CBRMA++/SR : On the Design of a MAN/WAN MAC Protocol for High–Speed Networks

Cesur Baransel Włodek Dobosiewicz Pawel Gburzynski

Department of Computing Science,
University of Alberta,
Edmonton, Alberta, Canada T6G 2H1
{cesur,dobo,pawel}cs.ualberta.ca

Abstract

In this paper, a new MAN/WAN MAC*-level protocol called* CBRMA++/SR *will be introduced. It is suitable for folded bus topology with optical fiber used as the transmission medium.*

The eminent features of CBRMA++/SR *are total fairness and full bandwidth utilization. The preemptive approach employed in bandwidth allocation enables* CBRMA++/SR *not to suffer from the deficiencies of* DQDB *which mostly stem from the inconsistency of the distributed queue at a given instant. The protocol is based on the concept of* **fair share** *which also provides a solid ground to support connection-oriented services. A powerful slot-reuse scheme complements the protocol.*

1 Introduction

The design of the MAC–level protocol is the most crucial phase in any network design since the decisions made at this level will determine the major functional characteristics of the network and set the upper limits of its capabilities in general. Since the activities of the upper layers will be more and more carried out by software as one goes upward in the network protocol hierarchy, any attempt to provide a functionality that counteracts these basic characteristics will be more and more costly, eventually bordering the infeasibiltiy. Especially with the advent of optical fiber as a transmission medium, more strict constraints on message/packet/slot processing times are imposed at the intermediate nodes thereby providing a new challange to MAC–level protocol designers.

In this paper, we consider optical fiber as the basic transmission medium and 1 Gbps or more as the basic transmission rate. Consequently, a MAC–level protocol designed to operate in this environment must satisfy certain requirements:

1. It must be simple enough to be implemented directly in hardware, so that it can exploit the bandwidth capacity offered by optical fiber.

2. It must be fair, i.e. the throughput of a station must be independent of its location within the network. Furthermore, a bursty station should not be served in detriment of the others and the protocol should not allow a station to usurp the available bandwidth capacity inadvertently. This does not mean that the bursty stations should be assigned to a lower priority since every station is entitled to use a reasonable portion of the available bandwidth. However, a bursty station should not be granted with additional bandwidth which is stolen from the moderately or lightly loaded users. Also, in the presence multiple bursty stations, available idle bandwidth should be distributed among them evenly.

3. The throughput of the protocol must be independent of the network's size and transmission rate. This item is related to well-known a-parameter. Any MAC protocol which is sensitive to this parameter limits its usefulness to networks up to a certain size and rate.

4. It must be flexible enough to satisfy hetereogenous traffic demands, such as high-priority, syn-

chronous, asynchronous traffic.

5. It must be predictable since the presence of unpredictability requires more resources to be allocated at the stations by the users to be able to cope with the unexpected and increases the complexity of the protocol.

6. The overall structure should be robust and therefore the malfunction of a switching element or a severed link should not take down the network with it. In other words, the effect of the malfunctions should be kept as local as possible and/or the network should be able to "heal" itself so that the integrity of the network is preserved.

7. In the presence of rich connectivity, the protocol should be able to make an effective use of the alternative routes. Therefore, the means of providing flow control and load distribution — at least the formation of an infrastructure to facilitate the provision of these services by upper layer protocols, should be considered.

2 Alternative Approaches

By definition, MAC protocols are required when a single carrier is shared by multiple nodes. In other words, it is essentially an arbitration mechanism that is responsible the resolution of the contentions by the nodes for a shared transmission medium. Designing a MAC protocol requires the information regarding the topology of the network, the type of the carrier and the signalling mechanism. Some important implications of these factors can be stated with respect to MAC protocol design: different topologies are suitable for different networks, in terms of geographical size and the number of the nodes involved. They also have different degrees of connectivity and some are designed with special routing schemes in mind. The type of the carrier is also crucial since they have different transmission rates and different repeaterless transmission range. Furthermore, the transmission on optical fiber is inherently unidirectional whereas for copper links this is not the case. As for the signalling mechanism, the major implication is the presence of a network that is based on baseband transmission versus broadband transmission. In case of the latter, MAC protocol should be able to handle multiple channels.

The type of the applications that the network is supposed to support is also important. If a certain traffic pattern and load can be stated, the MAC protocol can also be tailored accordingly to provide a better performance.

The previous remarks are included to clearly define the environment for which our protocol is designed. Furthermore, the obvious approach to demonstrate the "betterness" of a new protocol is to compare it with the ones in existence. In [12], we argued that such a comparison is only meaningful if a particular topology is chosen[1]. The bottomline of the argument presented in that paper is that *<topology,protocol>* pairs are the meaningful members of the universal set of the MAC protocols, not the protocols alone. To do otherwise is just another attempt to compare the proverbial apples with oranges. By the same token, we will compare our protocol with the recently established IEEE standard, DQDB protocol for the following reasons: both protocols are designed for high–speed MAN/WAN MAC networks; the set of the possible applications has a wide range and not restricted to data transmission; the transmission discipline is that of slotted bus and the tasks performed by the so-called headend stations are similar.

Before going into the details of our protocol, first we will evaluate DQDB according to the aforementioned criteria. However, it must be noted that the requirements 6 and 7 are excluded from these discussions. The reason for this is quite obvious since on a bus topology the issues related to multiple paths between station pairs are meaningless and a severed link effectively divides the network into two, shattering the integrity of the network. Furthermore, it is not our intention in this paper to address the problems regarding the handling of two or more smaller networks after such an incident.

3 DQDB

The *Distributed-Queue-Dual-Bus* (DQDB) medium access control (MAC) level protocol has been accepted as the IEEE 802.6 standard for high-speed (Gbps range) metropolitan area networks (MANs). DQDB is extensively defined in [1], so no attempt will be made to describe it here. Although this protocol has properties which make it appealing for many high-speed network applications[2], it satisfies condition 3 and (arguably) 1 , but fails to satisfy the other conditions:

[1] Topology in this context implies networks with identical physical layer characteristics.

