INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps. Each original is also photographed in one exposure and is included in reduced form at the back of the book.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

UMI

A Bell & Howell Information Company
300 North Zeeb Road, Ann Arbor MI 48106-1346 USA
313/761-4700 800/521-0600

On the Grouping of a Dual Bus Configuration and Its Applications

by

Weiping Meng

A Dissertation Submitted to the Graduate Faculty in Engineering in Fulfilment of the Requirement for the Degree of Doctor of Philosophy, The City University of New York

1997

This manuscript has been accepted by the graduate faculty in Engineering in satisfaction of the dissertation requirement for the degree of Doctor of Philosophy.

9/19/1997
Date:

Chairman of the Examining Committee:
Dr. Tarek N. Saadawi, Professor of Electrical Engineering, The City College of The City University of New York.

9/19/97
Date:

Executive Officer:
Professor Gerard Lowen, Dean of Graduate Studies at the Engineering School, The City College of New York.

Dr. Ibrahim W. Habib
Co-Mentor, Assistant Professor,
Department of Electrical Engineering,
The City College of New York

Dr. Myoung Lee
Assistant Professor,
Department of Electrical Engineering,
The City College of New York

Dr. Mitra Basu
Associate Professor,
Department of Electrical Engineering,
The City College of New York

Dr. Brian Schwartz
Professor,
Director of the CUNY Multimedia Lab,
The City University of New York

The City University of New York

ABSTRACT

On the Grouping of a Dual Bus Configuration and Its Applications

By

Weiping Meng

Advisor: Professor Tarek N. Saadawi

This dissertation focuses on improving performance of metropolitan area network (MAN) or local area network (LAN) protocols at media access control (MAC) layers which are based on a dual/fold bus topological configuration. In order to enhance the performance, a simple, efficient, and versatile scheme is presented in the dissertation. The scheme is called grouping transmission, denoted by *GT*. It comprises two main ideas: (1) Dividing all the nodes in the network into multiple logical groups and assigning a group address to each group. (2) Setting up a post-source group transmission rule by which a busy slot carrying a segment (or a request) may carry new information (segments or requests) after the slot departs from its source group.

Several schemes have been proposed for improving throughput of dual bus MANs/LANs before. They are *Destination Release (DR), Erasure Nodes (EN), Previous Slot Information (PSI)*, and *SP-DQDB*. The common advantage of the scheme, *GT*, over those schemes is: it can potentially support lower access delay and higher throughput in a simple and versatile (Slot Reuse, Pre-Use, or both) way.

Application of the *GT* scheme to two existing MAC protocols, DQDB and S++, produces six novel protocols. These protocols are group addressing distributed queue dual bus (DQDB-GA), group requesting distributed queue dual bus (DQDB-GR), non-post-request slot-pre-use distributed queue dual bus (DQDB-NSP), grouping distributed queue dual bus (DQDB-G), high throughput distributed queue dual bus (DQDB-H), and grouping addressing S++ (S++-GA).

Eight theorems on the throughput gains of dual bus MANs/LANs using the proposed scheme are founded and theoretical analysis results based on these theorems are exhibited in curves.

Simulation performance analysis studies 1) show that the new-designed protocols have higher performance or more flexible than existing ones and 2) verify that theoretical results are correct. Hence, the scheme can be efficiently utilized to support high speed data transmission.

Dedication

To My Family

Acknowledgments

"A journey of a thousand miles must begin with a single step."

—— Lao-Tzu (550 BC)

It is a great pleasure to acknowledge my deep appreciation to my advisor: Professor Tarek Saadawi for his guidance, encouragement, support, and various helps. It is specially worthwhile mentioning that I have been assigned to be his teaching assistant almost every semester since I registered here, during which I learned a lot in telecommunication network, including fundamental of the DQDB protocol that may be thought of as a starting point of this dissertation. The result from his guidance and support is that I am obtaining my Ph.D. degree with the excellent dissertation and 14 papers in such short time[1].

I would like to acknowledge Professor Ibrahim Habib, who is my co-advisor and has spent a lot of his time helping me with my work. His guidance is greatly appreciated.

I would like to thank two Reviewer As of IEEE/ACM Transaction on Networking and Journal of High Speed Networks for their careful and detailed comments in technology and verb, which imporved the dissertation greatly.

For all the members of my Ph.D. examination committee to take the time to review my dissertation and making valuable comments to this work, I would like to express my sincere thanks.

I do also hereby wish to take this opportunity to express my sincere desire to thank all of the other different people who have helped me on my way for the Doctor degree. It is impossible to mention everyone who has had some hand in such long way. Let me try to name some of the people who foremost come to mind in connection with the way for the Ph.D. degree.

Professor Kenneth M. Sobel, Advisor of the Ph.D. program of the Electrical Engineering department, and Professor Ahmed, former Head of the EE department, recommended me to Professor T. Saadawi. The result from his recommendation is as mentioned in the first paragraph. I greatly appreciate it.

I would like also to thank Professor Gerard G. Lowen, Dean of the Ph.D. Program of School of Engineering, for various helps during my Ph.D. study here. For example, he introduced me to CCNY-China Exchange Program when I just came in the college.

Special thanks should go to Professor Ngee-pong Zhang, Head of CCNY-China Exchange Program. He has identified me with excellent students and granted me financial aid during my Ph.D. study since I registered here. The result from his support is as mentioned in the first paragraph.

[1] Refer to the Postscript on the last page in this text.

I also feel indebted to all the friends of mine for their various helps and making me have a happy life during my study here.

I extend my gratitude to all faculty and staff of Electrical Engineering Department for their creating work and study atmosphere, and pleasant life together.

To be obtaining my Ph.D. degree in Computer Networks, I can not help thinking of Professor Oscar Steenhaut, former President of Vrije Universaite Brussel, Brussels, Belgium. He recognized me to be promising in computer science, accepted me as a Master student, and provided me with various key helps even though my undergraduate major is machinery engineering. He tried his best to do me favors even in the last days before I left there. I could feel that he helped a Chinese student with his heart and love. I appreciate it deeply.

I also wanted to thank Professor Tiberghien, Head of Department of Information Science, Vrije Universaite Brussel, Brussels, Belgium, my Master promoter, for his various helps during my study there.

Doctor as the highest academic degree for a student is a result based on undergraduate and primary education. Without the education, nobody could reach a Doctor degree. I am very grateful to the University of Science and Technology of China, Hefei, China, and No.46 High School of Beijing, Beijing, China, for their providing outstanding fundamental education and training. The education and training include not only knowledge and skills of existing science and technology but also scientific methods of thinking and research for things unknown and new.

In addition, I owe lots of thanks to Professor Brian Schwartz, Director of the CUNY Multimedia Center, for his every effort in my striving for a Ph.D. rewarding position.

Last but most importantly, I would like to thank my family's encouragement and supports deeply.

I like Lao-Tzu's philosophy very much. It influences me and my study imperceptibly. The following are a few of Lao-Tzu's words :

"The way that can be experienced is not true. The world that can be constructed is not true. The way manifests all that happens and may happen. The world represents all that exists and may exist. To experience without intention is to sense the world. To experience with intention is to anticipate the world. These two experiences are indistinguishable. Their construction differs but their effect is the same. Beyond the gate of experience flows the way, which is ever greater and more subtle than the world."

"The way that can be testified does not keep unchanged forever. The world that can be identified by its own features does not remain itself the same for good."

Contents

List of Figures

Chapter 1 Introduction

1.1 A Dual Bus Network: Motivation, Issues, Standard

In the past two decades, there have been dramatic changes and important successes in the field of telecommunication networks. The most remarkable are the widespread use of local area networks (LAN) and the emergengce of asynchronous transfer mode (ATM) as a popular solution to broadband integrated service digital networks (B-ISDN).

Most of the work in computer networks had been based on wide area networks twenty years ago, using the underlying telephone network for transmission and adapting data traffic to it. LANs have been a point of departure from the telephony-based model of data communication, which are based on a shared serial medium that runs at a sufficiently-high speed to accommodate the needs of many users at once. It provides for the interconnection of computing devices that are closely located within a single building or a collection of buildings such as a college campus. Such LANs are charaterized by the five main features:

1. Limited geographical coverage, typically in the range of a few meters to a few kilometers
2. Ownership by a single organization
3. Low bit-error rates
4. Simple network topologies, such as a unidirectional ring or a bus
5. Low price

ATM provides for a model of high speed and high effectiveness multimedia transmission such as hundreds of video channels, thousands of telephone calls, and tens

of thousands of data at the same time. It will very likely be used as a trunk networks intercity, nationwide or worldwide due to high cost multiple-input-and-output ATM switch and is charaterized by the seven main features:

1. Very large geographical coverage, typically in the range of hundreds of kilometers to thousands of kilometers

2. Public communication facility

3. Very low bit-error rates

4. Complicated network topologies, such as a unidirectional ring or a bus

5. Very high speed in the order of one Gbps to several Gbps

6. Multimedia services

7. Very high price

As noted, within a city, a different kind of networks is expected that can be used as a private networks or a public network to integrate mulitmedia service. This is, on one hand, because a single telephone network or a single cable TV netwrok or single-media computer data exchange network can not provide for high effectiveness and high quality multimedia services using their own network. On the other hand, this is also because ATM networks are high expensive. Naturally, metropolitan area network (MAN) will fill the blank in the perspective of multimdeia telecommunication network. MAN is charaterized by the following features:

1. Large geographical coverage, typically in the range of ten kilometers to one hundred kilometers

2. Public or private communication facility

3. Very low bit-error rates

4. Simple network topologies, such as a unidirectional ring or a bus

5. high speed in the order of one hundred Mbps to one Gbps

6. Multimedia services

7. Low price

8. Reliability

According to [1, 2], MAN implies that a network requires enhanced reliability charateristics such as faults tolerance and operates over larger distances on the order of 10 to 100 km at high speed on the order of 100 Mbps to 1 Gbps with multimedia services such as data, voice and video. The voice service has stringent requirements that do not apply to computer data, and which dictate much of the design of the MAN. These are for quaranteed bandwidth (64 kilobits per second per voice channel) and bounded delay (under worst case conditions the maximum is two milliseconds round trip). MAN, as part of integrated service data telecommunication networks, is to be used as backbone in metropolitan areas in future.

For the sake of better relaibility, dual bus configuration (DBNET) is one of prefered configuration features of MANs or LANs.

Throughput is one of key parameters of networks. Improving throughput may directly lessen access delay and jitter of a network system. Although some transmission media such as optical fiber provide a high capacity channel, it may not provide a high throughput for networks, which depends on how the media access control (MAC) protocol is designed. The goals of this dissertation are, firstly, to propose and analyze a new, simple, effective, and versatile scheme, *Group Transmission* (*GT*), for improving MAC (MAN or LAN) throughput based on a DBNET, and secondly to apply *GT* to some new DBNET protocols.

Several MAC protocols have been suggested on such configurations denoted

by *DB-MAC*. They include DQDB[1, 3], S++[4], CRMA[5], and CBRMA[6]. The common performance characteristics of these protocols are: 1) They all provide slot-access only once for each slot, which conducts that the throughput is at most equal to its channel slot-capacity(slots per second) under heavy load and 2) Their slot-time-utilization is one fourth or so on the average under uniform heavy traffic and uniform source downstream-destination distribution and uniform node distribution(Refer to the simulation results in Chapter 3). The slot-time utilization is the sum of busy time intervals over the slot lifetime during which a busy slot is loaded with a segment that has not arrived at its destination yet.

Potential chances for improving the throughput of *DB-MAC* can be observed in a slot lifetime: with DQDB as an example, it leaves about three fourth of a slot lifetime idle on the average under uniform heavy load and uniform source downstream-destination distribution and uniform node distribution, of which there are, by our estimate, an one fourth or less prior to loading a slot by distributed queue (DQ) head nodes, which can be pre-used, an one fourth or so post a slot destination, which can be re-used, and the other one fourth or more, which has no chance to be used prior to loading a slot by DQ head nodes or post a slot destination.

1.2 Schemes for Improving Throughput of DBNET Proposed By Others

Some schemes were put forward to take these chances for further improving the throughput of these protocols and lessening the access delay such as *Destination Release (DR)*[7, 8, 9], *Erasure Nodes (EN)*[7, 10, 11], *Previous Slot Information (PSI)*[12], and SP-DQDB[13]. It is noticed that *DR*, *EN*, and *PSI* need an explicit destination address in every slot header, or a message identifier (MID) in every slot payload, or special addressing infomation in every node when they are incorporated

with DQDB, a future MAN standard that can easily be interconnected with ATM WAN. As a matter of fact, this address can only be found in an initial MAC protocol (IMP) (or called as "frame" in this paper) sublayer data header format of DQDB and this MID in a derived MAC protocol (DMP) sublayer data header format. Neither of them can be found in a queued arbitrated (QA) slot sublayer data header format of DQDB. This means that the three schemes depend on either IMP sublayer header or on DMP sublayer header and do not work without extracting either IMP destination address or DMP MID from a QA slot payload. SP-DQDB only needs a QA slot sublayer header access control field (ACF) so it works at a QA slot sublayer independently. These schemes stated above, actually, are make-up schemes for MAC, denoted by *MU*, and should work with one of MAC protocols mentioned in the last paragraph. Naturally, *DB-MAC-MU* stands for a dual bus MAC protocol incorporated with a make-up scheme stated here. Further, DQDB-PSI is an element of set *DB-MAC-MU*. Other elements of *DB-MAC-MU* we use in the paper should be understood according to the same naming way as that for DQDB-PSI but SP-DQDB is only an exception, kept unchanged.

There are two kinds of *MU*s that have been suggested so far for *DB-MAC*. One are *Slot Reuse* [7]-[12] and the other *Slot Pre-Use* [13]. Among *Slot Reuse*, [8]-[10] are referred to as *DR*, [7, 10, 11] as *EN*, and [12] as *PSI*.

Due to the symmetry of a DBNET (Fig.1), any scheme introduced in this paper only concerns one side situation : Bus A for segment transmission and Bus B for "auxiliary" transmission/operartion. By "auxiliary", we implies such information transmission that helps set up segment transmission on Bus A complying with a *DB-MAC*. Request transmission on Bus B of DQDB is an auxiliary transmission. As for a fold bus network, a special DBNET, the upper-side bus is used to send segments,

which is called as "upper-side sending". Because the upper-side bus is folded back at the end node of the lined nodes of the network as the lower-side bus, so a node can send segments to any other nodes in the network only through the upper-side sending. And for a fold bus *DB-MAC*, downstream nodes from a node always are all the other nodes in this paper. Connecting the two ends of Bus A and Bus B at node 1 in Fig.1 makes the configuration become a fold bus DBNET.

Before going on, let's introduce a special node for DBNET, first right node denoted by FRN. FRN is such a node in DBNET that has right to use the first free slot in some time according to a *DB-MAC*. At any instant there must be one and only one FRN in a DBNET. It can be any node in DBNET. A fair *DB-MAC* provides equal chances for all nodes to be a FRN. The way to let a node become a FRN is different from one *DB-MAC* another. The DQ head node of DQDB is a FRN.

1.2.1 About Slot Reuse Schemes

The basic idea of slot reuse is: loading a slot with a new segment after an old segment loaded by a FRN has reached its destination node. The slot time utilization of a *DB-MAC* may be improved by a *Slot Reuse* scheme by almost one fourth on the average under uniform heavy traffic and uniform soucre downstream-destination distribution and uniform node distribution (Refer to the simulation results in Chapter 3).

1.2.1.1 Destination Release The *DR* scheme requires that every node releases the slots carrying information to itself for further reuse by downstream nodes. This scheme offers the maximum possible capacity release. However, the latency caused by *DR* is also biggest since it requires buffering capability as part of the transmission medium

in order to read a destination address. For example, the latency in a DQDB-DR node should be 19 bytes while the original DQDB latency is only 5 bytes.

1.2.1.2 Erasure Node The *EN* scheme requires that a limited number of special nodes distributed on the network, known as erasure nodes, perform slot release. These nodes store every traversing slot plus portion of the following slot that contains a "read" bit, check the bit, and release the ones containing data that have already passed their destination. A node marks a current slot in the slot's "read" bit for such a previous slot that contains data destined to itself, thus signaling the erasure node that the previous slot can be released. With a small number of erasure nodes, this scheme overcomes the long-latency disadvantages of *DR*. In [7, 10, 11], it was shown that in DBNETs and under the assumption of uniform source destination distribution, indeed only a small number of erasure nodes is sufficient for a throughput gain close to that of *DR*. However, it requires more complex hardware and considerable latency in erasure nodes than *DR*. More important, such hardware can conduct its lower physical layer capacity than *DR* in spite of their being placed in a limit number of nodes. For example, a DQDB-EN erasure node requires 54 bytes latency though a DQDB-EN ordinary node 5 bytes.

1.2.1.3 PSI (Previous Slot Information) The *PSI* scheme requires that every node keeps the destination address of the segment in the previous slot and checks if the destination of the segment in the current slot is beyond the destination of the previous slot in order to reuse the slot. Upon receipt of a busy slot, a node performs reuse if both of the following rules hold: i) the destination of the previously received slot is before this node and ii) the destination of the segment in the current received slot is before the destination of the segment in the previous slot. Obviously, a *PSI* only

needs one bit flag R in a slot header to indicate if the destination of a segment in the slot is beyond that of the previous slot. Although *PSI* is less efficient than *DR*, the scheme needs no extra latency (only 5 bytes for DQDB-PSI) and is simple when a slot carries an explicit destination address [12]. Note that it is not that simple for DQDB-PSI because neither a QA slot header of DQDB nor a DMP data unit header contains an explicit destination address.

1.2.2 About Slot Preuse Schemes

The basic idea of slot pre-use is: loading a slot with a segment before a FRN loads the slot if the loading will not be seen by the FRN. This implies that the segment pre-using a slot should not be destined to (or through) the FRN.

SP-DQDB is only one protocol for *Slot Pre-use* so far. It inserts a new transmission rule under keeping DQDB operations, "neighborhood segment transmission rule" (named in this paper). The ideas of the new rule are: (1) a SP-DQDB node can load its segment for a neighbor node of its (or called as its neighbor segment) into a free slot when the node is not the DQ head at that time. (2) a SP-DQDB node should reduce a non-zero neighbor address of a current received busy slot by one with taking an incoming segment with number one in the slot neighbor address field off the bus. (3) a DQ head node at that time can load a segment of its into a slot (free or not) with storing a busy slot segment in its queue temporarily. SP-DQDB uses some bits (2 or 3) of ACF of a QA slot as its neighbor address for its neighbor nodes. It need not refer to infomation in a slot payload. Number of significant addressable downstream neighbors from a node is six for SP-DQDB with total number of 15 nodes network. It has been demonstrated that SP-DQDB is particularly well suited to two environments: networks consisting of a small number of nodes (up to 15) and networks in which

there are local community interests. It is noticed that the needed local queue results from that a SP-DQDB node does not know if its neighbor nodes include the FRN at that time and that the queue plays double-role: impoving throughput and lowering physical layer capacity. The latency per SP-DQDB node is also only 5 bytes.

1.3 GT: A Novel Theoretically-Proven-Effective Scheme

This dissertation introduces a novel technique that can support either Slot Reuse or Slot Pre-Use or both, named as *Group Transmission* protocol and denoted by *GT*. The proposed *GT* scheme has three alternatives for its working sublayer when used in DQDB: 1) at the QA slot sublayer independently; 2) at the IMP sublayer independently; 3) at all the three sublayers: QA slot, DMP, and IMP like *DR, EN*, and *PSI*. *GT* divides nodes in DBNET into several groups and assigns every group a unique code as their group address in order along a bus. Obviously, the number of bits for a group address is much fewer than that for an IMP address. And They can be put in access control field (ACF) of a QA slot, or in an IMP destination address, or in the both, which depends on which alternative for its working sublayer stated above it chooses. It can potentially achieve higher capacity and fewer latency because it can work at the QA slot sublayer independently, or because it needs to recognize only a group address for slot release rather than an IMP address or a MID like *DR, EN*, and *PSI*, or because it potentially does not need so long a local queue to store neighbor segments from a bus temporarily like SP-DQDB. It can be used to support *Slot Reuse*, *Slot Pre-Use*, or the both as long as a proper protocol is designed based on it. It will also be introduced in this paper that a *DQDB-GT* is a unified form in improving throughput for DQDB, DQDB-DR, and DQDB-EN.

1.4 Organization of the Proposal

The dissertation is organized as follows. In Chapter II, basic concepts are introduced, including ideas, theorems, and application-promises of the novel *GT* scheme. Theoretical curves are presented. In Chapter III, a series of new DQDB-like protocols based on *GT* are presented, including overview, specification, and simulation analysis for each protocol proposal. In Chapter IV, a new S++-like protocol on *GT* is presented, also including overview, specification, and simulation analysis. Comprehensive analysis, conclusions, and future work are described in Chapter V.

Chapter 2 *GT*: A New Versatile MU Scheme for High Throughput DBNET

2.1 Basic Concepts of *GT*

First let us introduce the concept of throughput gain. By throughput gain, we mean the average number of segments conveyed by a slot of a *DB-MAC* during the slot lifetime over that of DQDB, both of which work under uniform heavy load and uniform source downstream-destination distribution, with reference to the DQDB throughput gain as 1. By chance, the maximum of segments a DQDB slot can convey during the slot lifetime under the same conditions as above is just 1. Actually, any MAC can be used as the base for the comparisons as long as all the compared MACs are discussed or observed under the same conditions.

2.1.1 Basic Ideas for *GT* Schemes

The basic ideas of *GT* include two folds: 1) the grouping of all the nodes in the system and 2) a post-source-group slot-reuse rule.

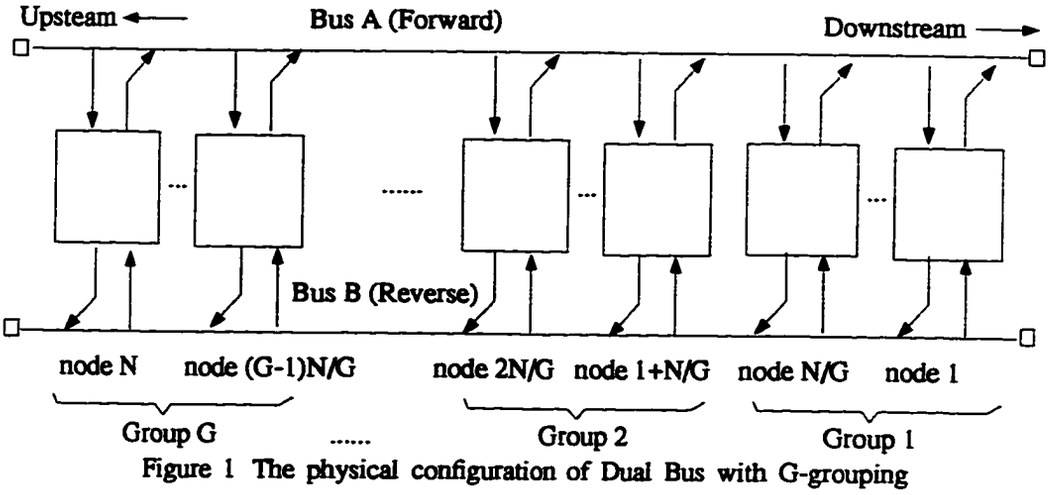


Figure 1 The physical configuration of Dual Bus with G-grouping

To explain the ideas, taking DQDB as an example, we recall that once a segment is sent by a node onto the media, it will not be taken off until the slot reaches the

end of the bus. This feature is thought of as a broadcast among all the downstream nodes from the sending node in the network, and all nodes in the network as a single group within which a slot can carry only one segment at most during the slot lifetime. *GT* generalizes this feature into a multiple-group concept: within each group a slot can carry only one segment during the slot in-group time. After the slot comes into a different group, it may carry a new load, i.e., "be reused".

The grouping is done by dividing all nodes on the network into a limited number of groups and assigning each group a group code, or say group address, in order along a bus. Figure 1 shows the physical configuration of a G-group DBNET network such as *DQDB-GT*, where $G = 2^i - 1$ and i is a positive integer greater than or equal to 1. The G groups are coded $1, 2, \ldots\ldots, G$ from end-node (right-end) to head-node (left-end) along Bus A and may contain the same number of nodes as or a different number from one another. Nodes in a group have the same group code. Note that hereinafter the left, i.e., high position index, of the network is thought of as upstream whereas the right as downstream.

The post-source-group slot-reuse rule is that the condition for a busy slot to be allowed "reuse" is that the slot should come in a different group from its source. This description makes the rule include an important alternative derived from it. The alternative is post-destination-group slot-reuse rule that a busy slot may be reused after the slot departs from its destination group. Why this alternative is very important for *GT* is that it supports *Slot Reuse* independently while its original rule has to be incorporated with it in order for *Slot Pre-Use*.

It should be noticed that an application of the rule depends both on what kind of scheme *GT* is used as (*Slot-Reuse*, *Slot-Pre-Use*, or the both) and on what MAC protocol *GT* is incorporated with (DQDB, S++, or CRMA, etc.). For example,

for *DQDB-GT* used as *Slot-Reuse*, the implementation based on the post-destination group slot-release rule is only required while for *DQDB-GT* used as *Slot Pre-use*, the implementation on the post-source-group slot-release rule is optionally required that allows a request-busy slot to be released for carrying a new request after the slot departs from its source group, besides the previously-stated implementation of the post-destination-group slot-release rule. In addition, *GT* can also be incorporated with Metaring[14] even though it is not a DBNET but a ring configuration.

What new fields are required to insert a slot format for *GT* also depends on the same two conditions given in the last paragraph. For example, the slot format for the *DQDB-GT* used as *Slot-reuse* at least requires a destination group field, denoted by GD(i), to carry a destination group address, where the *i*s in GD(i) and the appearing GR(i) mean that there are i bits in their own fields, which is just big enough to address ($2^i-1=G$) groups. While the slot format for the *DQDB-GT* as *Slot-Preuse* optionally requires a request group field, denoted by GR(i), to carry a request group address besides the previously-stated GD(i).

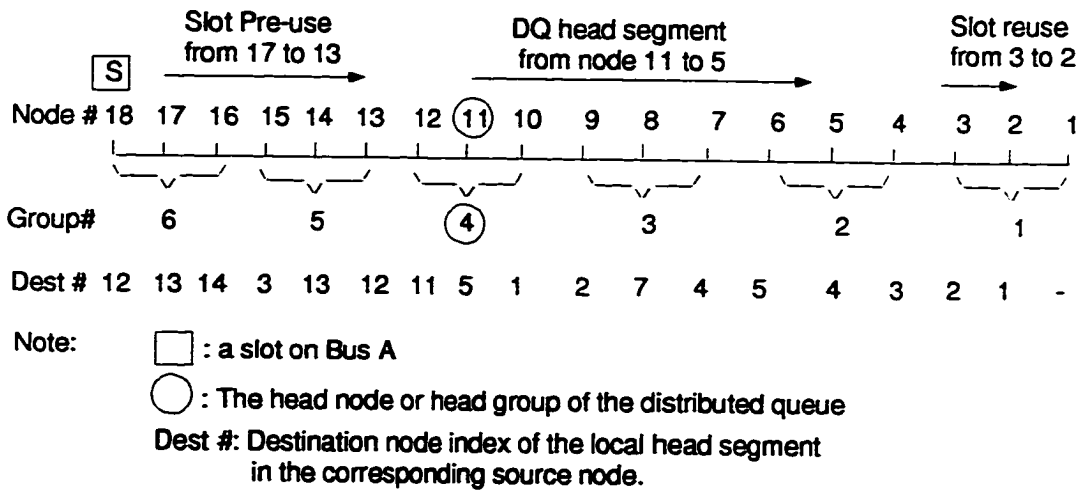
As stated above, the forms in which a *GT* scheme is implemented are different from one application another. The concrete machine research and design based on *GT* are found in Chapter III and IV of this dissertation.

The *GT*'s concept has a significant impact on the delivered throughput. The higher throughput is achieved due to two reasons. One is that nodes situated downstream from a slot destination group will have chances of reusing released slots. The other is that nodes situated upstream from a FRN group (where the group contains a FRN at that time) will have chances of pre-using free slots before the FRN group uses it.

2.1.2 Illustration of Working Features of *GT* Schemes

Fig.2 illustrates, using an example of a 18–node DBNET system in 6 groups, 3 nodes each, how the *DB-MAC-GT* takes the throughput-improving chances introduced in Section I for *Slot Reuse*, *Slot Pre-use*, or the both. Note that we ignore technical detail of how a *DB-MAC-GT* node operates such as requesting machine and sending machine but concentrate on segment transmission phenomena by a single slot on bus A caused by using *GT*.

Figure 2 The transmission phenomena of a 18–node 6–group *DQDB-GT* system.



We randomly select node 11 in group 4 to be the FRN at some instant. The first segment of the node is for node 5 in group 2. An empty slot S arrives at node 18 from slot generator. But node 18 has no right to transmit because the local head segment of node 18 is for node 12 in group 4 which group is holding the FRN. Node 17 has right to transmit because the local head segment of node 17 is for node 13 in group 5, which is before the FRN group. Nodes 16, 15, 14, and 13 have no right to transmit because they see a busy slot. Node 12 has no right to transmit although it sees a released slot S because the node is in the same group as the FRN. So far the slot is

preused once since the slot-use happend before the FRN. After the FRN transmits its segment for node 5, nodes 10, 9, 8, 7, 6, and 4 have no right to transmit because they all but 4 can not see a released slot S. The reason for node 4 is that it is in the same group as the destination group of slot S at that time. Node 3 will reuse slot S when slot S comes into group 1. Node 1 sees a busy slot S. Except node 11, all the other nodes in group 4 have no right to transmit because they are not the FRN, all the nodes in groups 5, 3, and 2 have no right to transmit. Slot S in this situation is pre-used once and conveys two segments if the *GT* is used as *Slot Pre-use*, reused once and conveys two segments if used as *Slot Reuse*, and pre-used and reused once respectively and conveys three segments if used for both *Slot Pre-use* and *Slot Reuse*. In the example, a slot may convey only one segment if the following two conditions are satisfied: 1) The FRN has a segment for the most-downstream group and 2) All the nodes in all the upstream groups from the FRN have no requesting segment or only segments either for the FRN group or for a downstream group from the FRN. A slot may also convey two, three, four, or five segments with their own probability. The maximum number of segments a slot may convey in the example is six.

2.1.3 Characteristics of *GT* Schemes

2.1.3.1 More Transmission Chances: High Throughput The throughput gains caused by using *GT* are obviously determined by the number of groups designed, intuitively because more groups means more chances for transmissions. For instance, if a *DQDB-GT* used as *Slot-Reuse* is divided into two groups, a slot traversing the two different groups can carry two segments totally at most during a slot lifetime when both segments are destined to their own group. Only one segment can it carry, of course, if the segment from the upstream group is destined to the other group. According to

the numerical results in Section V, the throughput gain is approximately improved by seven fifth on the average and at least one bit is needed as a group address. For a 7-group of the *DQDB-GT*, throughput gain by three second or so and at least three bits as a group address, ... and so on. In this way, *GT* improves throughput using a flexible mechanism. The interesting averages of the throughput gains are analyzed theoretically as follows.

2.1.3.2 Less Overhead: Potential High Speed and Less Latency *GT* only needs buffer destination group address rather than the whole destination node address that *DR* and *EN* have to buffer. Per-node latency of *DB-MAC-GT* is different from one *DB-MAC-GT* application another, too. For DQDB, the latency of *DQDB-GT* is 5 bytes when used at the QA slot sublayer independently or 9–10 bytes when at the IMP sublayer. Unlike a SP-DQDB scheme, *GT* does not need so long a local queue in each node as a SP-DQDB node does. As a result, *DB-MAC-GT* is simple and can possess higher system capacity.

2.1.3.3 Flexibility: Potential High Throughput It is worthwhile emphasizing that *DB-MAC-GT* not only is capable for *Slot-Pre-Use*, *Slot-Reuse*, or the both but also is a unified form in improving throughput for *DB-MAC-DR*, *DB-MAC-EN*, and *DB-MAC*. From the viewpoint of *DB-MAC-GT*, *DB-MAC-DR* and *DB-MAC* are two extreme cases. One can use one node per group or combine all nodes in one group. The former is *DB-MAC-DR* and the latter *DB-MAC*. *DB-MAC-EN* is non-extreme groupings. Concrete to say, *DQDB-GT*, DQDB-DR, DQDB-EN, and DQDB are subset or elements of the above corresponding set and possess the corresponding same features. Referring to the definitions of concepts on *G*-grouping and complete grouping given in the next section, a complete grouping *GT* system is a *DR* system

while a $(M+1)$ grouping *GT* system is equivalent to the corresponding distributed M erasure-nodes *EN* system in improving throughput. Therefore, the studies done on the relation between throughput gain and erasure-node distribution like [7] and on the relation between maximum throughput and transmission permission assignment like [15] still hold for *GT*.

2.1.3.4 Lower Working Layer: Potential High Speed Unlike *PSI*, *GT* may be used at the slot sublayer of MAC as long as an access control field is defined in the corresponding lower sublayer message format such as a DQDB slot format. As a result, *GT* remains simple and can support higher system capacity.

2.2 Throughput Gain Theorems for *GT* Schemes

In this section, we present four lemmas and eight theorems on the throughput gains of *GT* from probability analysis based on three assumptions. Let N = {1, 2, ..., N} be a set of nodes which represents a N-node *DB-MAC* network and let G = {K_1, K_2, ..., K_g, ..., K_G} be a set of groups with at least one node in each group, where G's size ||G|| = G ($G \in I^{+1}$, a positive integer greater than or equal to 1.) and K_g are subsets of G, whose sizes $||K_g|| = K_g \in I^{+1}$, g=1,2,...,G and $\sum_{g=1}^{G} K_g = N$. For these sets, a high-index denotes upstream position.

2.2.1 Definitions and Assumptions

Definition 0: A node in DBNET is refered to as a FRN when the node has right to use the first free slot in a specified time according to a DB-MAC and the group containing a FRN as a FRN group.

Definition 1: A G-grouping for N is defined if $N/G \in R^{+1}$, a positive real greater than or equal to one.

Definition 2: A complete grouping for N is defined if N/G=1.

Definition 3: A non-grouping for N is defined if N/G=N.

Definition 4: An equal-size weak G-grouping for N is defined if

$$N/G \in I^{+2} \quad I^{+2} - a\ positive\ integer\ set\ excluding\ 1.$$

Definition 5: Throughput gain of a N-node G-grouping DB-MAC-GT network is the average number of segments conveyed by a slot during its lifetime over that of the corresponding non-grouping DB-MAC under the same conditions, denoted by $\overline{x}(G)$.

Assumption 1 (Uniform Traffic): The probability p_n *that a node in a N-node DB-MAC-GT network becomes a FRN is equal and independent:*

$$p_n = \frac{1}{N}.$$

The feasibility and practicality of this assumption may be verified in [16] and [4].

Assumption 2 (Uniform Downstream-Destination) : The probability that a segment in a N-node DB-MAC-GT network is addressed from node i *to itself is zero and that a segment is addressed from node* i *to its downstream nodes* j *are equal :*

$$p_{ij} = \begin{cases} 0 & ,j = i \\ \frac{1}{i-1} & ,j = i-1, i-2, \ldots, 1 \end{cases}.$$

for a DB-MAC network and

$$p_{ij} = \begin{cases} 0 & ,j = i \\ \frac{1}{N-1} & ,j \neq i \end{cases}.$$

for a fold bus upper-side-sending DB-MAC network.

Assumption 3 (Heavy Load): There is at least one message arriving at one node for transmission in a N-node DB-MAC-GT network during a specified sufficient small time interval.

Before presenting on, we need to declare function $u(z)$ as below because the function is helpful to express a formula. *Hereinafter, we have*

$$u(z) = \begin{cases} 0 & , z \leq 0 \\ 1 & , z > 0 \end{cases}$$

where z is an integer.

2.2.2 *GT* Throughput Gain Theory

2.2.2.1 Lemmas *Lemma 1 (Express of Assumption 1 under G-Grouping): The probability P_g that a group in a G-grouping DB-MAC-GT network becomes a FRN group is complete:*

$$\sum_{g=1}^{G} P_g = 1.$$

Proof: The proof is too obvious to be given out here.

Lemma 2 (Express of Assumption 2 under G-Grouping): The probability, defined by $P_{g,k}$ below, that group g in a G-group DB-MAC-GT network transmit to its downstream groups k is complete, that is: for a *DB-MAC-GT* network

$$\sum_{k=1}^{g} P_{g,k} = 1 \tag{7}$$

and for a folded bus upperside sending *DB-MAC-GT* network

$$\sum_{k=1}^{G} P_{g,k} = 1 \tag{8}$$

where the first subscript g is the position index of source group, the second subscript k is the position index of destination group and

$$P_{g,k} = \begin{cases} 1 & , g = k = 1; \\ \frac{1}{K_g} \sum_{n=1}^{K_g} \frac{n-1}{\sum_{d=1}^{g-1} K_d + n - 1} & , g = k > 1; \\ \frac{K_k}{K_g} \sum_{n=1}^{K_g} \frac{1}{\sum_{d=1}^{g-1} K_d + n - 1} & , g > k \text{ and } g = 1, 2, \ldots, G \\ 0 & , g < k \end{cases}$$

for a DB-MAC-GT network and

$$P_{g,k} = \begin{cases} \frac{K_k - 1}{N-1} & , g = k \\ \frac{K_k}{N-1} & , g \neq k \end{cases}$$

for a fold bus upper-side-sending DB-MAC-GT network. K— the number of nodes in a group.

Proof: Refer to Appendix I. Unless specified otherwise, deduction is a main method for proofs in this dissertation.

Lemma 3 (Slot Reuse Probability Completion at any Group): The probability, defined by $P_r(m,k)$ below, that groups after group k in a G-group DB-MAC-GT network transmit m segments with slot reuse to their downstream groups is complete, that is:

$$\sum_{m=0}^{\infty} P_r(m,k) = 1 \tag{11}$$

where, for a DB-MAC-GT network, we have

$$P_r(m,k) = u(2-m)P_{k,1} + u(m-1)\sum_{j=m}^{k} P_{k,j}P_r(m-1, j-1) \tag{12}$$

where subscript j on the right side of equation (12) is the position index of a destination group.

Proof: Refer to Appendix II.

Lemma 4 (Slot Pre-Use Probability Completion at any Group): The probability, defined by $P_p^G(p,g)$ below, that groups n before group g transmit p segments by slot pre-use to the groups j, j_2 before group g is complete for a G-grouping DB-MAC-GT network, that is:

$$\sum_{p=0}^{\infty} P_p^G(p,g) = 1 \tag{13}$$

where $g < G$ *and*

$$P_p^G(p,g) = u(G-g-p+1)$$
$$\begin{cases} \prod_{k=g+1}^{G} \sum_{j=1}^{g} P_{k,j} & , p = 0 \\ \sum_{n=g+1}^{G} \sum_{j=g+1}^{n} \left(\prod_{k=n+1}^{G} \sum_{m=1}^{g} P_{k,m} \right)^{u(G-n)} P_{n,j} \left(\prod_{k=g+1}^{j-1} \sum_{m=1}^{g} P_{k,m} \right)^{u(j-g-1)} & , p = 1 \\ \sum_{n=g+1}^{G} \sum_{j=g+1}^{n} \left(\prod_{k=n+1}^{G} \sum_{m=1}^{g} P_{k,m} \right)^{u(G-n)} P_{n,j} P_{pi}(p,g,j) & , p > 1 \end{cases} \tag{14}$$

where

$$P_{pi}(p,g,j) \overset{\substack{j_1=j-1 \\ p_0=p-1}}{=}$$
$$\begin{cases} \sum_{n=g+1}^{j_1} \sum_{j_2=g+1}^{n} \left(\prod_{k=n+1}^{j_1} \sum_{m=1}^{g} P_{k,m} \right)^{u(j_1-n)} P_{n,j_2} \left(\prod_{k=g+1}^{j_2-1} \sum_{m=1}^{g} P_{k,m} \right) & , p_0 = 1 \\ \sum_{n=g+p_0}^{j_1} \sum_{j_2=g+p_0}^{n} \left(\prod_{k=n+1}^{j_1} \sum_{m=1}^{g} P_{k,m} \right)^{u(j_1-n)} P_{n,j_2} P_{pi}(p_0,g,j_2) & , p_0 > 1 \end{cases} \tag{15}$$

where subscript p_0 *on the right side of equation (15) is the number of slot pre-use chances left, whereas* p_0*=p-1 below the equal-symbol means to reduce 1 slot preuse chance due to the last slot preuse, subscript n is the position index of a possible slot-preuse source group from* $g+p_0$ *through* j_1*, whereas* j_1*=j-1 above the equal-symbol means to let a group next to the destination group j of the last slot-preuse be the new first possible slot preuse source group,* j_2 *is the position index of a possible slot-preuse destination group from* $g+p_0$ *through n.*

Proof: Refer to Appendix III.

2.2.2.2 Theorem 1: Slot Reuse Probability Completion Theorem *Theorem 1: The probability, defined by* $P_R^G(i)$ *below, that a G-grouping DB-MAC-GT network for Slot Reuse transmits i segments during one slot lifetime is complete, that is:*

$$\sum_{i=0}^{\infty} P_R^G(i) = 1. \tag{16}$$

where

$$P_R^G(i) = \sum_{g=i}^{G} P_g P_r(i,g) \tag{17}$$

where for DB-MAC-GT networks

$$P_r(i,g) = u(2-i)P_{g,1} + u(i-1)\sum_{k=i}^{g} P_{g,k} P_r(i-1,k-1) \tag{18}$$

and whereas for folded bus upper-side-sending DB-MAC-GT networks

$$\begin{aligned} P_r(i,g) =& u(2-i)[(\frac{1}{2})^{u(2-g)} P_{g,1} + \sum_{h=g}^{G} (\frac{1}{2})^{u(g+1-h)u(h-g+1)} P_{g,h}] \\ &+ u(i-1)\sum_{k=i}^{g} (\frac{1}{2})^{u(g+1-k)u(k-g+1)} P_{g,k} P_r(i-1,k-1) \end{aligned} \tag{19}$$

And $(\frac{1}{2}P_{g,g})$ in Eq.(19) implies that within a group the sum of the probabilities that nodes are destined to downstream equals to half of the sum of the probabilities that nodes are destined to the group itself. For upstream, the same relation exists, hereinafter $(\frac{1}{2}P_{g,g})$ keeps the same meaning..

