CTR_k > 0$ or c) the highest priority segment queued for transmission is of priority "k+1", even though $RG\_CTR_{k+1}=0$. The part of the operation of the erasure nodes which is related to the reverse bus, for both P_BWB_DQDB

and P_ITU_NSW, remains identical to the one described in the previous sections. That is, whenever a request of priority "k" is seen and ES_CTR_k is greater than 0, the request is reset and ES_CTR_k is decreased by 1.

In Table 6.4 we examine the behavior of P_ITU_NSW_TNE and P_BWB_DQDB under overload conditions. We consider the same network configuration as in Table 6.3, but with the following bandwidth balancing parameters: $M^h=M^m=M^l=2$ for P_ITU_NSW_TNE and $M^h=M^m=M^l=8$ for P_BWB_DQDB. We consider, again, both cases, that is, when there is no erasure node and when there is one erasure node between stations "2" and "3". Table 6.4 shows that in the case of P_ITU_NSW_TNE the entire bandwidth is allocated to the highest priority class unless there is a lower priority user which is the only one that can reuse erased slots. For instance, such a case is shown by the third row entry of Table 6.4 in which the low priority class of station "3" uses all the erased slots. Similar is the behavior of P_BWB_DQDB with the difference that down-stream lower priority classes can utilize the idle bandwidth that higher priorities are forced to allow to pass by.

Table 6.4: Throughput comparison with and without slot reuse. Absolute priority mechanism. In the case of slot reuse 50% of the slots passing in front of the erasure node node are erased.

	P_ITU_NSW No slot reuse				P_ITU_NSW_TNE M=2				P_BWB_DQDB M=8			
Active classes	1,H	2,M	3,H	3,L	1,H	2,M	3,H	3,L	1,H	2,M	3,H	3,L
2,M	-	1.00	-	-	-	1.00 E	-	-	-	0.88 E	-	-
1,H 2,M	1.00	0.00	-	-	1.00	0.00 E	-	-	0.88	0.11 E	-	-
1,H - 2,M 3,L	1.00	0.00	-	0.00	1.00	0.00 E	-	0.50	0.88	0.11 E	-	0.45
1,H - 2,M 3,H - 3,L	0.50	0.00	0.50	0.00	0.67	0.00 E	0.67	0.00	0.35	0.00 E	0.73	0.08

In the remaining figures we consider three priority classes of traffic and compare the delay performance of P_ITU_NSW_TNE, P_BWB_DQDB and BWB_DQDB with the adaptive over classes BWB mechanism. We have assumed the same network configuration as in Fig 6.9. In this case all classes have the same value for the bandwidth balancing parameter M, i.e. $M^h=M^m=M^l=2$ in the case of P_ITU_NSW_BWB and $M^h=M^m=M^l=8$ in the case of P_BWB_DQDB.

In Fig 6.10 and 6.11 we compare the delay performance of P_ITU_NSW against BWB_DQDB and P_BWB_DQDB against BWB_DQDB respectively. Fig 6.10 shows that in the case of P_ITU_NSW users of the same priority class located at the same bus segment have similar delays with the delay of the priority classes located at the second bus segment being less than that of the priority classes located at the first bus segment. Furthermore, in the case of P_ITU_NSW all priority classes encounter significantly lower delays than in the case of BWB_DQDB. Fig 6.11 shows that P_BWB_DQDB has a similar delay performance with that of P_ITU_NSW. However, the delay variation among the stations of the same priority class is higher in this case. These two figures clearly show the superiority of the P_ITU_NSW priority scheme.

6.5 Conclusions

In this chapter we have investigated the performance of the various NSW and BWB_DQDB schemes under slot reuse. In the case of NSW we have considered two different variations, i.e. TNE and TE. In the TNE variation, the erasure node does not reset the TAR bits in the slots it releases. In the TE variation, the erasure node resets the TAR bits in the released slots. We have investigated the performance of both variations and have compared it with the corresponding performance of the current BWB mechanism of DQDB. We have found that both NSW mechanisms provide a higher aggregate throughput than BWB_DQDB; with the throughput of NSW_TE being, in many cases, higher that the throughput of NSW_TNE. All three schemes have the ability to provide