[2] Such as being able to sustain an aggregate transmission rate near the total bus capacity independently of the network's size and transmission rate.

1. DQDB violates condition 2, at least to some extent[3]. The BWB (Bandwidth Balancing) mechanism, which has been introduced as a countermeasure, is proven to be effective but only after a rather long transition time [4,5], a fact that renders it almost useless in case of short-lived bursts. DQDB also permits a single heavy-user to dominate the overall throughput, consequently increasing the access delays of the low-traffic users significantly. [4,5,6].

2. DQDB does not satisfy condition 4. DQDB makes ineffective priority assignments; it is observed that the priority mechanism is not effective under heavy load when the low-priority traffic generating nodes lie between the headend and the high-priority node [6]. Furthermore, the protocol may even exhibit inverse priority behavior [4,6].

3. DQDB violates condition 5, since the status of the distributed queue is globally inconsistent at any given instant [2].

Numerous variations of DQDB protocol have been proposed, some of them satisfying conditions 2 and 4 (e.g. [16]). We will not discuss these variations here restricting ourselves to the standard DQDB.

4 CBRMA

Our aim is to design a scheme which can satisfy the perfomance requirements stated in section 1. In the rest of the paper, we will argue that *Cyclic-Balanced Reservation Multiple Access ++ with Slot Reuse* (CBRMA++/SR) MAC protocol has the necessary qualities to be so. The design of the protocol is a three-step process each being the content of a previous paper [7,8,9]. As the first step towards the aforementioned goal, we introduced CBRMA [7], which we claim to satisfy the basic requirements and then improved it twice [8,9]. The first improvement involves getting the throughput of dual bus topology on a folded bus, therefore, the elimination of the second headend station in the configuration. In [8], we explained how this can be done and named the second variation of our protocol CBRMA++. Later we augmented CBRMA++ with an efficient slot reuse mechanism and in [9] introduced CBRMA++/SR. The slot mechanism is unique in a way, since beside slot reuse, it also provides the upper layer protocols with helpful information about future incoming traffic profile. In the remainder of this section, first, we will state the important differences between our protocol and CRMA – a protocol which is previously introduced by IBM, to prevent a possible confusion. Later, a basic description of the protocol along with the code of the basic algorithms, an example cycle and some simulation results will be provided. The next section is devoted to the discussion related to the unnecessity of the second headend station and the details of the slot reuse mechanism.

4.1 Comparison with CRMA

Although the name of the protocol is somewhat akin to *Cyclic Reservation Multiple Access* (CRMA) protocol developed at IBM, only the idea of cyclic reservation is similar and even it is handled in a different way. CRMA requires more complex structures both at the nodes and at the headend station. The reservations made by the nodes are not certain and need to be confirmed by the headend to be valid. They can be rejected and in such a case a retry is necessary by the nodes[4]. Consequently, nodes have to maintain three different message queues, namely *Confirmed Reservation Queue* (CRQ), *Tentative Reservation Queue* (TRQ) and *Entry/Reentry Queue* (ERQ). The headend station also maintains two queues called *Global Reservation Queue* (GRQ) and *Elasticity Buffer* (EB), which is also a queue despite its name. Multiple reservation slots co-exist on the bus at a given instant and the number of control slots is six (*Reserve, Confirm, Start, Reject, unused and Noop*). Since a high generation rate of reserve slots can result in considerable access delays, a backpressure mechanism is developed in order to reduce the worst-case access delay. The details of this protocol can be found in [3,13,14,15].

Our protocol differs from CRMA in a number of ways: firstly, there can be only one reservation slot on the bus at a given instant and it is read/updated by a node twice, once on the forward bus and once on the backward bus. There is no confirm and reject slots, therefore no backpressure mechanism either. Consequently, message queues are simpler at the nodes and the headend station acts only as a slotter and maintains no queues. The reservations are final and are not subject to the further approval of the headend, removing the decision-making responsibility from headend and distributing it amongst the nodes. Every station

[3] When oversaturated DQDB is patently unfair. At more realistic traffic loads, the significance of the variation in the throughput rates of the transport layer protocol due to the location of the station becomes arguable.

[4] A reservation slot contains the reservations made by a number of nodes and when a reject slot is issued by the headend station the outcome is the invalidation of the reservation slots that are present on the bus altogether.

can get assertive information (not a close approximation) about the global traffic profile by simply inspecting the contents of the reservation slot. The information at the disposal of the MAC and upper layer protocols cover the traffic load and available bandwidth for the next cycle with utmost certainty. The coherency of the basic mechanism makes a powerful slot reuse mechanism possible — a mechanism that can not be applied to CRMA directly because of the uncertainties involved in the reservation process.

4.2 Basic Algorithms and Data Structures used by CBRMA

CBRMA is a slot reservation scheme suitable for both the folded bus and the dual bus topologies. The protocol will be explained for the folded bus configuration, since its counterpart on the double bus requires only the duplication of the data structures and the processes or the finite state machines in which these processes are hardwired along with some minor additions both at the headends and the nodes. That issue will be addressed at the end of the subsection.

In CBRMA, the bandwidth allocation process is organized in cycles. At the beginning of every cycle, each station is granted the same number of slots. This default number is called **fair share**. A **reservation cycle** begins at the headend with the headend station issuing a *reservation slot* which passes by each station twice, once on the forward bus and once on the backward bus (figure 1).

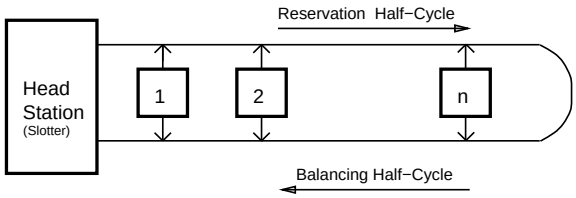

Figure 1: Half-Cycles of CBRMA

Since the traffic loads of stations may vary, a station can either be contented with its fair share (or a part of it) or ask for more. If a station uses only a portion of its fair share, the remaining slots are recorded as *unused*. These are gathered together and distributed evenly among users with heavy slot demands. The distribution of unused slots is done while the reservation slot is passing by the stations on the backward bus (**balancing cycle**).