Proof: Refer to Appendix IV.

2.2.2.3 Theorem 2: Slot Pre-Use Probability Completion Theorem *Theorem 2: The probability, defined by $P_P^G(i)$ below, that a G-grouping DB-MAC-GT network for Slot Pre-use transmits i segments during one slot lifetime is complete, that is:*

$$\sum_{i=0}^{\infty} P_P^G(i) = 1. \tag{20}$$

where

$$P_P^G(i) = \sum_{g=1}^{G} P_g P_p^G(i,g) \tag{21}$$

Proof: refer to the corresponding decomposition parts of *Theorem 3* proof.

2.2.2.4 Theorem 3: Slot Reuse and Pre-Use Probability Completion Theorem

Theorem 3: The probability, defined by $P^G(i)$ below, that a G-grouping DB-MAC-GT network for both Slot Reuse and Slot Pre-use transmits i segments during one slot lifetime is complete, that is:

$$\sum_{i=0}^{\infty} P^G(i) = 1. \tag{22}$$

where

$$P^G(i) = \sum_{g=1}^{G} P_g \{P_{g,1}[P_p^G(i-1,g)]^{u(i-1)u(G-g)} P_p^G(0,g)^{u(2-i)u(G-g)} + u(i-1) \sum_{m=1}^{i-1} u(g-m) \sum_{k=m+1}^{g} P_{g,k} P_r(m,k-1) P_p^G(i-1-m,g)^{u(G-g)}\} \tag{23}$$

Proof: Refer to Appendix V.

2.2.2.5 Theorem 4: Dual Bus Slot Reuse Throughput Gain Theorem

Theorem 4: Throughput gain $\overline{x}(G)$ of a G-grouping DB-MAC-GT network for Slot Reuse is :

$$\overline{x}(G) = \sum_{g=1}^{G} P_g P_{g,1} + \sum_{i=2}^{G} i\{\sum_{g=i}^{G} P_g \sum_{k=i}^{g} P_{g,k} P_r(i-1,k-1)\} \tag{24}$$

Proof: This can be proved by applying average theorem of probability theory to*Theorem 1*.

2.2.2.6 Theorem 5: Fold Bus Slot Reuse Throughput Gain Theorem

Theorem 5 : Throughput gain $\overline{x}(G)$ of a G-grouping fold bus upper-side-sending DB-MAC-GT network for Slot Reuse is :

$$\overline{x}(G) = \sum_{g=1}^{G} P_g (P_{g,1} (\frac{1}{2})^{u(2-g)} + \sum_{h=g}^{G} (\frac{1}{2})^{u(g+1-h)u(h-g+1)} P_{g,h}) + \sum_{i=2}^{G} \sum_{g=i}^{G} \{i \sum_{k=i}^{g} (\frac{1}{2})^{u(g+1-k)u(k-g+1)} P_{g,k} P_r(i-1,k-1)\} \tag{25}$$

*Proof:*This proof is the same as that of *Theorem 4.*

2.2.2.7 Theorem 6: Slot Pre-Use Throughput Gain Theorem *Theorem 6: Throughput gain $\overline{x}(G)$ of a G-grouping DB-MAC-GT network for Slot Pre-use is :*

$$\overline{x}(G) = \sum_{g=1}^{G} P_g\{P_{g,1}P_p^G(0,g)^{u(2-i)u(G-g)} + u(i-1)i\sum_{k=1}^{g} P_{g,k}P_p^G(i-1,g)^{u(G-g)}\} \tag{26}$$

Proof: This proof is done by applying average theorem of probability theory to*Theorem 2.*

2.2.2.8 Theorem 7: Slot Reuse and Slot Pre-Use Throughput Gain Theorem *Theorem 7: Throughput gain $\overline{x}(G)$ of a G-grouping DB-MAC-GT network for both Slot Reuse and Slot Pre-use is :*

$$\begin{aligned}\overline{x}(G) = &\sum_{g=1}^{G} P_g\{P_{g,1}P_p^G(0,g)^{u(G-g)} + \sum_{i=2}^{G} i\sum_{g=1}^{G} P_g\{P_{g,1}P_p^G(i-1,g)^{u(G-g)} + \\ &\sum_{m=1}^{i-1} u(g-m)\sum_{k=m+1}^{g} P_{g,k}P_r(m,k-1)P_p^G(i-1-m,g)^{u(G-g)}\}\end{aligned} \tag{27}$$

Proof: This proof is by applying average theorem of probability theory to*Theorem 3.*

2.2.2.9 Theorem 8: *GT*-Effectiveness Theorem *Theorem 8 : The throughput gain of a DB-MAC-GT network with G-grouping must be higher than that of the corresponding non-grouping DB-MAC network.*

Proof: Refer to Appendix VI.

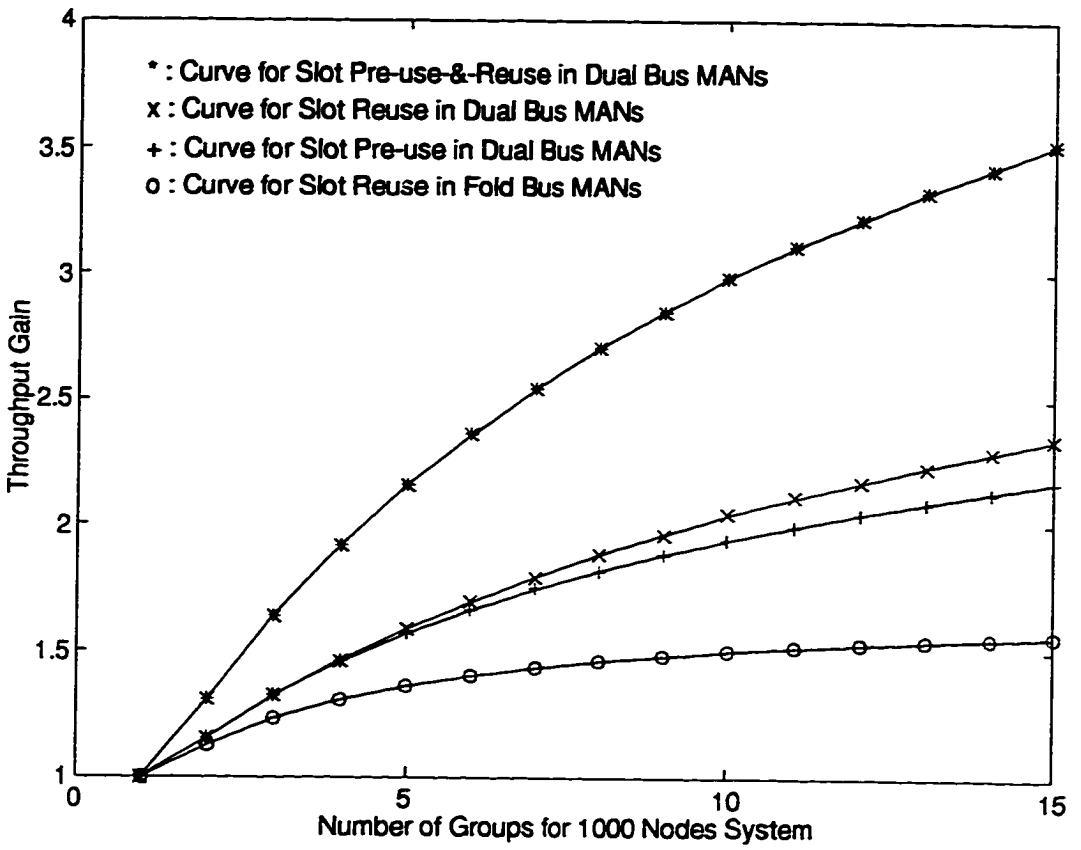
Theorem 8 guarantees theoretically that *GT* is significant in improving throughput gain for a non-grouping *DB-MAC* protocol.

2.3 Theoretical Computation Results

Fig.3 shows theoretical throughput gains of the various *GT* systems with the change of the number of groups under uniform heavy traffic, uniform source downstream-destination distribution, and uniform node (group size) distribution. Four curves are seen: all with a *DB-MAC-GT*, one for *Slot Reuse* with a fold bus upper-side-sending such as S++-GT, denoted by "o" curve; one for *Slot Pre-use* such as *DQDB-GT*, denoted by "+" curve; one for *Slot Reuse*, denoted by "x" curve; the other for both *Slot Reuse* and *Slot Pre-use*, denoted by "*" curve. These curves result from the calculation of a 1000–node MAN system based on *Theorem 4, Theorem 5, Theorem 6, and Theorem 7*, respectively.

As seen, the highest throughput gains obtained from *DB-MAC-GT*s is the one for both *Slot Reuse* and *Slot Pre-use*. The reason for this is that the scheme for both *Slot Reuse* and *Slot Pre-use* provides more transmission chance by using both pre-FRN slot idle time and post-FRN-destination slot idle time, whereas schemes for *Slot-Reuse* or *Slot-Pre-use* uses only the corresponding pre-FRN slot idle time or post-FRN-destination slot idle time, respectively. The throughput gains of the *Slot Reuse* (non-fold-bus) is a little bit higher than that of the *Slot Pre-use*, which results from the fact that the *Slot Pre-use* requires one more transmission constraints than the *Slot Reuse*: segments pre-using a slot should not be destined to or through the group which has made its request. Now that in the fold bus *DB-MAC-GT* scheme for the *Slot Reuse* a segment may be destined to upstream nodes through its fold bus, the probability that the slot can be reused is low. This results in the lowerest throughput gain as seen in the figure.

Figure 3 Throughput Gain vs. Number of Groups for 1000 Nodes DBNET



It is also observed from the curves for those *DB-MAC-GT*s that for a fixed grouping number the throughput gain on the curve for the *Slot Reuse and Preuse* is almost always equal to the sum of the one for the *Slot Reuse* and the one for the *Slot Preuse* minus 1. This verfiies the theorems for these schemes in computation results from another angle.

It should be noted that throughput gains of the various *GT* systems non-linearly increase with the number of groups dividing a DBNET system increasing. When the number of groups in a *GT* system is one, a *DB-MAC-GT* becomes its corresponding non-grouping network and the throughput gain is one.

It is worthwhile specially mentioning: 1) that all the curves demonstrate two important characteristics: monotonic increment of the curves and monotonic decrement

of their derivatives in the given range. Based on the characteristics it is predicted that the throughput gain upper bounds for them exist in the range of 1000. Unfortunately, we have not started to look for them. But due to a complete grouping *Slot Reuse GT*'s equivalence to a *DR* in improving throughput, the maximum equal throughput $e = 2.718...$ obtained in [15] seems to hold for the upper bound of the *GT* scheme. 2) that there exists an optimized group size distribution for maximum throughput for each *GT* scheme. For example, the optimized group size distribution for a *GT* used for *Slot Reuse* and *Slot Pre-Use* seems to be an equal-group-size uniform distribution. However, it need studying further.

Figures (4) through (7) show the probability distributions of segments conveyed during a slot lifetime under the *GT* systems. As seen, the peaks of the various probability distribution curves move to the right as the grouping number increases, meanwhile, the left ends of the curves fall down and the middle parts rise, which leads

Figure 4 Probability distribution of segments conveyed during a slot lifetime in Slot Pre-use & Reuse *DB-MAC-GT* networks

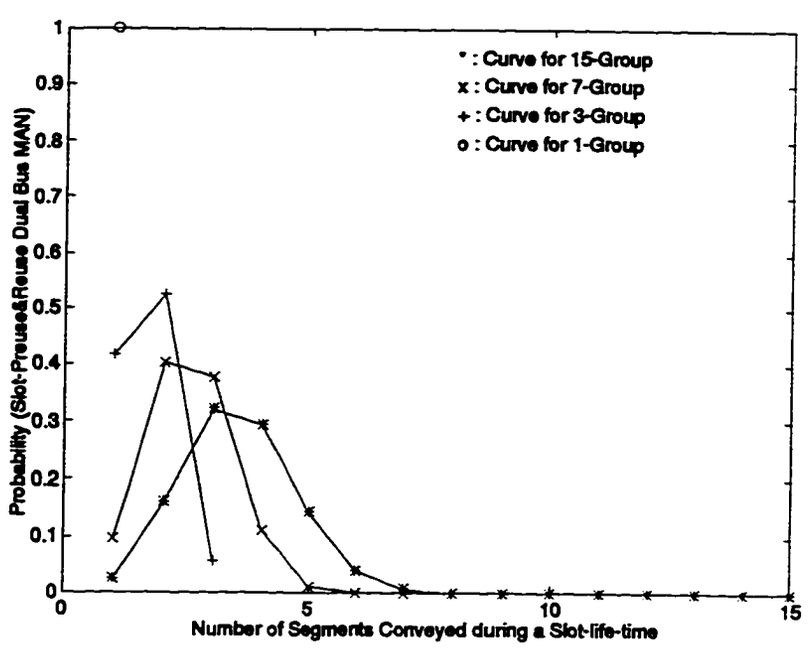


Figure 5 Probability distribution of segments conveyed during a slot lifetime in Slot Reuse *DB-MAC-GT* networks

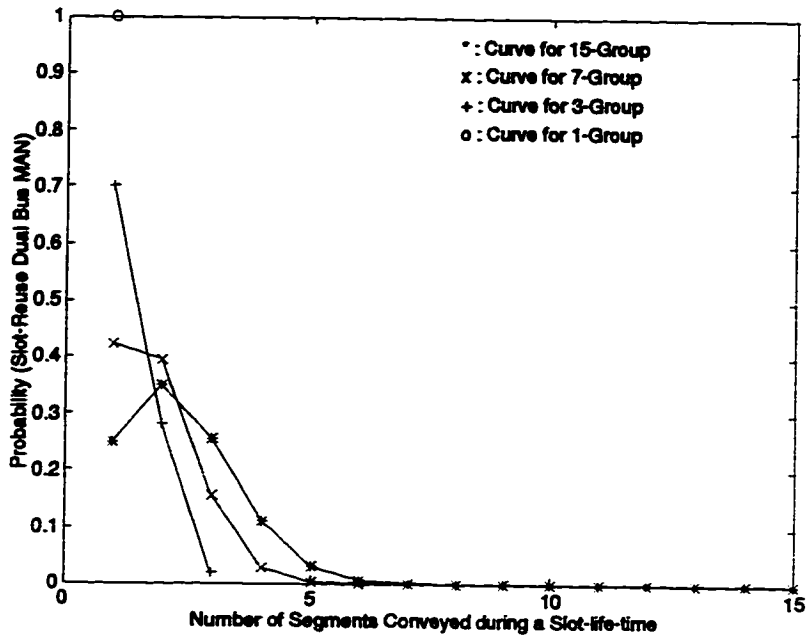


Figure 6 Probability distribution of segments conveyed during a slot lifetime in Slot Pre-use *DB-MAC-GT* networks

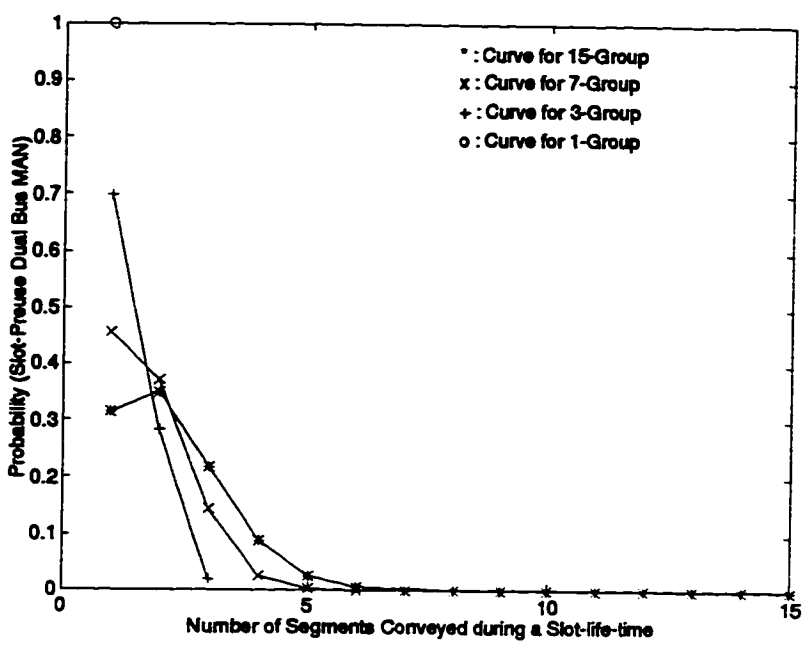
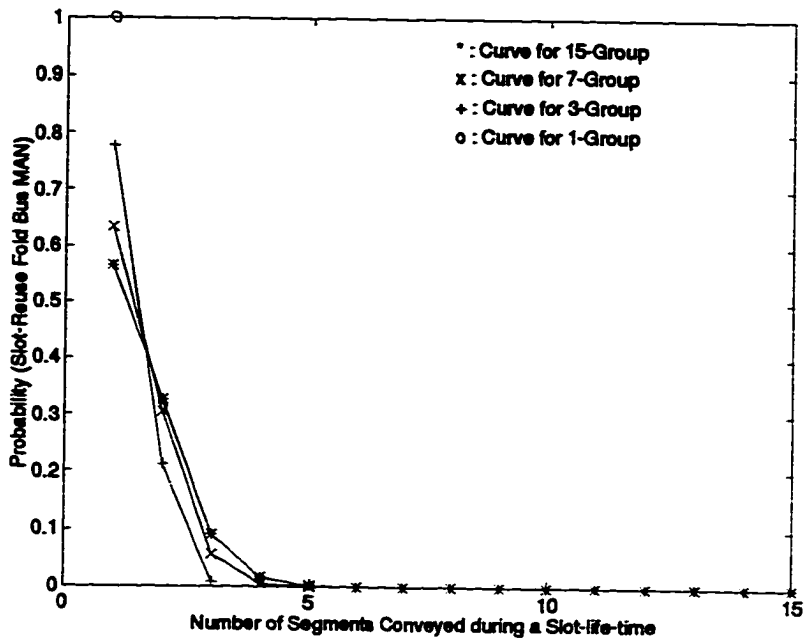


Figure 7 Probability distribution of segments conveyed during a slot lifetime in Slot-Reuse fold bus *DB-MAC-GT* networks



to an increase in their corresponding throughput gains. Among the peak-shift-distances of various *GT* schemes, the one for both *Slot Reuse* and *Pre-use* with a *DB-MAC-GT* is greatest, the one for *Slot Reuse* is second, the one for *Slot Pre-use* is third, the one for *Slot Reuse* with a fold bus upper-side sending *DB-MAC-GT* is smallest. These verify the explanation we have made in the previous paragraphs.

2.4 Conclusion for This Chapter

A new make-up scheme for improving throughput of *DB-MAC*, *GT*, is introduced and theoretical analysis of the throughput gains of *GT* is presented. The main features of this scheme are: 1) it divides all the nodes in a DBNET into multiple groups and assigns each group a group code and 2) either a segment-busy slot can be reloaded with a new segment for *Slot-reuse* after the slot departs from its destination group or

a request-busy slot can be reloaded with a new request for *Slot-Pre-use* after the slot departs from its source group.

GT is a flexible scheme for *Slot Reuse*, *Slot Pre-use*, or the both and a unified form for *DR* and *EN* in improving throughput. The implementation of a *GT* scheme is application-dependent.

Eight theorems for *GT* schemes to obtain higher throughput gains than the corresponding non-grouping system are proved. These theorems result from probability analysis based on three assumptions: uniform traffic, heavy traffic, and uniform source downstream-destination distribution. The theoretical analysis under uniform node distribution demonstrates that throughput gains of the systems increases as the number of groups dividing the system increases. The throughput gain upper bounds of a *GT* system have not been found and the study for optimilzed distributions of group-sizes for maximum throughput has not started.

A *GT* scheme can work at a QA slot sublayer of DQDB independently for slot-reuse whereas *DR*, *EN*, and *PSI* can not. The scheme does not need such a long local queue in each node as that in SP-DQDB and support higher throughput under uniform traffic. It also remains simpler than *PSI* when incorporated with the QA slot sublayer of DQDB independently because it does not need such special addressing information from IMP and DMP sublayers that are required by *PSI*. The comparison between *GT* and *PSI* when they both work within the three sublayers (QA slot, DMP, and IMP) of DQDB is subtle while *PSI* seems simpler than *GT* when at a single-frame-sublayer *DB-MAC* such as S++.

GT's being compared with *DR*, *EN*, and *PSI*, the per-node latency of *DB-MAC-DR* is biggest, *DB-MAC-EN* second, *DB-MAC-GT* third, and *DB-MAC-PSI* smallest, usually. For DQDB, a special case, however, we get two orders : DQDB-DR (19

bytes) >DQDB-EN (>5 bytes) > *DQDB-GT* (5 bytes) = DQDB-PSI (5 bytes) when *DQDB-GT* works at the QA slot sublayer independently and DQDB-DR (19 bytes) >*DQDB-GT* (9–10 bytes) > DQDB-EN (5–9 bytes) > DQDB-PSI (5 bytes) when *DQDB-GT* at the sublayers of the QA slot, DMP, and IMP.

In overall, a *GT* scheme is capable of obtaining higher system capacity and throughput and is a versatile scheme for DBNET. It is foreseen that future DBNET will be interconnected with backbone high speed MAN or WANs employing ATM and conveying multimedia traffic. The *GT* scheme, therefore, is useful in applications involving isochronous traffic (such as video and voice) where high throughput is imperative.

Chapter 3 New DQDB-Like Protocols

In this chapter, five new DQDB-like protocols based on the *GT* scheme proposed in the last chapter will be presented in each section below, respectively, after a brief introduction to ISO/IEC 8802-6 (ANSI/IEEE 802.6) Std. The five new DQDB-like protocols are Group Addressing Distributed Queue Dual Bus (DQDB-GA), Group Requesting Distributed Queue Dual Bus (DQDB-GR), Non-Post-Request Slot-Pre-Use Distributed Queue Dual Bus (DQDB-NSP), Grouping Distributed Queue Dual Bus (DQDB-G), and High Throughput Distributed Queue Dual Bus (DQDB-H).

All the five new DQDB-like protocols will be verified and analyzed by simulation at slot-level written in the C programming language. In the other words, I will simulate two buses, all the designed fields of ACF of a slot and node-operations of the MAC layer. Due to symmetry of the networks, I only use one bus for segment transfer and the other for a request transfer. The assumptions for the simulation are that the DQDB-like protocols have no slot waste [16], provide equal chance to obtain equal bandwidth for the same priority at each node, and work under Poisson traffic load with a set of average arrival rates (segments per second). In order to satisfy the simulation assumptions, I limit the size of its waiting queue (*Q) in a node to 48 segments and its outstanding queue (*CD) to 3 for DQDB-GR and 48 for DQDB-GA. Whenever a node reaches the (*Q) limited number of buffer, the node will reject an arrival of a segment. Whenever a node finishes the (*CD) limited number of requests, the node will stop requesting until all other nodes complete their own limited number of requests. Hence, the simulations support fair service to all nodes. Before the nodes reach the limited number of requests, they comply with the request rule described in IEEE 802.6 standard.

The basic parameters for the network simulation are: the total number of nodes in the system is 60; the length of optical fiber is 7.7 kilometers; the channel capacity is 1 giga bits per second; the slot size is ATM standard of 53 bytes; the slot rate is 2,358,490 slots per second.

In addition, I am going to compare and analyze all the simulative results with theoretical results obtained in the previous chapter.

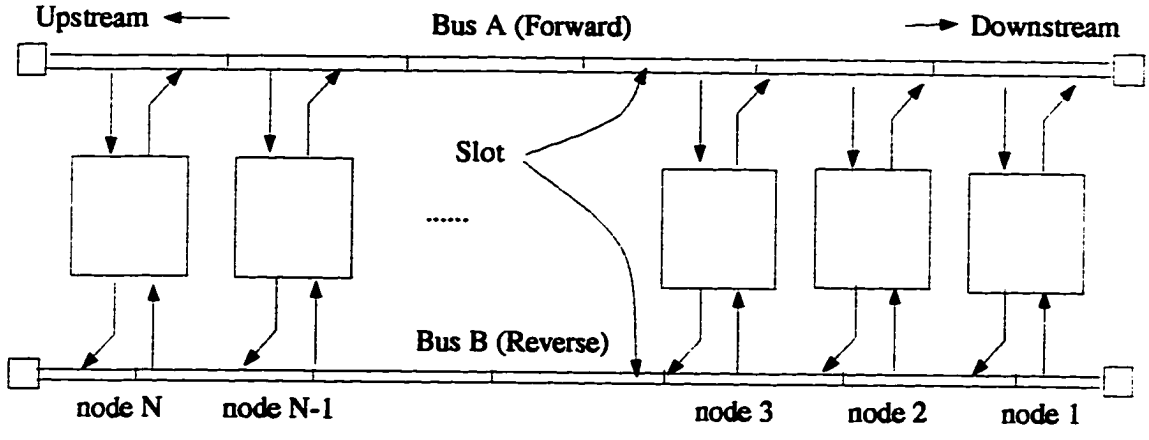
3.1 Introduction to IEEE 802.6

A DQDB network consists of two unidirectional buses with data flow in opposite directions. One bus is denoted by bus A or Forward bus and the other by bus B or Reverse bus as shown in Fig.(8). A slot generator at the head of each bus emits free fixed-size slots at a constant rate. Each node is connected to both buses. A node transmits data by filling in a free slot on a particular bus. Note that, due to the topology of the dual bus, each node has to make a routing decision whether to use bus A or bus B for transmission dependent on the physical location of the destination node. Since the architecture of a dual bus network is symmetric, we will focus on segment transfer on bus A.

The DQDB MAC protocol prevents nodes closer to the head of a bus from acquiring all free slots by implementing a reservation scheme. A node having a segment ready for transmission on bus A notifies the nodes closer to the head of bus A by sending a reservation request on bus B. Each node keeps a waiting queue of untransmitted segments. Only the segment at the head of a waiting queue is allowed to submit a reservation request. Note that setting the request bit may be delayed since a node has to wait for a slot which has a request bit not set. A node determines its turn to transmit a segment with two counters: the request counter *(RQ)* and the

countdown counter *(CD)*. If a node does not have segments queued for transmission, it increments its *(RQ)* counter for each passing slot on bus B, having the request bit set. It decrements *(RQ)* for each free slot the node detects on bus A. Upon arrival of a segment to the node, *(RQ)* is copied to *(CD)* and then reset. Then, *(RQ)* is incremental for slots on bus B having the request bit set. The *(CD)* is decremental for each free slot passing by the node on bus A. When *(CD)* reaches zero, the segment is allowed to take the next free slot for transmission.

Figure 8 The Dual Bus Configuration



Now let us take a look at slot-time utilization and throughput of a DQDB network. It is clear that there is only one distributed queue (DQ) in the whole system for the common media access. Naturally, the segment at the DQ head is the only one that is waiting for the next free slot. In case the segment is loaded in a free slot , the loaded slot will not be unloaded at any node until it reaches the end of the bus. The time interval after the loaded slot reaches its destination node is wasted. The performance characteristics of DQDB are: 1) the throughput is at most equal to its channel slot-capacity under heavy load, 2) the slot-time utilization is one fourth on the average under uniform traffic, uniform downstream-destination, and uniform node

distributions (Refer to the corresponding simulation results below), and 3) some slots are left unused. The slot-time utilization is the sum of busy time intervals over the slot lifetime during which a busy slot is loaded with a segment that has not yet reached its destination. The slot lifetime means the duration a slot takes to travel from a bus head to the bus end.

It is observed from the above characteristics that there is chance to improve the throughput by reusing a slot. The basic concepts of *Slot Reuse* are: (1) loading a slot with a new segment after the slot loaded by a DQ head has reached its destinations or (2) after (1) occurs, loading a slot with a new segment after the slot has reached its destinations.

3.2 A DQDB-GA Protocol for Slot Reuse

I am going to describe the overview and specifications of the DQDB-GA protocol in the next two subsections seperately, and then exhibit simulation results.

3.2.1 Overview of A DQDB-GA Protocol

The proposed DQDB-GA protocol for *Slot Reuse* provides for higher throughput via three folds: 1) the grouping of all the nodes in the system, 2) slot post-destination group releasing rule, and 3) group transmission rule.

Refer to Chapter 2 for the principle of grouping.

As we shall illustrate later, the grouping concept has a significant impact on the delivered throughput. The higher throughput is achieved because nodes situated downstream from a destination-group will have higher chance of using slots released at the destination group when compared with DQDB. Moreover, grouping requires less number of bits in access control field (ACF) since we need only identify the group-

address whereas other schemes (i.e., DQDB-DR and DQDB-EN) need to identify the whole node address[10].

It is worthwhile mentioning that the distributivity of the DQDB-GA protocol can be guaranteed by one adequate management protocol for the DQDB-GA protocol so that in case of the occurrence of one fault on the DQDB-GA, the DQDB-GA management protocol is capable of re-grouping the nodes and activating proper group code for each node in its group. As a result, DQDB-GA can be also one-fault tolerant.

A new field, GD, is defined in ACF of a slot format, which carries its destination group code loaded when a slot is loaded with a segment. The size of field GD is determined by the number i of grouping given previously.

The slot post-destination group release rule works as follows: within each group, each free slot can only carry a segment. A slot carrying a segment will become free when the slot departs from its destination group. A node resets the Busy-status field when the local group code of the node matches the contents of the GD field of a coming slot. Hence a busy slot can be free for post-destination reuse by lower index nodes that are located downstream from a destination group.

The group transmission rule works as follows. We define two types of free slots. One is a slot that is free for an outstanding queue (*CD). Note that (*yx) stands for a first-in-first-out (FIFO) queue naming yx, hereinafter. By the outstanding queue, we imply that all segments in the queue have made their requests and are waiting for free slots. The other is a slot that is free for slot reuse. The determination of a free slot is done by a traditional field of a slot: Busy field. The determination of type of free slots is done by the DESTINATION_GROUP field (GD). A slot is free for (*CD) only if the values in the both fields are NULL, whereas a slot is free for slot reuse if field Busy is NULL and if field GD is neither NULL nor the local group code of the node. Similar

to the countdown-*CD*-to-zero-for-transmission rule of DQDB, the DQDB-GA rule for transmission is: countdown (CD) at the head of its (*CD) by one when a node with non-empty (*CD) receives on bus A a non-Busy slot with NULL in field GD until the (CD) becomes zero, then the node may load its segment addressed by the pointer field of the first position of the (*CD) into a coming "free" slot. A node with non-empty (*CD) and non-empty (*Q) receiving a slot free for reuse transmits a segment from the head of the (*CD) without care about a (CD) value and moves a segment from the head of the (*Q) into the tail of the (*CD) without care about a (RQ) value. A node with an empty (*CD) and non-empty (*Q) receiving a slot free for reuse transmits a segment directly from the head of the (*Q) without care about a (RQ) value.

The group request rule works as follows. A node holding the segment has to send a request to the upstream head-end of Bus A through Bus B. The REQ field of a slot will be set by a node when the node with segment to send detects value NULL in the REQ field of the slot. The RQ counter increases by one when a node receives a slot with non-NULL value in the REQ field on bus B. The RQ counter decreases by one when a node with RQ>0 and empty (*CD) receives on bus A a non-Busy slot.

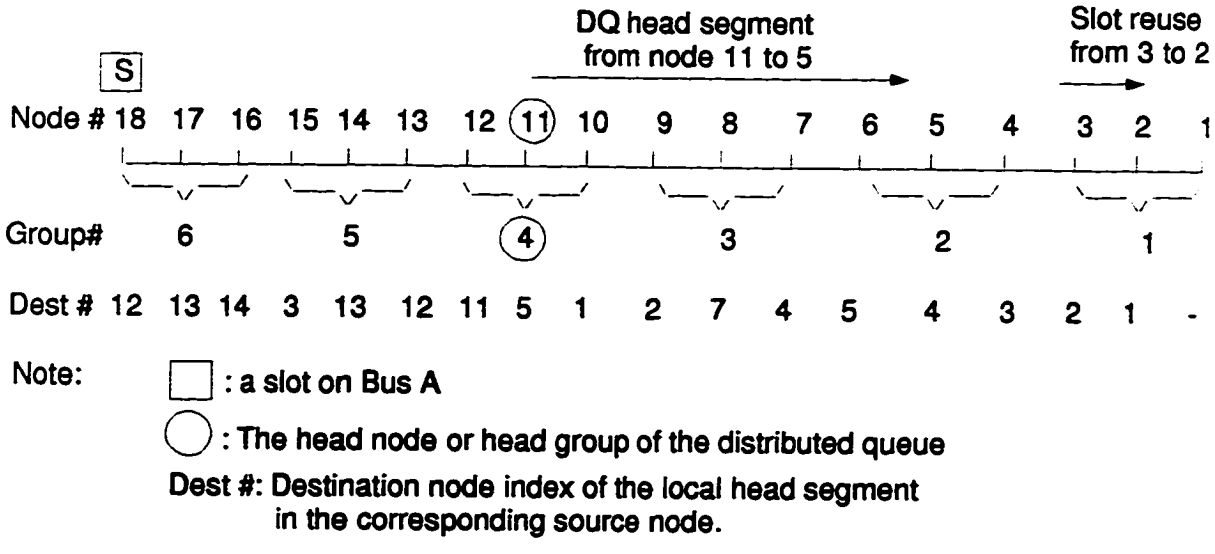
It is worthwhile pointing out that referring to the theory in Chapter 2, a complete grouping DQDB-GA system is a DQDB-DR network, while a $(M + 1)$ grouping DQDB-GA network may be considered as a correspondingly-distributed M erasure nodes DQDB-EN network in throughput with less complexity and less latency than a real M erasure nodes DQDB-EN network.

Fig.(9) illustrates how a DQDB-GA protocol works for *Slot Reuse* by an example of a 18–node DQDB-GA system in 6 groups and 3 nodes each. Note that we ignore the detail of how a node of DQDB operates in some fields such as transmission request but only focus on segment transmission phenomena on a bus caused by a DQDB-GA

protocol.

We randomly selected node 11 in group 4 to be the DQ head. The head segment of the node is for node 5 in group 2. Like DQDB, a free slot S carry no segment before it reaches the head. After the DQ head transmits its local head segment for node 5 in group 2, nodes 10, 9, 8, 7, 6, 4 have no right to transmit in slot S because they recognize that S is busy. Node 3 can reuse S after S has been released, but nodes 1 and 2 see a busy slot S. Except for node 11, all the other nodes in group 4 have no right to transmit because they are not in the head position, all the nodes in group 3 , group 2 have no right to transmit because they all see a busy slot.

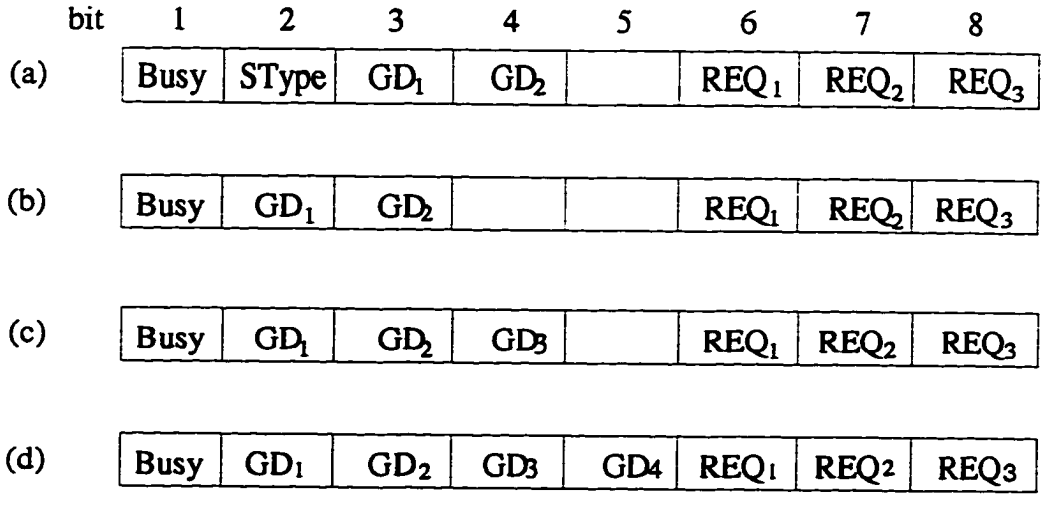
Figure 9 The transmission principle of a 18-node 6-group DQDB-GA system.



It is clear that slot S in this situation is reused once and conveys two segments. One segment was transmitted from node 11 in group 4 to node 5 in group 2. The second segment was transmitted from node 3 in group 1 to node 2 in the same group by slot reuse. In the above example, a slot may convey only one segment if the DQ head has a segment for its downstream-most group, which is one of transmission chances.

A slot may convey either two segments, or more up to maximum six segments with their own probability. For example, when a slot meets such a case that nodes the slot meets first in each group send their segments to their own groups (sending in-group), it may convey at most six segments. But what is the average number of segments a slot m. But what is the average number of segments a slot may convey in the system? I have exhibited throughput gain theorems and theoretical curves on *Slot-Reuse* in Chapter 2. Simulation results can be found in the 3.2.3 section following Section 3.2.2: Specifications.

Figure 10 The ACF Formats of A DQDB-GA Slot.



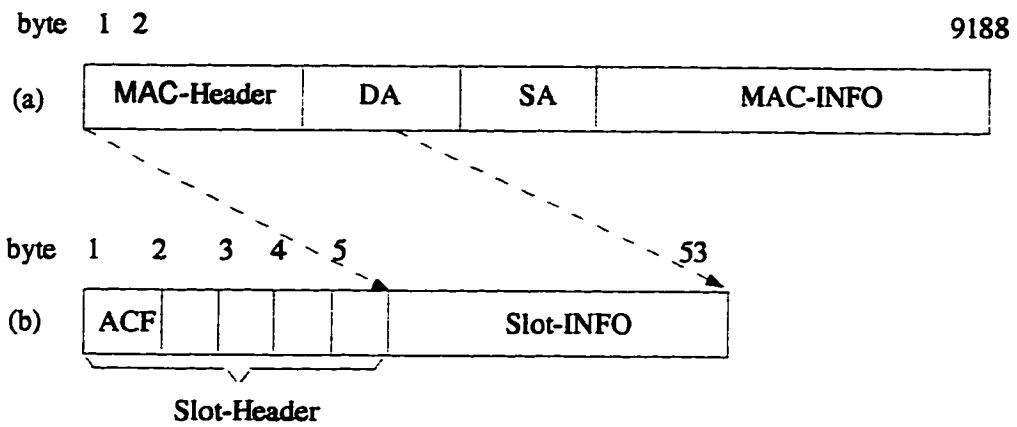
Note: GD -- Group Destination bit

3.2.2 Specifications

The ACF, node structure, and operations of a DQDB-GA protocol are specified as the following, respectively.

3.2.2.1 Modified ACF of a DQDB Slot In Fig.(10), I show the ACF format of DQDB-GA protocol. I maintain the busy status field, Busy, in the ACF format like that in DQDB and add one new field in the ACF format, a group destination field GD(i). The GD(i) field is used to carry a destination group address. The i in GD(i) means that the i-bit group destination field has i bits and can address 2^i–1) groups. In Fig.(10), four options of the ACF format are provided.

Figure 11 (a) MAC frame format. (b) Slot format.