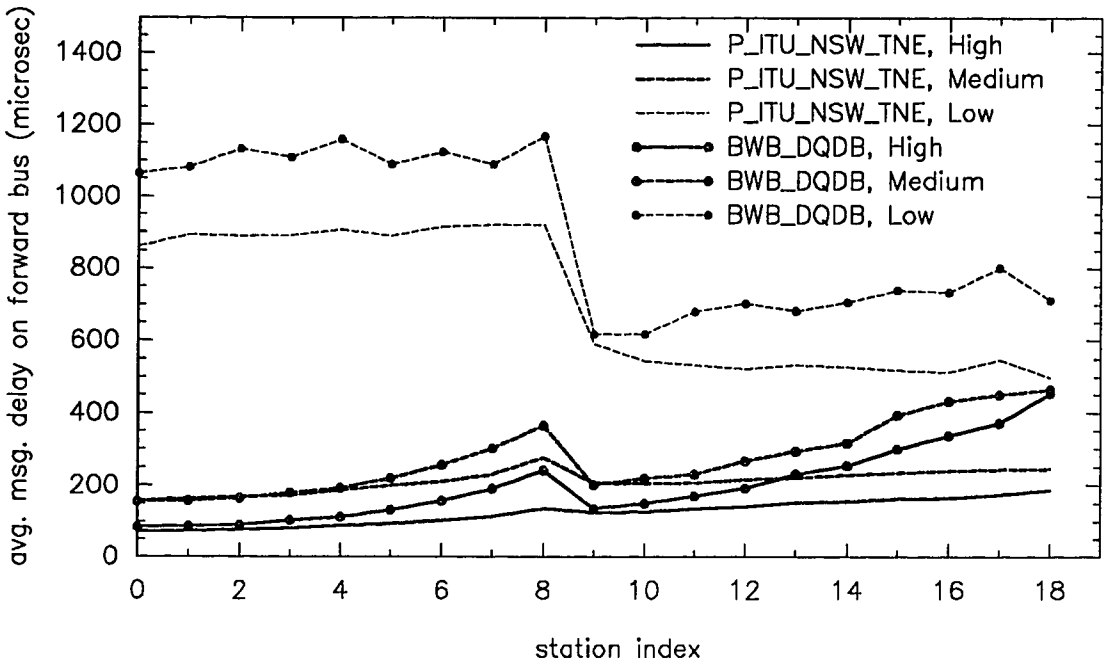

Fig.6.10:Delay comparison of P_ITU_NSW_TNE and adaptive over classes BWB_DQDB. A total load of 1.11 us evenly distributed among the priority classes. Erasure node is station 9. Constant message size of 20 segments.

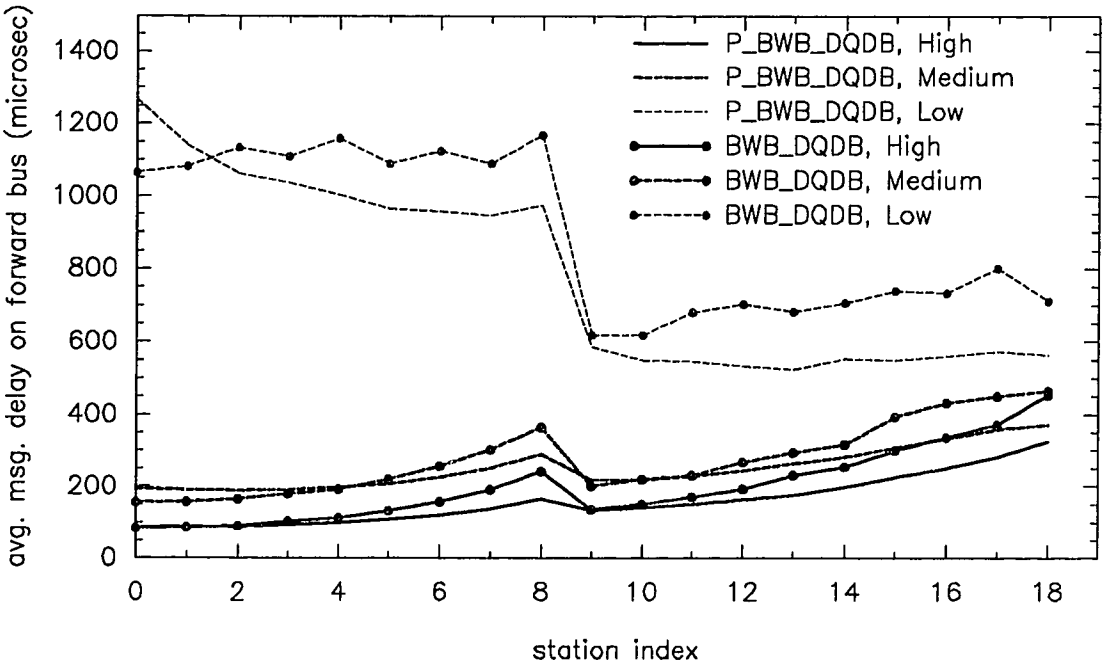

Fig.6.11:Delay comparison of P_ITU_NSW_TNE and P_BWB_DQDB. A total load of 1.11 us evenly distributed among the priority classes. Erasure node is station 9. Constant message size of 20 segments.

the stations that are located on the same bus segment with throughputs which are proportionate to their values of *M*. However, NSW_BWB_TNE has the ability to do that even among stations that belong to different bus segments. Based on these properties we have derived analytic expressions for the station throughput for both NSW_TNE and NSW_TE. In terms of message delay ITU_NSW demonstrates a substantial superior performance than all the other schemes. On the other hand BWB_DQDB has the highest message delay.