A reservation slot has five subfields:

1. Cycle number (C)
2. Number of slots reserved so far (R)
3. Number of stations that need more slots than their fair share (ES)
4. Total number of extra slots requested by ES stations so far (ER)
5. Total number of slots that left unused by lightly-loaded users (U)

At the nodes, the necessary data structure is composed of a single message queue and four counter variables named *CycleStartCounter, XMitCounter, QueueSize and ExtraRequestCounter*. CycleStartCounter is used as a countdown counter and indicates the number of slots that the node let pass by before transmitting. XMitCounter contains the number of slots that the node will use in the next cycle. QueueSize is the number of packets backlogged at the node. ExtraRequestCounter is set to the number of slots that the node can use if given the chance.

4.3 Reservation Algorithm

Upon detecting the reservation slot on the forward bus, a node executes the following algorithm. In the algorithm, the fields that belong to the reservation slot are marked with the prefix *Slot->* to distinguish them from the local variables. As can be seen, there are three *if-statements* that correspond to the three possible cases: if the node needs more than its fair share, it claims its fair share and then requests more to transmit the rest of the queue by incrementing the ER field of the slot. If its current need is exactly equal to the fairshare, it simply claims it and sets the local counters to appropriate values. If the node needs less than its fair share, it marks the remainind slots as unused by incrementing U field of the slot accordingly.

• RESERVATION ALGORITHM.

```
ExtraRequestCounter = 0 ;
CycleStartCounter = Slot->R + 1 ;

if (QueueSize > FairShare) {
    XMitCounter = FairShare ;
    ExtraRequestCounter = QueueSize - FairShare ;
    Slot->ES ++ ;
    Slot->ER += ExtraRequestCounter ;
    Slot->R  += FairShare ;
    exit() ;
}

if (QueueSize == FairShare) {
    XMitCounter = FairShare ;
```

```
    Slot->R  += FairShare ;
    exit() ;
}

if (QueueSize < FairShare) {
    XMitCounter = QueueSize ;
    Slot->R += QueueSize ;
    Slot->U += (FairShare - XMitCounter) ;
    exit() ;
}
```

4.4 Balancing Algorithm

As the reservation slot turns the fold and starts to propagate on the reverse bus, the balancing cycle begins. The reservation slot is handled by the nodes according to the following algorithm.

•BALANCING ALGORITHM.

```
if (Slot->U == 0) exit() ;

if (ExtraRequestCounter == 0) {
    CycleStartCounter += Slot->U ;
    QueueSize -= XMitCounter ;
    exit() ;
}

if (Slot->U >= Slot->ER) {
    Slot->U -= ExtraRequestCounter ;
    Slot->R += ExtraRequestCounter ;
    Slot->ES -- ;
    Slot->ER -= ExtraRequestCounter ;
    CycleStartCounter += Slot->U ;
    XMitCounter  += ExtraRequestCounter ;
    QueueSize -= XMitCounter ;
    exit() ;
}

else {
    int othersneed, difference, fair, share ;

    othersneed = Slot->ER - ExtraRequestCounter ;
    fair = floor (Slot->U / Slot->ES) ;
    difference = Slot->U - othersneed ;

    if (difference > fair)
         share = difference ;
    else share = fair ;

    if (share > ExtraRequestCounter)
        share = ExtraRequestCounter ;

    Slot->U -= share ;
    Slot->R += share ;

    XMitCounter += share ;
    CycleStartCounter += Slot->U ;

    Slot->ES -- ;
    Slot->ER -= ExtraRequestCounter ;
    QueueSize -= XMitCounter ;
    exit() ;
}
```

As can be seen, in the absence of any unused slot, there is nothing to do, since the balancing algorithm deals with the fair distribution of the unused slots amongst users operating under heavy traffic load. Otherwise, the nodes that are not requesting extra slots simply adjust their counters to ensure the slot contiguity, by executig the body of the second if statement. The third or fourth if is to be executed by the node that placed a request for extra slots in the reservation phase. The third if deals with the case where the number of unused slots is greater than or equal to the requested amount. In this case there is no problem and all requests can be fully met. Otherwise, there are calculations to perform to ensure fairness[5].

By the basic definition of the protocol, the following points are clear: firstly, a station must make a reservation before starting to transmit. Therefore, even under light traffic conditions zero medium access delay is not possible. Secondly, since it is not possible to perform arithmetic operations on a field of reservation slot without storing it at least partially, a delay is inevitable. The amount of the delay is totally dependent upon the speed of the of the arithmetic circuitry at the disposal of the node, but, it is clear that the whole traffic on the bus will be delayed by the same amount. In the simulations, we used a shorter bus length for DQDB since the delay can be regarded as increasing the distance betweeen station pairs. For this reason, the bus length is shortened by 6850 bits (from 68250 to 61400) giving approximately 214-bit transmission time delay per station (see section 4.6).

As for the operation on the dual bus, two minor additions are necessary to the protocol: in order to complete the reservation cycle, a headend station should repeat the reservation slot issued by the other headend station on the outgoing bus. Therefore a mechanism is necessary at the headends to differentiate between its own reservation slot and the reservation slot of the other headend — a problem that can be solved at the expense of a single bit in the reservation slot structure. The other addition involves a decision mecha-

[5] In which a division operation is necessary. In the absence of a fast division circuitry, a simple look-up table that can be accessed on *(how many slots are available, how many stations are contending for them)* basis can be used to alleviate this problem.

nism at the nodes which is necessary to select one of the available two buses since the nodes can transmit on both of them. In simulations, nodes use the forward bus for transmitting to downstream nodes and the backward bus for transmitting to upstream nodes, consequently minimizing the distance between station pairs and yielding a higher throughput.