Figs.(10.a) and (10.b) both support a three-group DQDB-GA protocol. In Fig.(10.a), bits 1, 2, 6, 7, 8 follow the definitions of the IEEE 802.6 DQDB standard. The two undefined bits 3, 4, in the ACF format are defined as field GD(2), GD_1 and GD_2 standing for the two bits, respectively. The meanings of their four values are 00 for NULL, 01 for group 1, 10 for group 2, and 11 for group 3. Bit 5 is left undefined. In Fig.(10.b), a redefined ACF format is shown: bit 1 is for Busy state of a slot: 0 for FREE, 1 for BUSY. Bits 2, 3 are the same as bit 3, 4 in Fig.(10.a). The only difference is that the NULL state of field GD and BUSY state of field Busy is used to broadcast management messages, like TYPE bit in Fig.(10.a). Bits 4, 5 are left undefined. Fig.(10.c) defines a seven-group DQDB-GA protocol, and Fig.(10.d)

defines a up to fifteen-group DQDB-GA protocol. Both formats have similar functions to Fig.(10.b). The REQ fields in all four versions have the same functions as those of the DQDB standard which provides for priority requests. Fig.(11.a) and (11.b) show

Figure 12 Structure of A DQDB-GA Node

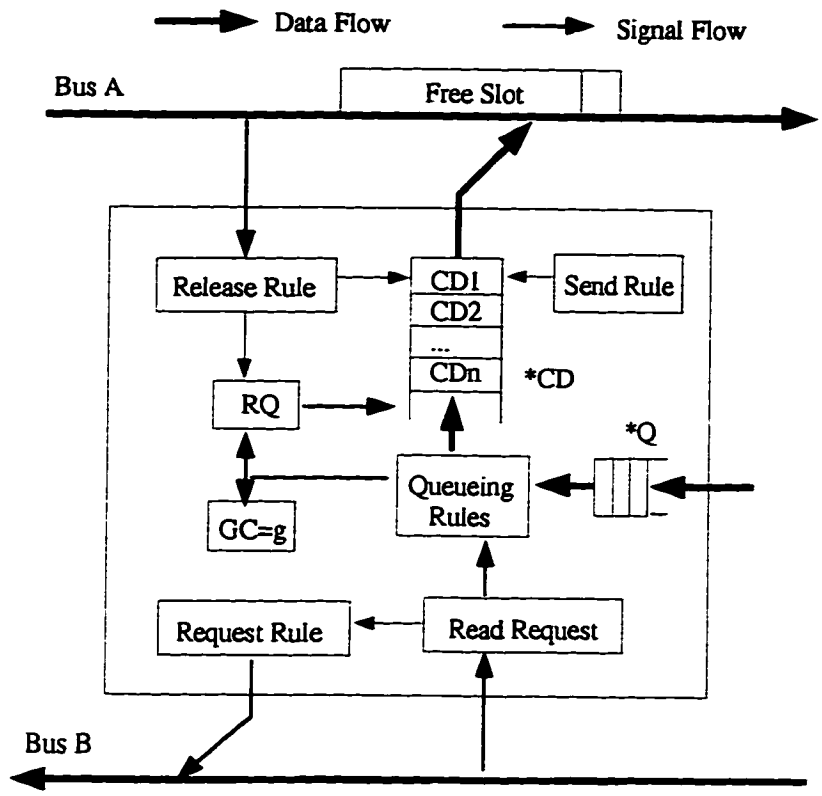
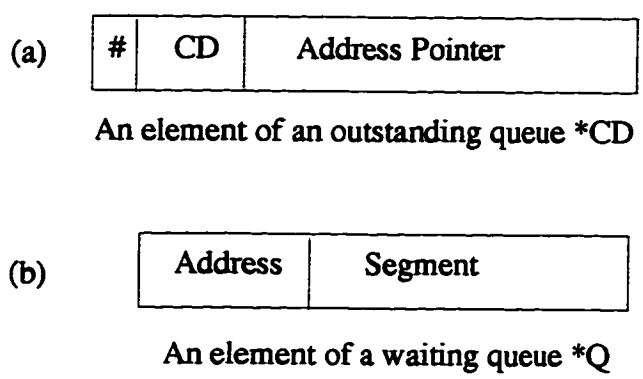


Figure 13 Elements of the (*CD) and the (*Q) for DQDB-GA



a MAC frame format and a slot format, respectively. It is clear that the MAC frame

is segmented and encapsulated into a number of slots. Part of the destination address field (DA) of the MAC frame format can be used to define any number of groups (e.g., more than fifteen groups).

3.2.2.2 Node Structure of DQDB-GA Fig.(12) shows the structure of a typical DQDB-GA node. A waiting queue (*Q) used to keep segments from the local host and a request rule machine with a request counter *RQ* follow the same functionality as those of DQDB. However, we add the following devices in each node. One register (GC) is in order that the node keeps its own group code. An outstanding queue (*CD) is used to keep segments that have made a request for transmission. Each element of the (*CD) consists of two parts: a countdown counter (*CD*) and an address-pointer (see Fig.(13.a)) that points to a segment in the (*Q) (see Fig.(13.b)). One release rule machine decides when the node may release a busy slot. One transmission rule machine decides when the node may send its segment in the (*CD) or in the (*Q).

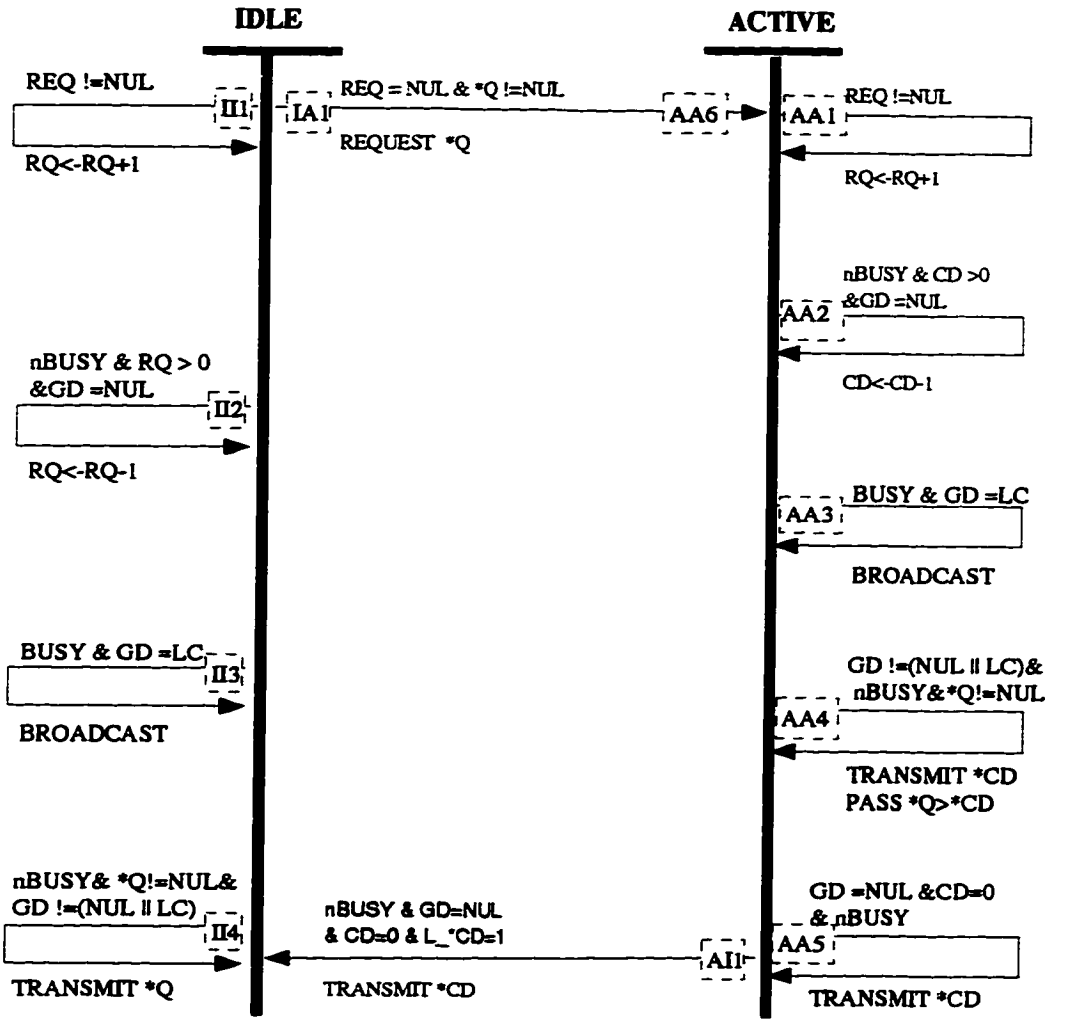
3.2.2.3 Operations of A DQDB-GA Node Fig.(14) demonstrates operations of the nodes. As usual, vertical bold lines are for states of the nodes, the names of states can be found at top of the lines; lines with arrows are for transitions of states; letters above these lines are for events that enable the transition; letters below these lines are for actions that execute when the events above the transition line occur. Tables (15) and (16) provide the explanation of the events and the actions in Fig.(14).

The nodes have two states: state IDLE under which there is no segment in their (*CD) waiting for channel and state ACTIVE under which there is at least one segment in their (*CD) waiting for channel.

The node in state IDLE takes loop-transition Π1 that raises RQ by one when the node receives a non-null request nNUL in field REQ in a coming slot on Bus B.

The node in IDLE with its RQ greater than zero takes loop-transition II2 that reduces RQ by one when the node receives a nBUSY slot on Bus A and a null group code NUL in field GD of the slot.

Figure 14 DQDB-GA Node Operations



The node in IDLE takes loop-transition II3 that broadcasts a coming slot on Bus A by marking nBUSY in field Busy of the slot when the node receives a BUSY slot on Bus A carrying the node's local group code LC in field GD of the slot.

The node in IDLE takes loop-transition II4 that transmits a segment in the head of its (*Q), shifts the (*Q) toward the head by one position, and marks BUSY in field Busy of a slot on Bus A when the node with non-empty (*Q) receives the slot on bus A with nBUSY value in field Busy and neither the node's local group code nor NUL in field GD of the slot.

The node in IDLE takes transition IA1 that changes the state from IDLE to ACTIVE by putting the address of a segment from the head of the (*Q) and its request counter value RQ into part Pointer and part CD of the tail of its (*CD) respectively and setting flag REQ of a slot on Bus B when the node with non-empty (*Q) receives a non-request NUL in field REQ of the slot.

Figure 15 List Of Events

EVENT	Description
BUSY	Busy slot on Bus A
GR	Group request code in a slot on Bus B
GD	Group destination code in a slot on Bus A
xy =/!= cc	The content of the counter or register xy is or is not the value cc
(xy || yz)	Either event xy or yz happans and xy have priority if both happan
*xy =/!= NUL	The queue *xy is or is not empty

Note : (1) The letter n in front of an event means that the event does not happan
(2) L_*XY means the length of queue *XY.

The node in state ACTIVE takes loop-transition AA1 that raises RQ by one when the node receives a non-null request nNUL in field REQ of a coming slot on Bus B.

The node in ACTIVE with its head CD greater than zero state takes loop-transition AA2 that reduces CD in the head of the (*CD) by one when the node receives a nBUSY slot on Bus A carrying a null group code NUL in field GD of the slot.

Figure 16 List Of Actions

ACTION	Description
SHIFT *R,*xy,...	Shift every element in their own queues *R,*xy,... toward the tops by one position and the old segments at the topd are discarded.
TRANSMIT *XY	Mark the current slot as busy and insert the local group code in GD field of the slot and transmit the segment from the position pointed by pointer part in the top of queue *XY, then SHIFT *XY.
BROADCAST	Mark the current slot as empty.
xy <- xy +/- 1	Increase/decrease the counter or the value xy by 1
yz <- wz	Transfer the content of the counter or the value wz to the counter or the value yz
*xy++ <- cc	Put the value cc at the tail of the queue xy then increase its pointer
*xy <- *yz	Transfer all the elements of the queue yz to the queue xy in the same order
*xy,yz <- NUL	Clean the queue xy and counter yz.
FF (*XY || *YZ)	The queue XY have a higher priority to take the action FF to the queue YZ if *XY is not empty
REQUEST *XY	Put the address of the segment at the top of queue *XY in pointer part of queue *CD and do actions : GR <-LC, *CD_CD++ <-RQ, RQ <-0
PASS *X>*Y	Put the address of the segment at the top of queue *X in pointer part of *Y

Note : (1) *xy means queue xy.
(2) *xy_z means part z of an element of queue *xy.
(3) LC means the local group code, C1,C2,... codes for group 1, 2,... .

The node in ACTIVE takes loop-transition AA3 that broadcasts a coming slot on Bus A by marking nBUSY in field Busy of the slot when the node receives a BUSY slot on Bus A carrying the node's local group code LC in field GD of the slot.

The node in ACTIVE takes transition AA4 that transmits a segment in the head of its (*CD), shifts the (*CD) toward the head by one position, marks BUSY in field Busy of a slot on Bus A, and put the address of a segment at the head of the (*Q) into part Pointer of the tail of its (*CD) when the node with non-empty (*Q) receives a nBUSY slot on Bus A with neither null group code NUL nor the node's local group code LC.

The node in ACTIVE takes loop-transition AA5 that transmits a segment in the head of its (*CD) with the corresponding CD equal to zero, shifts the (*CD) toward the head by one position and marks BUSY in field Busy of a slot on Bus A when the node receives the slot on bus A with state nBUSY in field Busy and group code NUL in field GD of the slot.

The node in ACTIVE takes transition AI1 that transmits a segment in the head of its (*CD) with the corresponding CD equal to zero, shifts the (*CD) toward the head by one element and marks BUSY in field Busy of a slot on Bus A, and changes its state from ACTIVE to IDLE when the node with the last one left in the (*CD), (i.e., L_*CD=1), receives the slot with state nBUSY in field Busy and group code NUL in field GD of the slot.

3.2.3 Simulation Results

3.2.3.1 Simulation Parameters The simulation collects data from 60,000 full life-time slots. These data measure the throughput gain, media access delay, slot-time utilization, and segment loss ratio of the whole system under Poisson traffic with

Figure 17 Segment loss ratio vs number of grouping of DQDB-GA under the offered segment-arrival rates

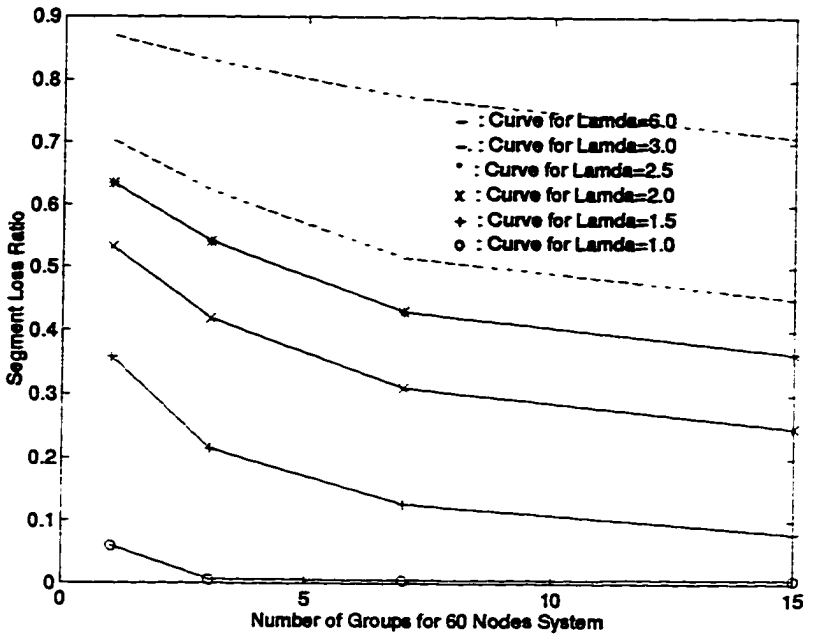


Figure 18 Access delay vs number of grouping of DQDB-GA under the offered segment-arrival rates

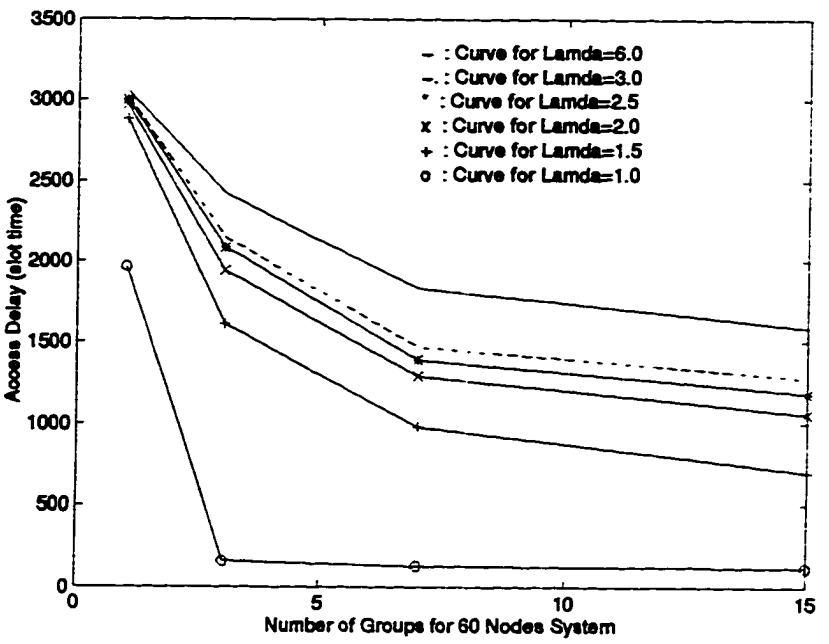


Figure 19 Slot-time utilization vs number of grouping of DQDB-GA under the offered segment-arrival rates

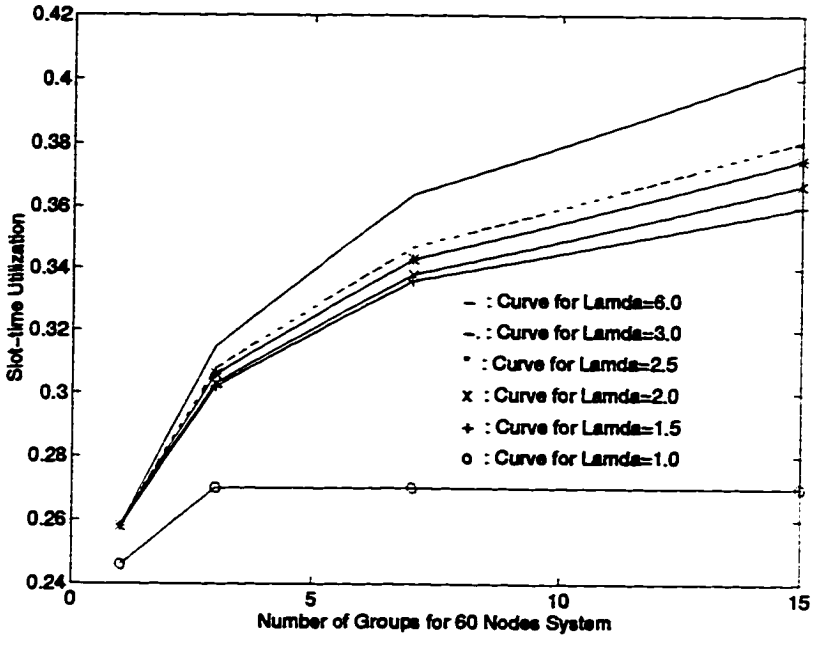
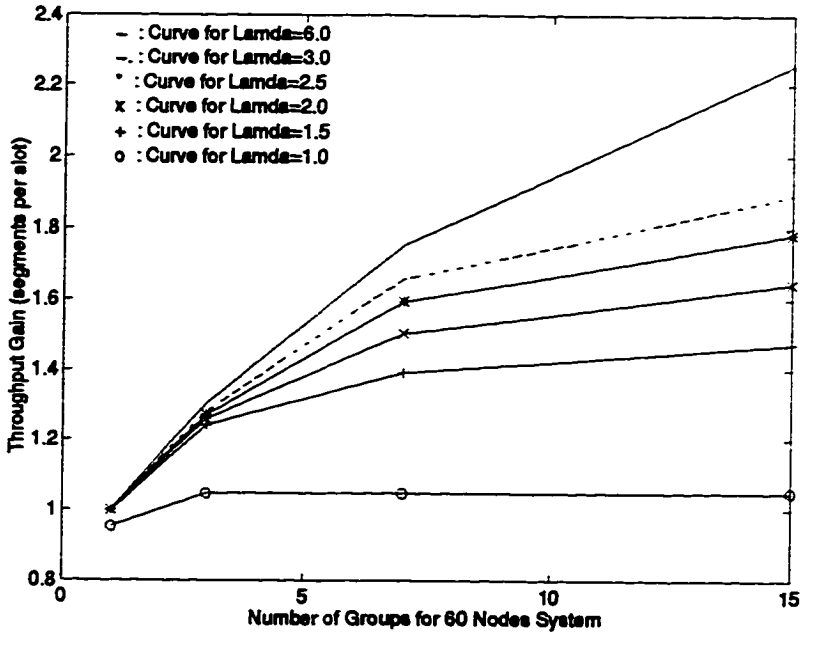


Figure 20 Throughput gain vs number of groups of DQDB-GA under the offered message-arrival rates



average arrival ratio λ of 1.0, 1.5, 2.0, 2.5, 3.0, and 6.0 segment per slot-time. A

cmparison among various grouping method was performed (i.e., performance measures for one-group system up to fifteen groups system.).

3.2.3.2 Performance Results In Fig.(17), simulation results of the segment loss ratio are shown. From the figure, it is seen that the loss ratio decreases as the number of groups increases. This is due to the fact that as the number of groups increases, throughput increases so that the probability that the system buffers are full decreases.

Fig.(18) show simulation results of the average media access delay. From the figure, it is clear that the delay decreases as the number of groups increases. The reason that causes this happen is that more groups in the system directly provides more chances to every segment waiting in the system to be carried by a slot so that the waiting time for a segment to stay in the system with more groups becomes shorter than the system with less groups.

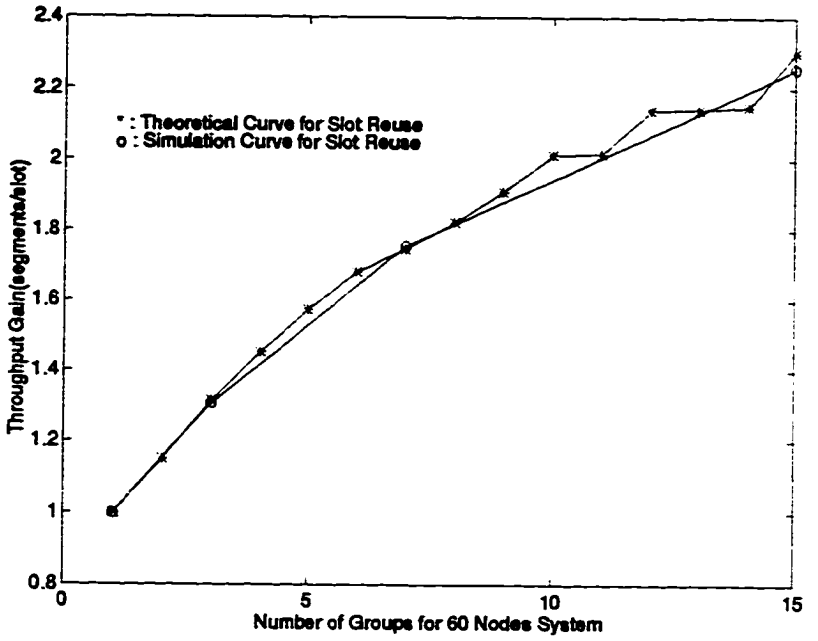
Fig.(19) show simulation results of the slot-time utilization. From the figure, it is obvious that the utilization increases as the number of groups increases. When grouping number of a DQDB-GA system is 1, the utilization is 0.26.

Fig.(20) show simulation results of the throughput gain. It is observed that the throughput gains increase with the increment of number of groups on average under Poisson traffic with average arrival rate λ of 1.0, 1.5, 2.0, 2.5, 3.0, 6.0 segments per slot-time. When grouping number of a DQDB-GA system is 1, the throughput gain is 1.

Fig.(21) shows two throughput gain curves: one from theorem, the other from simulation, both of which are for the same grouping network as the described above. We see that theoretical curve is higher than simulative curve. The chief reason for this is that the machine we presented here itself does not make full use of transmission

chance that is expected by our theorem. In addition, theoretical curve in the figure is obtained by calculating *Theorem 2*.

Figure 21 Comparison of theoretical throughput gain with simulation result



3.2.4 Remarks

DQDB-GA is a novel high performance MAC protocol. It is designed based on the grouping transmission *GT* concepts proposed in Chapter 2 and actually is an application of *GT*. The presented results of the simulation of DQDB-GA show that the throughput gain of a DQDB-GA network (1) increases as the number of groups being divided of the network increases and (2) almost matches the theoretical curve from Chapter 2. Of course, too large number of groups makes the system complex so that the capacity of the system comes down. The impact on the system capacity caused by this complexity of a DQDB-GA system is absolutely less than those caused by the corresponding DQDB-EN system since a node of the DQDB-GA system only need buffer a group code rather than the whole address like a DQDB-EN system.

3.3 A DQDB-GR Protocol for Slot Pre-Use

I am going to describe overview and specifications of the protocols in the next two subsections, and then exhibit simulation results.

3.3.1 Overview of A DQDB-GR Protocol

The proposed DQDB-GR protocol for *Slot Pre-Use* provides for higher throughput gain via four folds: 1) the grouping of all the nodes in the system; 2) slot post-destination group releasing rule; 3) slot pre-use group transmission rule; and 4) group request rule.

Refer to Chapter 2 for the principle of grouping.

It is worthwhile mentioning that the distributivity of the DQDB-GR protocol can be guaranteed by an adequate management protocol such that in case of the occurrence of one fault on the DQDB-GR, the DQDB-GR management protocol is capable of re-grouping the nodes and activating proper group code for each node. As a result, DQDB-GR can be also one-fault tolerance.

The group request rule works as follows. Whereas the request rule of basic DQDB only uses one-bit request field in a slot, the DQDB-GR request rule uses i bits as the GR(i) field. The i in GR(i) means that GR has$i$ bits and address ($2i$-1) groups. A node holding the segment has to send a request to the upstream head-end of Bus A through Bus B. The field GR(i) of a slot will be filled with the local group code by a node when the node compare the content in the GR(i) field with the destination group code of the first segment in the local waiting queue (*Q) and find that the destination group code of the segment is not going to reach the position indicated by the GR(i) field of the slot. A node will set a flag to remember that the node itself is the first to have used a slot for request if it receives the slot with NULL value in field GR(i) of the slot. The

flag is called HD. The RQ counter increases by one when a node receives on bus A a slot with non-NULL value in the GR(i) field and the non-NULL value is greater than the destination group code of the segment. The RQ counter decreases by one when a node with RQ>0 and empty (*CD) receives on bus A either a non-Busy slot with H unset, or a Busy slot with H unset and GD(i)'s content equal to the local group code. This request signals upstream-nodes that the group region indicated by the GR(i) field and its downstream regions are not reachable. In other words, this request signals upstream-nodes that a node in the upstream groups, holding a segment that is destined to a group before the group indicated by the GR(i) field, may send its transmission request by replacing the existing group code in the slot with its own local group code.

The slot post-destination group release rule works as follows. Within each group, each free slot can only carry a segment. A slot carrying a segment will become free when the slot departs from its destination group. A node resets the Busy-status field to zero when the local group code of the node matches the contents of field GD(i) of a coming slot so that the slot is in non-busy state. A node resets field GD(i) of a slot to NULL when the node receives a non-busy slot with a non-NULL value not equal to its local group code in field GD(i) of the slot. The i bits in GD(i) has the same functions as GR(i) groups. In this way, a busy slot becomes free for post-destination groups reuse by lower index nodes that are situated downstream from the destination group.

The group transmission rule works as follows: The determination of a free slot is done by three fields of a slot rather than only one field (two of them are newly added): field Busy, field Group_Destination denoted by GD(i), and field H. The H Field is used to indicate if a slot has passed by all nodes that requested it and may be set only by a DQ-head node. Note that no node can use a slot with field H set. A slot is free only if the Busy field and the H field are NULL and field GD(i) does not

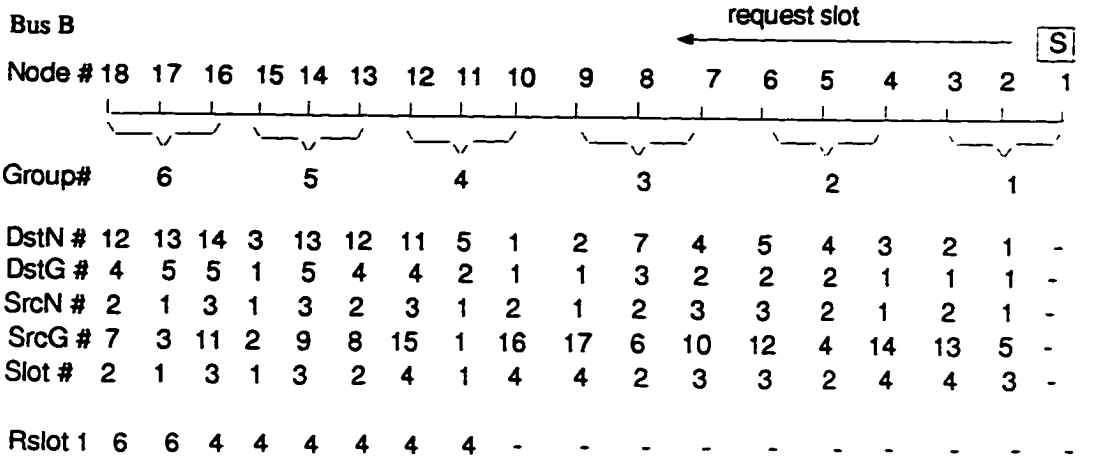
contain the local group code of the node. Similar to the countdown-CD-to-zero-for-transmission rule of DQDB, the DQDB-GR transmission rule is: countdown (CD) at the head of its outstanding queue (*CD) by one when a node with non-empty (*CD) receives on bus A either a non-Busy slot with H unset, or a Busy slot with H unset and GD content equal to the local group code until the CD reaches zero, then the node may loads its segment addressed by the pointer field of the first position of the (*CD) into an incoming free slot.

Figure 22 List of symbols in Figs.(28,23,24,25,26 and 27)

Symbols :	Meanings.
□	: a slot on Bus B to carry request information
◯	: The head node or head group of the distributed queue
DstN #	: Destination node index of the local head segment in the corresponding node.
SrcN #	: Source node order in a group when a slot meets the node with a segment.
SrcG #	: Source group order when a slot meets the group with a segment.
Slot #	: The slot the corresponding nodes will meet.
Rslot 1	: updating information in group request field of slot 7 with the local group code by the corresponding nodes the slot 7 meets in the order from node 1 to 18.
Rslot #	: similar meaning to Rslot 7.
-	: null.

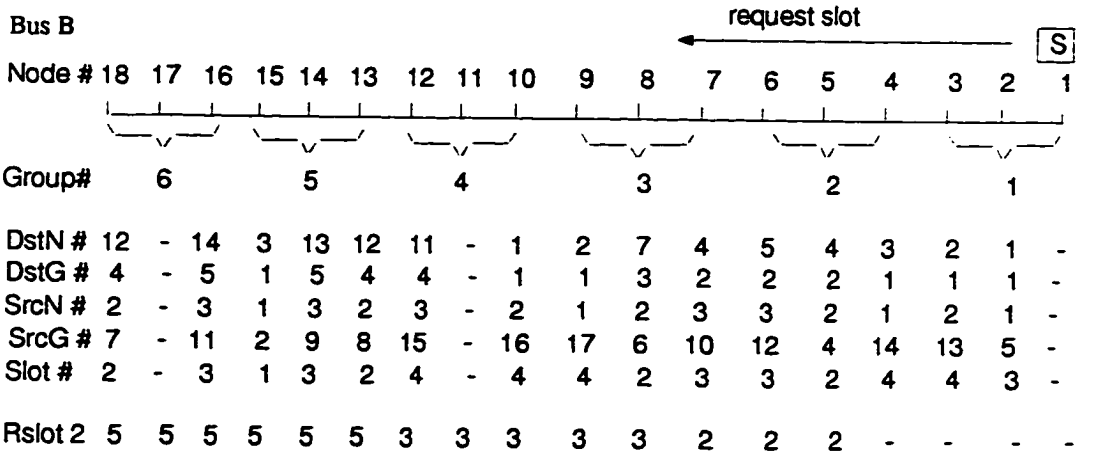
Figs.(23,24,25,26) and (27) with legend table (22) illustrate how the DQDB-GR protocol establishes requests by an example of a 18–node DQDB system in 6 groups 3 nodes each with 15 segments to send. We assume that the arrival process of segments in system is a stochastic process. Note that we ignore technical detail of how a node of DQDB-GR operates such as sending request and slot group release but focus on request phenomena on bus B and segment transmission on bus A. About the technical detail we will present in the next section.

Figure 23 The request principle of a 18-node 6-group DQDB-GR system: for Request slot 1 on Bus B



Before we explain the Fig.(23), it is worthwhile pointing out that the node that is first in the whole network system receiving a segment to send must not be the node that is first receiving a slot for request usually. This depends both on time order of

Figure 24 The request principle of a 18-node 6-group DQDB-GR system, for Request slot 2 on Bus B



segment-arrival-at-node and the position of a slot running on Bus B. Obviously, in the following illustration, we further assume that when node 11 receives a segment to send, the node receives slot 1 by chance simultaneously and the interval between any two successive slots is random. In addition, we do not show values in request

register and countdown register queue in any node, whose operations are similar to DQDB and shown in detail in the next section: specification.

Figure 25 The request principle of a 18-node 6-group DQDB-GR system, for Request slot 3 on Bus B

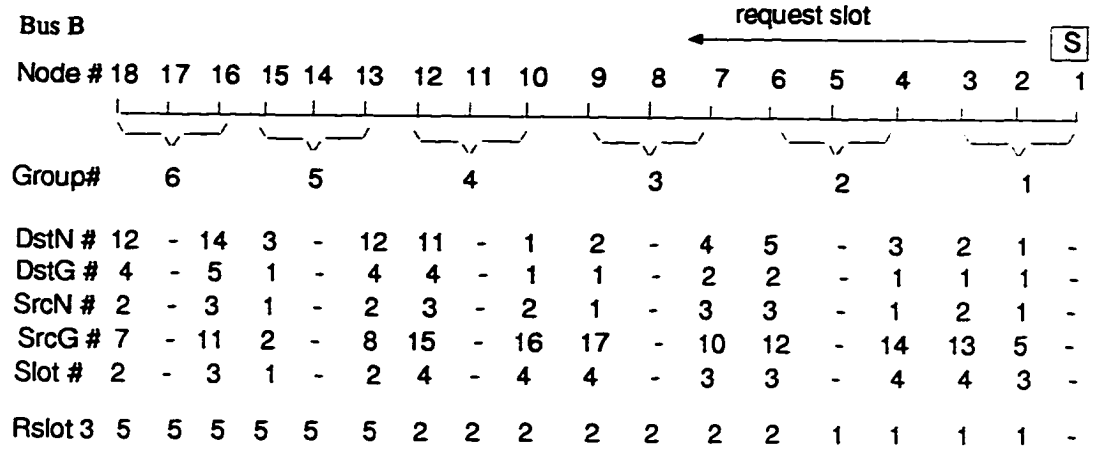
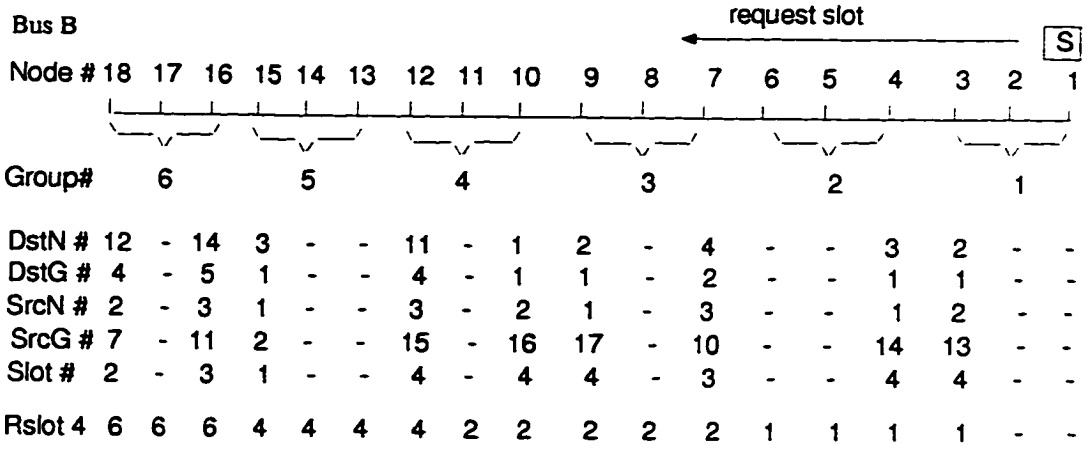


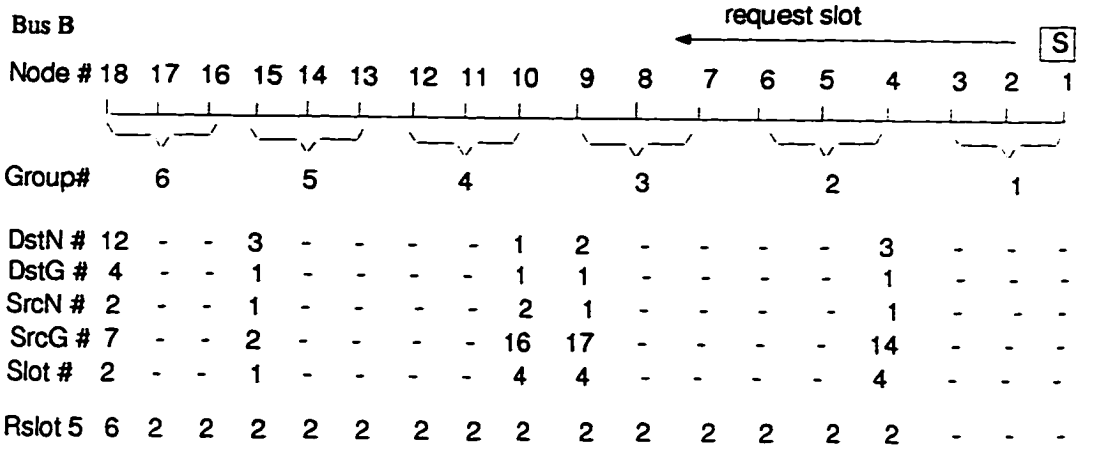
Figure 26 The request principle of a 18-node 6-group DQDB-GR system, for Request slot 4 on Bus B



Focus on the second column in Fig.(23) first. According to the list of symbols, we interpret the column every entry from top to bottom. Node 18 is in group 6. The node has a segment for node 12 in group 4. It is the second node in time order within group 6 to receive a segment for transmission. Its group is the seventh group in time order

within the whole system to receive a segment for transmission. Its segment will meet the second slot on Bus B and the group request code in the first slot received on Bus B is 6. The meanings of the other columns may be obtained by similar translation. The row Slot # with the row Node # means that segments in nodes 11, 15, and 17 will meet slot 1, segments in 5, 8, 13, and 18 slot 2, in 2, 6, 7, 14, and 16 slot 3, and in 3, 4, 9,

Figure 27 The request principle of a 18-node 6-group DQDB-GR system, for Request slot 5 on Bus B



10, and 12 slot 4. The row Rslot 1 means that slot 1, running from node 1 to 18 in order, receives null group request code at node 1 through 10, 4 at 11 through 16 and 6 at 17 through 18. The reason that node 15 that met slot 1 can not send its own request is that the destination group code 1 of the segment node 15 is holding determines that its segment is going to traverse the area that has been requested by group 4 first.

The Fig.(24) shows that after slot 1 passes through the whole system, nodes 11 and 17 have no segment for request. The meanings of all the columns in the Fig.(24) may be obtained by similar translation of those in Fig.(23). It is worthwhile pointing out that node 15 that checked slot 1 checks slot 2 definitely because the node is still holding a segment. In Figs.(25 and 26), you can see similar operations for slot 3 and slot 4. In Fig.(27), all the remaining segment transmission requests will be collected in similar way by slot 5, slot 6,... .

Figure 28 The transmission of a 18-node 6-group DQDB-GR system

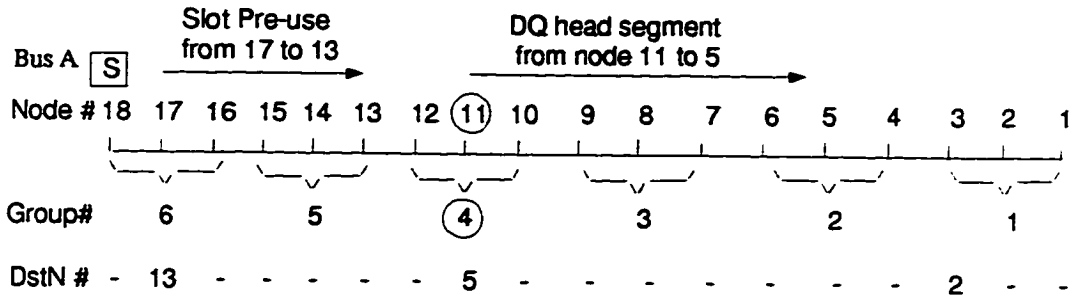


Fig.(28) illustrates how the DQDB-GR network makes segment transmissions for *Slot Pre-Use*. Referring to Figs.(23) and (28), after slot 1 on Bus B finally conveys group 6 request to the head of Bus A with requests by nodes 11 and 17, node 17 in group 6 is the first node and group of the system with a requesting segment to send the free slot S meets. The head segment of the node is for node 13 in group 5. As soon as the slot loaded with the segment of node 17 for node 13 come into group 4, the slot is marked "broadcast". Node 11 in group 4 is the first node and group of the system with another requesting segment to send the released slot S meets. As a matter of fact, node 11 is the head node of distributed queue at the moment, so node 17 pre-uses the slot. Node 11 loads its segment for group 2 in the released slot and sets the H field of the slot. Although according to destination group release rule, the slot carrying the segment of node 11 for node 5 is marked free again when the slot departs the destination group 2, Node 3 can not use the slot again because the H field of the slot is set.

It is clear that slot S in this situation is pre-used once and conveys two segments. One segment was transmitted from node 17 in group 6 to node 13 in group 5 by *Slot Pre-Use*. The second segment was transmitted from node 11 in group 4 to node 5 in

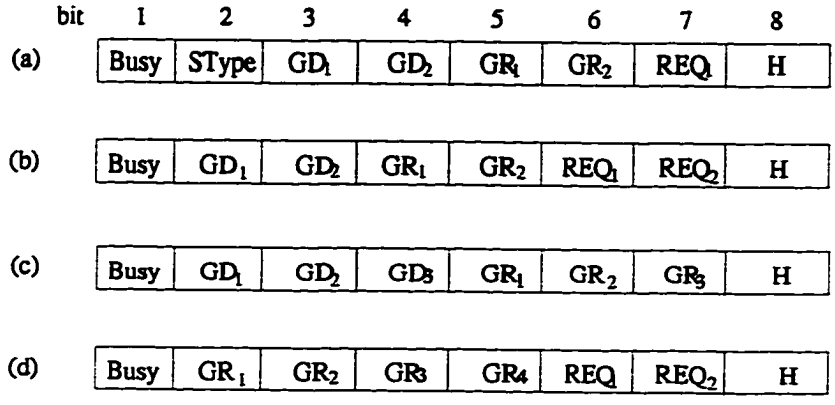
group 2. In the above example, a slot may convey only one segment if all the nodes in all the upstream groups have no requesting segment or only have segments either for the head group of DQ or for a downstream group from the head group. There exists a certain probability for one-segment transmission in the system. A slot may convey either two segments, or more up to maximum six segments with their own probability. For an instant, when the slot meets such a case that the node that slot meets first in each group sends its segment to its own group, a slot may convey at most six segments. But what is the average number of segments a slot may convey in the system? This problem has theoretically been addressed in the previous chapter.