We have also investigated the performance of the various schemes in the presence of multiple priority classes of traffic. We have considered the non-adaptive, adaptive and absolute priority mechanisms. We have compared the performance of these schemes using simulation results and have provided a throughput analysis in the cases of NSW_TNE and P_ITU_NSW_TNE. Furthermore, we have compared their delay performance and have found that the absolute priority mechanism is the most effective one. It is the only scheme, under which higher priority users have better delay characteristics than lower priority users, regardless of their location on the bus. Finally, P_ITU_NSW_TNE demonstrates significantly lower delays than P_BWB_DQDB and smaller delay variation among the users of the same priority class.

CHAPTER 7

CONCLUDING REMARKS

In this chapter we outline the major points that have been presented in this dissertation. Then, we discuss some open issues that can serve as the basis for future research. We briefly describe a variation of the ITU_NSW mechanism, in which stations are able to delay the incoming busy slots. We outline this scheme and provide a sample of indicative preliminary results that describe its performance. Then, we identify additional related research topics which are propelled by the work presented in this dissertation.

7.1 Conclusions

The limitations in DQDB and BWB_DQDB, along with the importance of supporting a high variety of services over high speed MANs, have motivated our research interest in this area. We have shown that DQDB does not have the ability to provide fairness among the stations, both under overload and underload conditions. Furthermore, its unfairness is intensified as the network size increases. BWB_DQDB can provide the requested bandwidth to the lightly loaded stations and evenly distribute the remaining bandwidth among the overloaded ones. However, it slowly converges to the fair state and in order to do so, requires the wastage of channel slots. The less the number of stations or the higher the convergence speed, the greater the bandwidth wastage. In the case of underload conditions the station location drastically affects its performance. Furthermore, the average station is significantly higher than that of DQDB.

In this dissertation we have proposed a geometric solution to the problem of the DQDB fairness, the RSG scheme. With RSG the responsibility for generating slots rotates around the bus, as stations take turns being slot generators. We have investigated two switching mechanisms for the SG, the IS_SG and the SS_SG. We have shown that RSG has the ability to provide all stations with similar delays and throughputs. In the presence

of multiple priority classes of traffic, RSG continues to be fair. Moreover, the average segment delays of the higher priority users are significantly lower than those of the lower priorities. However, RSG is a fair solution for the DQDB network in the "long term", that is, when a full rotation of the SG is considered. In this respect, RSG, as well as any other topological solution, may not be appropriate to support services with very bursty loads and strict delay requirements.

Motivated by the challenge to investigate MAC mechanisms that can support real time traffic, we have proposed the NSW_BWB and ITU_NSW schemes that can react fast to different load configurations and reach a fair state within a few slots. NSW_BWB and ITU_NSW have similar complexity with that of BWB_DQDB, but much better performance. Their operation does not require the wastage of any channel slots and for this reason they can converge very fast to the fair state. We have thoroughly investigated their performance and provided analytical estimates for the station throughputs. We have also verified their faster convergence and ability to distribute the available bandwidth in any arbitrary way among the stations. Moreover, we have demonstrated that their delay behavior is superior to that of BWB_DQDB. Finally, we have investigated their ability to support multi-priority traffic and have shown that existing priority mechanisms can be easily implemented by both ITU_NSW and NSW_BWB. Furthermore, the resulting priority schemes in the case of ITU_NSW are superior to the ones in the case of BWB_DQDB. However, all these priority mechanisms in essence require by the high priority users to use larger values for their bandwidth balancing parameter M. As a result they slow down their convergence speed towards the fair state and make high priority vulnerable to transient low priority overloads. This significantly limits their effectiveness to support real time traffic. For this reason, we have proposed a novel, very effective priority mechanism that enables high priority users to have almost immediate access on to the medium, regardless of the presence of lower priority traffic. We have shown how the new priority mechanism can be implemented in the cases if ITU_NSW, NSW_BWB

and BWB_DQDB. We have also compared, through simulation, the performance of these priority schemes. Our simulation results have shown that, under any aspect, the new priority mechanism is superior than the existing ones. Under overload conditions, it evenly distributes the entire bandwidth among the highest priority users. In underload conditions, provides higher priority users with significantly better performance characteristics than lower priority ones and the delay variation among users of the same priority becomes minimal. Our simulation results have also shown that the P_ITU_NSW version of the new priority mechanism demonstrates the best performance. It provides the lowest delays to the high priority class, makes the presence of lower priority traffic almost transparent to the higher priority one, and minimizes the effect of the station location on both throughput and delay performance. The efficiency of P_ITU_NSW becomes stronger under the presence of voice and video traffic. We have shown that the performance of the various voice and video sources is not affected by their distribution on the bus, nor the presence of data traffic. Based on these appealing performance characteristics, we have analytically estimated the maximum number of voice and video sources P_ITU_NSW can support.

Finally, in this dissertation we have investigated the performance of the NSW schemes under the presence of erasure nodes. We have considered two variations, i.e. the TNE and TE, and compared their performance with the corresponding performance of BWB_DQDB under single and multiple priority classes of traffic. Our analysis has once again verified the superior performance of the NSW mechanism.