4.5 An Example Cycle

For a certain network with 5 stations, a bus length of 30 slots and a fair share set to 6 slots per station, the contents of the reservation slot is depicted in figure 2, as it passes by each station, assuming that at the beginning of reservation cycle i, the following reservation requests are valid,

Station number	*The length of queue-F*	*Unused (-) Extra (+)*
1	8	2
2	2	-4
3	5	-1
4	7	1
5	8	2

The negative numbers in the last column denote the number slots that are left unused by a station out of its fairshare. The positive numbers indicates extra slot requests.

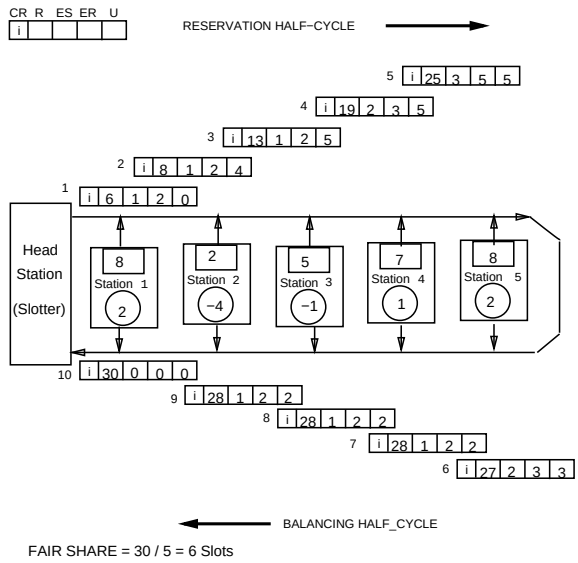

Figure 2: An example cycle

The appendix contains another example in which the available bandwidth and the slot requests are not aligned so nicely. This example also shows how forward and backward bus is controlled by a single head-end station. Therefore, the reader is encouraged to examine it after reading section 5.1.

4.6 Performance of CBRMA

In figure 3 and 4 some simulation results are provided. As the other simulations in this paper, they are obtained for a network which is composed of 32 stations placed in equidistant intervals on a folded or dual bus. The transmission rate is 1 Gbps and bus length is 68250 bits or 160 slots. Considering the 5 ns/m propogation rate of optical waveguides, this corresponds a bus length of roughly 13650 m. The fair share is set to 5 slots per cycle. Each slot is 416 bits long and on the transmission link they are seperated by 2-bit interslot gaps. Slots have 32-bit headers and 384-bit segment payloads. The throughput rates are calculated in terms of segment payload as opposed to slot length. Therefore, without slot reuse, the maximum throughput rate per bus is calculated to be

$$\tau = 384/(416+2) = 0.919$$

Stations operate under asymptotic conditions and every other station is equally likely to be addressed. Each message is exactly one slot long and therefore no bandwidth is wasted due to the internal fragmentation.

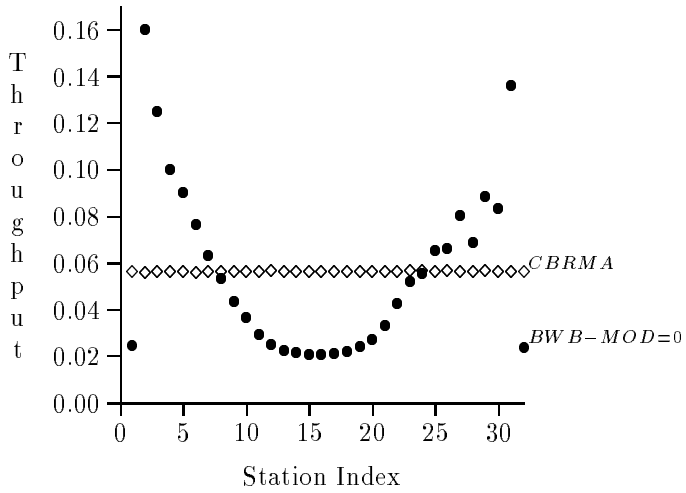

Figure 3: CBRMA vs. DQDB with BMW–MOD=0 on dual bus.

Figure 3 and 4 show the throughput of CBRMA and DQDB on a dual bus under asymptotic conditions. In figure 3, BWB-MOD value is 0 whereas it is set to 1 in figure 4. Note that the throughput rates shown in the

figures are the total of the two buses. Since S_1 and S_n can only use one of them, their throughput appear to be low. In fact, the throughput of S_n is much higher than that of any other station on the reverse bus. The same argument also applies to S_1 for the throughput on the forward bus.

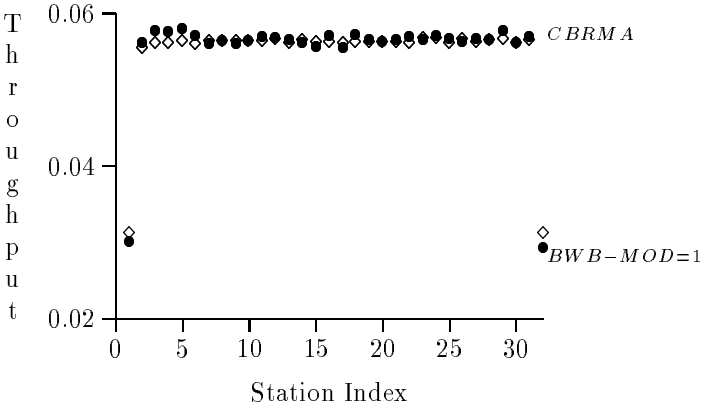

Figure 4: CBRMA vs. DQDB with BWB−MOD=1 on dual bus.

5 Improvements

5.1 CBRMA++: Elimination of the Second Headend Station

In a folded bus configuration, S_1 will use only forward bus for transmission since all the other stations are located downstream with respect to its position, and consequently are reachable on the forward bus. Therefore every slot filled by S_1 can be regarded as read and emptied on the forward bus by the time it reaches to the folding point and therefore, there is no reason for not to put them into use after they turn the fold. If the same condition can be guaranteed for the slots that are transmitted by the other stations as well, it becomes possible to use a single slot twice, first on the forward bus and later on the backward bus. This is the main idea behind the CBRMA++ protocol and it can also be regarded as a simple slot reuse mechanism. In the remainder of this section, we will explain how it can be done and provide the related simulation results.