3.3.2 Specifications

3.3.2.1 Modified ACF of a DQDB Slot In Fig.(29), we show the ACF format of the DQDB-GR protocol. Three new fields as introduced in the last section are added in the ACF format: field GD(*i*) to carry a destination group address, field H to signal if the slot has be used by a DQ-head node, and field GR(*i*) to indicate which group is making request. In Fig.(29), four options of the ACF format are provided: Figs.(29.a, b, c, and d). Figs.(29.a and 29.b) both support a three-group DQDB-GR protocol. In Fig.(29.a), bits 1, 2, and 7 follow the definitions of the IEEE 802.6 DQDB standard. The three undefined bits, bits 3, 4, and 5, in a DQDB ACF format and the redefined bits, bits 6 and 8, are defined as, bits 3 and 4 denoted by GD_1 and GD_2 are a GD(2) field : 00 for NULL, 01 for group 1, 10 for group 2, and 11 for group 3. Bits 5 and 6 denoted by GR_1 and GR_1 are a GR(2) field: 00 for NULL, 01 for group 1, 10 for group 2, and 11 for group 3. Bit 8 is field H: 0 for NULL implying that the slot can be used and 1 for SET implying that the slot can not be used by a node. In Fig.(29.b), bit 1 is for a Busy state: 0 for empty and 1 for BUSY. Bits 2 and 3 are the same as

bits 3 and 4 in Fig.(29.a). The only difference of bits 2 and 3 from bits 3 and 4 is that their NULL state together with state BUSY of field Busy is for a special type of slots that are used to broadcast management messages, their acting the same as bit 2

Figure 29 ACF Formats of A DQDB-GR Slot.



in Fig.(29.a). Bits 4 and 5 act the same as bits 5 and 6 in Fig.(29.a). Bit 8 acts the same as in Fig.(29.a). Fig.(29.c) is for a seven-group DQDB-GR protocol, and Fig. (29.d) is for a fifteen-group DQDB-GR protocol. The REQ fields in the corresponding versions are used for priority requests. Figs.(11 a and b) show a MAC frame format and a slot format, respectively. It is clear that the MAC frame is segmented and encapsulated into a number of slots. Part of the destination address field (DA) of the MAC frame format can then be used to define more number of groups as field GD(*i*).

3.3.2.2 Node Structure of DQDB-GR Fig.(30) shows the structure of a typical DQDB-GR node. A waiting queue (*Q) is used to keep segments from the local host and a request counter *RQ* follow the similar functionality to that of a standard DQDB. However, we add the following devices in each node. One register (GC) is used for the node to keep its own group code (LC). An outstanding queue (*CD) is used to keep segments that have made a request. Each element of the (*CD) consists of three parts: a countdown counter (*CD#*) whose value is from the request counter

RQ when the local node sends a request for the corresponding segment transmission,

Figure 30 Structure of A DQDB-GR Node

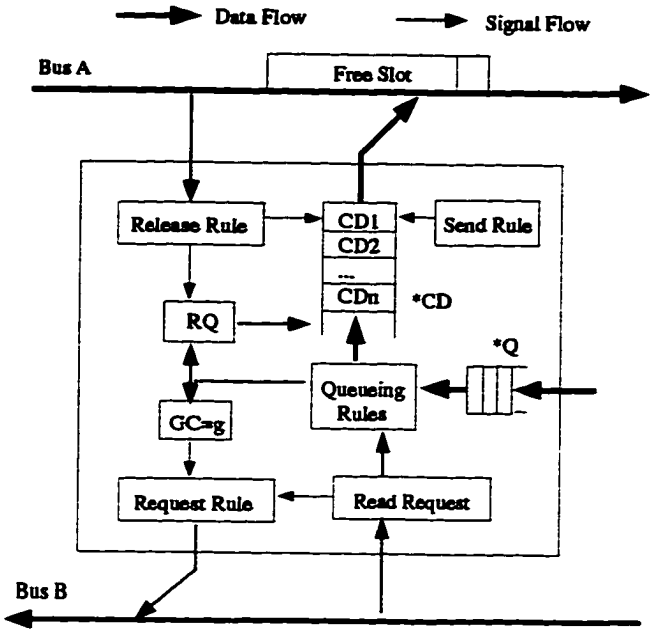
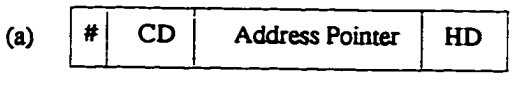
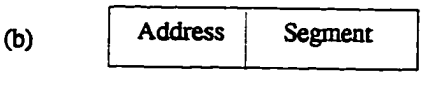


Figure 31 Elements of (*CD) and (*Q) for DQDB-GR



An element of an outstanding queue *CD



An element of a waiting queue *Q

an address-pointer (see Fig.31a) that points to a segment in the (*Q) (see Fig.31b), and a flag HD to indicate if the segment made its request by a slot with value NULL in the GR field. If HD is SET, the segment made its request by a slot with value NULL in the GR field. If HD is NULL, the segment made its request by a slot with a non-NULL value in the GR field. One slot group request rule machine decides when the node may mark a request in a slot. One slot group release rule machine decides

when the node may release a busy slot. One group transmission rule machine decides when the node may send its segment in the (*CD) or in the (*Q).

3.3.2.3 Operations of DQDB-GR Fig.(32) demonstrates operations of a DQDB-GR node machine. As usual, vertical bold lines are for states of the machine and the names of states can be found at top of the lines. Arrow-head lines are for transitions of states, letters above these lines are for events that enable the transition, and letters below these lines are for actions that execute when the events above the transition line occur. Tables (15 and 16) provide the explanation of the events and the actions in Fig.(32).

All the machines have two states: IDLE under which there is no segment in their (*CD) waiting for channel and ACTIVE under which there is at least one segment in their (*CD) waiting for channel.

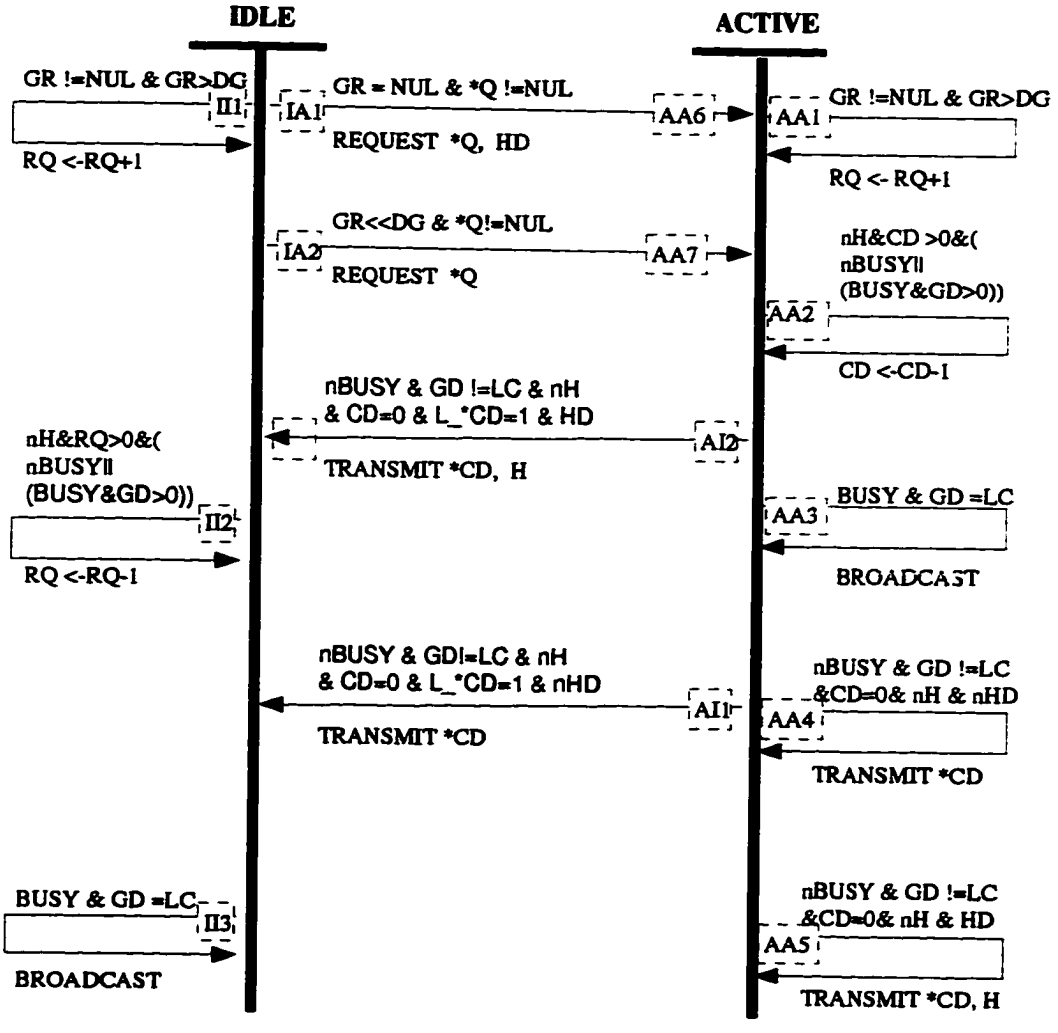
All the machines have two states: IDLE under which there is no segment in their (*CD) waiting for channel and ACTIVE under which there is at least one segment in their (*CD) waiting for channel.

The machine in IDLE takes loop-transition Π1 that raises RQ by one when the machine receives a non-null request nNUL in field GR in a coming slot on Bus B and the request group code in field GR of the slot is greater than the destination group code DG of the segment at the head position in the (*Q).

The machine in IDLE takes loop-transition Π2 that reduces RQ by one when the machine with its RQ greater than zero receives either a nBUSY slot with value NULL in field H of the slot on Bus A or a BUSY slot with a non-null group code in field GD and value NULL in field H on Bus A.

The machine in IDLE takes loop-transition II3 that broadcasts a coming slot on Bus A by marking nBUSY in field Busy of the slot when the machine receives a BUSY slot on Bus A with the local group code LC in field GD of the slot.

Figure 32 DQDB-GR Machine



The machine in ACTIVE takes loop-transition AA1 that raises RQ by one when the machine receives a non-null request nNUL in field GR of a coming slot on Bus B and the request group code in field GR of the slot is greater than the destination group code DG of the segment at head position in the (*Q).

The machine in ACTIVE takes loop-transition AA2 that reduces CD in the head of the (*CD) by one when the machine with its first CD value in its (*CD) greater than zero receives either a nBUSY slot with value NULL in field H of the slot on Bus A or a BUSY slot with a non-null group code in field GD and value NULL in field H on Bus A.

The machine in ACTIVE takes loop-transition AA3 that broadcasts a coming slot on Bus A by marking nBUSY in field Busy of the slot when the machine receives a BUSY slot carrying the local group code LC in field GD of the slot.

The machine in ACTIVE takes loop-transition AA4 that transmits a segment in the head of its (*CD), shifts the (*CD) toward the head by one position and marks BUSY in field Busy of a slot on Bus A when the machine with value NULL in the first HD field of and value zero in the corresponding CD field of its (*CD) receives the nBUSY slot with non-local-group-code in field GD of and value NULL in field H of the slot.

The machine in ACTIVE takes loop-transition AA5 that transmits a segment in the head of its (*CD), shifts the (*CD) toward the head by one position, marks BUSY in field Busy of a slot on Bus A and sets field H of the slot when the machine with value NULL in the first HD field of and value zero in the corresponding CD field of its (*CD) receives the nBUSY slot with non-local-group-code in field GD of and value NULLin field H of the slot.

The machine in ACTIVE takes loop-transition AA6 that does the same as IA1 under the same events as IA1. For brief, the transition is only marked in name but not in a line.

The machine in ACTIVE takes loop-transition AA7 that does the same as IA2 under the same events as IA2. For brief, the transition is only marked in name but

not in a line.

The machine in IDLE takes transition IA1 that changes its state from IDLE to ACTIVE by putting the address of a segment from the head of its (*Q), its request counter RQ value into part pointer, part CD of the tail of its (*CD) respectively and setting part HD of the (*CD), and marking with its local group code LC in field GR of the slot on Bus B when the machine with non-empty queue (*Q) !=NUL receives a null-group-request-code NUL in field GR of a slot on Bus B.

The machine in IDLE takes transition IA2 that changes its state from IDLE to ACTIVE by putting the address of a segment from the head of its (*Q) and its request counter RQ value into part pointer and part CD of the tail of its (*CD) respectivley and marking with its local group code LC in field GR of the slot on Bus B when the machine with non-empty queue (*Q) !=NUL receives a group request code in field GR of a slot on Bus B and the group code is less than the destination group code DG of the first segment in its (*Q).

The machine in ACTIVE takes transition AI1 that changes its state from ACTIVE to IDLE by transmitting a segment in the head of its (*CD), shifting the (*CD) toward the head by one position, and marking BUSY in Busy field of a slot when the machine with NULL value in the first HD field and zero value in the first CD field that is the last one left in the (*CD), (i.e., L_*CD=1), receives on Bus A the nBUSY slot with non local group code in field GD of and value NULL in field H of the slot.

The machine in ACTIVE takes transition AI2 that changes its state from ACTIVE to IDLE by transmitting a segment in the head of its (*CD), shifting the (*CD) toward the head by one position, marking BUSY in field Busy of a slot, and setting field H of the slot to one when the machine with non-NULL value in the first HD field and value zero in the first CD field that is the last one left in the (*CD), (i.e., L_*CD=1),

receives the nBUSY slot on Bus A with non local group code in field GD of and value NULL in field H of the slot.

3.3.3 Simulation Results

3.3.3.1 Simulation Parameters The simulation collects data from 60,000 full life-time slots. These data measure throughput gain, media access delay, utilization, and segment loss ratio of the whole network systems under Poisson traffics with average arrival rate of 1.0, 1.5, 2.0, 4.0, 7.0, 10.0 segments per slot-time. A comparison among various grouping method was performed (i.e., performance measures for one-group system up to fifteen groups system.).

3.3.3.2 Performance Results In Fig.(33) we show simulation results of the segment loss ratio. From the figure, we see that the loss ratios decrease as the number of groups

Figure 33 Segment Loss Ratio vs Number of Grouping of DQDB-GR under The Offered Segment-Arrival Rates

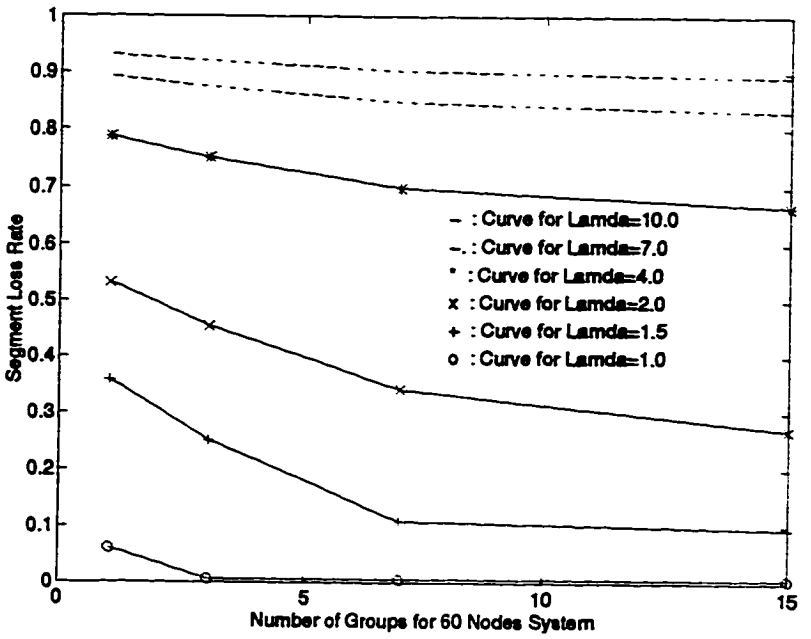


Figure 34 Access Delay vs Number of Grouping of DQDB-GR under The Offered Segment-Arrival Rates

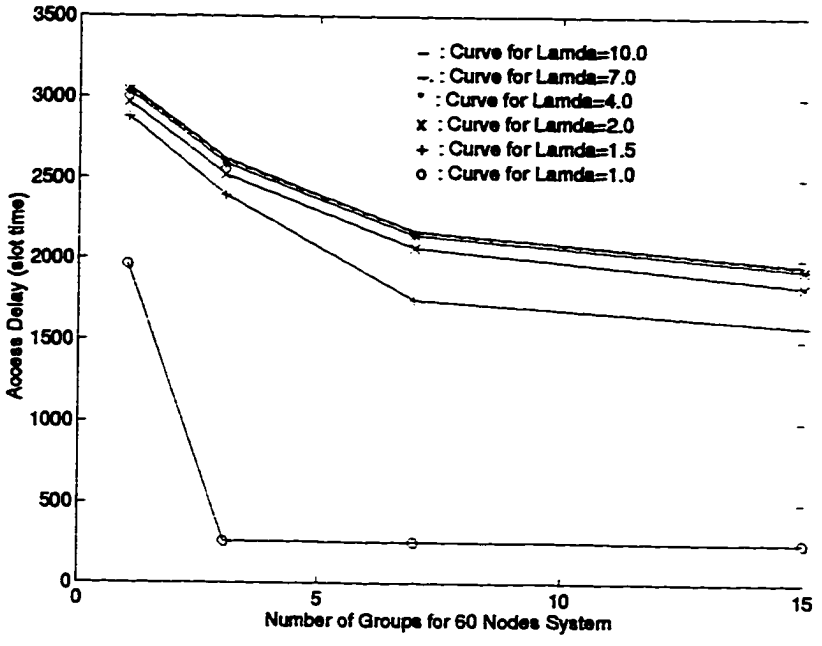


Figure 35 Slot-time Utilization vs Number of Grouping of DQDB-GR under The Offered Segment-Arrival Rates

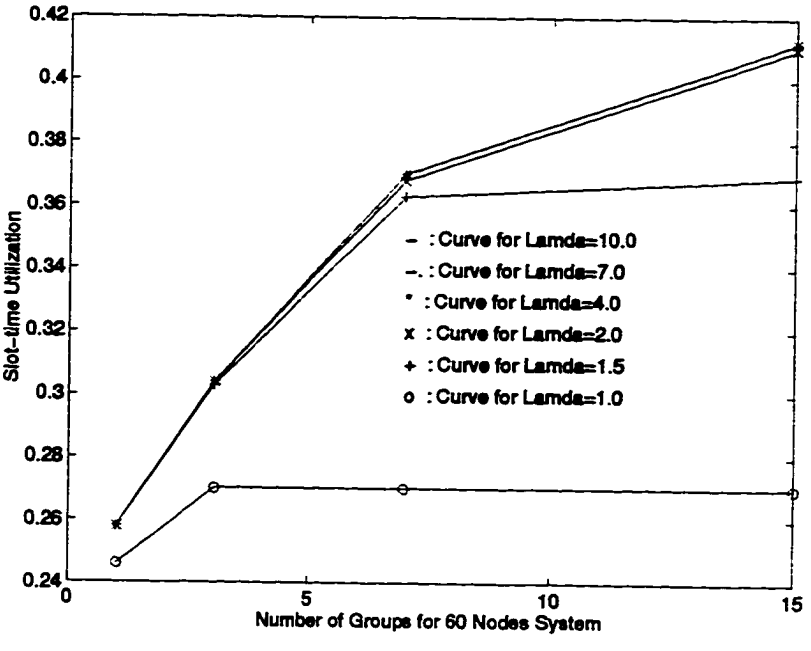
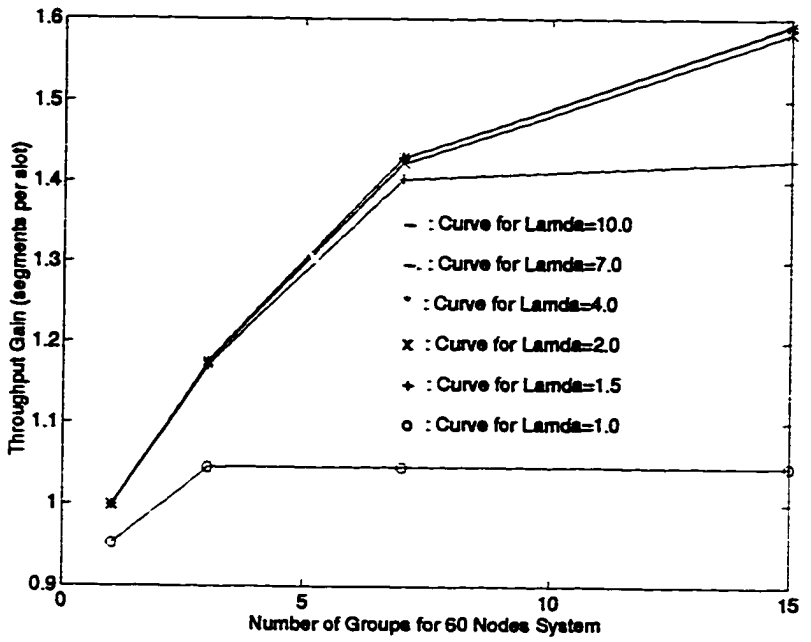


Figure 36 Throughput Gain vs Number of Groups of DQDB-GR under The Offered Message-Arrival Rates



increases. This is due to the fact that as the number of groups increases, throughput increases so that the probability that the system buffers are full decreases.

Fig.(34) show simulation results of the average media access delay. From the figure, it is clear that the delay decreases as the number of groups increases. The reason that causes this is that more groups in the system directly provides more chances to every segment waiting in the system to be carried by a slot so that the waiting time for a segment to stay in the system with more groups becomes shorter than the system with less groups.

Fig.(35) shows simulation results of the slot-time utilization. From the Figures, it is clear that the utilization increases as the number of groups increases. When grouping number of a DQDB-GR system is 1, the utilization is 0.26.

Fig.(36) shows simulation results of the throughput gain. From the figure, we see that the throughput gains increase with the increment of number of groups under

Poisson traffic with average arrival rate of 1.0, 1.5, 2.0, 4.0, 7.0, 10.0 segments per slot-time.

Figure 37 Comparisons of Results of Simulation & Theory for DQDB-G

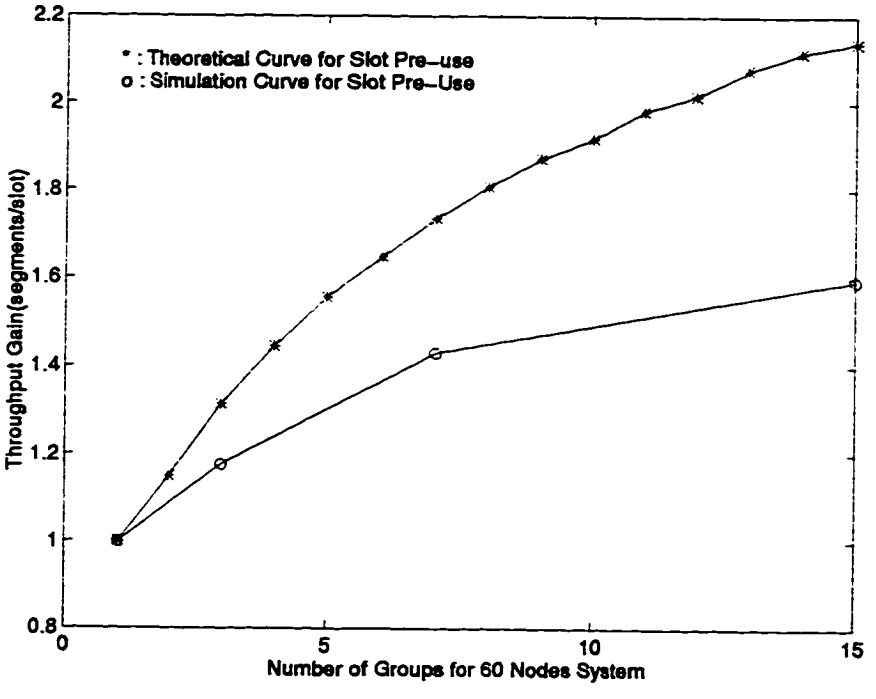


Fig.(37) shows two throughput gain curves: one from theorem, the other from simulation, both of which are for the same grouping network as the described above. We see that theoretical curve is higher than simulative curve. The chief reason for this is that the machine we presented here itself does not make full use of transmission chance that is expected by our theorem. In addition, theoretical curve in the figure is obtained by calculating *Theorem 6*.

3.3.4 Remarks

DQDB-GR is a new protocol for high throughput and little access delay at the MAN-MAC layer. Its throughput obtains improvement through the grouping concepts, its group request rule, its slot group release rule, and its group transmission rule. The throughput gain of a DQDB-GR system increases as the number of groups increases. Of course, too large number of groups being used makes the system complex so that the capacity of the system goes down. Unlike SP-DQDB, DQDB-GR does not need to buffer neighbor traffic segments that SP-DQDB has to buffer. Hence, DQDB-GR is simple and achieves higher system capacity than the SP-DQDB system.

3.4 DQDB-G Protocol for Slot Pre-Use&Reuse

I am going to describe overview and specifications of the protocol in the next two subsections, and then exhibit simulation results.

3.4.1 Overview of DQDB-G Protocol

The proposed DQDB-G protocol both for *Slot Pre-Use* and for *Slot Reuse* provides for higher throughput via four folds: 1) the grouping of all the nodes in the system; 2) slot post-destination group releasing rule; 3) slot multiple-use group transmission rule and 4) group request rule.

Refer to Chapter 2 for the principle of grouping.

It is worthwhile mentioning that the distributivity of the DQDB-G protocol can be guaranteed by an adequate management protocol for the DQDB-G protocol so that in case of the occurrence of one link-fault on the DQDB-G, the DQDB-G management protocol is capable of re-grouping the nodes and activating proper group code for each node in its group. As a result, DQDB-G can be also one-fault tolerance.

The group request rule works as follows. Distinguishing from the request rule of DQDB that only uses the one-bit request field in a slot, the DQDB-G request rule uses i bits as the GR(i) field. The i in GD(i) means that the GR has$i$ bits and addresses ($2i$-1) groups. A node holding the segment has to send a request to the upstream head-end of Bus A through Bus B. The GR(i) field will be filled with its local group code by a node when the node compare the content in the GR(i) field of a slot with the group destination code of the first segment in its waiting queue (*Q) and find that the group destination code of the waiting segment is not going to reach the position indicated by the GR(i) field of the slot. A node will set a flag to remember that the node itself is the first to have used a slot for request if it receives the slot with NULL value in field GR(i) of the slot. The flag is called HD. The RQ counter increases by one when a node receives on bus B a slot with non-NULL value in the GR(i) field that is greater than the group destination code of the first segment in the waiting queue (*Q). The RQ counter decreases by one when a node with RQ>0 and empty (*CD) receives on bus A either a non-Busy slot with H unset, or a Busy slot with H unset and GD content equal to the local group code. This request group code signals upstream-nodes that the group region denoted by the group code and its downstream region are not available. In the other words, this request group code signals upstream-nodes that a node in the upstream groups holding a segment that is destined to a group before the group indicated by the GR(i) field may send its transmission request by replacing the existing group code in the GR(i) field in the passing slot with its own local group code.

The slot post-destination group release rule works as follows. Within each group, each free slot can only carry a segment. A slot carrying a segment will become free when the slot departs from its destination group. A node resets the Busy-status field to zero when the local group code of the node matches the contents of field GD(i) of

a coming slot so that the slot is in non-busy state. A node resets field GD(i) of a slot to NULL when the node receives a non-busy slot with a non-NULL value not equal to its local group code in field GD(i) of the slot. In this way, a busy slot becomes free for post-destination groups reuse by lower index nodes that are situated downstream from the destination group.

The group transmission rule works as follows. The determination of a free slot is done by two fields of a slot rather than only one field: field Busy and field Destination_Group (GD(i)). The i in GD(i) has the same functions as GR(i) groups. A slot is free only if the Busy field is NULL and the GD field is not the local group code of the node. A one-bit field is required in a slot format to indicate whether the slot is free either for its outstanding queue (*CD) slot pre-use/use or for slot reuse, denoted by H. In other words, field H is to indicate if the slot passed all the nodes that requested for it. The H field of a slot may be set only by such nodes that have flag HD set when the nodes are permitted to transmit. A free slot with H unset is free for queue (*CD), allowing pre-use/use the slot. Whereas a free slot with H set is free for slot reuse. If a slot is free for slot pre-use/use, similar to the countdown-CD-to-zero-for-transmission rule of DQDB protocol, the DQDB-G rule for transmission is: countdown (CD) at the head of its (*CD) by one when a node with non-empty (*CD) receives on bus A either a non-Busy slot with H unset, or a Busy slot with H unset and GD content equal to the local group code until the CD reaches zero, then the node may loads its segment from the first position of the (*CD) into a coming free slot. If a slot is free for slot reuse, the machine will check if the (*CD) is ready to transmit first. If the (*CD) is non-empty, a transmission may be made without reference to any CD value from (*CD). If (*CD) is not empty, the machine will check if its waiting queue (*Q) is empty. If no, a transmission may be made from (*Q).

Figs.(23,24,25,26 and 27) with legend table (22) illustrate how the DQDB-G protocol make requests in the system by an example of a 18–node DQDB system in 6 groups 3 nodes each with 15 segments to send. In this illustration, we assume that the arrival process of segments in system is a stochastic process. Note that we ignore technical detail about how a node of DQDB-G operates such as transmission request and slot group release but focus on request phenomena on bus B and segment transmission on bus A caused by using a DQDB-G. About the technical detail we will present in the next section.

Before we explain the Fig.(23), it is worthwhile pointing out that the node that is first in the whole network system receiving a segment to send must not be the node that is first receiving a slot for request usually. This depends both on time order of segment-arrival-at-node and the position of a slot running on Bus B. Obviously, in the following illustration, we further assume that when node 11 receives a segment to send, the node receives slot 1 by chance simultaneously and the interval between any two successive slots is random. In addition, we do not show values in request register and countdown register queue in any node, whose operations are similar to DQDB and shown in detail in the next section: specification.

Refer to Section 3.3.1 for the illustration of the requesting principle.

Fig.(2) illustrates how the DQDB-G protocol makes transmissions both for *Slot Pre-Use* and for *Slot Reuse*. In Figs.(23 and 2), after slot 1 on Bus B finally conveys group 6 request to the head of Bus A with requests by nodes 11 and 17, node 17 in group 6 is the first node and group of the system with a requesting segment to send the free slot S meets. The head segment of the node is for node 13 in group 5. As soon as the slot loaded with the segment of node 17 for node 13 comes into group 4, the slot is marked "free". Node 11 in group 4 is the first node in the system with

requesting segment that the released slot S meets. As a matter of fact, node 11 is the DQ head node at the moment. So, node 17 pre-uses the slot. Node 11 loads its segment for group 2 in the released slot and sets flag H field of the slot. Although node 3 did not make a request for its segment for node 2, node 3 can reuse the slot because the H flag of the slot was set and the slot carrying the segment of node 11 for node 5 is marked free again when the slot departs the destination group 2 according to the destination group release rule.

It is clear that slot S in this situation is pre-used and reused once respectively and conveys three segments. One segment was transmitted from node 17 in group 6 to node 13 in group 5 by *Slot Pre-Use*. The second segment was transmitted from node 11 in group 4 to node 5 in the group 2, which is by the DQ head node transmission. The other segment was transmitted from node 3 in group 1 to node 2 in the same group by *Slot Reuse*. In the above example, a slot may convey only one segment if all the nodes in all the upstream and downstream groups have no requesting segment or all the nodes in all the upstream groups only have requesting segments either for the DQ head group or for a downstream group from the DQ head group and the DQ head node holds the segment for mostdownstream group. There exists a certain probability for one-segment transmission in the system. A slot may convey either two segments, or more up to maximum six segments with their own probability. For an instant, when the slot meets such a case that the node the slot meets first in each group sends its segment to its own group, a slot may convey at most six segments.

3.4.2 Specifications

The ACF, node structure, and operations of a DQDB-G protocol are specified in the following, respectively.

3.4.2.1 Modified ACF of a DQDB Slot In Fig.(29), we show the ACF format of our DQDB-G protocol. Refer to Section 3.3.2 for the details.

3.4.2.2 Structure of a DQDB-G Node Fig.(30) shows the structure of a typical DQDB-G node. Refer to Section 3.3.2 for the details.

3.4.2.3 Operations of The Node of The DQDB-G Fig.(32) demonstrates operations of a DQDB-GR node machine. As usual, vertical bold lines are for states of the machine and the names of states can be found at top of the lines. Arrow-head lines are for transitions of states, letters above these lines are for events that enable the transition, and letters below these lines are for actions that execute when the events above the transition line occur. Tables (15 and 16) provide the explanation of the events and the actions in Fig.(32).

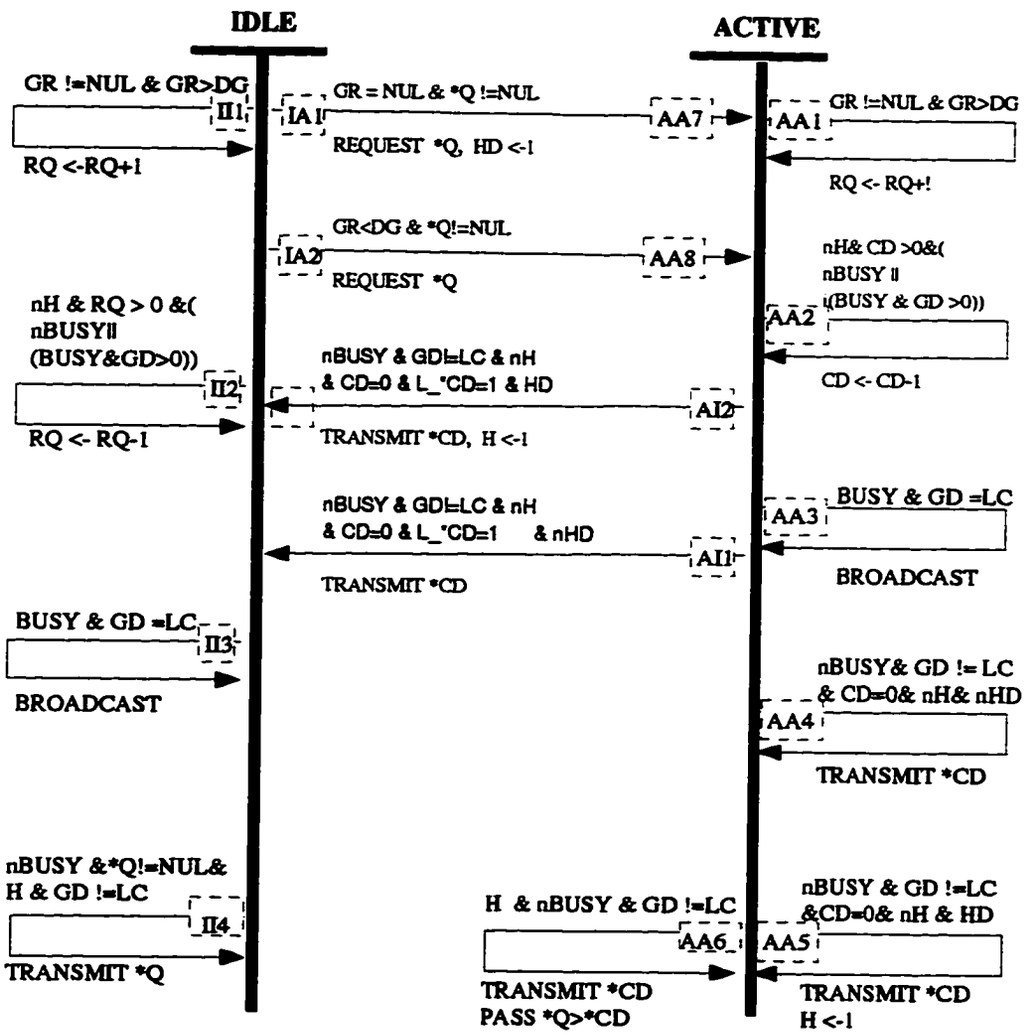
All the machines have two states: IDLE under which there is no segment in their (*CD) waiting for channel and ACTIVE under which there is at least one segment in their (*CD) waiting for channel.

The machine in IDLE takes loop-transition Π1 that raises RQ by one when the machine receives a non-null request nNUL in field GR in a coming slot on Bus B and the request group code in field GR of the slot is greater than the destination group code DG of the segment at the head position in the (*Q).

The machine in IDLE takes loop-transition Π2 that reduces RQ by one when the machine with its RQ greater than zero receives either a nBUSY slot with value NULL in field H of the slot on Bus A or a BUSY slot with a non-null group code in field GD and value NULL in field H.

The machine in IDLE takes loop-transition Π3 that broadcasts a coming slot on Bus A by marking nBUSY in field Busy of the slot when the machine receives a BUSY slot on Bus A with the local group code LC in field GD.

Figure 38 DQDB-G Machine



The machine in IDLE takes loop-transition Π4 that transmits a segment at the head of its (*Q), shifts the (*Q) toward the head by one position, marks BUSY in field Busy of a slot on Bus A when the machine with non-empty (*Q) receives the nBUSY slot with non-local-group-code in field GD of and value SET in field H.

The machine in ACTIVE takes loop-transition AA1 that raises RQ by one when the machine receives a non-null request nNUL in field GR of a coming slot on Bus B and the request group code in field GR of the slot is greater than the destination group code DG of the segment at head position in the (*Q).

The machine in ACTIVE takes loop-transition AA2 that reduces CD in the head of the (*CD) by one when the machine with its first CD value in its (*CD) greater than zero receives either a nBUSY slot with value NULL in field H of the slot on Bus A or a BUSY slot with a non-null group code in field GD and value NULL in field H.

The machine in ACTIVE takes loop-transition AA3 that broadcasts a coming slot on Bus A by marking nBUSY in field Busy of the slot when the machine receives a BUSY slot carrying the local group code LC in field GD.

The machine in ACTIVE takes loop-transition AA4 that transmits a segment in the head of its (*CD), shifts the (*CD) toward the head by one position and marks BUSY in field Busy of a slot on Bus A when the machine with value NULL in the first HD field of and value zero in the corresponding CD field of its (*CD) receives the nBUSY slot with non-local-group-code in field GD of and value NULL in field H.

The machine in ACTIVE takes loop-transition AA5 that transmits a segment in the head of its (*CD), shifts the (*CD) toward the head by one position, marks BUSY in field Busy of a slot on Bus A and sets field H of the slot when the machine with value NULL in the first HD field of and value zero in the corresponding CD field of its (*CD) receives the nBUSY slot with non-local-group-code in field GD of and value NULLin field H.

The machine in ACTIVE takes loop-transition AA6 that transmits a segment at the head of its (*CD), shifts the (*CD) queue toward the head by one position, marks

BUSY in field Busy of a slot on Bus A, sets field Hof the slot to one, passes a segment from the head of the (*Q) to the tail of the (*CD), and shifts the (*Q) toward the head by one position when the machine with non-empty (*Q) receives a nBUSY slot with non local group code in field GD of and value SET in field H.

The machine in ACTIVE takes loop-transition AA7 that does the same as IA1 under the same events as IA1. For brief, the transition is only marked in name but not in a line.

The machine in ACTIVE takes loop-transition AA8 that does the same as IA2 under the same events as IA2. For brief, the transition is only marked in name but not in a line.

The machine in IDLE takes transition IA1 that changes its state from IDLE to ACTIVE by putting the address of a segment from the head of its (*Q), its request counter RQ value into part pointer, part CD of the tail of its (*CD) respectively and setting part HD of the (*CD), and marking with its local group code LC in field GR of the slot on Bus B when the machine with non-empty queue (*Q) !=NUL receives a null-group-request-code NUL in field GR of a slot on Bus B.

The machine in IDLE takes transition IA2 that changes its state from IDLE to ACTIVE by putting the address of a segment from the head of its (*Q) and its request counter RQ value into part pointer and part CD of the tail of its (*CD) respectivley and marking with its local group code LC in field GR of the slot on Bus B when the machine with non-empty queue (*Q) !=NUL receives a group request code in field GR of a slot on Bus B and the group code is less than the destination group code DG of the first segment in its (*Q).

The machine in ACTIVE takes transition AI1 that changes its state from ACTIVE

to IDLE by transmitting a segment in the head of its (*CD), shifting the (*CD) toward the head by one position, and marking BUSY in Busy field of a slot when the machine with NULL value in the first HD field and zero value in the first CD field that is the last one left in the (*CD), (i.e., L_*CD=1), receives on Bus A the nBUSY slot with non local group code in field GD of and value NULL in field H of the slot.

The machine in ACTIVE takes transition AI2 that changes its state from ACTIVE to IDLE by transmitting a segment in the head of its (*CD), shifting the (*CD) toward the head by one position, marking BUSY in field Busy of a slot, and setting field H of the slot to one when the machine with non-NULL value in the first HD field and value zero in the first CD field that is the last one left in the (*CD), (i.e., L_*CD=1), receives the nBUSY slot on Bus A with non local group code in field GD of and value NULL in field H of the slot.