7.2 Current Related Research Activity

In the case of NSW, a downstream station receives its bandwidth by requesting extra slots (from the upstream stations) at the rate it observes TAR=1 bits on the forward channel. In order for the downstream station to write into it has to wait (in the worst case) for the request to go upstream and make the reservation and then for the extra slot to arrive

at the station. It is evident that we can improve the performance of NSW, if we allow a station to store and overwrite a passing slot and then reinsert it in the channel. Since the main objective is to balance the bandwidth as soon as possible we propose that a station can only store and overwrite slots that carry TAR=1 bits, which the station intented to erase. In this way stations have immediate access to the bus whenever they erase a TAR=1 bit and send an extra request. In order to emphasize the immediate transmission ability of this scheme we use the term Immediate Transmission NSW (ITR_NSW) to refer to it.

We have already built the basic simulator of ITR_NSW and initial simulation results demonstrate its ability to fairly distribute the available bandwidth among the stations. It is evident that ITR_NSW can converge faster than NSW to the fair state. For instance, when two stations are overloaded the convergence is immediate regardless of the initial loading.

In Fig 7.1 we compare the delay performance of ITR_NSW and ITU_NSW under independent segment transmissions. We have considered a network consisted of 20 stations with interstation distance equal to 2 slots. The total offered load is 0.9 and is linearly distributed. In Fig 7.1 we plot the average segment delay of the stations. We define as segment delay the time interval from the arrival of a segment at the station's queue until the transmission of the first bit on the bus. Fig 7.1 shows We see that for ITR_NSW the delay variation is lower than that of ITU_NSW. However, in the case of ITR_NSW the above defined value of delay may not provide the complete picture on the performance of a station since, the transmitted segments maybe delayed by intermediate stations before they reach their destination. In the case of ITR_NSW we should also consider additional performance parameters. It is evident that additional performance measures are needed and these may include the additional delay that is added to a segment until it reaches the end of the bus, or until it is read by its destination and the average or maximum number of delayed segments which are present at a given instant inside a

station. For the system we examine in Fig 7.1 the additional segment delay in the intermediate stations is not significant. Segments originated by station "0" are those with the highest additional delay, which is equal to 1.13 μsec. Furthermore, the maximum number of segments that were buffered into a particular station was only 3 (in station "12"). This result, together with the higher convergence speed of ITR_NSW over ITU_NSW demonstrate its high potential. It is interesting to examine in detail its performance characteristics under single and multiple priority classes and investigate its suitability to accommodate real time traffic.

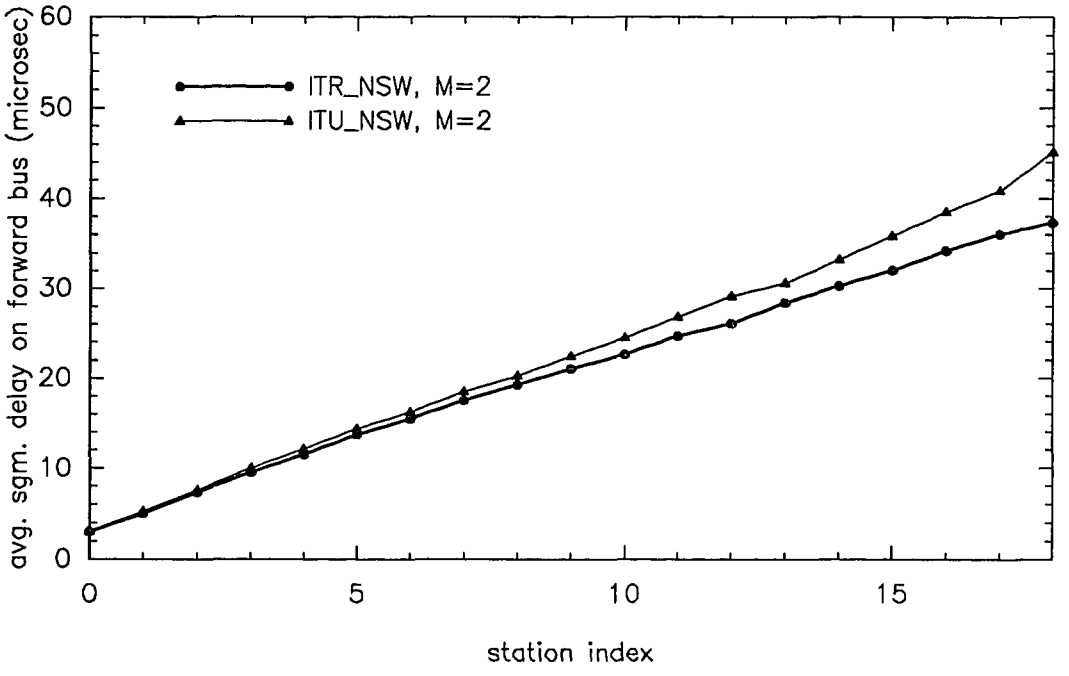

Fig.7.1:Delay comparison of ITR_NSW, and ITU_NSW. Bus utilization 0.9. Independent segment transmission.