A very simple mechanism is sufficient to quarantee the aforementioned condition: in CBRMA++, we assume that stations maintain two different message queues; one for the messages that are to be transmitted to downstream nodes (queue-F) and one for the messages that are destined to upstream nodes (queue-B). The messages that come from queue-F will be transmitted on the forward bus while the messages that reside in queue-B will use the backward bus. Therefore it is guaranteed that the slots transmitted on the forward bus will reach their destination before the fold. This being the case, it is necessary for a station to make seperate reservations for each bus. Therefore, we doubled the number of counter fields in the reservation slot and used one set for each bus exclusively. As a result it became possible for a single slot to carry two segments in a cycle. Therefore it is intuitively obvious that the global performance the network will be doubled under asymptotic conditions (i.e. when each station has as many messages as it can use in each queue). It is also obvious that the data structures and the processes are the same with that of what is necessary for CBRMA to operate on the double bus. One difference is that, there is no need for the second headend station.

On the other hand, under normal conditions we expect the number of messages in each queue to be different. To begin with, it is clear that S_1 will use only forward bus (since there is no upstream node to it) and S_n will use always backward bus for a similar reason. Also at S_2 queue-F will probably be longer than queue-B. However this does not degrade the performance of the protocol, since for each station that wants to use the forward bus more frequently, there will be another station looking for extra free slots on the backward bus. The reservation scheme is capable of effectively distributing the bus capacity amongst the stations, so that, for example, S_2 will be able to gain some more bandwidth on the forward bus in exchange of the slots it left unused out of its fairshare on backward bus.

In the remainder of this section we will present simulation results to show that the performance of CBRMA++ on a folded bus is equal to the performance of CBRMA and DQDB on a dual bus.

Figure 5 depicts the throughput of CBRMA++ vs. CBRMA and DQDB on folded bus under asymptotic conditions. As can be seen in the figure, the throughput rate of CBRMA++ is approximately twice as high when compared to that of CBRMA and DQDB. Actually it is equal to the throughput rates of these two schemes on dual bus. Figure 6 depicts the performance of CBRMA++ on folded bus vs. the performance of CBRMA and DQDB on dual bus.

Figure 6 illustrates also one additional mechanism of the protocol. Note that in this figure there is no irregularity regarding the throughput of S_1 and S_{32}. Remember that S_1 never uses its fair share on back-

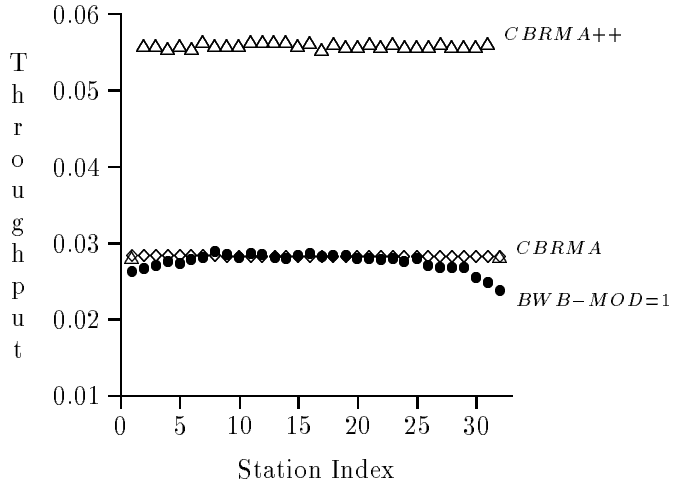

Figure 5: CBRMA++ vs. CBRMA and DQDB on folded bus.

ward bus and the same is true for S_{32} when its fair share on forward bus is concerned. We can transfer the fair share of S_{32} on forward bus to S_1 simply by doubling the fair share of S_1 and setting the fair share of S_{32} to zero. Carrying out the same operation for backward bus will produce the desired affect.

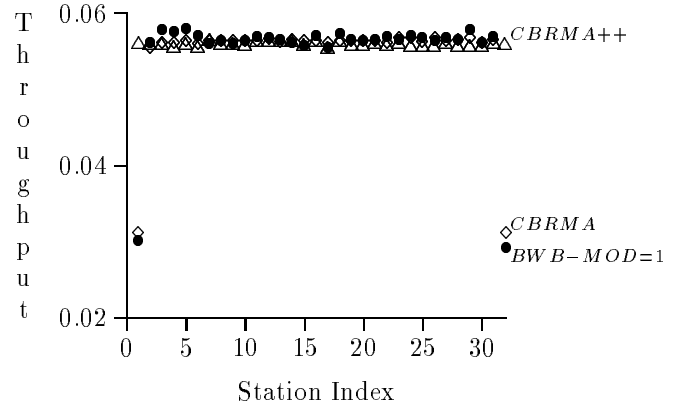

Figure 6: CBRMA++ on folded bus vs CBRMA and DQDB on dual bus.

This approach can be used whenever it is necessary to allocate excessive bandwidth to special stations, such as file servers.

5.2 CBRMA++/SR: An Efficient Slot Reuse Mechanism

A deficiency of DQDB, CRMA and CBRMA is that once a slot is filled by a segment will have to propagate to the end of the bus. The use of erasure nodes is suggested to overcome this problem in [9] for DQDB. In this section we will propose a method to serve the same purpose for CBRMA++.