3.4.3 Simulation Results

3.4.3.1 Simulation Parameters The simulation collects data from 60,000 full life-time slots. These data measure throughput gain, media access delay, slot time utilization, and segment loss ratio of the whole system under Poisson traffics with average arrival rate of 1.0, 1.5, 2.0, 3.0, 6.0, 16.0 segments per slot-lifetime. A comparison among various grouping method was performed (i.e., performance measures for one-group system up to fifteen groups system.).

3.4.3.2 Performance Results Fig.(39) show simulative results of the segment loss ratio. From the Fig., we see that the loss ratios decrease with the increment of number of groups on average under Poisson traffics. This is due to the fact that as the number of groups increases, throughput increases so that the probability that the system buffers are full decreases.

Fig.(40) show simulative results of the average media access delay. From the Fig., we see that the delays decrease with the increment of number of groups on average under Poisson traffics. The reason that causes this happen is that more groups in the system directly provides more chances to every segment waiting in the system to be carried by a slot so that the waiting time for a segment to stay in the system with more groups becomes shorter than the system with less groups.

Fig.(41) show simulative results of the channel utilization. From the Fig., we see that the utilizations increase with the increment of number of groups on average under Poisson traffics. When grouping number of a DQDB-G system is 1, the utilization is 0.26.

Fig.(42) show simulative results of the throughput gain. From the Fig., we see that the throughput gains increase with the increment of number of groups under Poisson traffic with average arrival rate λ of 1.0, 1.5, 2.0, 3.0, 6.0, 16.0 segments per slot-

Figure 39 Segment Loss Ratio vs Number of Grouping of DQDB-G under The Offered Segment-Arrival Rates

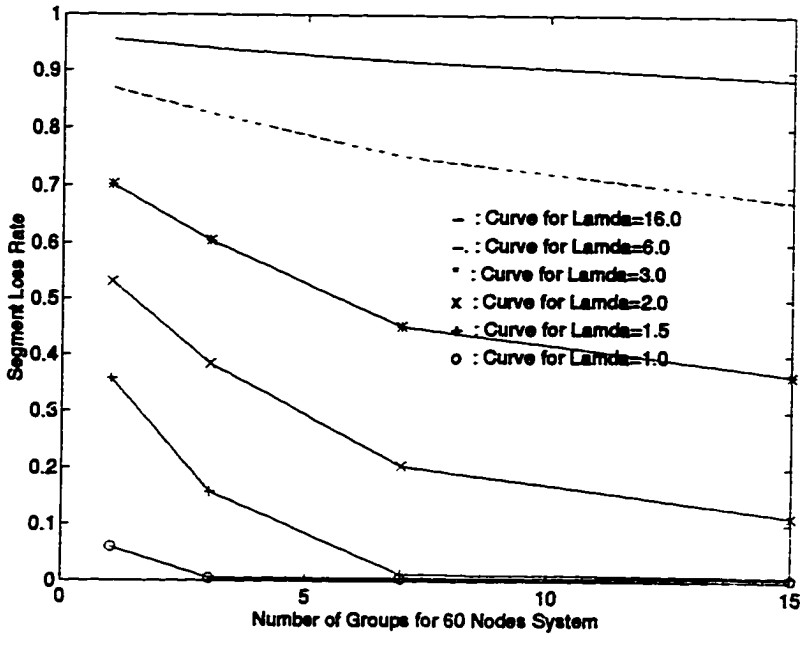


Figure 40 Access Delay vs Number of Grouping of DQDB-G under The Offered Segment-Arrival Rates

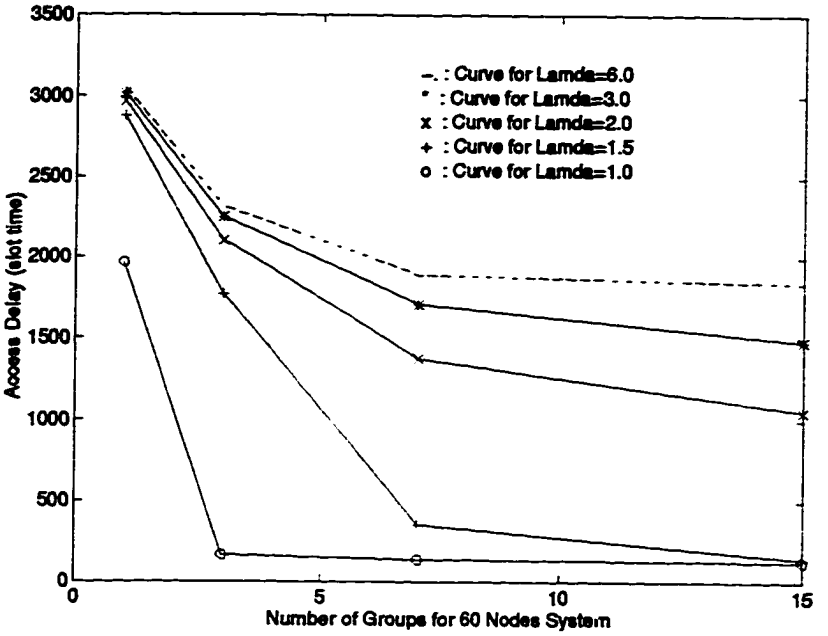


Figure 41 Slot Time Utilization vs Number of Grouping of DQDB-G under The Offered Segment-Arrival Rates

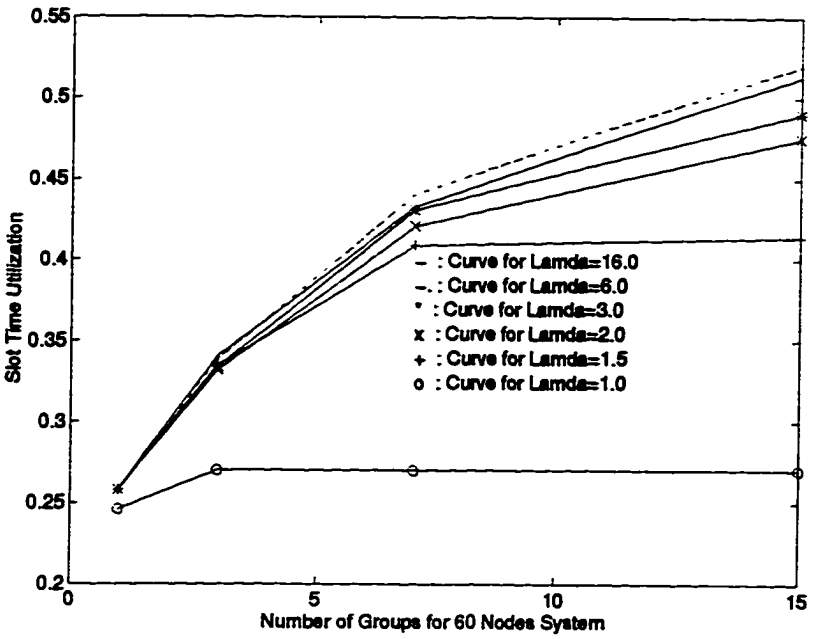
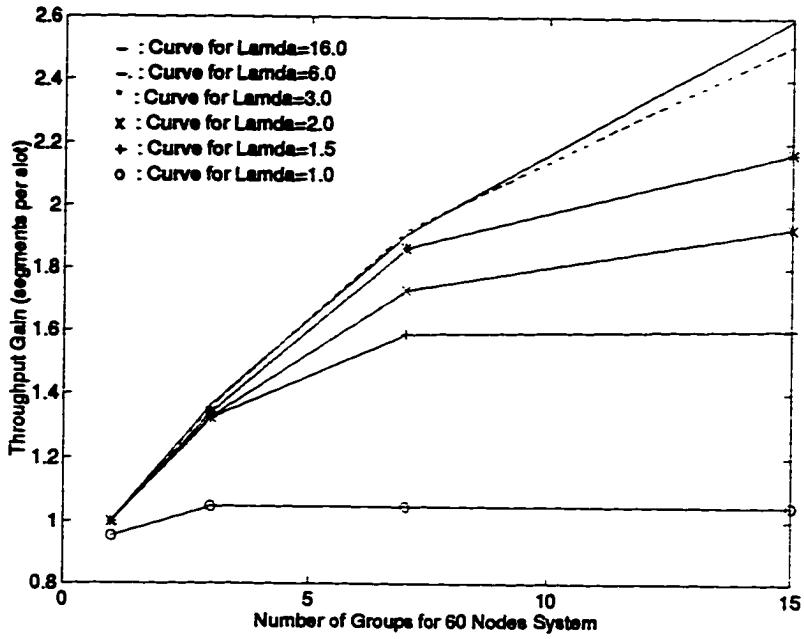


Figure 42 Throughput Gain vs Number of Grouping of DQDB-G under The Offered Message-Arrival Rates



time. When grouping number of a DQDB-G system is 1, the system become a DQDB system and the throughput gain is 1.

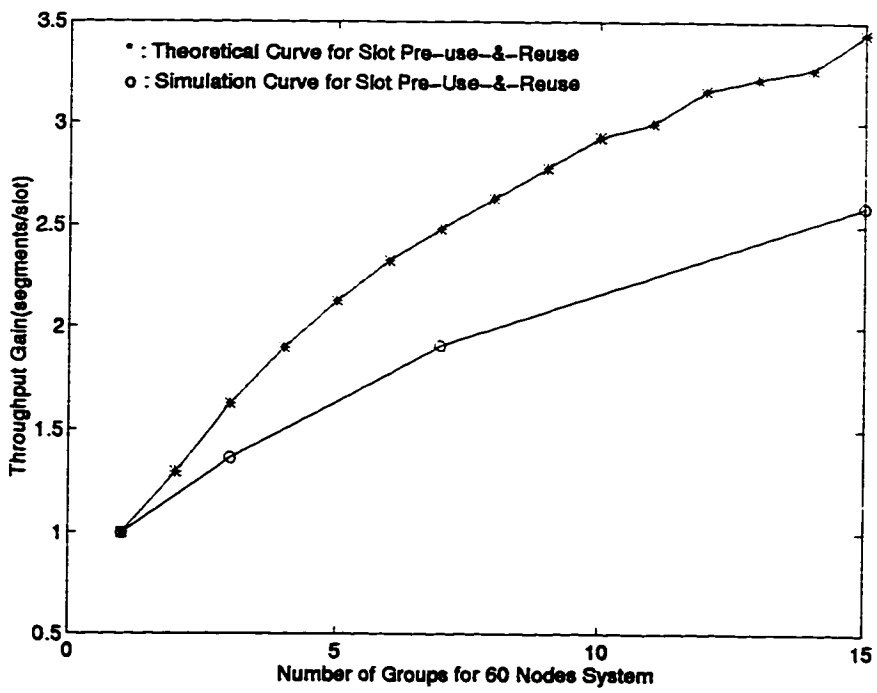
Fig.(43) shows two throughput gain curves: one from our theory, the other from simulation, both of which are for the same grouping network as the described above. We see that theoretical curve is higher than simulative curve. The chief reason for this is that the machine we presented here itself does not make full use of transmission chance that is expected by our theorem. In addition, theoretical curve in the figure is obtained by calculating *Theorem 6*.

3.4.4 Remarks

A new protocol for high throughput and little access delay at a MAC layer, a DQDB-G protocol, is presented. It supports for both *Slot Reuse* and *Slot Pre-Use* by inserting a simple scheme, resulting in high throughput. The main features of this protocol include: 1) Dividing all the nodes in the system into multiple groups and

assigns a group code to each group, 2) a node may send its transmission request when the node find that the destination group code of its segment does not reach the group indicated by the request field GR of a slot, 3) a busy slot can be reused after the slot pass through its destination group, and 4) a slot can be pre-used before the slot reaches the group that is first requesting for it. The throughput gain of a DQDB-G system increases as the number of groups of the system increases. Of cause, too large number of groups being used makes the system complex so that the capacity of the system goes down.

Figure 43 Comparisons of Results of Simulation & Theory for DQDB-G



3.5 DQDB-NSP Protocol for Slot Pre-Use

I am going to describe the overview and specifications of the protocol in the next two subsections, and then exhibit simulation results.

3.5.1 Overview of DQDB-NSP Protocol

The proposed DQDB-NSP protocol for *Slot Pre-Use* provides for higher throughput gain via five folds: 1) the grouping of all the nodes in the system; 2) slot post-destination group releasing rule; 3) slot pre-use multi-transmission rule; 4) group request rule; and 5) slot request status record/erasure rule.

Refer to Chapter 2 for the principle of grouping.

As we shall illustrate later, the grouping concept has a significant impact on the delivered throughput. The higher throughput is achieved because nodes located upstream from a head-group of the DQ will have a chance of using free slots before a head-node in a DQ head-group uses it.

It is worthwhile mentioning that the distributivity of the DQDB-NSP protocol can be guaranteed by an adequate management protocol such that in case of the occurrence of one fault on the DQDB-NSP, the DQDB-NSP management protocol is capable of re-grouping the nodes and activating proper group code for each node. As a result, DQDB-NSP can be also one-fault tolerance.

Three new fields are defined in ACF of a slot format. They are GROUP_DESTINATION, denoted by GD, which carries destination group code, GROUP_REQUEST, denoted by GR, which carries requesting group code, and DQ_HEAD, denoted by H, which indicates if the slot has been used by a DQ-head node. The sizes of fields GD and GR are determined by the number of grouping. The DQ_HEAD is one-bit-wise.

The slot post-destination group release rule works as follows. Within each group, each free slot can only carry a segment. A slot carrying a segment will become free when the slot departs from its destination group. A node resets the Busy-status field

to zero when the local group code of the node matches the contents of the GD field of a coming slot so that the slot is in non-busy state. A node resets the GD field of a slot to NULL when the node receives a non-busy slot with a non-NULL value not equal to its local group code in the GD field of the slot. In this way, a busy slot becomes free for post-destination groups reuse by lower index nodes that are situated downstream from the destination group.

The group transmission rule works as follows: The determination of a free slot is done by three fields of a slot rather than only one field: field Busy and field DESTINATION_GROUP denoted by GD(i), and field H. The i in GD(i) means that i-bit Group Destination field covering ($2i$-1) groups. The H Field is used to indicate if a slot has passed by all nodes that requested it and may be set only by such node that is the first to request the slot. Note that no node can use a slot with field H set. A slot is free only if the Busy field and the H field are NULL and the GD field is not the local group code of the node. Similar to the countdown-CD-to-zero-for-transmission rule of DQDB, the DQDB-NSP transmission rule is: countdown (CD) at the head of its outstanding queue (*CD) by one when a node with non-empty (*CD) receives on bus A either a non-Busy slot with H unset, or a Busy slot with H unset and GD content equal to the local group code until the CD reaches zero, then the node may loads its segment addressed by the pointer field of the first position of the (*CD) into an incoming free slot. A node can pre-use a slot when the node find that the destination group of the segment at the heads of its (*CD) or (*Q) is not going to reach the group that is requesting the slot. Note this slot request information is from the head position of the slot queue (*SQ) of the node (refer to the next paragraph).

The slot request status record/erasure rule works as follows. In each node, we insert a slot queue, denoted by (*SQ), that can store every slot field GR on bus B.

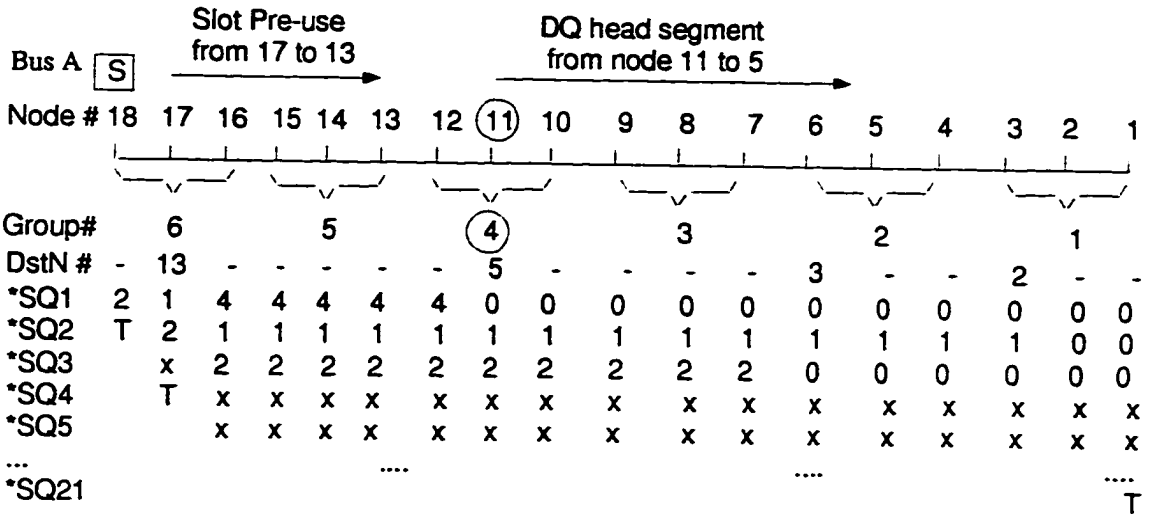
When a node detects a slot on bus B passing by it, the node copies the GR field into the tail position of its (*SQ). When a node detects a slot on bus A passing by it, the node shift the (*SQ) toward the head, i.e., erase a slot record.

The group request rule works as follows. Whereas the request rule of basic DQDB only uses one-bit request field in a slot, the DQDB-NSP rule for request uses i bits as the GR(i) field. The i bits in GR(i) covers ($2i$-1) groups. A node holding the segment has to send a request to the upstream head-end of Bus A through Bus B. The GR field of a slot will be filled with the local group code by a node when the node detect NULL value in the GR field of the slot. The RQ counter increases by one when a node receives on bus B a slot with non-NULL value in the GR field. The RQ counter decreases by one when a node with RQ>0 and empty (*CD) receives on bus A either a non-Busy slot with H unset, or a Busy slot with H unset and GD content equal to the local group code.

Unlike *SP_DQDB* protocol, the DQDB-NSP protocol provides more possible slot for pre-use and remote traffic service under uniform load. The throughput gain is determined by the number of groups designed. Intuitively, more groups means more chances for transmissions. For instance for a two-group scheme, a slot traversing the two different groups can carry two segments totally at most when both segments are destined to their own groups. Of course, it can only carry one segment destined to the other group. Throughput gain is improved by one fourth or so and at least one bit is needed as group address code. For a seven groups scheme, throughput gain by 0.8 times or so and at least three bits as group address code, ... and so on. In this way. the DQDB-NSP scheme improves throughput gain using a simpler and flexible mechanism. However, there is a limit on the throughput gain achievable by the number of groups.

Fig.(44) illustrates how the DQDB-NSP protocol makes slot record/erasure and segment transmissions for *Slot Pre-use* by an example of a 18–node DQDB system in 6 groups 3 nodes each group at the moment when Bus B finishes emitting slot 20. As you see in Fig.(44), #1 node records 20 slot request status exactly, and #2 node 19, #3 node 18, and so on until #16 node 5. Whereas #17 node and #18 node have 3 not 4 records and 1 not 3 records respectively since #17 node and #18 node have shifted out one slot and two slot respectively already. We suppose that DQ head at the moment is node 11, the next in the DQ is node 2, and the third is node 6. Row *SQ1 tells us that nodes from 1 to 11 detected that slot 1 was not carrying a request, node 11 made a request for slot 1, but nodes from 12 to 16 detected that a node in group 4 was requesting slot 1. #17 and #18 nodes both do not keep the GR of 4 in their (*SQ) even though they both detected the information because it has been shifted out by the corresponding slots on bus A. Row *SQ2 tells us that nodes from 1 to 2 detected that slot 2 was not carrying a request, node 2 made a request for slot 2, but nodes from 4 to 17 detected that a node in group 1 was requesting slot 2. #18 nodes does not keep the GR of 1 in its (*SQ) even though it detected the information because it has been shifted out by the corresponding slots on bus A. Row *SQ3 tells us that nodes from 1 to 6 detected that slot 3 was not carrying a request, node 6 made a request for slot 3, but nodes from 7 to 18 detected that a node in group 2 was requesting slot 3. Row *SQ4 tells us that nodes from 1 to 17 detected slot 4 but node 18 not. We do not need the information in the GR after slot 3 for our illustration. Now node 17 has had a chance to pre-use slot 1 since node 17 1) had a slot request status record at the head of its (*SQ) indicating that group 4 was requesting the slot; 2) had a segment whose destination group was not going to reach group 4; and 3) had received the slot on bus A.

Figure 44 The slot record/erasure of a 18-node 6-group DQDB-NSP system



Note: 1) *SQ# means the positions in the queue *SQ;
2) T means a tail of the queue;
3) x in row *SQ# means a value in the GR of the correponding slot we don't care;
4) 4 in row *SQ# is got from the GR field of the corresponding slot.

It is clear that slot S in this situation is pre-used once and conveys two segments. One segment was transmitted from node 17 in group 6 to node 13 in group 5 by slot pre-use. The second segment was transmitted from node 11 in group 4 to node 5 in the group 2. In the above example, a slot may convey only one segment if all the nodes in all the upstream groups have no requesting segment or only have requesting segments either for the head group or for a downstream group from the head group. There exists a certain probability for one-segment transmission in the system. A slot may convey either two segments, or more up to maximum six segments with their own probability. For an instant, when the slot meets such a case that the node the slot meets first in each group sends its segment to its own group, a slot may convey at most six segments. But what is the average number of segments a slot may convey in the system? We address this question in the following subsection.

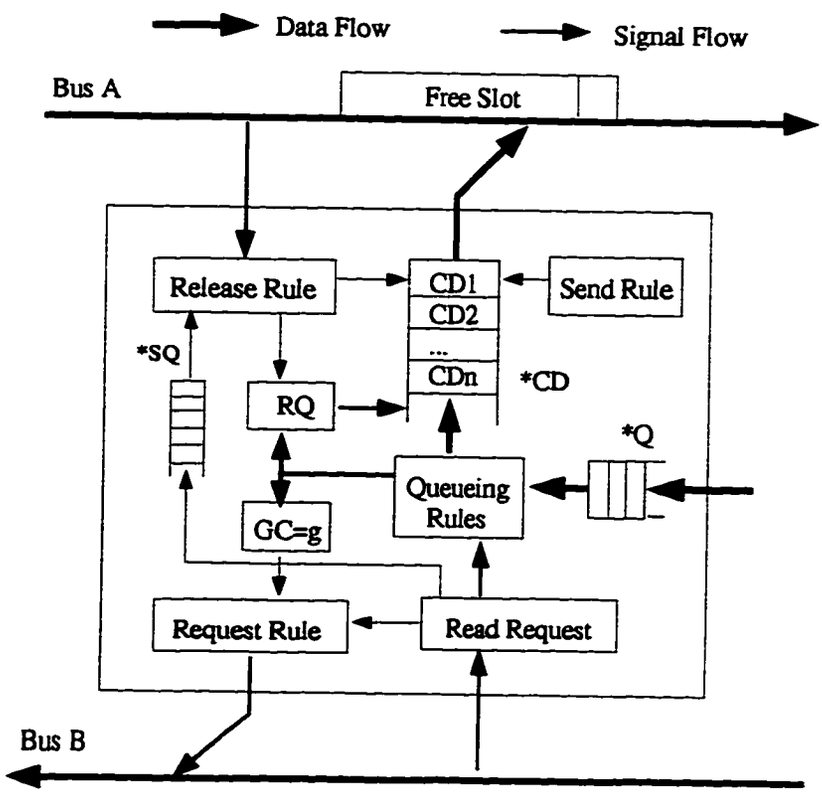
3.5.2 Specifications

The ACF, node structure, and operations of a DQDB-NSP protocol are specified in the following, respectively.

3.5.2.1 Employing the ACF of a DQDB Slot Refer to Section 3.3.2

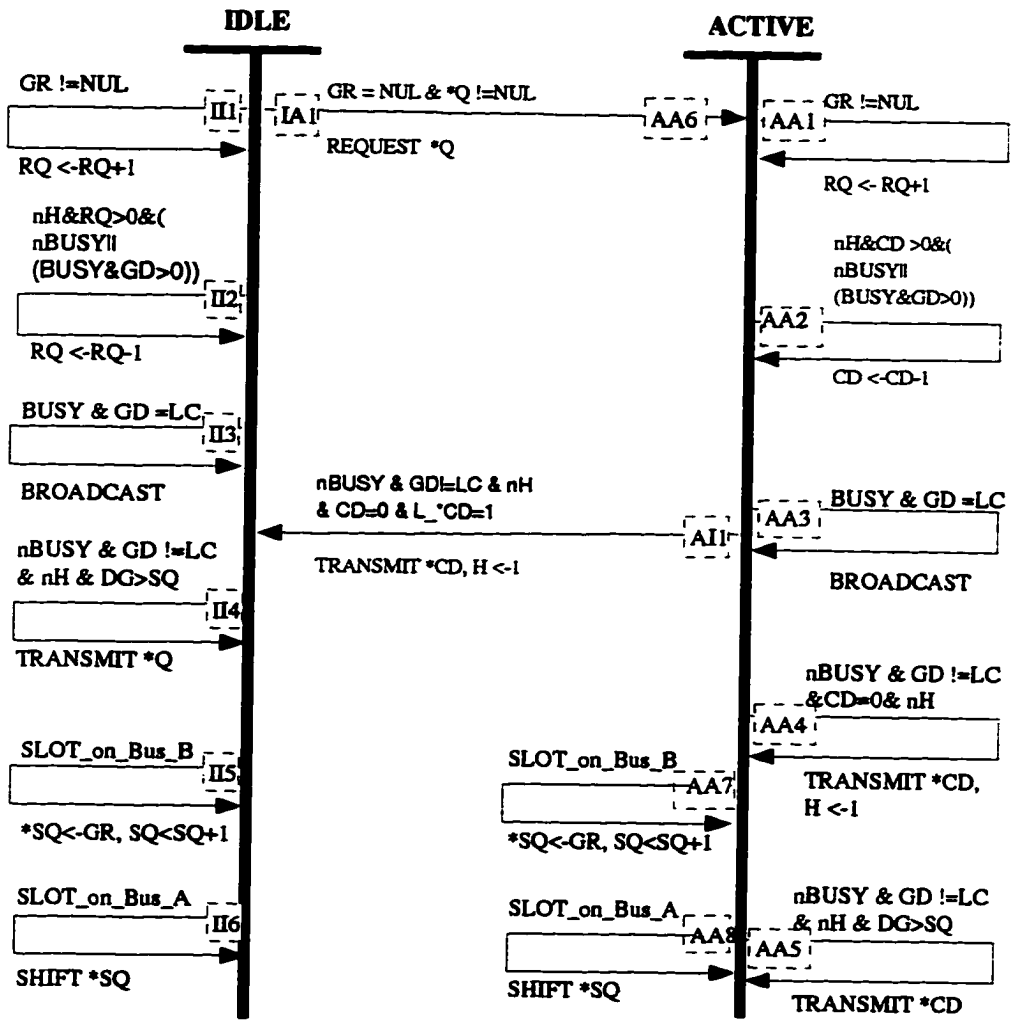
3.5.2.2 Node Structure of DQDB-NSP Fig.(45) shows the configuration of a typical node. A waiting queue (*Q) used to keep segments from the local host and a request rule machine with a request counter *RQ* follow the similar functionality as those of standard DQDB. However, we add the following devices in each node. One register (GC) is in order that the node keeps its own group code. An outstanding queue (*CD) is used to keep segments that have made a request for transmission. Each element of the (*CD) consists of two parts: a countdown counter (*CD*) and an

Figure 45 Configuration of A DQDB-NSP Node



address-pointer (see Fig.(13.a)) that points to a segment in the waiting queue (*Q) (see Fig.(13.b)). A queue (*SQ) is for storing slot request status. One release rule machine decides when the node may release a busy slot. One transmission rule ma-

Figure 46 DQDB-NSP Machine



chine decides when the node may send its segment in the outstanding queue (*CD) or in the waiting queue (*Q).

3.5.2.3 Operations of The DQDB-NSP The following is a specification of operations of the DQDB-NSP protocol. As mentioned previously, each node keeps its own group code (LC). Fig.(46) demonstrates operations of the machine. As usual,

vertical bold lines are for states of machine, the names of states can be found at top of the lines; arrow-head lines are for transitions of states; letters above these lines are for events that enable the transition; letters below these lines are for actions that execute when the events above the transition line occur. Tables (15 and 16) provide the explanation of the events and the actions in Fig.(46).

All the machines have two states: IDLE state under which there is no segment in their (*CD) waiting for channel, ACTIVE state under which there is at least one segment in their (*CD) waiting for channel.

Machine in IDLE state takes loop-transition Π1 that raises RQ by one when the machine receives a non-null request nNUL in field GR in a coming slot on Bus B.

Machine in IDLE state takes loop-transition Π2 that reduces RQ by one when the machine with its RQ greater than zero receives either a nBUSY slot with NULL value in field H of the slot on Bus A or a BUSY slot with a non-null group code in field GD and NULL value in field H on Bus A.

Machine in IDLE state takes loop-transition Π3 that broadcasts a coming slot on Bus A by marking nBUSY in field Busy of the slot when the machine receives a BUSY slot on Bus A with a local group code LC in field GD of the slot.

Machine in IDLE state takes loop-transition Π4 that transmits a segment in the head of (*Q), shifts the (*Q) toward the head by one position, marks BUSY in field Busy of a slot on Bus A when the machine with the first DG in its (*Q) greater than the first slot request record SQ in its (*SQ) receives the nBUSY slot with non-local-group-code in field GD of and NULL value in field H of the slot.

Machine in IDLE state takes loop-transition Π5 that copies field GR of the incoming slot into the tail of its (*SQ) when the machine detects a slot on bus B.

Machine in IDLE state takes loop-transition II6 that shifts out the head GR of its (*SQ) when the machine detects a slot on bus A.

Machine in ACTIVE state takes loop-transition AA1 that raises RQ by one when the machine receives a non-null request nNUL in field GR of a coming slot on Bus B.

Machine in ACTIVE state takes loop-transition AA2 that reduces CD in the head of its (*CD) by one when the machine with its first CD value in its (*CD) greater than zero receives either a nBUSY slot with NULL value in field H of the slot on Bus A or a BUSY slot with a non-null group code in field GD and NULL value in field H on Bus A.

Machine in ACTIVE state takes loop-transition AA3 that broadcasts a coming slot on Bus A by marking nBUSY in field Busy of the slot when the machine receives a BUSY slot carrying a local group code LC in field GD of the slot.

Machine in ACTIVE state takes loop-transition AA4 that transmits a segment in the head of its (*CD), shifts the (*CD) queue toward the head by one position, and marks BUSY in busy field of and sets field H of the slot on Bus A when the machine with zero value in the corresponding CD field of its (*CD) receives the nBUSY slot with non local group code in field GD of and NULL value in field H of the slot.

Machine in ACTIVE state takes loop-transition AA5 that transmits a segment in the head of (*CD), shifts the (*CD) toward the head by one position, marks BUSY in field Busy of a slot on Bus A when the machine with the first DG in its (*CD) greater than the first slot request record SQ in its (*SQ) receives the nBUSY slot with non-local-group-code in field GD of and NULL value in field H of the slot. If its (*Q) is non-empty, it transfers one segment from the head of its (*Q) to the tail of its (*CD).

Machine of group three in ACTIVE state takes loop-transition AA6 that does the

same as IA1 under the same events as IA1. For brief, the transition is only marked in name but not in a line.

Machine in ACTIVE state takes loop-transition AA7 that copies field GR of the incoming slot into the tail of its (*SQ) when the machine detects a slot on bus B.

Machine in ACTIVE state takes loop-transition AA8 that shifts out the head GR of its (*SQ) when the machine detects a slot on bus A.

Machine in IDLE state takes transition IA1 that changes the state from IDLE to ACTIVE by putting the address of a segment of the head of its (*Q) and its request counter RQ value into part Pointer and part CD of the tail of its (*CD) respectively and marking its local group code LC in field GR of a slot on Bus B when the machine with non-empty queue (*Q) !=NUL receives a null group request code NUL in field GR of a slot.

Machine in ACTIVE state takes transition AI1 that transmits a segment in the head of its (*CD), shifts the (*CD) toward the head by one position and marks BUSY in field Busy of a slot, sets field H of the slot and changes its state from ACTIVE to IDLE when the machine with zero value in the first CD field that is the last one left in the (*CD), (i.e.,L_*CD=1), receives the nBUSY slot with non-local-group-code in field GD of and NULL value in field H of the slot.

3.5.3 Simulation Results

3.5.3.1 Simulation Parameters The simulation collects data from 60000 full life-time slots. These data measure throughput gain, media access delay, utilization and segment loss ratio of the whole system under Poisson traffics with average arrival rate of 1.0,1.3,1.7,2.0,3.0,6.0 segments per slot-time. A comparison among various

grouping method was performed (i.e.: performance measures for one-group system up to fifteen groups system.).

3.5.3.2 Performance Results In Fig.(47) we show simulation results of the segment loss ratio. From the figure, we see that the loss ratios decrease as the number of groups increases. This is due to the fact that as the number of groups increases, throughput increases so that the probability that the system buffers are full decreases.

Fig.(48) shows simulation results of the average media access delay. From the figure, it is clear that the delay decreases as the number of groups increases. The reason that causes this happen is that more groups in the system directly provides more chances to every segment waiting in the system to be carried by a slot so that the waiting time for a segment to stay in the system with more groups becomes shorter than the system with less groups.

Figure 47 Segment Loss Ratio vs Number of Grouping of DQDB-NSP under The Offered Segment-Arrival Rates

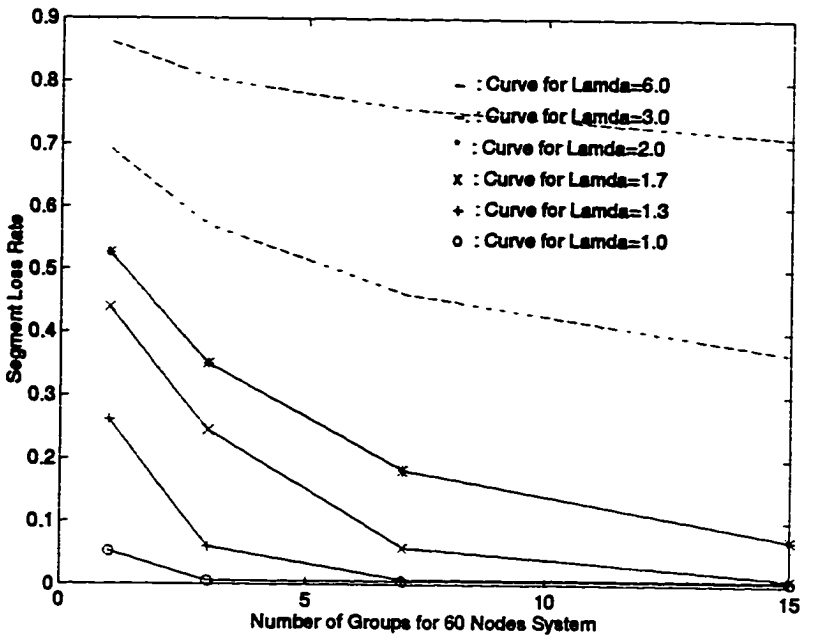


Figure 48 Access Delay vs Number of Grouping of DQDB-NSP under The Offered Segment-Arrival Rates

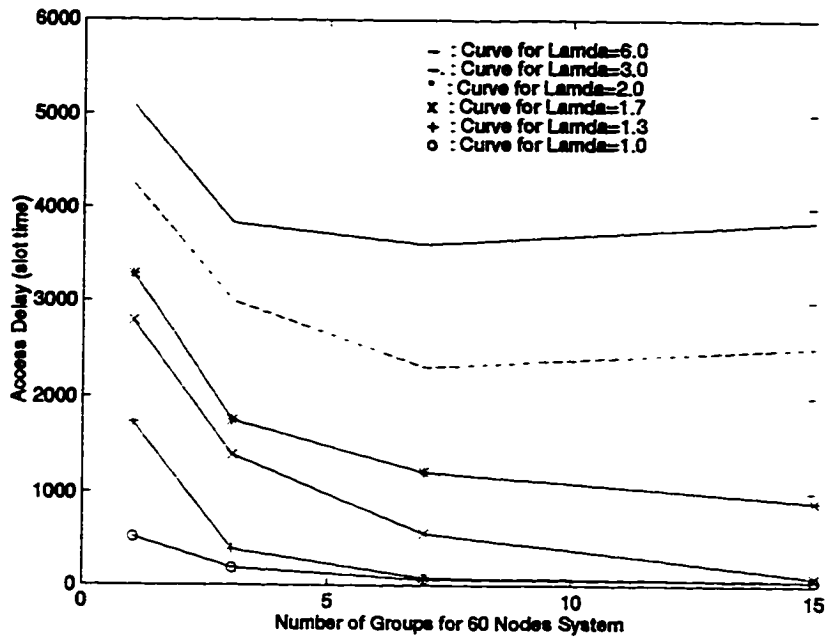


Figure 49 Slot-time Utilization vs Number of Grouping of DQDB-NSP under The Offered Segment-Arrival Rates

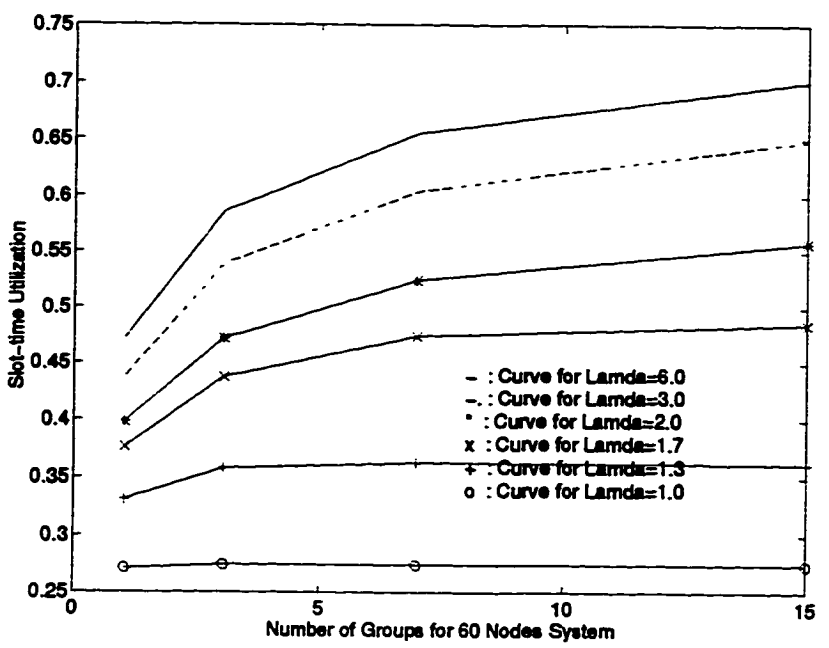


Figure 50 Throughput Gain vs Number of Groups of DQDB-NSP under The Offered Message-Arrival Rates

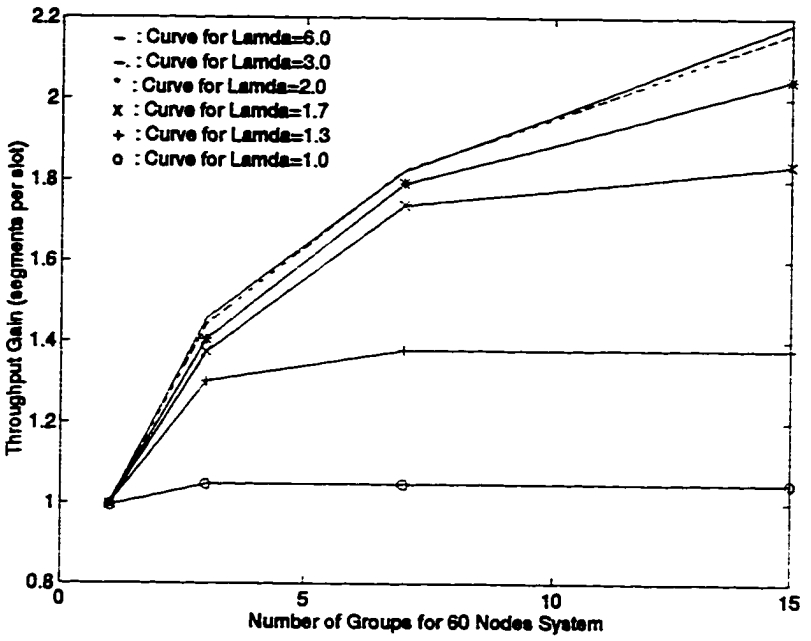


Fig.(49) shows simulation results of the slot-time utilization. From the Figures, it is clear that the utilization increases as the number of groups increases. When grouping number of a DQDB-NSP system is 1, the utilization is 0.26.

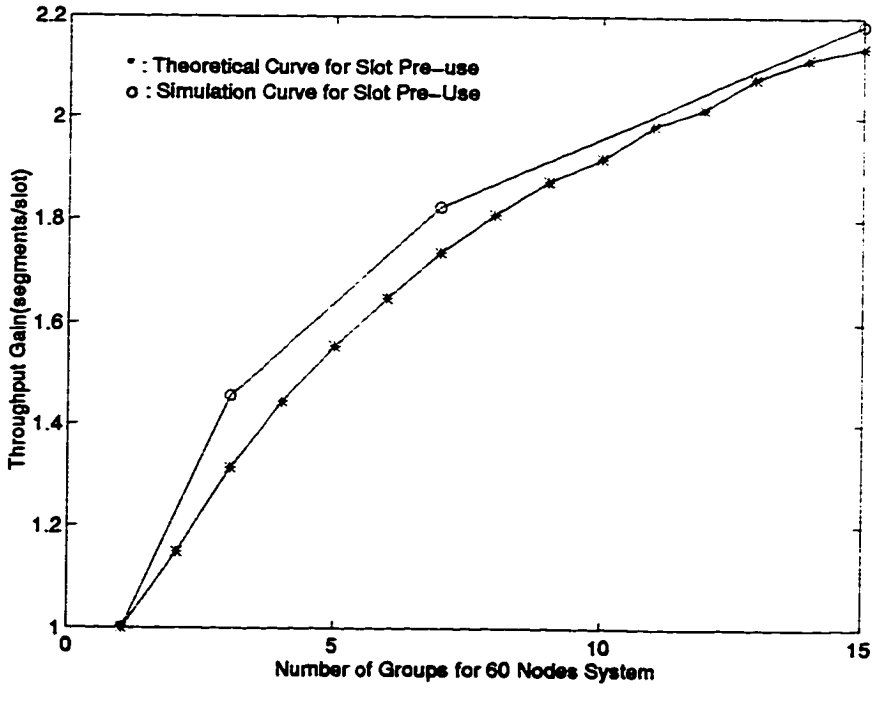
Fig.(50) shows simulation results of the throughput gain. From the Figure, we see that the throughput gains increase with the increment of number of groups on average under Poisson traffic with average arrival rate of 1.0,1.3,1.7,2.0,3.0,6.0 segments per slot-time.