7.3 Directions for Future Research

The main objective in the area of MAC mechanisms for MANs is to propose and investigate efficient control mechanisms which are appropriate for supporting applications with very diverse traffic requirements. P_ITU_NSW (or potentially P_ITR_NSW) is a very significant step in this direction and can serve as the basis for future research.

In this dissertation we have provided an analytic queuing model for ITU_NSW in the case of segment transmissions and in the absence of the CD_CTR. It would be very interesting to try to extend this analysis to the case of message transmissions in both the presence or absence of the CD_CTR. Furthermore, it would also be extremely interesting and useful to provide a queuing analytic model for the P_ITU_NSW priority mechanism.

A very important research area is the investigation of flow and congestion control mechanisms that are appropriate for high seed MANs. The need for such mechanisms arises from the fact that the majority of the traffic in future networks will be bursty. The size of the traffic bursts and the inter burst time will be different for various sources and may also vary with time for the same source. One approach for serving these sources could be to accept a new one into the network only if the network could provide its peak rate. However, since a bursty source only for a small interval during its connection time transmits at peak rate, this method would be inefficient and the cost of the connection too high. Therefore, the need for statistical multiplexing naturally arises because it can achieve a more efficient use of the channel bandwidth and keep the cost of the connection low. It is obvious that in this case congestion may arise and some kind of control is needed to prevent it or deal with it if it occurs. P_ITU_NSW alone cannot deal with this situation. P_ITU_NSW can guarantee that if there is available bandwidth the highest priority users will acquire it. However, there is no much it can do when the aggregate requested bandwidth by all the highest priority users is greater than the channel bandwidth. It will distribute what is available among them equally and make everybody equally unhappy. Therefore, it becomes evident that additional control, on top of P_ITU_NSW, is needed to decide on the acceptance of a call. The investigation of such a type of control is a very interesting and challenging problem. It is also one of the main objectives of my research activity in the near future.

REFERENCES

[1] F. Tobagi, F. Borgonovo, and L. Fratta, "EXPRESSNET: A High Performance Integrated Services Local Area Network", *Journal Select. Areas Commun.*, vol SAC-1, No 5, Nov. 1983

[2] J.O. Limb and L.E. Flamm, "A Distributed Local Area Network Packet Protocol for Combined Voice and Data Transmission", *Journal Select. Areas Commun.*, vol SAC-1, No 5, Nov. 1983

[3] F.E. Ross "FDDI - A Tutorial", *IEEE Communications Magazine,* May 1986

[4] V.I. Iyer and S. Joshi, "FDDI's 100 Mbps protocol improves on 802.5 spec's 4 Mbps limit", EDN, May 2, 1985

[5] Fiber Distributed Data Interface (FDDI) - Token Ring Media Access Control (MAC), ANSI, REV-10, Febr. 1986

[6] F.E. Ross, "An Overview of FDDI: The Fiber Distributed Data Interface", *Journal Select. Areas Commun.*, vol SAC-7, No 7, September 1989

[7] IEEE 802.6 Working Group, "Proposed Standard: Distributed Queue Dual Bus (DQDB) Subnetwork of a Metropolitan Area Network," Unapproved Draft P80.6/D12, February 1990.

[8] Z.L. Budrikis *et. al.*, "QPSX: A Queue Packet and Synchronous Circuit Exchange," in *Proc. 8th Int. Conf. Comput. Commun. ICCC '86,* Munich, West Germany, pp. 288-293, Sept. 15-19, 1986.

[9] M. Conti *et. al.*, "A Methodological Approach to an Extensive Analysis of DQDB Performance and Fairness," *IEEE J. Select. Areas Commun.*, vol. SAC-9, no. 1, pp. 76-87, January 1991.

[10] D. Karvelas and M. Papamichail, "A Simulation Model for DQDB", *Proceed. of Twenty-Second Annual Pittsburgh Conference on Modeling and Simulation,* Pittsburgh, pp. 2265-2272, May 1991.

[11] J.W. Wong, "Throughput of DQDB networks under heavy load," in *Proc. 7th Europ. Fibre Optic Commun. Local area Networks Expo. EFOC/LAN '89,* Amsterdam, The Netherlands, pp. 146-151, June 1989.

[12] D. Davids and T. Welzel, "Performance analysis of DQDB based on simulation," in *3rd IEEE Workshop on Metropolitan Area Networks,* San Diego, CA, pp. 7.6.1-7.6.15, March 28-30, 1989.

[13] H.R. van As, J.W. Wong, and P. Zafiropoulo, "Fairness, Priority and Predictability of the DQDB MAC Protocol under Heavy Load", *1990 Intl. Seminar on Digital Communications,* Zurich, Switzerland, pp. 410-417, May 5-8, 1990.

[14] M. Conti *et. al.*, "DQDB under heavy load: Performance evaluation and

unfairness analysis", *Proc. INFOCOM '90,* San Francisco, CA, pp. 313-320, June 5-7, 1990.