Although a slot is used twice in CBRMA++, it is clear that the performance of the protocol can be improved further by using a slot more than once on forward and/or backward bus. The improvement in the throughput rate will be more pronounced as the number of stations increases. Figure 7 depicts a situation of interest for the following description of slot reuse (SR) mechanism. At this moment, the negotiations for the bandwidth allocation for $cycle_i$ was completed (i.e. reservation slot of $cycle_i$ has reached to headend station). Therefore, each station knows how many slots it will use and where to send them. It also has complete information regarding the number of backlogged packets and their destinations, in other words, the information related to its future slot request for $cycle_{i+1}$. However, it does not know how many slots it will receive in $cycle_i$. Providing that this information is also made available to the station, it will be able to use the slots that it will receive for transmission and to resize its request for $cycle_{i+1}$.

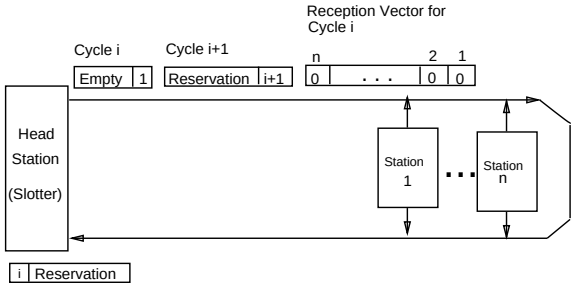

Figure 7: Slot Reuse Mechanism

The SR mechanism works as follows: as soon as the reservation slot reaches to the headend station, the headend station begins to transmit **Reception Vector** (RV) on the forward bus with its content is set to all zeroes. RV has a counter field for every station and therefore the overhead of SR is one counter field per station[6]. RV(i) contains the number of slots to be received by S_i in $cycle_i$. Consequently, S_1 sets the corresponding members of RV according to the receiver addresses of the packets scheduled for transmission in $cycle_i$. When RV arrives at S_2, it inspects RV(2) to

[6]Current slot length is the same as that of DQDB and current counter field length is 8 bits, so that a single slot contains 48 counter fields. If network contains more than 48 stations, additional slots are used to accommodate RV. The number of slots required by RV is equal to $\lceil n/48 \rceil$, where n denotes the number of stations in the network.

obtain the number of slots it will receive, increments the count of the slots scheduled for transmission by that number and increments the appropriate elements of RV accordingly. This operation will be repeated by every station until S_n is reached. S_n sets RV contents to all zeroes on the forward bus and on the backward bus the same operation is started anew for backward transmission. Note that RV is also reused and there is no need for a second RV.

Suppose that S_1 was granted 6 slots in $cycle_i$ and it will use them to transmit 3 slots to S_2, 1 slot to S_5 and 2 slots to S_8. To inform S_2, S_5 and S_8 about the number of slots they will receive, S_1 sets the second, fifth and eighth elements of the vector to 3, 1 and 2, respectively. The other stations will use increment operation.

When RV reaches S_2, it will be informed of the future transmission of 3 slots from S_1 in $cycle_i$. Assuming it is provided with 5 slots already during the previous balancing phase, the effective bandwidth allocated to S_2 becomes 8 slots for $cycle_i$. Using this information, S_2 is able to update RV for not only 5 but also for the additional 3 slots. Therefore, the 3 slots that will be carrying a message from S_1 in our current example can be reused not only S_2 but also by the downstream station that S_2 will be transmitting as well, since S_2 updated RV to convey this information to this particular destination station.

In the case that a station is not able to use all of these slots for transmission it passes the excessive slots to its immediate successor by simply changing their destination address and indicating the invalidity of information that they carry flipping the empty/full bit of the slots.

In order to illustrate the performance of SR mechanism, we provide three simulation results. The results are drawn against the performance of DQDB on dual bus configuration[7] to provide a comparison basis. Figure 8 shows the results obtained under asymptotic conditions. In this case, the traffic pattern is artificial so that every station has as many packets as it can use on either bus. Therefore, figure 8 gives the bandwidth available to stations. On the other hand, this bandwidth is not the same for forward and backward bus. Figure 9 shows the available bandwidth on forward and backward bus, separately.

The global throughput of DQDB in figures 8 and 9 is 1.76. The CBRMA++ scores slightly better, since there is no loss due to a corresponding bandwidth

[7]Note that the performance of DQDB on the folded bus will be the half of what is shown in the figure, assuming that slots are reused only at the (single) headend station.

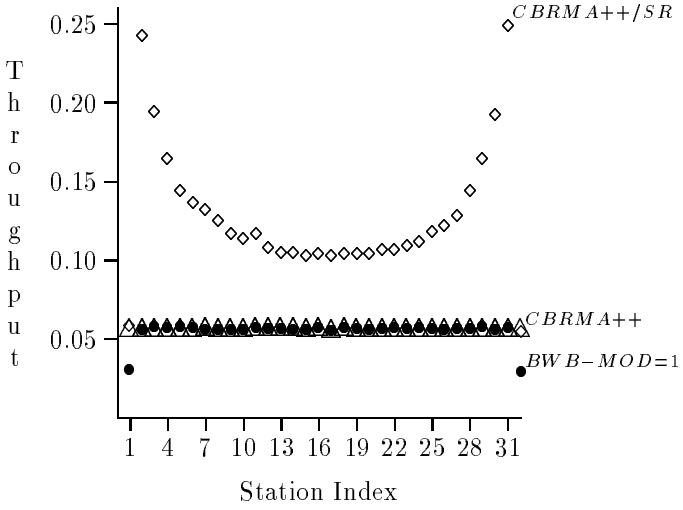

Figure 8: Performance under asymptotic conditions. Available bandwidth on the forward and backward buses are shown together.

balancing mechanism. The global throughput rate of CBRMA++/SR is calculated as 4.09 under asymptotic conditions. However, under uniform traffic distribution, this figure drops to 3.65, as can be seen in figure 10. The reason for the difference is quite obvious, since the stations closer to the both ends of the bus do not have a sufficient number of messages queued to make full use of the available bandwidth.

A number of methods for the implementation of erasure nodes for DQDB are suggested and analyzed in [9,10]. However, these proposed schemes are not as effective as the SR mechanism of CBRMA++/SR, since the station that clears the slot and the one that uses it are different stations. In other words, the slot that is cleared by S_i can be reused by S_{i+1} (or S_{i-1} depending on the direction of transmission) at the earliest. In our protocol, a station can begin to reuse a slot as soon as completing the inspection of destination address field.