Fig.(51) shows two throughput gain curves: one from theorem, the other from simulation, both of which are for the same grouping network as the described above. We see that the theoretical curve is very close to the simulative curve. In addition, theoretical curve in the figure is obtained by calculating *Theorem 2*.

3.5.4 Remarks

DQDB-NSP protocol is a new protocol for high throughput and little access delay at MAN-MAC layer. The main features of this protocol are: 1) the protocol divides all the nodes in the system into multiple groups and assigns a group code to each group, 2) a node of the protocol may put its group code in request field GR of a slot when the node get request-chance, 3) a node of the protocol has a queue to record all slot request GR, 4) a slot can be pre-used before the slot reaches the group that is first requesting for it. In this way, chance of transmission is increased. The

Figure 51 Comparisons of Results of Simulation & Theory for DQDB-NSP



throughput gain of a DQDB-NSP system increases as the number of groups increases. Of course, too large number of groups being used makes the system complex so that the capacity of the system goes down. Unlike SP-DQDB, DQDB-NSP handles remote traffic under uniform load. Hence, DQDB-NSP is generally significant in enhancing network throughput.

3.6 DQDB-H Protocol for Slot Pre-Use&Reuse

I am going to describe the verview and specifications of the protocol in the next two subsections, and then exhibit simulation results.

3.6.1 Overview of DQDB-H Protocol

The proposed DQDB-H protocol both for *Slot Pre-use* and for *Slot Reuse* provides for higher throughput via five folds: 1) the grouping of all the nodes in the system; 2) slot post-destination group releasing rule; 3) slot multi-transmission rule; 4) group request rule; and 5) slot request status record/erasure rule.

Refer to Chapter 2 for the principle of grouping.

As we shall illustrate later, the grouping concept has a significant impact on the delivered throughput. The higher throughput is achieved because nodes situated both upstream and downstream from a DQ head group will have chances of both pre-using free slots before a DQ head node uses it and reusing free slots after a segment sent by a DQ head node departs from its destination group. In this way, the DQDB-H both obtains high throughput and maintains the order of original distributed queue.

It is worthwhile mentioning that the distributivity of the DQDB-H protocol can be guaranteed by an adequate management protocol for the DQDB-H protocol so that in case of the occurrence of one fault on the DQDB-H, the DQDB-H management protocol is capable of re-grouping the nodes and activating proper group code for each node in its group. As a result, DQDB-H can be also one-fault tolerance.

Three new fields are defined in ACF of a slot format. They are GROUP_DESTINATION, denoted by GD, which carries destination group code, GROUP_REQUEST, denoted by GR, which carries requesting group code, and DQ_HEAD, denoted by H, which indicates if the slot has been used by a DQ-head

node. The sizes of fields GD and GR are determined by the number of grouping. The DQ_HEAD is one-bit-wise.

The slot post-destination group release rule works as follows. Within each group, each free slot can only carry a segment. A slot carrying a segment will become free when the slot departs from its destination group. A node resets the Busy-status field when the local group code of the node matches the contents of the field GD of a coming slot so that the slot is in non-busy state. In this way, a busy slot becomes free for post-destination groups by lower index nodes that are situated downstream from the destination group.

The group transmission rule works as follows: The determination of a free slot is done by three fields of a slot rather than only one field: field Busy and field DESTINATION_GROUP denoted by GD(i), and field H. The i in GD(i) means that i-bit Group Destination field covering ($2i$-1) groups. The H Field should be set only by DQ head node to indicate if a slot can be free for reuse and . Nodes may reuse a slot only if field H of the slot is set. A slot is free only if the Busy field is NULL and the GD field is not the local group code of the node. Similar to the countdown-CD-to-zero-for-transmission rule of DQDB protocol, the DQDB-H transmission rule is: countdown (CD) at the head of its (*CD) by one when a node with non-empty (*CD) receives on bus A either a non-Busy slot with H unset, or a Busy slot with H unset and GD content equal to the local group code until the CD reaches zero, then the node may loads its segment from the first position of the (*CD) into a coming free slot. A node can pre-use a slot when the node find that the destination group of the segment at the heads of its (*CD) or (*Q) is not going to reach the group that is requesting the slot. Note this slot request information is from the head position of the slot queue (*SQ) of the node (refer to the next paragraph). If a slot is free for

slot reuse, the machine will check if the (*CD) is ready to transmit first. If the (*CD) is non-empty, a transmission may be made without reference to any CD value from (*CD). If (*CD) is not empty, the machine will check if its waiting queue (*Q) is empty. If no, a transmission may be made from (*Q).

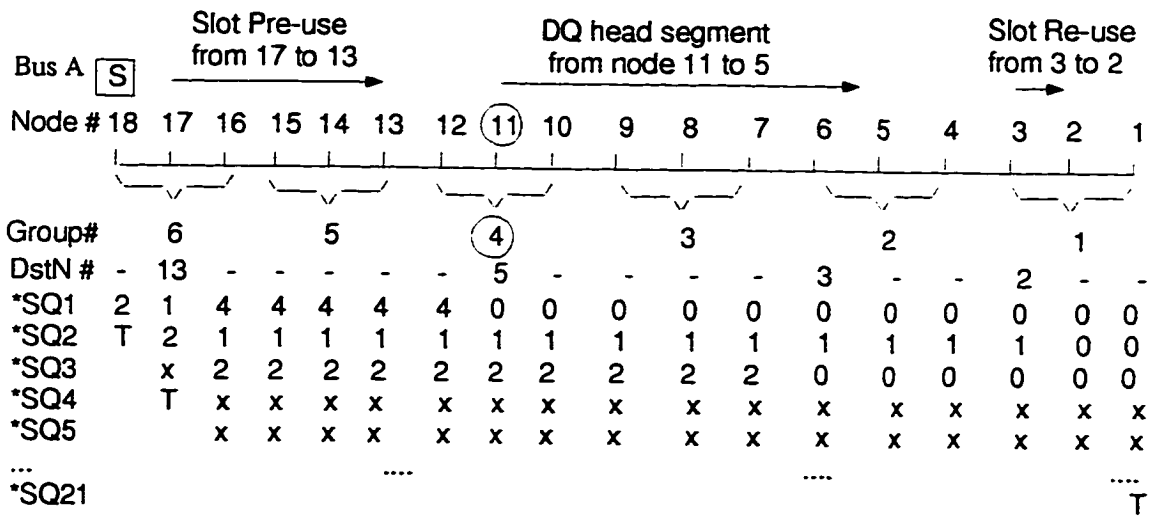
The slot request status record/erasure rule works as follows. In each node, we insert a slot queue, denoted by (*SQ), that can store every slot field GR on bus B. When a node detects a slot on bus B passing by it, the node copies the GR field into the tail position of its (*SQ). When a node detects a slot on bus A passing by it, the node shift the (*SQ) toward the head, i.e., erase a slot record.

The group request rule works as follows. Whereas the request rule of basic DQDB only uses one-bit request field in a slot, the DQDB-H rule for request uses i bits as the GR(i) field. The i bits in GR(i) covers ($2i$-1) groups. A node holding the segment has to send a request to the upstream head-end of Bus A through Bus B. The GR field of a slot will be filled with the local group code by a node when the node detect NULL value in the GR field of the slot. The RQ counter increases by one when a node receives a slot with non-NULL value in the GR field on bus B. The RQ counter decreases by one when a node with RQ>0 and empty (*CD) receives on bus A either a non-Busy slot with H unset, or a Busy slot with H unset and GD content equal to the local group code.

Like none of SP_DQDB, *Destination Release*, *Erasure Nodes* and *PSI*, the DQDB-H protocol provides both *Slot Pre-use* and *Slot Reuse*, which makes full use of slot pre-idle time and post-idle time. The throughput gain is determined by the number of groups. Intuitively, more groups means more chances for transmissions. For instance of a two-group scheme, a slot traversing the two different groups can carry two segments totally at most when both segments are destined to their own group. Of

course, it can only carry one segment destined to the other group. Throughput gain is improved by one third on average by roughly estimation and at least one bit is needed as group address code. For a seven group scheme, throughput gain by 2.0 times or so and at least three bits as group address code, ... and so on. In this way. the DQDB-H scheme improves throughput gain. As a cost of this, the DQDB-H protocol, however, is a little bit more complex than others.

Figure 52 The slot record/erasure and transmission of a 18-node 6-group DQDB-H system



Note: 1) *SQ# means the positions in the queue *SQ;
2) T means a tail of the queue;
3) x in row *SQ# means a value in the GR of the correponding slot we don't care;
4) 4 in row *SQ# is got from the GR field of the corresponding slot.

Fig.(52) illustrates how the DQDB-H protocol makes slot record/erasure and segment transmissions for *Slot Pre-use* by an example of a 18-node DQDB system in 6 groups 3 nodes each group at the moment when Bus B finishes emitting slot 20. As you see in Fig.(52), #1 node records 20 slot request status exactly, and #2 node 19, #3 node 18, and so on until #16 node 5. Whereas #17 node and #18 node have 3 not 4 records and 1 not 3 records respectively since #17 node and #18 node have shifted out one slot and two slot respectively already. We suppose that DQ head at the

moment is node 11, the next in the DQ is node 2, and the third is node 6. Row *SQ1 tells us that nodes from 1 to 11 detected that slot 1 was not carrying a request, node 11 made a request for slot 1, but nodes from 12 to 16 detected that a node in group 4 was requesting slot 1. #17 and #18 nodes both do not keep the GR of 4 in their (*SQ) even though they both detected the information because it has been shifted out by the corresponding slots on bus A. Row *SQ2 tells us that nodes from 1 to 2 detected that slot 2 was not carrying a request, node 2 made a request for slot 2, but nodes from 4 to 17 detected that a node in group 1 was requesting slot 2. #18 nodes does not keep the GR of 1 in its (*SQ) even though it detected the information because it has been shifted out by the corresponding slots on bus A. Row *SQ3 tells us that nodes from 1 to 6 detected that slot 3 was not carrying a request, node 6 made a request for slot 3, but nodes from 7 to 18 detected that a node in group 2 was requesting slot 3. Row *SQ4 tells us that nodes from 1 to 17 detected slot 4 but node 18 not. We do not need the information in the GR after slot 3 for our illustration. Now node 17 has had a chance to pre-use slot 1 since node 17 either had a slot request status record at the head of its (*SQ) indicating that group 4 was requesting the slot or had a segment whose destination group was not going to reach group 4; and 3) had received the slot on bus A. And Node 3 will have a chance to reuse the slot since the slot is going to be marked in the H field by node 11, DQ head node, and to depart its destination group, group 2.

It is clear that slot S in this situation is pre-used and reuse once respectively and conveys three segments. One segment was transmitted from node 17 in group 6 to node 13 in group 5 by slot pre-use. The second segment was transmitted from node 11 in group 4 to node 5 in the group 2, which is DQ head node transmission. The other segment was transmitted from node 3 in group 1 to node 2 in the same group

by slot reuse. In the above example, a slot may convey only one segment if all the nodes in all the upstream and downstream groups have no requesting segment or all the nodes in all the upstream groups only have requesting segments either for the DQ head group or for a downstream group from the DQ head group and the DQ head node holds the segment for mostdownstream group. There exists a certain probability for one-segment transmission in the system. A slot may convey either two segments, or more up to maximum six segments with their own probability. For an instant, when the slot meets such a case that the node the slot meets first in each group sends its segment to its own group, a slot may convey at most six segments.

3.6.2 Specifications

The ACF, node structure, and operations of a DQDB-H protocol are specified in the following, respectively.

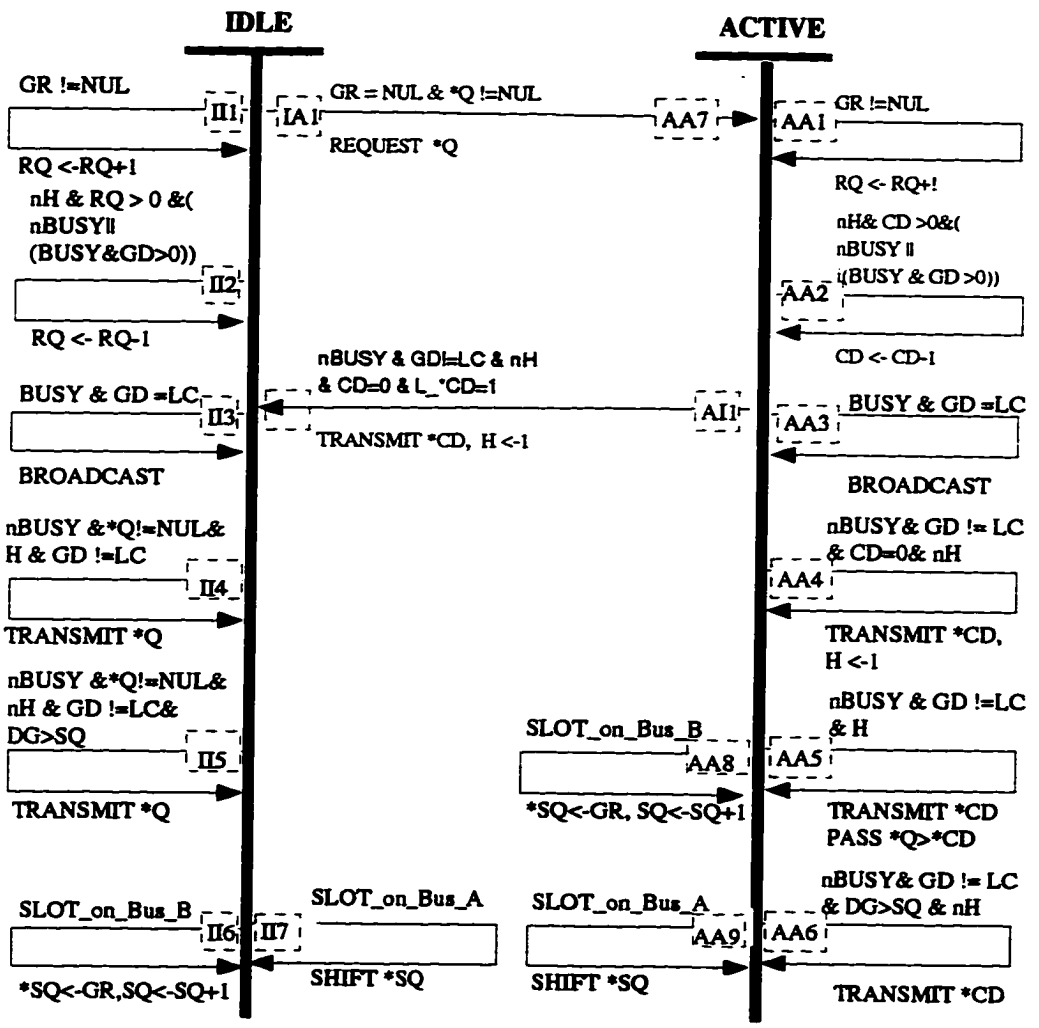
3.6.2.1 Employing the ACF of a DQDB Slot Refer to Section 3.5.2.

3.6.2.2 Node Structure of The DQDB-H Refer to Section 3.5.2.

3.6.2.3 Operations of The DQDB-H The following is a specification of operations of the DQDB-H protocol. As mentioned previously, each node keeps its own group code. Fig.(53) demonstrates operations of the machine. As usual, vertical bold lines are for states of machine, the names of states can be found at top of the lines; lines with arrows are for transitions of states; letters above these lines are for events that enable the transition; letters below these lines are for actions that execute when the events above the transition line occur. Tables (15 and 16) provide the explanation of the events and the actions in Fig.(53).

All the machines have two states: IDLE state under which there is no segment in their (*CD) waiting for channel, ACTIVE state under which there is at least one segment in their (*CD) waiting for channel.

Figure 53 DQDB-H Machine



Machine in IDLE state takes loop-transition II1 that raises RQ by one when the machine receives a non-null request nNUL in field GR in a coming slot on Bus B.

Machine in IDLE state takes loop-transition II2 that reduces RQ by one when the machine with its RQ greater than zero receives either a nBUSY slot with NULL value

in field H of the slot on Bus A or a BUSY slot with a non-null group code in field GD and NULL value in field H on Bus A.

Machine in IDLE state takes loop-transition II3 that broadcasts a coming slot on Bus A by marking nBUSY in field Busy of the slot when the machine receives a BUSY slot on Bus A with a local group code LC in field GD of the slot.

Machine in IDLE state takes loop-transition II4 that transmits a segment at the head of its (*Q), shifts the (*Q) toward the head by one position, marks BUSY in field Busy of a slot on Bus A when the machine with non-empty (*Q) receives the nBUSY slot with non-local-group-code in field GD and SET value in field H of the slot.

Machine in IDLE state takes loop-transition II5 that transmits a segment in the head of (*Q), shifts the (*Q) toward the head by one position, marks BUSY in Busy field of a slot on Bus A when the machine with the first DG in its (*Q) greater than the first slot request record SQ in its (*SQ) receives the nBUSY slot with non-local-group-code in field GD of and NULL value in field H of the slot.

Machine in IDLE state takes loop-transition II6 that copies field GR of the incoming slot into the tail of its (*SQ) when the machine detects a slot on bus B.

Machine in IDLE state takes loop-transition II7 that shifts out the head GR of its (*SQ) when the machine detects a slot on bus A.

Machine in ACTIVE state takes loop-transition AA1 that raises RQ by one when the machine receives a non-null request nNUL in field GR of a coming slot on Bus B.

Machine in ACTIVE state takes loop-transition AA2 that reduces CD at the head of its (*CD) by one when the machine with its first CD value in its (*CD) greater than zero receives either a nBUSY slot with NULL value in field H of the slot on Bus A or a BUSY slot with a non-null group code in field GD and NULL value in

field H on Bus A.

Machine in ACTIVE state takes loop-transition AA3 that broadcasts a coming slot on Bus A by marking nBUSY in field Busy of the slot when the machine receives a BUSY slot carrying a local group code LC in field GD of the slot.

Machine in ACTIVE state takes loop-transition AA4 that transmits a segment at the head of its (*CD), shifts the (*CD) toward the head by one position, sets field H and marks BUSY in field Busy of a slot on Bus A when the machine with zero value in the corresponding CD field of its (*CD) receives the nBUSY slot with non-local-group-code in field GD of and NULL value in field H of the slot.

Machine in ACTIVE state takes loop-transition AA5 that transmits a segment at the head of its (*CD), shifts the (*CD) queue toward the head by one position, marks BUSY in Busy field of a slot on Bus A, and transfers the segment at the head of (*Q) to the tail of (*CD) if (*Q) is non-empty when the machine receives the nBUSY slot with non-local-group-code in field GD of and SET value in field H of the slot.

Machine in ACTIVE state takes loop-transition AA6 that transmits a segment in the head of (*CD), shifts the (*CD) toward the head by one position, marks BUSY in Busy field of a slot on Bus A when the machine with the first DG in its (*CD) greater than the first slot request record SQ in its (*SQ) receives the nBUSY slot with non-local-group-code in field GD of and NULL value in field H of the slot. If its (*Q) is non-empty, it transfers one segment from the head of its (*Q) to the tail of its (*CD).

Machine of group three in ACTIVE state takes loop-transition AA7 that does the same as IA1 under the same events as IA1. For brief, the transition is only marked in name but not in a line.

Machine in ACTIVE state takes loop-transition AA8 that copies field GR of the

incoming slot into the tail of its (*SQ) when the machine detects a slot on bus B.

Machine in ACTIVE state takes loop-transition AA9 that shifts out the head GR of its (*SQ) when the machine detects a slot on bus A.

Machine in IDLE state takes transition IA1 that changes the state from IDLE to ACTIVE by putting the address of a segment at the top of its (*Q) and its request counter value RQ into part Pointer and copying and part CD of the tail of its (*CD) respectively and marking its local group code LC in field GR of a slot on Bus B when the machine with non-empty (*Q)!=NUL receives a null group request code NUL in field GR of a slot.

Machine in ACTIVE state takes transition AI1 that transmits a segment at the head of its (*CD), shifts the (*CD) toward the head by one position and marks BUSY in Busy field of a slot, sets field H of the slot and changes its state from ACTIVE to IDLE when the machine with zero value in the first CD field that is the last one left in the (*CD) (i.e.,L_*CD=1) receives the nBUSY slot with non-local-group-code in field GD of and NULL value in field H of the slot.

3.6.3 Simulation Results

3.6.3.1 Simulation Parameters The simulation collects data from 60000 full life-time slots. These data measure throughput gain, media access delay, utilization and segment loss ratio of the whole system under Poisson traffics with average arrival rate of 1.0, 1.5, 2.0, 3.0, 6.0, 16.0 segments per slot-time. A comparison among various grouping method was performed (i.e.: performance measures for one-group system up to fifteen groups system.).

3.6.3.2 Performance Results Fig.(54) shows simulative results of the segment loss ratio. From the Fig., we see that the loss ratios decrease with the increment of

number of groups on average under Poisson traffics. This is due to the fact that as the number of groups increases, throughput increases so that the probability that the system buffers are full decreases.

Fig.(55) shows simulative results of the average media access delay. From the Fig., we see that the delays decrease with the increment of number of groups on average under Poisson traffics. The reason that causes this happen is that more groups in the system directly provides more chances to every segment waiting in the system to be carried by a slot so that the waiting time for a segment to stay in the system with more groups becomes shorter than the system with less groups.

Fig.(56) shows simulative results of the channel utilization. From the Fig., we see that the utilizations increase with the increment of number of groups on average under Poisson traffics. When grouping number of a DQDB-H system is 1, the utilization is 0.26.

Figure 54 Segment Loss Ratio vs Number of Grouping of DQDB-H under The Offered Segment-Arrival Rates

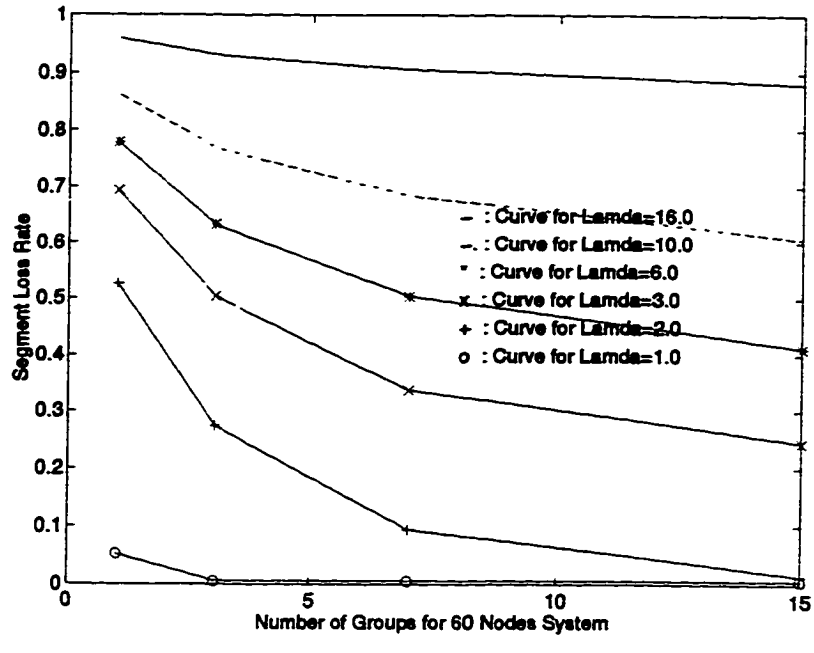


Figure 55 Access Delay vs Number of Grouping of DQDB-H under The Offered Segment-Arrival Rates

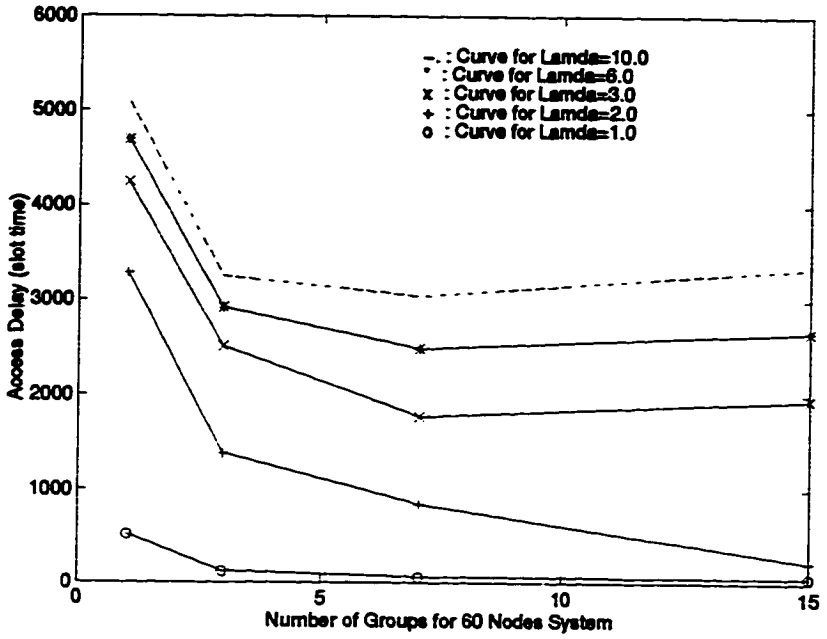


Figure 56 Slot Time Utilization vs Number of Grouping of DQDB-H under The Offered Segment-Arrival Rates

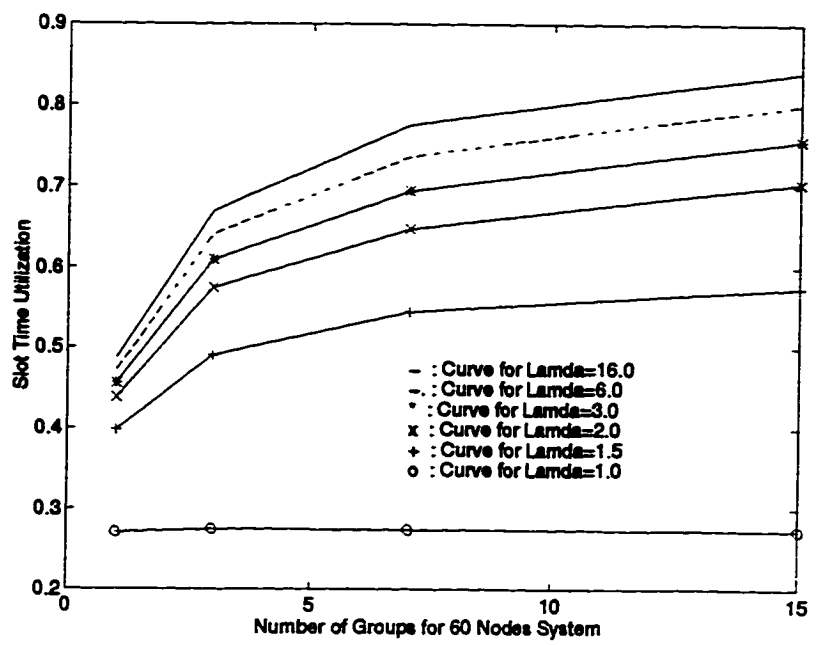


Figure 57 Throughput Gain vs Number of Grouping of DQDB-H under The Offered Message-Arrival Rates

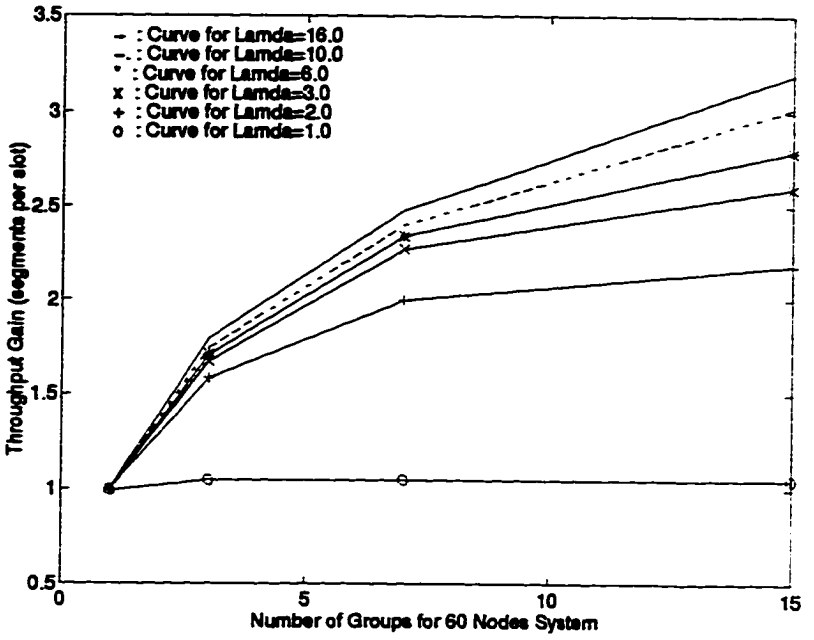


Fig.(57) shows simulative results of the throughput gain. From the Fig., we see that the throughput gains increase with the increment of number of groups under Poisson traffic with average arrival rate λ of 1.0, 2.0, 2.0, 3.0, 6.0, 16.0 segments per slot-time. When grouping number of a DQDB-H system is 1, the system become a DQDB system and the throughput gain is 1.

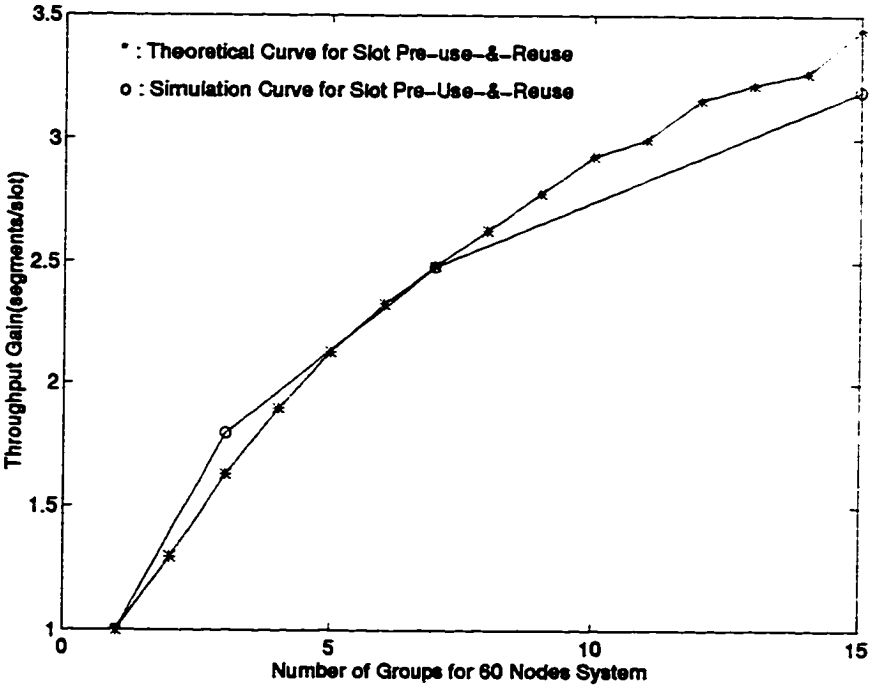
Fig.(58) shows two throughput gain curves: one from our theory, the other from simulation, both of which are for the same grouping network as the described above. We see that theoretical curve basically matches simulative curve. In addition, theoretical curve in the figure is obtained by calculating *Theorem 6.*

3.6.4 Remarks

A new protocol for high throughput and little access delay at MAN-MAC layer, DQDB-H protocol, is presented. The main features of this protocol are: 1) the protocol divides all the nodes in the system into multiple groups and assigns a group code to

each group, 2) a node records all slot request status, 3) a busy slot can be freed after the slot departs its destination group, 4) a busy slot can be reused after the slot carrying a DQ head segment pass through its destination group, and 5) a slot can be pre-used before the slot reaches the group that is first requesting for it. It is a united form for the two kinds of schemes: *Slot Reuse* and *Slot Pre-use* and make full use of slot pre-idle time and post-idle time. The throughput gain of a DQDB-H system reaches what our theory expects. As a cost of this, the DQDB-H is a little bit more complex than the others.

Figure 58 Comparisons of Results of Simulation & Theory for DQDB-H



Chapter 4 New S++-Like Protocols

I am going to introduce the S++ protocol first, then describe the overview and specifications of the protocol in the next two subsections, and finally exhibit simulation results.

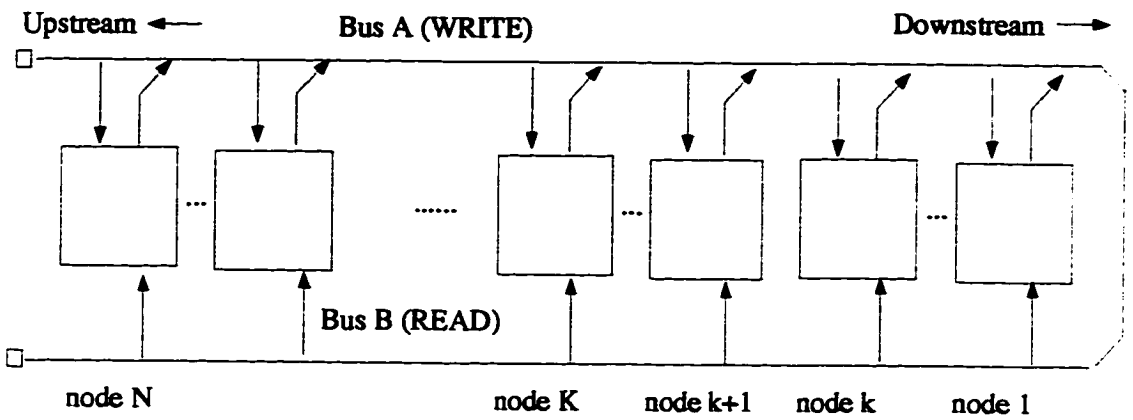
4.1 Introduction to the S++ Protocol

A S++ network consists of either two unidirectional buses with data flow in opposite directions or a single folded bus. If used on a folded bus, the bus is actually folded at one end node (node 1) into two sides: upper side and lower side. The upper side of the bus is denoted by bus A or a WRITE bus and the low side by bus B or a READ bus as shown in Fig.59. A slot generator at the head of the bus emits empty fixed size slots at a constant rate. Each node is connected to both buses. A node may only send data by filling in an empty slot on the WRITE bus and receive data on the READ bus though the node can also read signals from the WRITE bus.

The S++ prevents the nodes close to the head of the WRITE bus from acquiring all empty slots by implementing a transmission limitation scheme that requires one limitation counter P and one period flag F in each node. A node determines its turn (i.e., being the first right node (FRN) for using an empty slot) to transmit segments by recognizing an empty slot when the node is with its $P>0$ or with its $P=0$ and $F=1$. A node having segments ready for transmission may send them and subtract the P until the limitation is reached (i.e., $P=0$) when the node keeps on receiving empty slots on the WRITE bus. After the node reaches the limitation, the node will pass all empty slots to downstream nodes until flag F of the node is set. The F flag will be set to one when the node receives a 1-bit signal, RESTART, in a slot header on the READ

bus. The RESTART signal in a slot was initially set by the head-end node. The node can not find the signal in any slot on the READ bus until the last node finishes its P transmissions. When the node with the F flag set receives an empty slot on the WRITE bus, it sets up limitation (i.e., $P=P_{max}$) for new transmission period and uses the slot.

Figure 59 The Basic S++ Single Fold Bus Network



Now let us take a look at throughput of S++. The nodes in a S++ network empty their P value for sending their segments in the order from upstream to downstream. If a slot is filled, then it will not be released until it finishes running the trip along the whole bus. It means that maximum number of segments a slot can carry during a slot lifetime is one and so the maximum throughput under heavy load is equal to the channal capacity, the slot generating rate.

4.2 S++-GA Protocol for Slot Reuse

4.2.1 Overview of a S++-GA Protocol

The proposed S++-GA protocol with *Slot Reuse* provides for higher throughput via a four-fold approach: 1) the grouping of all the nodes in the system, 2) slot post-destination group releasing rule, 3) group transmission rule, and 4) cyclic bandwidth restart rule.

The S++ network working feature is as follows: once a segment is sent by a node onto the media, it will propagate through the media until it reaches the end of the bus. This feature is thought of as a broadcast among all nodes in the network and the nodes are viewed as a single group within which a slot can only carry one segment. S++-GA generalizes this feature into a multi-group concept: dividing all the nodes on the networks into a limited number of groups and assigning each group a group code or say group address in order along the bus.

Figure 60 The physical configuration of S++-GA Network

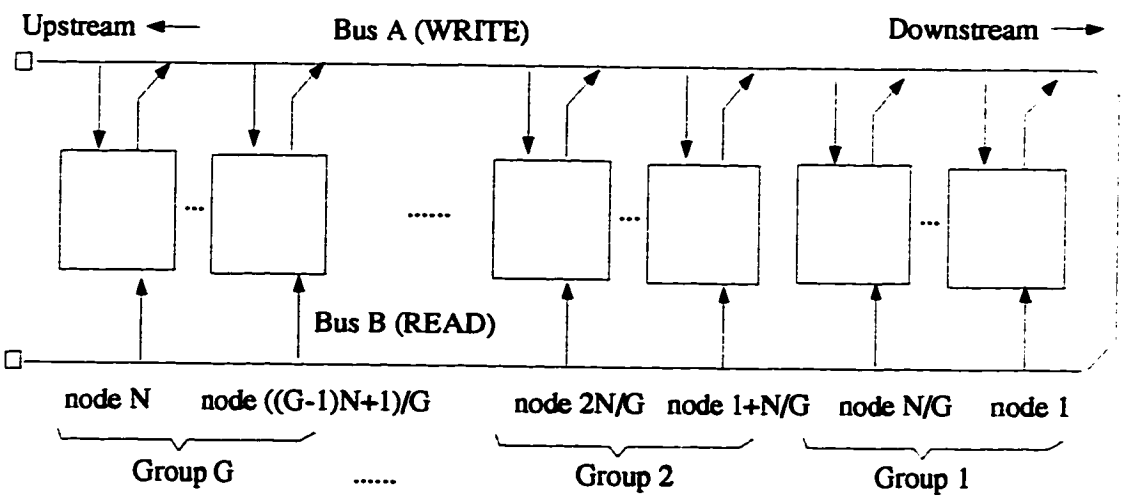
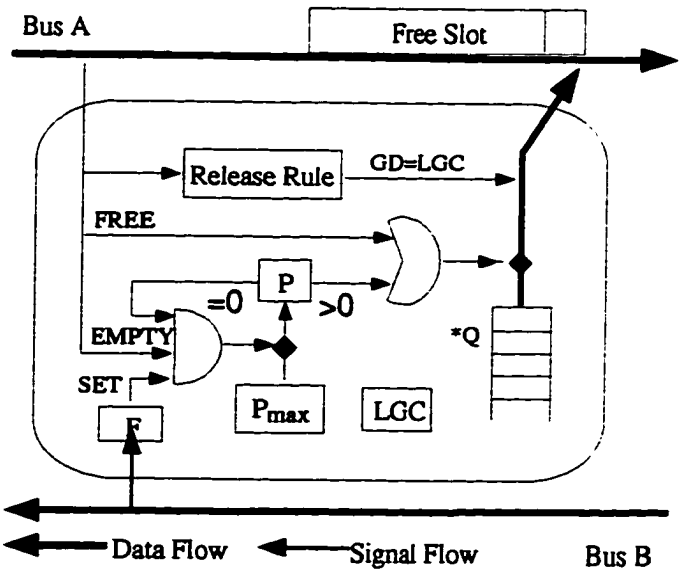


Figure 60 shows the physical configuration of a G-group S++-GA network. All the nodes in the S++-GA network are divided into $G(G = 2^i - 1, (i \in I^+ | I^+ \textit{ are all positive integers.}))$ groups in order, numbered group 1, group 2, ... , group G from the bus-folded-end node to the head-end node. Each group is assigned a group code. In each group there are N/G nodes, where N is the total number of nodes in the network. Note that hereinafter the left, high index position, of the networks is thought of as upstream whereas the right as downstream. The upside bus (Bus A) is a WRITE bus and the downside bus (Bus B) a READ bus.

A new field, destination group (GD), is added in a S++ slot header in order to change S++ into S++-GA. The GD has i bits standing for $(2^i - 1)$ groups, initially is NULL and then loaded with a group code by a node when the node uses the slot. Whether a slot is available is determined by two fields of a slot rather than only one: a traditional Busy status field and the GD field. A slot is EMPTY (or free for first-use) if the both fields are NULLs and free for reuse if the Busy is NULL and the GD carries neither NULL nor the local group code (LGC) with which a node is receiveing the slot. A slot is in the FULL state if its Busy field is BUSY and its GD contains non-NULL and in the REACH state if its Busy field is NULL and its GD contains non-NULL.