[15] E.L. Hahne, A. Choudhury, and N.F. Maxemchuk, "Improving the Fairness of Distributed Queue Dual Bus Networks", *Proceed. INFOCOM '90,* San Francisco, CA, pp. 175-184, June 1990.

[16] J. F. Mollenauer, "Standards for Metropolitan Area Networks", *IEEE Communications Magazine,* Vol. 26, No. 4, pp. 15-19, April 1984.

[17] M. Zukerman, "QPSX - The Effect of Circuit Allocation on Segment Capacity under Burst Switching", *Proceed. ICC '88,* Philadelphia, PA, pp. 599-603, June 1988.

[18] M. Zukerman, "Queueing Performance of QPSX", *Proceed. 12th Intern. Teletraffic Congress (ITC),* Torino, Italy, June 1988.

[19] M. Zukerman, "Overload Control of the Isochronous Traffic in QPSX", *Proceed. IEEE GLOBECOM '88,* pp. 1241-1245, Hollywood, FL, Nov. 1988.

[20] M. Zukerman, "Bandwidth Allocation for Bursty Isochronous Traffic in a Hybrid Switching System" *IEEE Trans. Commun.,* vol COM-37, No 12, December 1989.

[21] W. Jeon, C. Kim, and D.K. Kim, "Design of a Distributed Isochronous Channel Management Protocol for IEEE 802.6 MAN QPSX, *Proceed. IEEE INFOCOM '89,* pp. 268-275, Ottawa, Ont., Canada, April 1989.

[22] C. Bisdikian, "A Performance Analysis of the IEEE 802.6 (DQDB) Subnetwork with the Bandwidth Balancing Mechanism", *IBM Res. Report, RC 16450,* January 1991.

[23] M. Spratt, "Allocation of Bandwidth in IEEE 802.6 with Non-Unity Ratio Bandwidth Balancing", *Proceed. ICC '91,* Denver, CO, pp. 729-735, June 1991.

[24] M. Spratt, "Bandwidth Allocation over Several Networks with DQDB", *Proceed. EFOC/LAN '91 Conf.,* London, UK, June 1991.

[25] B. Mukherjee and S. Banerjee, "Alternatives Strategies for Fairness in and an Analytical Model of DQDB Networks", *Proceed. INFOCOM '91,* Bal Harbour , FL, pp. 879-888, April 1991.

[26] S.F. Chang, D.G. Messerschmitt, and A. Albanese, "Adaptable-Bit-Rate Video Services on DQDB Networks", *Proceed. ICC '91,* Denver, CO, pp. 836-841, June 1991.

[27] E.Y. Huang and L.F. Merakos, "On the Access Fairness of of the DQDB MAN Protocol", *Proceed. IEEE 1990 Phoenix Conference,* Phoenix, AZ, pp. 556-559, March 1990.

[28] K.M. Khalil and M.E. Koblentz, "A Fair Distributed Queue Dual Bus Method",

Proceed. IEEE 14th Conf. on Local Computer Networks, Minneapolis, MN, pp. 180-188, Oct. 1989.

[29] J. Filipiak, "Access Protection for Fairness in a DQDB MAN", *Proceed. ICC '89,* Boston, MA, pp. 635-639, June 1989.

[30] A. Lombardo, *et. al.*, "A Fair featured Metropolitan Area Network", *Proceed. IEEE 15th Conf. on Local Computer Networks,* Minneapolis, MN, pp. 426-430, Oct. 1990.

[31] M.C. Yuang and H.T. Chang, "AFair and Fast Protocol for DQDB Metropolitan Area Networks", *Proceed. IEEE 16th Conf. on Local Computer Networks,* Minneapolis, MN, pp. 535-543, Oct. 1991.

[32] H.R. van As, "Performance Evaluations of the Bandwidth Balancing in the DQDB MAC Protocol", *8th Annual EFOC/LAN Conference,* Munich, Germany, pp. 231-239, June 27-29, 1990.

[33] H.R. van As, "Major Performance Characteristics of the DQDB MAC Protocol", *SBT/IEEE Intl. Telecommunications Symposium (ITS '90),* Rio de Janeiro, Brazil, pp. 113-119, Sept. 3-6, 1990.

[34] V. Catania, *et. al.*, "Throughput Analysis of DQDB in Overload Conditions", *Proceed. ICC '91,* Denver, CO, pp. 741-747, June 1991.

[35] E.L. Hahne and N.F Maxemchuk, "Fair Access of Multi-Priority Traffic in DQDB", *Proceed. INFOCOM '91,* Bal Harbour , FL, pp. 889-900, April 1991.

[36] I.J. Hyun and K.J. Han, "Dynamic Bandwidth Balancing Mechanism for Improving DQDB Performance", *Proceed. ICC '91,* Denver, CO, pp. 1345-1349, June 1991.