As can be seen in the figures, the SR mechanism disturbs the fairness of the protocol. However, it is clear that no compromise is made out of the fair share. The irregularity stems from two factors: firstly, some stations receive more slots than the others. Secondly, the stations located closer to the both ends of the bus will find more empty slots at their disposal on forward or backward bus, depending on their location. This explains why the throughput of S_{31} on the forward bus and the throughput of S_2 on the backward bus are so high under asymptotic conditions. On the other hand, under more realistic traffic patterns, the first factor will be irrelevant, since the stations most likely

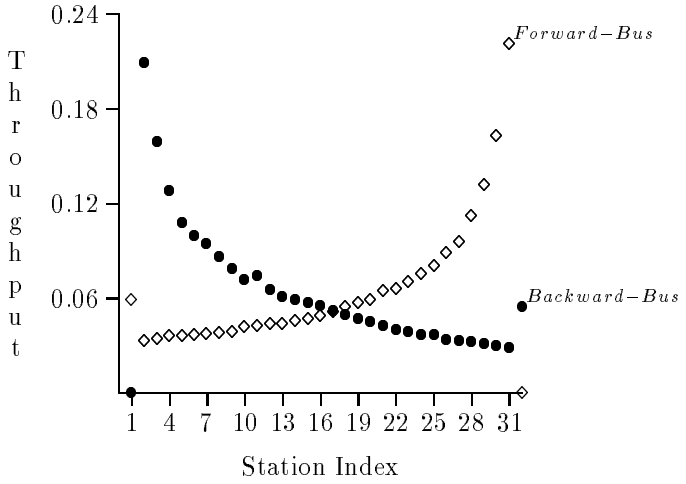

Figure 9: Performance under asymptotic conditions. Available bandwidth on the forward and backward buses are shown seperately.

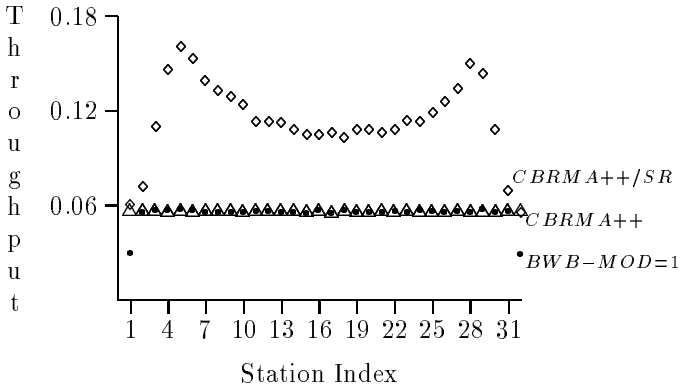

Figure 10: Performance under uniform traffic.

will not be able to fill all available empty slots, as can be seen in figure 10. As for the influence of the second factor, it is not strong enough to cause large variations in the throughput rates of the stations, since the bandwidth allocation mechanism does not allow a station to usurp the bus while the others are also contending for it. Therefore, under moderate or heavy load which is more or less uniform across the network, the effect of the SR will also follow the same pattern.

6 Conclusions

In this paper, we presented a new protocol called CBRMA++/SR which can sustain an aggregate data transmission rate while not compromising fairness. The relative success of the protocol is due to the preventive approach employed in the design rather than allowing the occurence of irregularities in the network and providing some countermeasures[8] to compensate the undesired side-effects. The latter approach is bound to be less effective since both the detection of the irregularity and the application of the countermeasure take time which is not negligible under high transmission rates.

A remark is also in order regarding the medium access delay. In our protocol, the delay between the beginning of the two consecutive transmission sessions[9] is always equal to the round trip delay of the bus when there is no SR mechanism in action. This is so, regardless of the global load on the network. However, the available bandwidth can change from the fair share up to the capacity of the whole cycle. In DQDB, shorter access delays are possible under light load. On the other hand, the SR mechanism has also the effect of reducing the access delays. The rate of the improvement is directly related to the number of slots that the stations receive.

When handling multislot packets, DQDB can not ensure that the slots forming the packet will not arrive interleaved with slots coming to the same receiver from other senders. The protocol has to keep a separate queue for each sender or the slots have to be reordered before being passed to the upper layer protocols. This may add to the complexity of the protocol design and certainly will decrease the performance, especially at high transmission rates. Besides ensuring the slot contiguity, CBRMA++/SR provides the upper layer protocols also with the number of slots to be transmitted and received, in advance. The authors believe in that, such information can proved to be quite useful in the design of more effective higher layer protocols.

In this study, slots are considered to be homogenous. Obviously, the ability to support various applications (therefore different traffic patterns) requires the efficient handling of different slot types. Priorities, multicast and broadcast support and isochronous and non–isochronous traffic handling policies will be addressed in a future paper.

[8] Such as the backpressure mechanism of CRMA.

[9] In which a number of slots are transmitted instead of one. Since the slot contiguity is guaranteed, using the delays between the transmissions of two slots can be misleading in the determination of the medium access delays.

References

[1] IEEE, "Std 802.6—1990, IEEE Standards for Local and Metropolitan Area Networks: Distributed Queue Dual Bus(DQDB) of a Metropolitan Area Network (MAN)", July 1991.

[2] M. Conti, E. Gregori and L. Lenzini, "A Methodological Approach to an Extensive Analysis of DQDB Performance and Fairness", IEEE Journal on Selected Areas in Communications, Vol.9, No.1, January 1991, pp. 76-87.

[3] H. R. Muller, M. M. Nassehi, J. W. Wong, E. Zurfluh, W. Bux and P. Zafiropulo, "DQMA and CRMA: New Access Schemes for Gbit/s LANs and MANs" in Proc. INFOCOM'90, San Francisco, June 1990, pp. 185-191.