Figure 61 Node Structure of S++-GA



Two new devices are added in a S++-GA node with retaining the P counter, the P_{max} register, the local queue (*Q), and the F flag of S++ in each node all with the same functions of theirs respectively as those in S++. One of the two new-added devices is a LGC register to keep local group code. The other is a post-destination group slot release rule that decides when to release a slot. In addition, we make a

minor modification of transmission rule that decides when and how to reuse a released slot. Fig.61 shows the implementation structure of a typical S++-GA node.

The post-destination group slot release rule works as follows: a node in any state sets the Busy field of an in-coming slot to NULL when the node reads a busy slot on the WRITE bus and its own group code in the GD field of the slot. As we expected previously, the slot with NULL in its Busy field and non-NULL in its GD field will become free for reuse when the slot gets into a different group.

The group transmission rule works as follows: a node with $P>0$ and $(*Q)>0$ may fill a segment of its own into a slot on the WRITE bus either with the decrement of its bandwidth counter P if the slot is empty for use or without action on its bandwidth counter P if free for reuse.

The explicit restart bit of S++ is retained for S++-GA almost in the same functions and values, RESTART or NO_RESTART. Let's recall here. Initially, this bit is set to RESTART by the head-end node. If a node with $P>0$ uses the slot, then the node changes the bit to NO_RESTART. A node may restart a new cycle immediately if $P=0$ and $F=1$ when using or reusing the slot, but does not change the bit to NO_RESTART. When a RESTART value is read on the READ bus, then all nodes start a new cycle by setting F to 1.

The state machine associated with S++-GA is shown in Fig.62. As usual, vertical bold lines are for states of machine, the names of states can be found at top of the lines; lines with arrows are for transitions of states; text above these arrow-head lines are events that enable the transition; text below these arrow-head lines are actions that are executed when the events above the transition line occur.

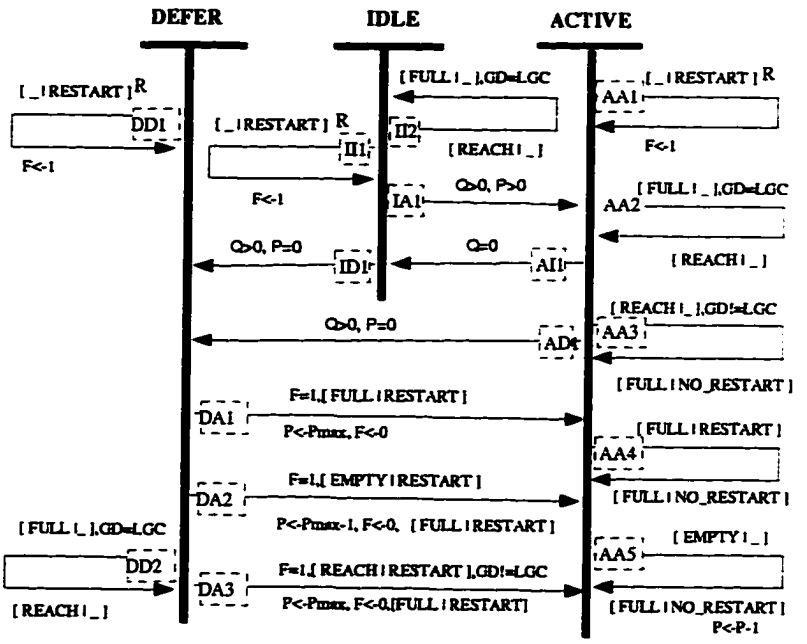
In the figure, the following notations have been used.

• The tuple [X | Y] denotes the state of a passing slot, where X indicates whether the slot is EMPTY, FULL, or REACH, and Y indicates the value of the restart bit (RESTART or NO_RESTART). A value of (_) indicates don't-care.

• A slot is assumed to be observed on the WRITE bus. If it is observed on the READ bus, then it has a superscript *R*, e.g., [REACH | RESTART]^R.

• Q denotes the length of the queue — if $Q > 0$, then the node has packets to send.

Figure 62 A S++-GA State Machine



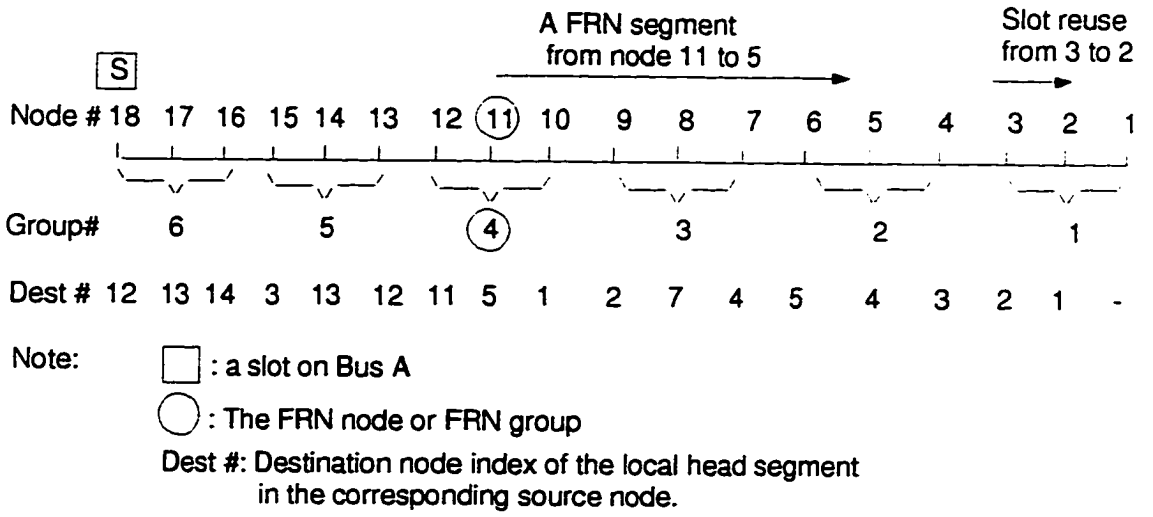
A S++-GA state machine has the same three states as S++: IDLE under which it has no packet waiting for channel (Q =0), ACTIVE under which it has at least one packet waiting for channel (Q>0) and bandwidth for transmission (P>0), and DEFER under which it has at least one packet waiting for channel but no bandwidth (P=0). The node is initially in the IDLE, and then moves to the ACTIVE if it has packets to send and its counter P is greater than zero. The node is forced to DEFER if it run out its counter P, but still has packets to send. The node leaves the DEFER when

its flag *F* is set to 1 and it observes a RESTART bit on the WRITE bus. A node does not reloaded *P* unless it is in the DEFER state, which prevents some possible unfairness conditions.

In order to support the recognition of field GD of a slot, a S++-GA node at least need buffer the bits before (and including) the GD field. In fact, it buffers one byte header. As a result, the latency per node of S++-GA is only 1 byte.

Fig.63 illustrates how S++-GA supports for *Slot Reuse* by an example of a 18-node S++-GA network in 6 groups with 3 nodes each. Note that we ignore the technical detail of how a node of S++-GA operates but focus on segment transmission phenomena on a bus caused by S++-GA.

Figure 63 The transmission principle of a 18-node 6-group S++-GA system.



We randomly selected node 11 in group 4 to be the FRN node for transmission. The first segment of the node is for node 5 in group 2. Like S++, a free slot S carries no segment before it reaches node 11. After node 11 transmits its segment for node 5 with loading the group code "2" of node 5 into the slot, nodes 10, 9, 8, 7, and 6 have no right to transmit in the slot because they recognize that it is busy. Node 6

sets the busy field of the slot to NULL because it has the same group code as the slot. Node 5 and 4 have no right to transmit in the slot because they recognize that the slot has the same group code as theirs. Node 3 can reuse the slot, but nodes 1 and 2 see their own group code in the slot.

It is clear that slot S in this situation is reused once and conveys two segments. One segment was transmitted from node 11 in group 4 to node 5 in group 2. The second segment was transmitted from node 3 in group 1 to node 2 in the same group by slot reuse. In the above example, a slot may convey only one segment if the FRN node has a segment either for the most-downstream group (i.e., for nodes 3, 2, or 1) or for an upstream node (i.e., for nodes 12, 13, ... or 18) of its, which is one of transmission chances. A slot may convey either two segments, or more up to maximum six segments with their own probability. For an instant, when a free slot meets such a case that active nodes the slot meets first in each group except group 1 send segments for their respective in-group downstream nodes whereas nodes in group 1 only need to have segments to send, a slot may convey at most six segments.

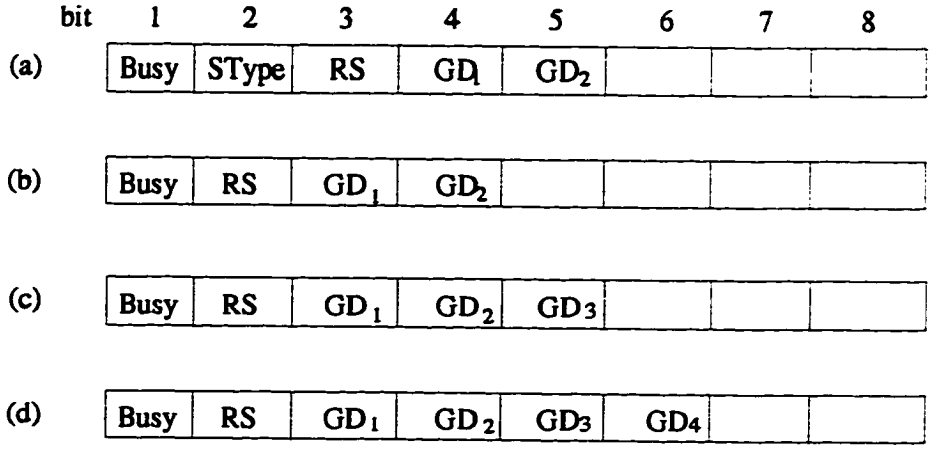
4.2.2 Specification of S++-GA Protocol

The first byte format of a slot and operations of S++-GA protocol are specified in the following, respectively.

4.2.2.1 The First Byte Format of a S++-GA Slot In Fig.64, we show the first byte (FB) format of a S++-GA slot. We maintain Busy field and Restart field Rs in the format like those in S++ protocol and add one new fields, destination group code, denoted by GD(i) and is used to carry a destination group address. The i in GD (i) means that the i-bit GD field covering (2^i-1) groups. In Fig.64, we provides four options of the FB format: Fig.64.a, b, c, and d.

Fig.64 (a) and (b) both support three-group S++-GA protocol. In Fig.(64.a), bit 1 is used as a traditional Busy field of the slot, Busy=0 means an empty slot while Busy=1 busy; bit 2 as Slot Type field SType of the slot, SType=0 means an ordinary message slot while SType=1 means a management message slot; bit 3 as restart field RS to convey cycle restart message, RS=1 means a permission to restart a new cycle while RS=0 means not; bit 4,5 as group-destination field GD to carry destination-group coding : 00 for null, 01 for group 1, 10 for group 2, and 11 for group 3.

Figure 64 The First Byte Formats of a S++-GA Slot.



4.2.2.2 Operation of S++-GA

Refering to the notations listed in the preious topic and figure 62, operation of a S++-GA node state machine is detailed as follows.

A S++-GA state machine have three states: IDLE under which there is no segment waiting for channel ($Q=0$), ACTIVE under which there is at least one segment waiting for channel ($Q>0$) and bandwidth for transmission ($P>0$), and DEFER under which there is at least one segment waiting for channel ($Q>0$) but no bandwidth ($P>0$).

The machine in IDLE takes loop-transition II1 that sets flag ($F=1$) when the machine receives a restart signal RS=1 on the READ bus.

The machine in IDLE takes loop-transition II2 that broadcasts a coming slot by marking nBUSY in field Busy of the slot when the machine receives a BUSY slot and its own group code in field GD of the slot.

The machine in IDLE takes transition ID1 to DEFER when the machine with no bandwidth ($P=0$) receives a segment for transmission ($Q>0$).

The machine in IDLE takes transition IA1 to ACTIVE when the machine with bandwidth ($P=0$) receives a segment for transmission ($Q>0$).

The machine in DEFER takes loop-transition DD1 that sets flag ($F=1$) when the machine receives a restart signal RS=1 on the READ bus.

The machine in DEFER takes loop-transition DD2 that broadcasts a coming slot by marking nBUSY in field Busy of the slot when the machine receives a BUSY slot and its own group code in field GD of the slot.

The machine in DEFER takes transition DA1 to ACTIVE that sets bandwidth $P=P_{max}$ and cleans period flag ($F=0$) when the machine with no bandwidth ($P=0$) and with flag $F=1$ receives BUSY and RS=1 in a slot.

The machine in DEFER takes transition DA2 to ACTIVE that sets bandwidth $P=P_{max}-1$, cleans flag ($F=0$), marks BUSY in the Busy field of a current slot, inserts the local group code into field GD, and transmits a segment from its (*Q) when the machine with no bandwidth ($P=0$) and flag $F=1$ receives BUSY and RS=1 in the slot.

The machine in DEFER takes transition DA3 to ACTIVE that sets bandwidth $P=P_{max}$, cleans flag ($F=0$), marks BUSY in the Busy field of a current slot, inserts the local group code into field GD, and transmits a segment from its (*Q) when the machine with no bandwidth ($P=0$) and flag $F=1$ receives REACH and RS=1 in the slot.

The machine in ACTIVE takes loop-transition AA1 that sets flag ($F=1$) when the

machine receives a restart signal RS=1 on the READ bus.

The machine in ACTIVE takes loop-transition AA2 that broadcasts a coming slot by marking nBUSY in field Busy of the slot when the machine receives a BUSY slot and its own group code in field GD of the slot.

The machine in ACTIVE takes loop-transition AA3 that marks BUSY in field Busy of a current slot, sets *RS=0*, inserts the local group code into field GD, and transmits a segment from its (*Q) when the machine receives REACH and *RS=1* in the slot.

The machine in ACTIVE takes loop-transition AA4 that sets *RS=0* of a current slot when the machine receives FULL and *RS=1* in the slot.

The machine in ACTIVE takes loop-transition AA5 that marks BUSY in field Busy of a current slot, sets *RS=0*, inserts the local group code into field GD, and transmits a segment from its (*Q) when the machine receives EMPTY and *RS=1* in the slot.

4.2.3 Simulation and Analysis

4.2.3.1 Model and Simulation Parameters The simulation of S++-GA networks is done only at the MAC-level, including the folded single bus, all the new-designed MAC fields of a slot, and node MAC finite state machines.

The basic parameters for the S++-GA protocol simulation are: the total number of nodes in the network is 60, 45, 30, or 15, the length of optical fiber is 37 kilometers, the channel capacity is 1 giga bits per second, the slot size is 256 bytes, and the server rate is 488,281 slots per second. We limit the size of a waiting queue (*Q) (or say buffer size) in a node to 48 segments and its P_{max} to 8.

The simulation collects data from 120,000 full lifetime slots. These data measure throughput gain, media access delay, slot-time utilization, and segment loss ratio of the whole system under uniform node distribution, uniform source-destination distribution,

and uniform Poisson traffic with average arrival rate λ of 1.0, 1.1, 1.2, 1.3, 1.4, 1.6, 1.7, and 6.0 segments per slot-time. The slot-time utilization is the sum of busy time intervals over the slot life-time during which a busy slot is loaded with a segment that has not yet reached its destination. Access delay is average time duration for a segment to wait in a node for transmission over all measured segments. A comparison among various grouping scheme was performed (i.e., performance measures for one-group system up to fifteen groups system.).

4.2.3.2 Performance Analysis Fig.(65) shows the theoretical and simulative results for the S++-GA network throughput gain. The theoretical curve is obtained by directly calculating based on the *Theorem 2* established in this paper. From the figure, it is noticed that the throughput gains increase with the increment of number of groups theoretically or simulatively and that simulative curves are close to the theoretical

Figure 65 Throughput Gain vs Number of Groups of S++-GA

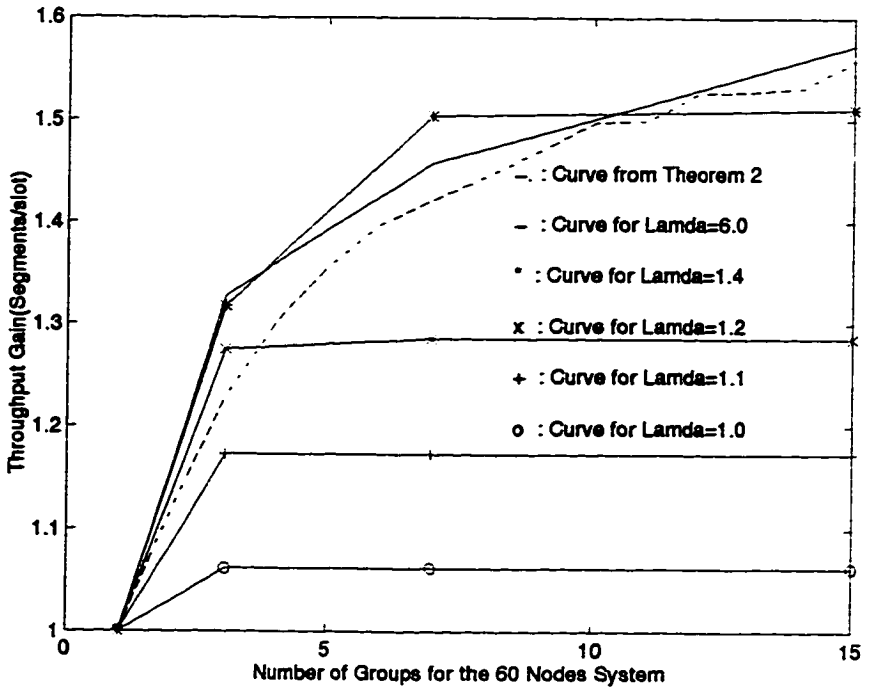


Figure 66 Segment Loss Ratio vs Number of Grouping of S++-GA

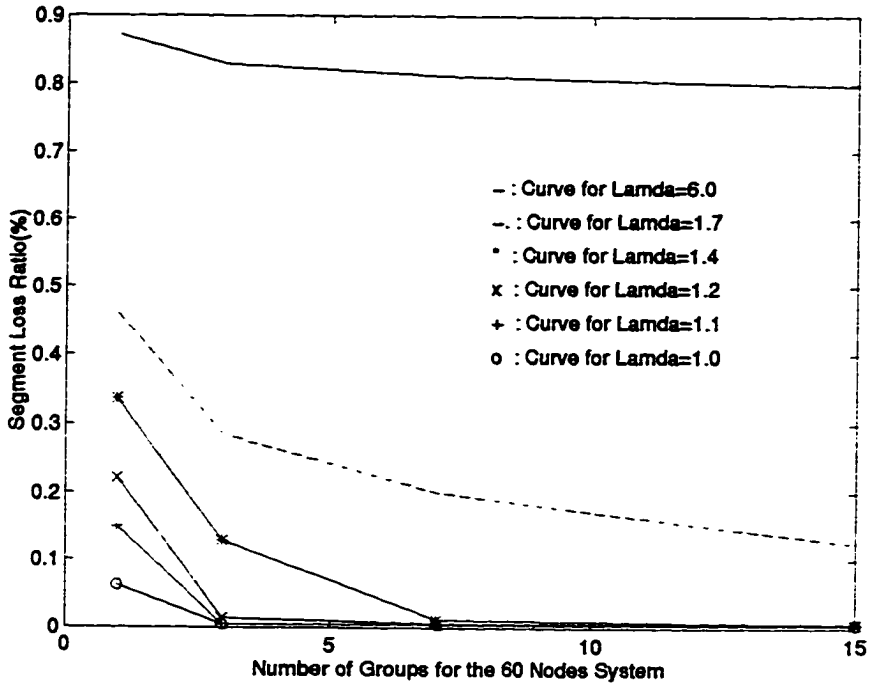


Figure 67 Access Delay vs Number of Grouping of S++-GA

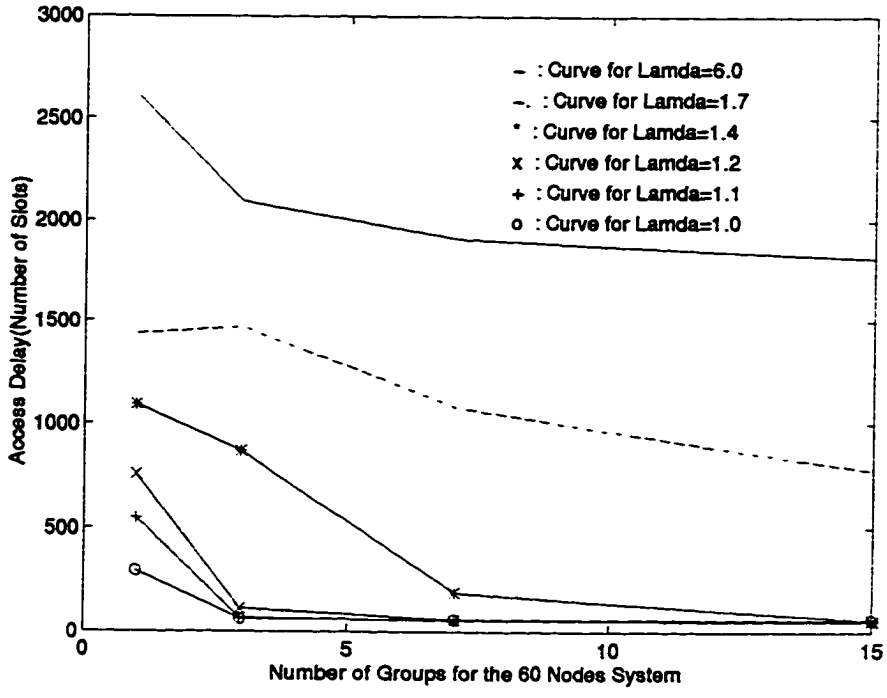
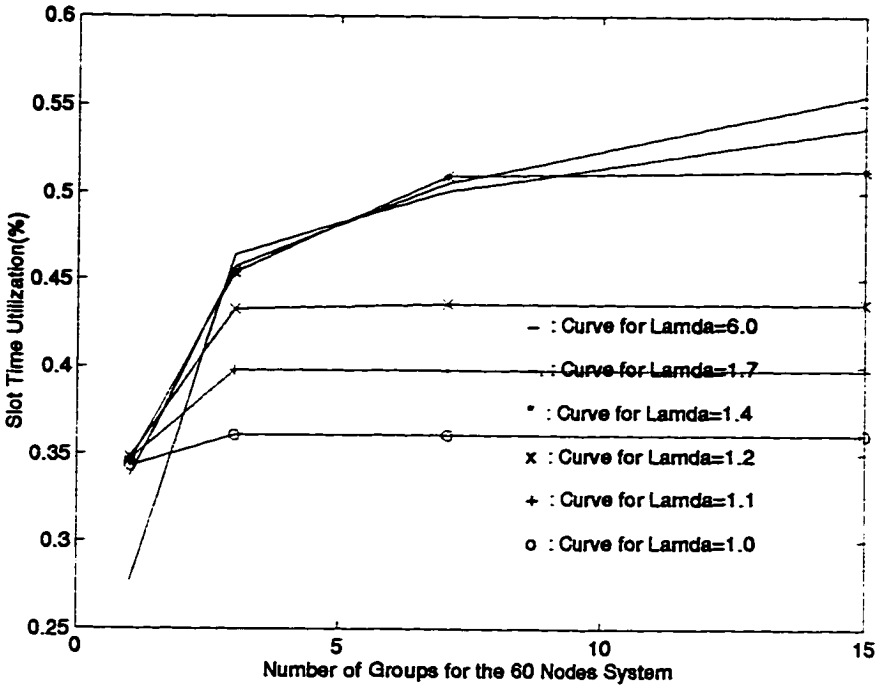


Figure 68 Slot-time Utilization vs Number of Grouping of S++-GA



curve. When the number of grouping in a S++-GA system is 1, the throughput gain is 1.

In Fig.(66) the simulation results of the S++-GA network segment loss ratio are shown. From the figure, it is seen that the loss ratios decrease as the number of groups increases. This is due to the fact that as the number of groups increases, throughput increases so that the probability that the system buffers are full decreases.

Fig.(67) shows simulative results of the S++-GA network average media access delay. It is observed that the delays decrease with the increment of number of groups. The reason that causes these happen is that more groups in the system directly provides more chances for every segment waiting in the system to be carried by a slot so that the waiting time for a segment to stay in the system with more groups becomes shorter than the system with less groups.

Fig.(68) shows the simulation results of the S++-GA network slot-time utilization. From the figure, it is clear that the utilizations increases as the number of groups increases. These are also direct results from the improved throughput.

Figs.(69 and 70) show the simulative and theoretical results for the throughput gains of the S++-GA networks with the different number (60, 45, 30, and 15) of nodes in the systems (The simulation under traffic λ=1.6 seg/s). It is found that as the number of nodes increases, the simulative throughput gains increase slightly. This phenomena can be explained by the theoretical curves, as you see in the figure (70).

Figure 69 Throughput Gains and Number of Nodes vs Number of Grouping of S++-GA under Heavy Traffic

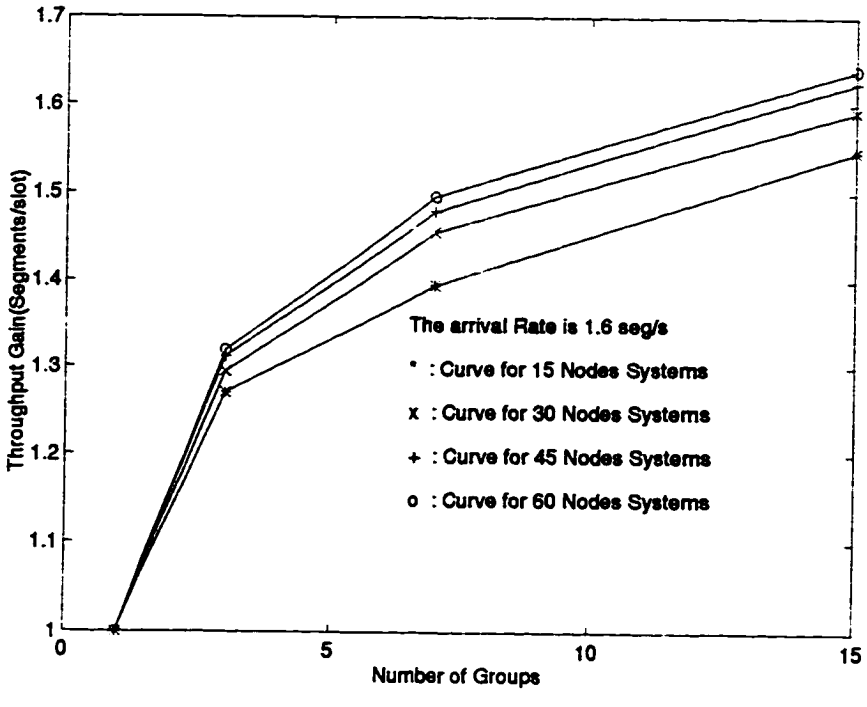
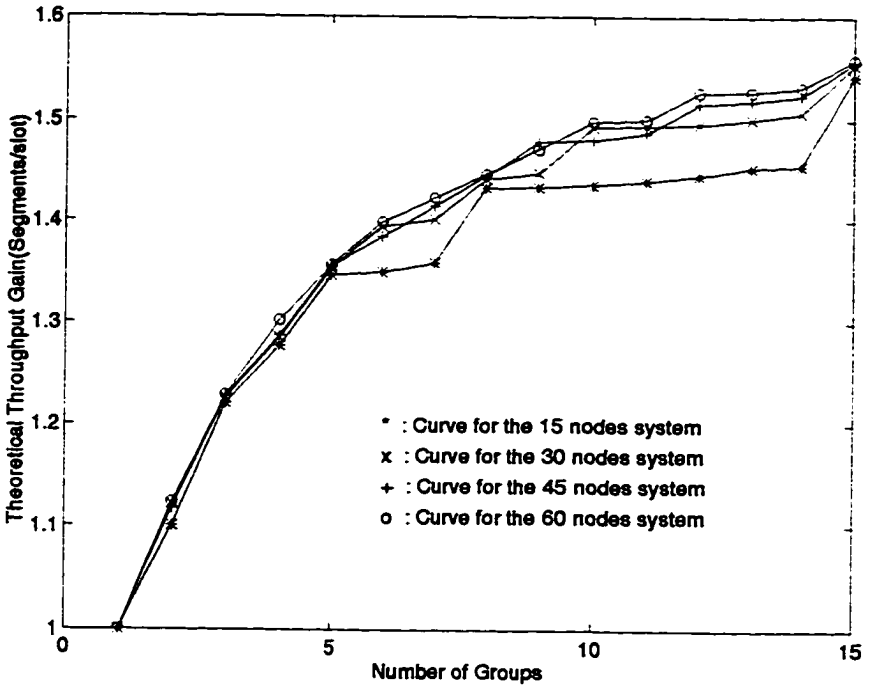


Figure 70 Theoretical Throughput Gains and Number of Nodes vs Number of Groups of S++-GA



4.3 Conclusion for This Chapter

S++-GA is a new protocol for high performance at the MAC layer. The main features that distinguish itself from S++ and improve the performance of S++ are: 1) that it divides all the nodes in the system into multiple groups and assigns a group code to each group and 2) that a S++-GA busy slot can be reused after the slot passes through its destination group on its WRITE bus.

From the theoretical analysis and calculations, S++-GA should always be effective in improving throughput of S++ and the throughput gains of a S++-GA network should increase as the number of groups increases.

From the simulation results, as the number of groups in the system increases, the throughput gain and the slot-time utilization of a S++-GA system do increase, while the access delays and segment loss ratio do decrease.

In fact, S++-GA is a general form for S++-DR, S++-EN, and S++. A complete grouping S++-GA system is a S++-DR system, a non-grouping S++-GA system is a S++ system, while a (M+1) grouping S++-GA system is considered as the corresponding distributed M erasure nodes S++-EN system in improving throughput gain. It is worthwhile stating further that any erasure node of a S++-EN protocol plays a role that groups the system and M erasure nodes splits a S++-EN system into $M+1$ divisions. The $M+1$ divisions are completely equivalent to the $M+1$ groups from S++-GA in improving throughput gain. A S++-EN erasure node, however, needs 257 bytes latency while any S++-GA node only needs 1 byte latency, which conducts that access delay of S++-GA is smaller than that of S++-EN.

In addition, the performance of S++-GA could also be improved further by using two sides "write", *Slot Pre-Use* or *PSI* like S++-EN.

Chapter 5 Conclusion

A simple scheme, named grouping transmission (*GT*), to enhance the throughput for a high speed dual bus configuration (DBNET) is proposed and analyzed. A theoretical analysis of the performances of the *GT* scheme is presented. The scheme has been applied to design six new protocols.

The main features of *GT* are: 1) it divides all the nodes in the system into multiple logical groups and assigns each group a group code and 2) a busy slot can be reloaded with a segment for *Slot-reuse* after the slot departs from its destination group, or a busy slot can be reloaded with a request for *Slot-Pre-use* after the slot departs from its source group. The *GT* scheme is a general form for *Slot Reuse* and *Slot Pre-Use*. The *Slot Reuse* includes *Destination Release* and *Erasure Nodes*. Being used as a *Slot-reuse* scheme, the *GT* either becomes *Destination Release* when every group in the system with a *GT* scheme has a single node or becomes *Erasure Nodes* when every group in the system with a *GT* scheme has more than one node. A *GT* scheme can employe less number of bits for slot-reuse than those of either of *Destination Release* and *Erasure Nodes*, does not need so long a local queue in each node as that in SP-DQDB protocol and can directly be used at lower layer than MAC at which *PSI* has to be implemented as long as an additional access control field (ACF) in the lower message format such as DQDB is defined. This means that a *GT* scheme is capable of obtaining higher system capacity and throughput than the others.

Four lemmas and eight theorems for *GT* schemes to obtain higher throughput gains than the corresponding non-grouping system are proved. The theoretical analysis and computation demonstrate that the throughput gains of performance of the systems increase as the number of groups dividing the system increases.

The implementations of a *GT* scheme are application-dependent. Based on the *GT* scheme, I have designed DQDB-GA, DQDB-GR, DQDB-NSP, DQDB-G, DQDB-H, and S++-GA protocols. Simulative curves verify both the protocols themselves and theoretical curves obtained from the *GT* throughput gain theorems.

In addition, *GT* can be inserted in CRMA and CBRMA resulting in CRMA-GA and CBRMA-GA. Furthermore, synchronous multimedia transmission is expected to be supported by DQDB-GA, DQDB-G, DQDB-H, and S++-GA protocols.

Appendix I Lemma 2 Proof

Proof:

(1) For a fold bus upper-side-sending DBNET system,

$$\sum_{k=1}^{G} P_{g,k} = \sum_{k=1}^{G} \frac{K_k - u(k-g+1)u(g-k+1)}{N-1} = \frac{\sum_{k=1}^{G} K_k - 1}{N-1} = \frac{N-1}{N-1} = 1.\square \tag{28}$$

(2) For a DBNET system:

(a) Let's, firstly, show that probability that a node in a DBNET sends segments to its downstream nodes is complete. In light of *Assumption 1*, we have

$$p_{ij} = \begin{cases} 0 & ,i = j \\ \frac{1}{i-1} & ,j = i-1, i-2, \ldots, 1 \end{cases}$$

and obviously

$$\sum_{j=1}^{i-1} p_{ij} = \sum_{j=1}^{i-1} \frac{1}{i-1} = \frac{i-1}{i-1} = 1, \quad i = 1, 2, \ldots, N$$

for each node in the system.

(b) Secondly, we derive Eq.(7) by substituting index i, j with $\sum_{d=1}^{g-1} K_d + n$ and $\sum_{d=1}^{k-1} K_d + m$ in the Eq.(30) above with reference to group index g, respectively. The index substitutuion $i = \sum_{d=1}^{g-1} K_d + n$ means that for the *GT* system described above, g be an index number of a group in the system, g=1,2,...,G, n be a relative index number of a node within group g, n=1,2,...,Kd and i is an absolute index in the whole system, i=1,2,...,N. For the g-th group, the probability that group g transmits to group

k, a downstream group, is

$$P_{g,k} = \sum_{i=\sum_{d=1}^{g-1} K_d+1}^{\sum_{d=1}^{g} K_d} \frac{1}{K_g} \sum_{j=\sum_{d=1}^{k-1} K_d+1}^{\sum_{d=1}^{k} K_d} p_{ij} = \sum_{i=\sum_{d=1}^{g-1} K_d+1}^{\sum_{d=1}^{g} K_d} \sum_{j=\sum_{d=1}^{k-1} K_d+1}^{\sum_{d=1}^{k} K_d} \frac{1}{K_g(i-1)}$$

$$= \sum_{n=1}^{K_g} \frac{1}{K_g} \sum_{j=\sum_{d=1}^{k-1} K_d+1}^{\sum_{d=1}^{k} K_d} \frac{1}{\sum_{d=1}^{g-1} K_d + n - 1} = \sum_{m=1}^{K_g} \frac{1}{K_g} \frac{K_k}{\sum_{d=1}^{g-1} K_d + n - 1}$$

$$= \frac{K_k}{K_g} \sum_{n=1}^{K_g} \frac{1}{\sum_{d=1}^{g-1} K_d + n - 1}, \quad g = 1,2,\ldots,G \text{ and } g > k \neq l$$

Let us consider probability $P_{g,g}$ of in-group transmission. Since great group-index means upstream, group 1 has no downstream group, which means that for g=1, we have

$$\sum_{i=1}^{K_g} \frac{1}{K_g} \sum_{j=1}^{i-1} p_{ij} = \sum_{i=1}^{K_g} \frac{1}{K_g} \sum_{j=1}^{i-1} \frac{1}{i-1} = \sum_{i=1}^{K_g} \frac{1}{K_g} \frac{i-1}{i-1} = \frac{\sum_{i=1}^{K_g}}{K_g} = 1 \ .$$

And for g >1, we have

$$P_{g,g} = \sum_{i=\sum_{d=1}^{g-1} K_d+1}^{\sum_{d=1}^{g} K_d} \frac{1}{K_g} \sum_{j=\sum_{d=1}^{g-1} K_d+1}^{i-\sum_{d=1}^{g-1} K_d-1} p_{ij} \underset{m=j-\sum_{d=1}^{g-1} K_d}{=} \sum_{i=\sum_{d=1}^{g-1} K_d+1}^{\sum_{d=1}^{g} K_d} \frac{1}{K_g} \sum_{m=1}^{i-\sum_{d=1}^{g-1} K_d-1} \frac{1}{i-1}$$

$$\underset{n=i-\sum_{d=1}^{g-1} K_d}{=} \sum_{n=1}^{K_g} \frac{1}{K_g} \sum_{m=1,}^{n-1} \frac{1}{\sum_{d=1}^{g-1} K_d + n - 1} = \frac{1}{K_g} \sum_{n=1}^{K_g} \frac{n-1}{\sum_{d=1}^{g-1} K_d + n - 1} \ .$$

So,

$$P_{g,g} = \begin{cases} 1 & ,g = 1; \\ \frac{1}{K_g} \sum_{n=1}^{K_g} \frac{n-1}{\sum_{d=1}^{g-1} K_d + n - 1} & ,g > 1; \end{cases}$$

$$P_{g,k} = \frac{K_k}{K_g} \sum_{n=1}^{K_g} \frac{1}{\sum_{d=1}^{g-1} K_d + n - 1}, \quad g = 1,2,\ldots,G \text{ and } g > l.$$

(c) Therefore,

$$\sum_{k=1}^{g} P_{g,k} = P_{g,g} + \sum_{k=1}^{g-1} P_{g,k}$$

$$= \frac{1}{K_g} \sum_{n=1}^{K_g} \frac{n-1}{\sum_{d=1}^{g-1} K_d + n - 1} + \sum_{k=1}^{g-1} \frac{K_k}{K_g} \sum_{n=1}^{K_g} \frac{1}{\sum_{d=1}^{g-1} K_d + n - 1}$$

$$= \frac{1}{K_g} \sum_{n=1}^{K_g} \frac{n - 1 + \sum_{k=1}^{g-1} K_k}{\sum_{d=1}^{g-1} K_d + n - 1} = \frac{\sum_{n=1}^{K_g}}{K_g} = \frac{K_g}{K_g} = 1.\square$$

Appendix II Lemma 3 Proof

Proof:

We simplify the problem. For $m > k$ $(k > 0)$ or $k < 1$, the *Lemma* is not significant; while $m < 1$ means actually no chance for use, the *Lemma* is not significant, either, that's

$$P_r(m,k) = 0, \quad (m > k,\ k > 0) \ \text{or}\ k < 1 \ \text{or}\ m < 1 \ .$$

So, the *Lemma* becomes

$$\sum_{m=0}^{\infty} P_r(m,k) = \sum_{m=1}^{k} P_r(m,k) = 1 \ .$$

We use mathematical induction as follows.

(a) we verify it by four initial cases: k=1, 2, 3, and 4, respectively.

For case $k = 1$, the left side of (11) becomes:

$$\sum_{m=1}^{k} P_r(m,k) = \sum_{m=1}^{1} P_r(m,1) = P_r(1,1) = P_{1,1}$$

and from *Lemma* 2, $\sum_{m=1}^{k} P_r(m,k) \overset{k=1}{=} \sum_{j=1}^{1} P_{1,j} = P_{1,1} = 1.$

For case $k = 2$, the left side of (11) becomes:

$$\sum_{m=1}^{k} P_r(m,k) = \sum_{m=1}^{2} P_r(m,2) = P_r(1,2) + P_r(2,2) = P_{2,1} + P_{2,2}P_{1,1}$$

and from $k = 1$ and *Lemma* 2, $\sum_{m=1}^{k} P_r(m,k) \overset{k=2}{=} \sum_{j=1}^{2} P_{2,j} = P_{2,1} + P_{2,2} = 1.$

For case $k = 3$, the left side of (11) becomes:

$$\begin{aligned}
\sum_{m=1}^{k} P_r(m,k) &= \sum_{m=1}^{3} P_r(m,3) = P_r(1,3) + P_r(2,3) + P_r(3,3) \\
&= P_{3,1} + \sum_{j=2}^{3} P_{3,j}P_r(1,j-1) + \sum_{j=3}^{3} P_{3,j}P_r(2,j-1) \\
&= P_{3,1} + P_{3,2}P_r(1,1) + P_{3,3}P_r(1,2) + P_{3,3}P_r(2,2) \\
&= P_{3,1} + P_{3,2} + P_{3,3} \sum_{m=1}^{2} P_r(m,2)
\end{aligned}$$

and from the result of $k = 2$ and *Lemma 2*, $\sum_{m=1}^{k} P_r(m,k) \overset{k\equiv 3}{=} 1$.

For case $k = 4$, the left side of (11) becomes:

$$\begin{aligned}
\sum_{m=1}^{k} P_r(m,k) &= \sum_{m=1}^{4} P_r(m,4) = P_r(1,4) + P_r(2,4) + P_r(3,4) + P_r(4,4) \\
&= P_{4,1} + \sum_{j=2}^{4} P_{4,j} P_r(1,j-1) + \sum_{j=3}^{4} P_{4,j} P_r(2,j-1) + \sum_{j=4}^{4} P_{4,j} P_r(3,j-1) \\
&= P_{4,1} + P_{4,2} P_r(1,1) + P_{4,3} P_r(1,2) + P_{4,4} P_r(1,3) + \\
&\qquad + P_{4,3} P_r(2,2) + P_{4,4} P_r(2,3) + P_{4,4} P_{3,3} P_r(2,2) \\
&= P_{4,1} + P_{4,2} P_{1,1} + P_{4,3} P_{2,1} + P_{4,4} P_{3,1} \\
&\qquad + P_{4,3} P_{2,2} + P_{4,4}(P_{3,2} + P_{3,3} P_{2,1}) + P_{4,4} P_{3,3} P_{2,2} \\
&= P_{4,1} + P_{4,2} + P_{4,3}(P_{2,1} + P_{2,2}) + P_{4,4}(P_{3,1} + P_{3,2} + P_{3,3}(P_{2,1} + P_{2,2})) \\
&= P_{4,1} + P_{4,2} \sum_{m=1}^{1} P_r(m,1) + P_{4,3} \sum_{m=1}^{2} P_r(m,2) + P_{4,4} \sum_{m=1}^{3} P_r(m,3)
\end{aligned}$$

and from the results of $k = 1, 2, and 3$ and *Lemma 2*, $\sum_{m=2}^{k} P_r(m,k) \overset{k\equiv 4}{=} 1$.