[37] D.G. Jeong, C.H. Choi and W.S. Jeon, "Fairness Improvement in DQ Protocol with Multiple Priority Classes", *Proceed. ICC '91,* Denver, CO, pp. 1340-1344, June 1991.

[38] S.Y. Kim and K.J. Han, "An Enhanced Bandwidth Balancing Mechanism of the DQDB MAN", *Proceed. ICC '91,* Denver, CO, pp. 423-427, June 1991.

[39] C. Bisdikian and A. Tantawy, "A Mechanism for Implementing Preemptive Priorities in DQDB Subnetworks", *Proceed. ICC '91,* Denver, CO, pp. 1062-1067, June 1991.

[40] M.A. Rodrigues, "Erasure Nodes: Performance Improvements for the IEEE 802.6 MAN", *Proc. INFOCOM '90,* San Francisco, CA, pp. 636-643, June 5-7, 1990.

[41] M.W. Garett and S.Q. Li, "A Study of Slot Reuse in Dual Bus Multiple Access Networks", *Proc. INFOCOM '90,* San Francisco, CA, pp. 617-629, June 5-7, 1990.

[42] S. Luigi, *et. al.*, "The Balanced Erasure Node: A Mechanism for Slot Reuse in DQDB protocol", *Proceed. ICC '91*, Denver, CO, pp. 1350-1354, June 1991.

[43] S Banerjee and B. Mukherjee, "Incorporating Continuation-of-Message (COM) Information, Slot Reuse, and Fairness in DQDB Networks", *Tech. Report, CSE-90-42*, Div. of Computer Science, University of California, Davis, CA, Oct. 1990.

[44] A.E. Kamal, "Efficient Multi-Segment Message Transmission with Slot Reuse in DQDB", *Proceed. INFOCOM '91*, Bal Harbour , FL, pp. 869-878, April 1991.

[45] B. Mukherjee and A.E. Kamal, "Scheduling Variable-Length Messages on Slotted High Speed Fiber Optic LASNs/MANs using the Continuation-Bit Approach", *Proceed. INFOCOM '91*, Bal Harbour , FL, pp. 678-687, April 1991.

[46] P.G. Potter and M. Zukerman, "A Discrete Shared Processor Model for DQDB", *Computer Networks and ISDN Systems*, vol. 20, pp. 217-222, Dec. 1990.

[47] L. Kleinrock, *Queueing Systems, Volume II: Computer Applications*, Jon Wiley, 1976.

[48] C. Bisdikian, "Waiting Time Analysis in a Single Buffer DQDB (802.6) Network", *Journal Select. Areas Commun.*, vol SAC-8, No 10, Oct. 1990.

[49] P. Tran-Gia and T. Stock, "Approximate Performance Analysis of the DQDB Access Protocol", *Computer Networks and ISDN Systems*, vol. 20, pp. 231-240, Dec. 1990.

[50] C. Bisdikian, "A Queueing Model with Applications to Bridges and the DQDB MAN", *IBM Res. Report, RC 15218*, Dec. 1989.

[51] C. Bisdikian, "A Queueing Model for a Data Station within the IEEE 802.6 MAN", *IBM Res. Report, RC 15587*, March 1990.

[52] R. Landry and I. Stayrakakis, "A Three Priority queueing Policy with Applications to Communication Networks" *Proceed. INFOCOM 1993* pp. 1067-1075, San Fransisco, CA, March 1993.

[53] I. Stayrakakis and S. Tsakiridou, " A Queueing Sysytem with Markov-stractured Service Availabilities" *Proceed. INFOCOM 1993* pp. 1083-1091, San Fransisco, CA, March 1993.

[54] J.O. Limb, "Load-Controlled Scheduling of Traffic on High-Speed Metropolitan Area Networks", *IEEE Trans. Commun.*, vol COM-37, No 11, Nov. 1989.

[55] P. Jacquet and P. Muhethaler, "Lightnet: A Class of Efficient High-Speed Access Protocols", *Proceed. ICC '90*, pp. 1673-1679, Atlanta, GA, April 1990.

[56] Y.S. Leu and D.H.C. Du, "Cycle Compensation Protocol: A Completely Fair Protocol for the Uni-Directional Twin-Bus Architecture", *Proceed. IEEE 15th Conf. on Local Computer Networks*, Minneapolis, MN, pp. 416-425, Oct. 1990.

[57] M. Conti, *et. al.*, "DCP: A Fully Distributed MAC Protocol Exploiting the Capabilities of Polling Systems", *Proceed. IEEE 15th Conf. on Local Computer Networks,* Minneapolis, MN, pp. 320-326, Oct. 1990.

[58] M. Papamichail and D. Karvelas, "A Simulation Study of DCP", in *Proceed. of Twenty-third Pittsburgh Conference on Modeling and Simulation,* Pittsburgh, pp. 2495-2502, April 30-May 1, 1992.

[59] M. Mehdi Nassehi, "CRMA: An Access Scheme for High-Speed LANs and MANs", *Proceed. ICC '90,* pp. 1697-1702, Atlanta, GA, April 1990.

[60] H.B. Muller *et. al.*, "DQMA and CRMA: New Access Schemes for Gbits/s LANs and MANs", *Proc. INFOCOM '90,* San Francisco, CA, pp. 185-191, June 5-7, 1990.

[61] B. Mukherjee and J.S. Meditch, "The π- Persistent Protocol for Unidirectional Broadcast Bus Networks", *IEEE Trans. Commun.,* vol COM-36, No 12, pp. 1277-1286, December 1988.

[62] B. Mukherjee and J.S. Meditch, "Integrating Voice with the π- Persistent Protocol for Unidirectional Broadcast Bus Networks", *IEEE Trans. Commun.,* vol COM-36, No 12, pp. 1287-1295, December 1988.

[63] B. Mukherjee, *et. al.*, "Dynamic Control of the p_i- Persistent Protocol Using Channel Feedback", *Proceed. IEEE INFOCOM '89,* pp. 858-865, Ottawa, Ont., Canada, April 1989.

[64] Y. Gong and M. Paterakis, "Performance Analysis of a Flexible Protocol Achieving User Fairness in High-Speed Dual-Bus Networks with Destination Release", *Proceed. INFOCOM '91,* Bal Harbour , FL, pp. 479-488, April 1991.

[65] I. Cidon and Y. Ofek, "Distributed Fairness Algorithm for Local Area Networks with Concurrent Transmissions", *Proceed. of the 3rd Intern. Workshop on Distr. Algorithms,* pp. 57-69, Sept. 1989.

[66] I. Cidon and Y. Ofek, "Metaring - A Full-Duplex Ring with Fairness and Spatial Reuse", *Proc. INFOCOM '90,* San Francisco, CA, pp. 969-981, June 5-7, 1990.

[67] J. Chen, H. Ahmadi and Y. Ofek, "Performance Study of Metaring with Gb/s Links", *Proceed. IEEE 16th Conf. on Local Computer Networks,* Minneapolis, MN, pp. 137-147, Oct. 1990.

[68] Special Report, "Gigabit Network Testbeds", *IEEE Computer Magazine,* vol. 24, No 9, pp. 77-80, 1990.

[69] Y. Ofek, "Integration of Multi-ring on the Metaring Architecture", *IBM Res. Report, RC 16058,* August 1990.

[70] D. Karvelas, M. Papamichail, and G. Polyzos, "Performance Analysis of the Rotating Slot Generator Scheme" in *Proceed. INFOCOM '92,* Florence, Italy, pp. 794-803, May 6-8 1992.

[71] D. Karvelas, M. Papamichail, and G. Polyzos, "The Rotating Slot Generator Dual Bus", in *Proceed. of Twenty-third Pittsburgh Conference on Modeling and Simulation,* Pittsburgh, pp. 2337-2344, April 30-May 1, 1992.

[72] D. Karvelas, M. Papamichail, "DQDB: A Fast Converging Bandwidth Balancing Mechanism that Requires No Bandwidth Loss", in *Proceed. ICC '92,* Chicago, pp. 142-146, June 14-18, 1992.

[73] D. Karvelas, M. Papamichail, "Performance Study of a New Bandwidth Balancing Mechanism under a Single and Multiple Priority Classes of Traffic", in *Proceed. of First International Conference on Computer Communications and Networks,* San Diego, California, pp. 102-107, June 8-10, 1992.

[74] D.E. Karvelas and M. Papamichail, "The No Slot Wasting Bandwidth Balancing Mechanism for Dual Bus Architectures", accepted and will appear in *IEEE Journal Selected Areas on Communications;* also CIS-92-15 Technical Report, New Jersey Institute of Technology.

[75] A. M. Viterbi, " Approximate Analysis of Time-Syncronous Packet Networks", *IEEE Journal on Sel. Areas of Commun.,* pp. 879-890, September 1986.

[76] J. Spragins, "Loop Transmission Sysytems-Mean Value Analysis" *Transaction on Communications,* pp. 592-602, June 1972.

[77] P. Brady, "A Technique for Investigation On-Off Patterns in Speech", *BSTJ, vol 47,* January 1968.

[78] P. Brady, "A Statistical Analysis of On-Off Patternsin 16 Conersations", *BSTJ, vol 47* January 1968

[79] P.Brady, "A Modele for generating On-Off patterns in Two Way Conersations", *BSTJ, vol.48,* January 1969

[80] B. Maglaris, "Performance Models of Statistical Multiplexing in Packet Video Communications", *Transactions on Communications,* July 1988

[81] D. Karvelas, M. Papamichail, "Performance Analysis of a New Bandwidth Balancing Mechanism under the Presence of Erasure Nodes", *Proceed. INFOCOM '93, San Francisco, pp. 1091-1098, March 30 - April 1, 1993.*

[82] M. Zukerman and P. G. Potter, "A Protocol for Eraser Node Implementation within the DQDB Framework" *Proceed. GLOBECOM '90,* San Diego, CA, pp. 1400-1404, December 2-5, 1990.