(b) Set up the following mathematical induction assumption:

$$\forall (k-1, m | (k-1, m) \in I^+, 1 < m \leq k-1), \exists \sum_{m=1}^{k-1} P_r(m, k-1) = 1.$$

(c) For case k, the left side of (11) becomes:

$$\sum_{m=1}^{k} P_r(m,k)$$

$$= P_r(1,k) + P_r(2,k) + P_r(3,k) + P_r(4,k) \ldots + P_r(k-2,k) + P_r(k-1,k) + P_r(k,k)$$

$$= P_{k,1} + \sum_{j=2}^{k} P_{k,j} P_r(1,j-1) + \sum_{j=3}^{k} P_{k,j} P_r(2,j-1)$$

$$\ldots + \sum_{j=k-2}^{k} P_{k,j} P_r(k-3,j-1) + \sum_{j=k-1}^{k} P_{k,j} P_r(k-2,j-1) + \sum_{j=k}^{k} P_{k,j} P_r(k-1,j-1)$$

$$= P_{k,1} + \sum_{j=2}^{k} P_{k,j} P_{j-1,1}$$

$$+ (P_{k,3} P_r(2,2) + P_{k,4} P_r(2,3) \ldots P_{k,k-1} P_r(2,k-2) + P_{k,k} P_r(2,k-1))$$

$\ldots$

$$+ (P_{k,k-2} P_r(k-3,k-3) + P_{k,k-1} P_r(k-3,k-2) + P_{k,k} P_r(k-3,k-1))$$

$$+ (P_{k,k-1} P_r(k-2,k-2) + P_{k,k} P_r(k-2,k-1))$$

$$+ P_{k,k} P_r(k-1,k-1)$$

$$= P_{k,1} + P_{k,2} P_{1,1} + P_{k,3}(P_{2,1} + P_{2,2} P_{1,1}) + P_{k,4}(P_{3,1} + P_{3,2} P_{1,1} + P_{3,3}(P_{2,1} + P_{2,2} P_{1,1}))$$

$\ldots\ldots$

$$+ P_{k,k-1}(P_{k-2,1} + P_{k-2,2} P_{1,1} + P_{k-2,3}(P_{2,1} + P_{2,2} P_{1,1})$$

$$+ P_{k-2,4}(P_{3,1} + P_{3,2} P_{1,1} + P_{3,3}(P_{2,1} + P_{2,2} P_{1,1})) \ldots + P_{k-2,k-2}(\ldots))$$

$$+ P_{k,k}(P_{k-1,1} + P_{k-1,2} P_{1,1} + P_{k-1,3}(P_{2,1} + P_{2,2} P_{1,1})$$

$$+ P_{k-1,4}(P_{3,1} + P_{3,2} P_{1,1} + P_{3,3}(P_{2,1} + P_{2,2} P_{1,1})) \ldots + P_{k-1,k-1}(\ldots))$$

$$= P_{k,1} + P_{k,2} \sum_{m=1}^{1} P_r(1,1) + P_{k,3} \sum_{m=1}^{2} P_r(m,2) +$$

$$\ldots\ldots + P_{k,k-1} \sum_{m=1}^{k-2} P_r(m,k-2) + P_{k,k} \sum_{m=1}^{k-1} P_r(m,k-1)$$

and from the mathematical induction assumption and *Lemma 2*, we conclude that

$$\forall (k,m | (k,m) \in I^+, 1 < m \leq k), \quad \exists \sum_{m=0}^{\infty} P_r(m,k) = \sum_{m=1}^{k} P_r(m,k) = 1. \square$$

Appendix III Lemma 4 Proof

Proof:

1) For $G = 1$, the *Lemma* is not significant and for $p > G - g$, the *Lemma* is not significant, either, that's

$$P_p^G(p,g) = 0, \quad (p > G-g,\ 0 < g < G) \ \ or \ \ G = g \ .$$

So, the *Lemma* becomes

$$\sum_{p=0}^{\infty} P_p^G(p,g) = \sum_{p=0}^{G-g} P_p^G(p,g) = 1 \ .$$

2) Consider the cases $G = 2$ and $G > 2$, respectively.

(a) $G = 2$. The Lemma is significant only for $g = 1$. So we have

$$\sum_{p=0}^{G-g} P_p^G(p,g) = \sum_{p=0}^{2-1} P_p^2(p,1) = P_p^2(0,1) + P_p^2(1,1)$$

$$= \prod_{k=1+1}^{2} \sum_{j=1}^{1} P_{k,j} + \sum_{n=1+1}^{2} \sum_{j=1+1}^{n} \Big(\prod_{k=n+1}^{2} \sum_{m=1}^{1} P_{k,m} \Big)^{u(2-2)} P_{n,j} \Big(\prod_{k=1+1}^{j-1} \sum_{m=1}^{1} P_{k,m} \Big)^{u(j-2)}$$

$$= P_{2,1} + P_{2,2}$$

and from *Lemma 2*, $\sum_{p=0}^{G-g} P_p^G(p,g) \overset{G=2}{=} \sum_{p=0}^{2-1} P_p^2(p,1) = \sum_{j=1}^{2} P_{2,j} = 1$.

(b) $G > 2$. Let $g = G - K$. We straightly expend the left side of (14),

$$\sum_{p=0}^{G-g} P_p^G(p,g) \overset{g=G-K}{=} \sum_{p=0}^{K} P_p^G(p, G-K)$$

$$= P_p^G(0,G-K) + P_p^G(1,G-K) + P_p^G(2,G-K) \ldots + P_p^G(K-1,G-K) + P_p^G(K,G-K) \tag{48}$$

We further expend each item of the right side of (48). For the first item, i.e., $p = 0$, we have

$$P_p^G(0,G-K) = \prod_{k=G-K+1}^{G} \sum_{j=1}^{G-K} P_{k,j} = \sum_{j=1}^{G-K} P_{G,j} \sum_{j=1}^{G-K} P_{G-1,j} \cdots \sum_{j=1}^{G-K} P_{G-K+2,j} \sum_{j=1}^{G-K} P_{G-K+1,j}$$

, and for the second item, i.e., $p = 1$, we have

$$P_p^G(1, G-K) = \sum_{n=G-K+1}^{G} \sum_{j=G-K+1}^{n} (\prod_{k=n+1}^{G} \sum_{m=1}^{G-K} P_{k,m})^{u(G-n)} P_{n,j} (\prod_{k=G-K+1}^{j-1} \sum_{m=1}^{G-K} P_{k,m})^{u(j-G+K-1)}$$

$$= (\sum_{m=1}^{G-K} P_{G,m} \sum_{m=1}^{G-K} P_{G-1,m} \cdots \sum_{m=1}^{G-K} P_{G-K+2,m})^{u(K-1)} P_{G-K+1,G-K+1}$$

$$+ (\prod_{k=G-K+3}^{G} \sum_{m=1}^{G-K} P_{k,m})^{u(K-2)} \sum_{j=G-K+1}^{G-K+2} P_{G-K+2,j} (\prod_{k=G-K+1}^{j-1} \sum_{m=1}^{G-K} P_{k,m})^{u(j-G+K-1)}$$

$$+ (\prod_{k=G-K+4}^{G} \sum_{m=1}^{G-K} P_{k,m})^{u(K-3)} \sum_{j=G-K+1}^{G-K+3} P_{G-K+3,j} (\prod_{k=G-K+1}^{j-1} \sum_{m=1}^{G-K} P_{k,m})^{u(j-G+K-1)}$$

.....

$$+ (\prod_{k=G-1}^{G} \sum_{m=1}^{G-K} P_{k,m})^{u(2)} \sum_{j=G-K+1}^{G-2} P_{G-2,j} (\prod_{k=G-K+1}^{j-1} \sum_{m=1}^{G-K} P_{k,m})^{u(j-G+K-1)}$$

$$+ (\prod_{k=G}^{G} \sum_{m=1}^{G-K} P_{k,m})^{u(1)} \sum_{j=G-K+1}^{G-1} P_{G-1,j} (\prod_{k=G-K+1}^{j-1} \sum_{m=1}^{G-K} P_{k,m})^{u(j-G+K-1)}$$

$$+ \sum_{j=G-K+1}^{G} P_{G,j} (\prod_{k=G-K+1}^{j-1} \sum_{m=1}^{G-K} P_{k,m})^{u(j-G+K-1)}$$

, and for the third item, i.e., $p = 2$, we have

$$P_p^G(2, G-K) = \sum_{n=G-K+2}^{G} \sum_{j=G-K+2}^{n} (\prod_{k=n+1}^{G} \sum_{m=1}^{G-K} P_{k,m})^{u(G-n)} P_{n,j} P_{pi}(2, G-K, j)$$

$$= (\sum_{m=1}^{G-K} P_{G,m} \sum_{m=1}^{G-K} P_{G-1,m} \cdots \sum_{m=1}^{G-K} P_{G-K+3,m})^{u(K-2)} P_{G-K+2,G-K+2} P_{G-K+1,G-K+1}$$

$$+ (\prod_{k=G-K+4}^{G} \sum_{m=1}^{G-K} P_{k,m})^{u(K-3)} \sum_{j=G-K+2}^{G-K+3} P_{G-K+3,j} P_{pi}(2, G-K, j)$$

$$+ (\prod_{k=G-K+5}^{G} \sum_{m=1}^{G-K} P_{k,m})^{u(K-4)} \sum_{j=G-K+2}^{G-K+4} P_{G-K+4,j} P_{pi}(2, G-K, j)$$

.....

$$+ (\prod_{k=G-1}^{G} \sum_{m=1}^{G-K} P_{k,m})^{u(2)} \sum_{j=G-K+2}^{G-2} P_{G-2,j} P_{pi}(2, G-K, j)$$

$$+ (\prod_{k=G}^{G} \sum_{m=1}^{G-K} P_{k,m})^{u(1)} \sum_{j=G-K+2}^{G-1} P_{G-1,j} P_{pi}(2, G-K, j)$$

$$+ \sum_{j=G-K+2}^{G} P_{G,j} P_{pi}(2, G-K, j)$$

, and

$\ddots \quad \ddots$

, and for the k-1-th item, i.e., $p = K-2$, we have

$$P_p^G(K-2, G-K) = \sum_{n=G-K+K+2}^{G} \sum_{j=G-2}^{n} (\prod_{k=n+1}^{G} \sum_{m=1}^{G-K} P_{k,m})^{u(G-n)} P_{n,j} P_{pi}(K-2, G-K, j)$$

$$= (\sum_{m=1}^{G-K} P_{G,m} \sum_{m=1}^{G-K} P_{G-1,m})^{u(2)} P_{G-2,G-2} P_{pi}(K-2, G-K, G-2)$$

$$+ (\sum_{m=1}^{G-K} P_{G,m})^{u(1)} \sum_{j=G-2}^{G-1} P_{G-1,j} P_{pi}(K-2, G-K, j)$$

$$+ \sum_{j=G-2}^{G} P_{G,j} P_{pi}(K-2, G-K, j)$$

$$= (\sum_{m=1}^{G-K} P_{G,m} \sum_{m=1}^{G-K} P_{G-1,m})^{u(2)} P_{G-2,G-2} P_{pi}(K-2, G-K, G-2)$$

$$+ (\sum_{m=1}^{G-K} P_{G,m})^{u(1)} (P_{G-1,G-2} P_{pi}(K-2, G-K, G-2) + P_{G-1,G-1} P_{pi}(K-2, G-K, G-1))$$

$$+ P_{G,G-2} P_{pi}(K-2, G-K, G-2) + P_{G,G-1} P_{pi}(K-2, G-K, G-1) + P_{G,G} P_{pi}(K-2, G-K, G)$$

, and for the k-th item, i.e., $p = K-1$, we have

$$P_p^G(K-1, G-K) = \sum_{n=G-K+K+1}^{G} \sum_{j=G-1}^{n} (\prod_{k=n+1}^{G} \sum_{m=1}^{G-K} P_{k,m})^{u(G-n)} P_{n,j} P_{pi}(K-1, G-K, j)$$

$$= (\sum_{m=1}^{G-K} P_{G,m})^{u(1)} P_{G-1,G-1} P_{pi}(K-1, G-K, G-1)$$

$$+ \sum_{j=G-1}^{G} P_{G,j} P_{pi}(K-1, G-K, j)$$

$$= (\sum_{m=1}^{G-K} P_{G,m})^{u(1)} P_{G-1,G-1} P_{pi}(K-1, G-K, G-1)$$

$$+ P_{G,G-1} P_{pi}(K-1, G-K, G-1) + P_{G,G} P_{pi}(K-1, G-K, G)$$

, and for the k+1-th item, i.e., $p = K$, we have

$$P_p^G(K, G-K) = \sum_{n=G-K+K}^{G} \sum_{j=G}^{n} (\prod_{k=n+1}^{G} \sum_{m=1}^{G-K} P_{k,m})^{u(G-n)} P_{n,j} P_{pi}(K, G-K, j)$$

$$= P_{G,G} P_{pi}(K, G-K, G) = P_{G,G} P_{G-1,G-1} P_{pi}(K-1, G-K, G-1) \ldots$$

$$= P_{G,G} P_{G-1,G-1} \ldots P_{G-K+2,G-K+2} P_{G-K+1,G-K+1} = \prod_{j=G-K+1}^{G} P_{j,j}.$$

After expending each function P_{pi} (Due to the limitation of length of the paper,

$$\sum_{p=0}^{G-g} P_p^G(p,g) \overset{g=G-K}{=} \sum_{p=0}^{K} P_p^G(p, G-K)$$

$$= \sum_{j=1}^{G-K} P_{G,j}(\sum_{j=1}^{G-K} P_{G-1,j}(\sum_{j=1}^{G-K} P_{G-2,j}(\ldots(\sum_{j=1}^{G-K} P_{G-K+2,j}(\sum_{j=1}^{G-K} P_{G-K+1,j} + P_{G-K+1,G-K+1})$$

$$+ P_{G-K+2,G-K+1} + P_{G-K+2,G-K+2}(\sum_{j=1}^{G-K} P_{G-K+1,j} + P_{G-K+1,G-K+1}))$$

$$\ldots) + P_{G-2,G-K+1} + P_{G-2,G-K+2}(\sum_{j=1}^{G-K} P_{G-K+1,j} + P_{G-K+1,G-K+1}) \ldots + P_{G-2,G-2}(\ldots))$$

$$\ldots + P_{G-1,G-K+1} + P_{G-1,G-K+2}(\sum_{j=1}^{G-K} P_{G-K+1,j} + P_{G-K+1,G-K+1}) \ldots + P_{G-1,G-1}(\ldots))$$

$$+ P_{G,G-K+1}$$

$$+ P_{G,G-K+2}(\sum_{j=1}^{G-K} P_{G-K+1,j} + P_{G-K+1,G-K+1})$$

$$\ldots\ldots$$

$$+ P_{G,G-2}(\ldots)$$

$$+ P_{G,G-1}(\sum_{j=1}^{G-K} P_{G-2,j}(\ldots(\sum_{j=1}^{G-K} P_{G-K+2,j}(\sum_{j=1}^{G-K} P_{G-K+1,j} + P_{G-K+1,G-K+1})$$

$$+ P_{G-K+2,G-K+1} + P_{G-K+2,G-K+2}(\sum_{j=1}^{G-K} P_{G-K+1,j} + P_{G-K+1,G-K+1}))$$

$$\ldots) + P_{G-2,G-K+1} + P_{G-2,G-K+2}(\sum_{j=1}^{G-K} P_{G-K+1,j} + P_{G-K+1,G-K+1}) \ldots + P_{G-2,G-2}(\ldots))$$

$$+ P_{G,G}(\sum_{j=1}^{G-K} P_{G-1,j}(\sum_{j=1}^{G-K} P_{G-2,j}(\ldots(\sum_{j=1}^{G-K} P_{G-K+2,j}(\sum_{j=1}^{G-K} P_{G-K+1,j} + P_{G-K+1,G-K+1})$$

$$+ P_{G-K+2,G-K+1} + P_{G-K+2,G-K+2}(\sum_{j=1}^{G-K} P_{G-K+1,j} + P_{G-K+1,G-K+1}))$$

$$\ldots) + P_{G-2,G-K+1} + P_{G-2,G-K+2}(\sum_{j=1}^{G-K} P_{G-K+1,j} + P_{G-K+1,G-K+1}) \ldots + P_{G-2,G-2}(\ldots))$$

$$\ldots + P_{G-1,G-K+1} + P_{G-1,G-K+2}(\sum_{j=1}^{G-K} P_{G-K+1,j} + P_{G-K+1,G-K+1}) \ldots + P_{G-1,G-1}(\ldots)) \tag{56}$$

we omit the expansion of functions of P_{pi}.), by substituting each item in (48) with

the corresponding expansions of the items above and combining them with reference to $P_{G,j}(j = 1, 2, \ldots, G)$, we obtain equation (56) above

Since we already have *Lemma 2*,

$$\sum_{p=0}^{G-g} P_p^G(p,g) \overset{g=G-K}{=} \sum_{p=0}^{K} P_p^G(p, G-K)$$

$$= \sum_{j=1}^{G-K} P_{G,j}(\sum_{j=1}^{G-1} P_{G-1,j}) + P_{G,G-K+1} + P_{G,G-K+2}(\sum_{j=1}^{G-K+1} P_{G-K+1,j})$$

.....

$$+ P_{G,G-2}(\sum_{j=1}^{G-3} P_{G-3,j}) + P_{G,G-1}(\sum_{j=1}^{G-2} P_{G-2,j}) + P_{G,G}(\sum_{j=1}^{G-1} P_{G-1,j})$$

$$= \sum_{j=1}^{G-K} P_{G,j} + P_{G,G-K+1} + P_{G,G-K+2} \ldots + P_{G,G-2} + P_{G.G-1} + P_{G,G} = \sum_{j=1}^{G} P_{G,j} = 1 \quad . \tag{57}$$

we, therefore, conclude that

$$\forall(G, p, g | (G, p, g) \in I^+, 0 < g \leq G), \quad \exists \sum_{p=0}^{\infty} P_p^G(p,g) = \sum_{p=1}^{G-g} P_p^G(p,g) = 1. \square$$

Appendix IV Theorem 1 Proof

Proof:

(1) For the proof for a DBNET system, refer to the corresponding decomposition part of *Theorem 3* proof.

(2) For the proof for a fold bus one-side-sending DBNET system,

(a)For arbitrary G, we expand the left side of 16

$$
\begin{aligned}
P_R^G(1) &= \sum_{g=1}^{G} P_g\{(P_{g,1}(\frac{1}{2})^{u(g)u(2-g)} + \sum_{h=g}^{G}(\frac{1}{2})^{u(g+1-h)u(h-g+1)}P_{g,h}) \\
&= P_1(\frac{P_{1,1}}{2} + \frac{P_{1,1}}{2} + P_{1,2} \ldots + P_{1,G}) \\
&+ P_2(P_{2,1} + \frac{P_{2,2}}{2} + P_{2,3} \ldots + P_{2,G}) \\
&+ P_3(P_{3,1} + \frac{P_{3,3}}{2} + P_{3,4} \ldots + P_{3,G}) \\
&\ldots \\
&+ P_{G-2}(P_{G-2,1} + \frac{P_{G-2,G-2}}{2} + P_{G-2,G-1} + P_{G-2,G}) \\
&+ P_{G-1}(P_{G-1,1} + \frac{P_{G-1,G-1}}{2} + P_{G-1,G}) \\
&+ P_G(P_{G,1} + \frac{P_{G,G}}{2})
\end{aligned}
\tag{59}
$$

$$
\begin{aligned}
P_R^G(2) &= \sum_{g=2}^{G} P_g \Big(\sum_{k=2}^{g} \big(\frac{1}{2}\big)^{u(g+1-k)u(k-g+1)} P_{g,k} P_r(2,k)\Big) \\
&= P_2\big(\frac{P_{2,2}}{2}\big) \\
&+ P_3\Big(P_{3,2} + \frac{P_{3,3}}{2}\big(P_{2,1} + \frac{P_{2,2}}{2} + P_{2,3} \ldots + P_{2,G}\big)\Big) \\
&\ldots \\
&+ P_{G-1}\Big(P_{G-1,2} + P_{G-1,3}\big(P_{2,1} + \frac{P_{2,2}}{2} + P_{2,3} \ldots + P_{2,G}\big) \\
&\ldots + \frac{P_{G-1,G-1}}{2}\big(P_{G-2,1} + \frac{P_{G-2,G-2}}{2} + P_{G-2,G-1} + P_{G-2,G}\big)\Big) \\
&+ P_G\Big(P_{G,2} + P_{G,3}\big(P_{2,1} + \frac{P_{2,2}}{2} + P_{2,3} \ldots + P_{2,G}\big) \\
&\ldots + \frac{P_{G,G}}{2}\big(P_{G-1,1} + \frac{P_{G-1,G-1}}{2} + P_{G-1,G} + P_{G-1,G}\big)\Big)
\end{aligned}
\tag{60}
$$

$$
\begin{aligned}
P_R^G(3) &= \sum_{g=3}^{G} P_g \Big(\sum_{k=3}^{g} \big(\frac{1}{2}\big)^{u(g+1-k)u(k-g+1)} P_{g,k} P_r(3,k)\Big) \\
&= P_3\big(\frac{P_{3,3}}{2}\frac{P_{2,2}}{2}\big) \\
&+ P_4\Big(P_{4,3}\frac{P_{2,2}}{2} + \frac{P_{4,4}}{2}\big(P_{3,2} + \frac{P_{3,3}}{2}\big(P_{2,1} + \frac{P_{2,2}}{2} + P_{2,3} \ldots + P_{2,G}\big)\Big) \\
&\ldots \\
&+ P_{G-1}\Big(P_{G-1,3}\frac{P_{2,2}}{2} + P_{G-1,4}\big(P_{3,2} + \frac{P_{3,3}}{2}\big(P_{2,1} + \frac{P_{2,2}}{2} + P_{2,3} \ldots + P_{2,G}\big)\big) + \ldots \\
&\ldots + \frac{P_{G-1,G-1}}{2}\Big(P_{G-2,3}\frac{P_{2,2}}{2} + P_{G-2,4}\big(P_{3,2} + \frac{P_{3,3}}{2}\big(P_{2,1} + \frac{P_{2,2}}{2} + P_{2,3} \ldots + P_{2,G}\big)\big) \ldots\Big) \\
&+ P_G\Big(P_{G,3}\frac{P_{2,2}}{2} + P_{G,4}\big(P_{3,2} + \frac{P_{3,3}}{2}\big(P_{2,1} + \frac{P_{2,2}}{2} + P_{2,3} \ldots + P_{2,G}\big)\big) + \ldots \\
&\ldots + \frac{P_{G,G}}{2}\Big(P_{G-1,3}\frac{P_{2,2}}{2} + P_{G-1,4}\big(P_{3,2} + \frac{P_{3,3}}{2}\big(P_{2,1} + \frac{P_{2,2}}{2} + P_{2,3} \ldots + P_{2,G}\big)\big) \ldots\Big)\Big)
\end{aligned}
\tag{61}
$$

.....

$$
\begin{aligned}
P_R^G(G-2) &= \sum_{g=G-2}^{G} P_g\Big(\sum_{k=G-2}^{g} (\frac{1}{2})^{u(g+1-k)u(k-g+1)} P_{g,k} P_r(G-2,k)\Big) \\
&= P_{G-2}\Big(\frac{P_{G-2,G-2}}{2}\frac{P_{G-3,G-3}}{2}\cdots\frac{P_{3,3}}{2}\frac{P_{2,2}}{2}\Big) \\
&+ P_{G-1}\Big(P_{G-1,G-2}\frac{P_{G-3,G-3}}{2}\cdots\frac{P_{3,3}}{2}\frac{P_{2,2}}{2} \\
&+ \frac{P_{G-1,G-1}}{2}\Big(P_{G-2,G-3}\frac{P_{G-4,G-4}}{2}\cdots\frac{P_{3,3}}{2}\frac{P_{2,2}}{2} + \frac{P_{G-2,G-2}}{2}P_{G-3,G-4}\frac{P_{G-5,G-5}}{2}\cdots\frac{P_{3,3}}{2}\frac{P_{2,2}}{2}\cdots\Big) \\
&+ P_G\Big(P_{G,G-2}\frac{P_{G-3,G-3}}{2}\cdots\frac{P_{3,3}}{2}\frac{P_{2,2}}{2} + P_{G,G-1}P_{G-2,G-3}\frac{P_{G-4,G-4}}{2}\cdots\frac{P_{3,3}}{2}\frac{P_{2,2}}{2}\cdots\Big)
\end{aligned}
\tag{62}
$$

$$
\begin{aligned}
P_R^G(G-1) &= \sum_{g=G-1}^{G} P_g\Big(\sum_{k=G-1}^{g} (\frac{1}{2})^{u(g+1-k)u(k-g+1)} P_{g,k} P_r(G-1,k)\Big) \\
&= P_{G-1}\Big(\frac{P_{G-1,G-1}}{2}\frac{P_{G-2,G-2}}{2}\cdots\frac{P_{3,3}}{2}\frac{P_{2,2}}{2}\Big) \\
&+ P_G\Big(P_{G,G-1}\frac{P_{G-2,G-2}}{2}\cdots\frac{P_{3,3}}{2}\frac{P_{2,2}}{2} + \frac{P_{G,G}}{2}P_{G-1,G-2}\frac{P_{G-3,G-3}}{2}\cdots\frac{P_{3,3}}{2}\frac{P_{2,2}}{2} \\
&\cdots + \frac{P_{G,G}}{2}\frac{P_{G-1,G-1}}{2}\cdots P_{3,2}\frac{P_{2,2}}{2} + \frac{P_{G,G}}{2}\frac{P_{G-1,G-1}}{2}\cdots P_{4,3}\frac{P_{2,2}}{2} + \\
&\frac{P_{G,G}}{2}\frac{P_{G-1,G-1}}{2}\cdots\frac{P_{3,3}}{2}P_{2,1}\Big)
\end{aligned}
\tag{63}
$$

$$
\begin{aligned}
P_R^G(G) &= \sum_{g=G}^{G} P_g\Big(\sum_{k=G}^{g} (\frac{1}{2})^{u(g+1-k)u(k-g+1)} P_{g,k} P_r(G,k)\Big) \\
&= P_G\frac{P_{G,G}}{2}\frac{P_{G-1,G-1}}{2}\frac{P_{G-2,G-2}}{2}\cdots\frac{P_{3,3}}{2}\frac{P_{2,2}}{2}
\end{aligned}
\tag{64}
$$

(b) We combine items with reference to P_g

$$\begin{aligned}
\sum_{i=1}^{G} P_R^G(i) = P_1 + P_2(P_{2,1} + \frac{P_{2,2}}{2} + P_{2,3} + \ldots + P_{2,G} + \frac{P_{2,2}}{2}) + \\
+ P_3(P_{3,1} + \frac{P_{3,3}}{2} + P_{3,4} + \ldots + P_{3,G} + \\
+ P_{3,2} + \frac{P_{3,3}}{2}(P_{2,1} + \frac{P_{2,2}}{2} + P_{2,3} + \ldots + P_{2,G}) + \\
+ \frac{P_{3,3}}{2}\frac{P_{2,2}}{2}) \\
+ P_4(P_{4,1} + \frac{P_{4,4}}{2} + P_{4,5} + \ldots + P_{4,G} + \\
+ P_{4,2} + \frac{P_{4,4}}{2}(P_{3,1} + \frac{P_{3,3}}{2} + P_{3,4} + \ldots + P_{3,G}) + \\
+ P_{4,3} + \frac{P_{4,4}}{2}(P_{3,2} + \frac{P_{3,3}}{2}(P_{2,1} + \frac{P_{2,2}}{2} + P_{2,3} + \ldots + P_{2,G})) + \\
+ \frac{P_{4,4}}{2}\frac{P_{3,3}}{2}\frac{P_{2,2}}{2}) \\
\ldots \\
+ P_{G-2}\sum_{k=1}^{G} P_{G-2,k} + P_{G-1}\sum_{k=1}^{G} P_{G-1,k} + P_G\sum_{k=1}^{G} P_{,G,k}
\end{aligned} \tag{65}$$

In terms of *Lemma 2*,*Lemma 1*, and*Assumption 3*,

$$\sum_{i=1}^{G} P_R^G(i) = 1.\square \tag{66}$$

Appendix V Theorem 3 Proof

Proof:

1) For $G < 1$, the *Lemma* is not significant, for $i > G$, the probability $P^G(i)$ is zeros, and for $i < 1$, the probability $P^G(i)$ is also zero due to *Assumption 3*, that's

$$P^G(i) = 0, \quad G < 1, \; G < i \; or \; i < 1 \; .$$

So, the *Lemma* becomes

$$\sum_{i=0}^{\infty} P^G(i) = \sum_{i=1}^{G} P^G(i) = 1 \; .$$

2) Consider the case $G = K$ $(K \geq 0)$. By expending the left side of (22), it becomes

$$\sum_{i=1}^{G} P^G(i) \overset{G=K}{=} \sum_{i=1}^{K} P^K(i) = P^K(1) + P^K(2) + P^K(3) \ldots + P^K(K-1) + P^K(K)$$

$$= \sum_{g=1}^{K} P_g P_{g,1} P_p^K(0,g) + \sum_{g=1}^{K} P_g [P_{g,1} P_p^K(1,g) + \sum_{k=2}^{g} P_{g,k} P_r(1,k-1) P_p^K(0,g)^{u(K-g)}]$$

$$+ \sum_{g=1}^{K} P_g [P_{g,1} P_p^K(2,g) + \sum_{m=1}^{2} u(g-m) \sum_{k=m+1}^{g} P_{g,k} P_r(m,k-1) P_p^K(2-m,g)^{u(K-g)}]$$

.....

$$+ \sum_{g=1}^{K} P_g [P_{g,1} P_p^K(K-2,g) + \sum_{m=1}^{K-2} u(g-m) \sum_{k=m+1}^{g} P_{g,k} P_r(m,k-1) P_p^K(K-2-m,g)^{u(K-g)}]$$

$$+ \sum_{g=1}^{K} P_g [P_{g,1} P_p^K(K-1,g) + \sum_{m=1}^{K-1} u(g-m) \sum_{k=m+1}^{g} P_{g,k} P_r(m,k-1) P_p^K(K-1-m,g)^{u(K-g)}]$$

$$= P_1 P_{1,1} P_p^K(0,1) + P_2 P_{2,1} P_p^K(0,2) + P_3 P_{3,1} P_p^K(0,3) \ldots + P_K P_{K,1} P_p^K(0,K)^{u(K-K)}$$

$$+ P_1 P_{1,1} P_p^K(1,1) + P_2 P_{2,1} P_p^K(1,2) + P_3 P_{3,1} P_p^K(1,3) \ldots + P_{K-1} P_{K-1,1} P_p^K(1,K-1)$$

$$+ P_2 P_{2,2} P_r(1,1) P_p^K(0,2) + P_3[P_{3,2} P_r(1,1) + P_{3,3} P_r(1,2)] P_p^K(0,3)$$

$$\ldots + P_K[P_{K,2} P_r(1,1) + P_{K,3} P_r(1,2) + P_{K,4} P_r(1,3) \ldots + P_{K,K} P_r(1,K-1)] P_p^K(0,K)^{u(0)}$$

$$+ P_1 P_{1,1} P_p^K(2,1) + P_2 P_{2,1} P_p^K(2,2) + P_3 P_{3,1} P_p^K(2,3) \ldots + P_{K-2} P_{K-2,1} P_p^K(2,K-2)$$

$$+ P_2 P_{2,2} P_r(1,1) P_p^K(1,2) + P_3[P_{3,2} P_r(1,1) + P_{3,3} P_r(1,2)] P_p^K(1,3)$$

$$\ldots + P_{K-1}[P_{K-1,2} P_r(1,1) + P_{K-1,3} P_r(1,2) + \ldots + P_{K-1,K-1} P_r(1,K-1)] P_p^K(1,K-1)$$

$$+ P_3 P_{3,3} P_r(2,2) P_p^K(0,3) + P_4[P_{4,3} P_r(2,2) + P_{4,4} P_r(2,3)] P_p^K(0,4)$$

$$\ldots + P_K[P_{K,3} P_r(2,2) + P_{K,4} P_r(2,3) + \ldots + P_{K,K} P_r(2,K-2)] P_p^K(0,K)^{u(0)}$$

$$\ddots \qquad \ddots$$

$$+ P_1 P_{1,1} P_p^K(K-1,1) + P_2 P_{2,2} P_r(1,1) P_p^K(K-2,2) + P_3 P_{3,3} P_r(2,2) P_p^K(K-3,3)$$

$$\ldots + P_{K-1} P_{K-1,K-1} P_r(K-2,K-2) P_p^K(1,K-1) + P_K P_{K,K} P_r(K-1,K-1) P_p^K(0,K)^{u(0)}$$

and then combine the corresponding items in the expansions above with reference to

P_g, we readily have

$$\sum_{i=1}^{G} P^G(i) \overset{G\equiv K}{=} \sum_{i=1}^{K} P^K(i)$$

$$= P_1 P_{1,1}[P_p^K(0,1) + P_p^K(1,1) + P_p^K(2,1) + P_p^K(3,1) \ldots + P_p^K(K-1,1)]$$

$$+ P_2[P_{2,1} + P_{2,2}P_r(1,1)][P_p^K(0,2) + P_p^K(1,2) + P_p^K(2,2) \ldots + P_p^K(K-2,2)]$$

$$+ P_3[P_{3,1} + P_{3,2}P_r(1,1) + P_{3,3}(P_r(1,2) + P_r(2,2))][P_p^K(0,3) + P_p^K(1,3) + P_p^K(2,3) \ldots + P_p^K(K-3,3)]$$

$$\ddots \quad \ddots$$

$$+ P_{K-1}[P_{K-1,1} + P_{K-1,2}P_r(1,1) + P_{K-1,3}(P_r(1,2) + P_r(2,2))$$

$$\ldots + P_{K-1,K-1}(P_r(1,K-2) + P_r(2,K-2) \ldots + P_r(K-2,K-2))][P_p^K(0,K-1) + P_p^K(1,K-1)]$$

$$+ P_K[P_{K,1} + P_{K,2}P_r(1,1) + P_{K,3}(P_r(1,2) + P_r(2,2))$$

$$\ldots + P_{K,K}(P_r(1,K-1) + P_r(2,K-1) \ldots + P_r(K-2,K-1) + P_r(K-1,K-1))]P_p^K(0,K)^{u(0)}$$

$$= P_1 P_{1,1} \sum_{p=0}^{K-1} P_p^K(p,1) + P_2 \sum_{k=1}^{2} P_{2,k} \sum_{r=1}^{k-1} P_r(r,k-1) \sum_{p=0}^{K-2} P_p^K(p,2) + P_3 \sum_{k=1}^{3} P_{3,k} \sum_{r=1}^{k-1} P_r(r,k-1) \sum_{p=0}^{K-3} P_p^K(p,3)$$

$$\ldots + P_{K-1} \sum_{k=1}^{K-1} P_{K-1,k} \sum_{r=1}^{k-1} P_r(r,k-1) \sum_{p=0}^{1} P_p^K(p,K-1) + P_K \sum_{k=1}^{K} P_{K,k} \sum_{r=1}^{k-1} P_r(r,k-1) \sum_{p=0}^{0} P_p^K(p,K)]P_p^K(0,K)^{u(0)}$$

and next, repeatedly using *Lemma 4*, *Lemma 3*, *Lemma 2*, and *Lemma 1*, we obtain

$$\sum_{i=1}^{K} P^K(i) = P_1 + P_2 + \ldots + P_{K-1} + P_K = \sum_{g=1}^{K} P_g = 1.$$

Since the K may be an arbitrary positive integer, we conclude that

$$\forall (G,i | (G,i) \in I^+, i \le G),\ \exists \sum_{i=0}^{\infty} P^G(i) = \sum_{i=1}^{G} P^G(i) = 1. \square$$

Appendix VI Theorem 8 Proof

Proof: As known, a non-grouping *DB-MAC* system is actually an one-grouping system from the viewpoint of *GT*. The theorem, therefore, means that $\overline{x}(G \mid G > 1) > \overline{x}(1)$.

In terms of *Theorem 1*, *Theorem 2*, and *Theorem 3*, when G=1,

$$\sum_{i=0}^{\infty} P_x^1(i) = \sum_{i=1}^{1} P_x^1(i) = P_x^1(1) = 1,$$

where $x = R, PorNone$,

$$\overline{x}(1) = \sum_{i=0}^{\infty} iP_x^1(i) = \sum_{i=1}^{1} iP_x^1(i) = P_x^1(1) = 1.$$

Note

$$\overline{x}(1) = 1 = \sum_{i=0}^{\infty} P_x^G(i) = \sum_{i=1}^{G} P_x^G(i), \quad for\ any\ G > 1.$$

and

$$\begin{aligned} \forall(G > 1), \exists\overline{x}(G) - \overline{x}(1) &= \sum_{i=1}^{G} iP_x^G(i) - \sum_{i=1}^{G} P_x^G(i) \\ &= \sum_{i=1}^{G} (i-1)P_x^G(i) = \sum_{i=2}^{G} (i-1)P_x^G(i) > 0. \end{aligned}$$

since $P_x^G(i) > 0$ when $1 \leq i \leq G$ under full load. Therefore, we obtain

$$\overline{x}(G \mid G > 1) > \overline{x}(1).\square$$

Bibliography

[1] T. N. Saadawi and M. H. Ammar with A. E. Hakeem. "Fundamentals of Telecommunication Networks". *John Wiley & Sons* , New York, New York, pp.299-309, 1994.

[2] James F. Mollenauer. "Standards for Metropolitan Area Networks". *IEEE Communications Magazine,* vol.26, no.4., pp.15-19, April 1988.

[3] ISO/IEC 8802-6 ANSI/IEEE std 802.6 Working Group. "Distributed Queue Dual Bus (DQDB) Access Method and Physical Layer Specifications". *The IEEE,Inc.* First Edition, Piscataway, New Jersey, Mar. 7, 1994.

[4] Greg C. Watson and Samir Tohme. "S++ – A New MAC Protocol for Gb/s Local Area Networks". *IEEE Journal On Selected Areas In Communications*,vol.11 no.4, pp.531-539, May 1993.

[5] M.M.Nassehi. "Cyclic-Reservation Multiple-Access Scheme for Gbit/s LANs and MANs based on Dual Bus Configuration". The 8th European Fibre Optical Communication's and Local Area Network's Conference, Munich, Federal Republic of Germany, pp.246-251, June 1990.

[6] C.Baransel and W.Dobosiewicz. "CBRMA (Cyclic Balanced Reservation Multiple Access) MAC Protocol". in *Proc.Sixth Int. Symp. Comput. and Inform, Sci.*, Antalya, Turkey, pp.537-545, Oct. 1991.

[7] Mark W. Garrett and San-Qi Li. "A Study Of Slot Reuse In Dual Bus Multiple Access Networks.". *IEEE Journal Selected Areas on Communications*, vol.9, no.2, pp.248-256., Feb. 1991.

[8] B. Mukherjee and S. Banerjee. "Incorporating Continuation-of-message(COM) Information, Slot Reuse, And Fairness In DQDB Networks". Tech. Rep. CSE-90-42, Div. Comput. Sci., Univ. California, Davis, CA, Oct. 1990.

[9] A.E.Kamal. "Efficient Multi-segment Message Transmission With Slot Reuse On DQDB". in *Proc. INFOCOM*, pp. 869-878., 1991.

[10]M.A.Rodriques. "Erasure Node: Performance Improvements For The IEEE 802.6 MAN". in *Proc. INFOCOM'90*, San Francisco, pp.636-643, June 1990.

[11]T. Yokotani, H. Sato, and S. Nakatsuka. "Proposed Erasure Nodes Algorithm In DQDB. Mitsubishi Electric Corp., Commun. Syst. Dev. Lab., Japan.

[12]Oran Sharon and Adrian Segall. "A Simple Scheme For Slot Reuse Without Latency For A Dual Bus Configuration". *IEEE/ACM Transactions On Networking*, vol.1, no.1., pp.96-104, Feb. 1993.

[13]Andrzej R. Pach, Sergio Palazzo. "Slot Pre-Using In IEEE 802.6 Metropolitan Area Networks". *IEEE Journal on Selected Areas in Communications*, vol.11, no.8., pp.1249-1258, Oct. 1993.

[14]I.Cidon and Y.Ofek. "Metaring-A Full Duplex Ring With Fairness And Spatial Reuse". in *Proc.IEEE INFOCOM'90*, San Francisco, CA, pp.969-981, June 1990.

[15]Oran Sharon and Adrian Segall. "On the Efficiency of Slot Reuse in the Dual Bus Configuration". *IEEE/ACM Transactions On Networking*, vol.2, no.1., pp.89-100, Feb. 1994.

[16]Dionysios Karvelas, Michail Papamichail. "The No Slot Wasting Bandwidth Balancing Mechanism For Dual Bus Architectures". *IEEE Journal on Selected Areas in Communications*, vol.11, no.8., pp.1214-1228, Oct. 1993.