[4] H.R. van As, "Major Performance Characteristics of the DQDB MAC Protocol", SBT/IEEE International Telecommunications Symposium '90, Rio de Janeiro, Sept. 1990, paper 6.3, pp. 113-120.

[5] H.R. van As, "Performance Evaluation of Bandwidth Balancing in the DQDB MAC Protocol", Eighth Annual European Fibre Optic and Local Area Networks Conference, EFOC/LAN 90, Munich, June 1990, paper 5.3.2, pp. 231-239.

[6] H.R. van As, J.W. Wong, P. Zafiropulo, "Fairness, Priority and Predictability of the DQDB MAC Protocol under Heavy Load", Int. Zurich Seminar on Digital Communications, Zurich, March 1990, pp. 410-417.

[7] C. Baransel, W. Dobosiewicz, "CBRMA (Cyclic Balanced Reservation Multiple Access) MAC Protocol", in Proc. The Sixth International Symposium on Computer and Information Sciences-ISCIS VI, Antalya, Türkiye, October 1991, Volume 1, pp. 537-545.

[8] C. Baransel, W. Dobosiewicz, P. Gburzynski,"CBRMA++: On How to Increase the Performance of a MAC Protocol Without a Second Headend Station", in Proc. Silicon Valley Networking Conference, California, April 1992, pp. 201-207.

[9] C. Baransel, W. Dobosiewicz, P. Gburzynski,"CBRMA++/SR: A High-Speed MAN/WAN MAC Protocol with An Efficient Slot Reuse Mechanism", *submitted.*

[10] M. A. Rodrigues, "Erasure Nodes: Performance Improvements for the IEEE 802.6 MAN", INFOCOM'90, pp. 636-643.

[11] M. W. Garret, S. Q. Li, "A Study of Slot Reuse in Dual Bus Multiple Access Networks", IEEE Journal on Selected Areas in Communications, Vol.9, No.2, February 1991, pp. 258-256.

[12] W. Dobosiewicz, P. Gburzynski,"The Topology Component of Protocol Performance", in Proc. Conference on Local Computer Networks, Minneapolis, October 1991, pp. 582-588.

[13] M.M. Nassehi, "CRMA: An Access Scheme for High-Speed LANs and MANs", IEEE International Conference on Communications, ICC 90, Atlanta, GA, April 1990, pp. 1697-1702.

[14] M.M. Nassehi, "Cyclic Reservation Multiple-Access Scheme for Gbit/s LANs and MANs based on Dual-Bus Configuration", Eighth Annual European Fibre Optic and Local Area Networks Conference, EFOC/LAN 90, Munich, June 1990, pp. 246-251.

[15] P. Tran-Gia, R. Dittmann, "Performance Analysis of the CRMA-Protocol in High-Speed Networks", Univ. of Würzburg, Institute of Computer Science Research Report Series, Report No. 23, December 1990.

[16] J. Liebeherr, I.F. Akyildiz, A.N. Tantawi, "An Effective Scheme for Pre-Emptive Priorities in Dual Bus Metropolitan Area Networks", SIGCOMM'92, August 17, Baltimore, Maryland.

Appendix: An example cycle

Consider a network with 8 stations and a bus length of 80 slots so that the fair share is 10 slots per station; assume also that at the beginning of reservation cycle i, the following reservation requests are valid (figure 11) :

Station number	*The length of queue-F*	*Unused (-) Extra (+)*
1	15	-
2	5	15
3	25	-
4	5	20
5	-	5
6	5	30
7	25	-
8	-	20

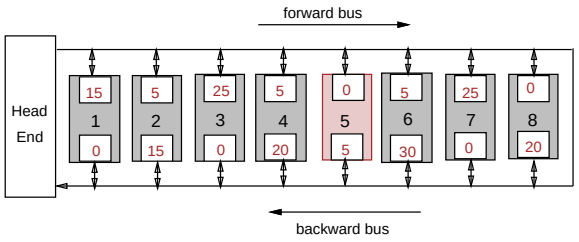

Figure 11: Outstanding reservation requests in cycle i

RESERVATION HALF-CYCLE

The headend station starts the reservation cycle by sending a reservation slot whose contents will be denoted as $< C, R_f, ES_f, ER_f, U_f, R_b, ES_b, ER_b, U_b >$ from now on. At the beginning, the slot contains $<i,0,0,0,0,0,0,0,0>$ and the stations update it in the following manner:

- Station 1: The first station has messages for the forward bus only. So it claims its fairshare first, which is 10. Then it states its willingness to use 5 additional slots, if it is given the chance. At this time, the number of stations that requires extra number of slots on forward bus is 1. Since it has nothing to transmit on the backward bus, it simply puts its fairshare into common use by incrementing U_b by 10.

  (Station 1 updates the reservation slot contents to $<i,10,1,5,0,0,0,0,10>$)

- Station 2: This station has messages for both upstream and downstream nodes. Its need on forward bus is less than its fairshare while it can use 5 more slots on the backward bus.

  (Station 2 updates the reservation slot contents to $<i,15,1,5,5,10,1,5,10>$)

  Following the same principles the other stations make their reservations as follows:

- Station 3: $<i,25,2,20,5,10,1,5,20>$
- Station 4: $<i,30,2,20,10,20,2,15,20>$
- Station 5: $<i,30,2,20,20,25,2,15,25>$
- Station 6: $<i,35,2,20,25,35,3,35,25>$
- Station 7: $<i,45,3,35,25,35,3,35,35>$
- Station 8: $<i,45,3,35,35,45,4,45,35>$

When the reservation slot reaches the fold, the situation of forward bus reservation is as good as it can be, since the number of extra slot requests and the number of available slots are exactly the same. However, on the backward bus, we are 10 slots short to meet all demands in their entirety. So, there has to be a compromise and the protocol will make it sure it will be a fair one, in the balancing half-cycle.

During the reservation cycle, the *CycleStartCounter*s and *XMitCounter*s are set as follows;

Counters for FORWARD BUS

Station number	*CycleStart Counter*	*XMit Counter*
1	1	10
2	11	5
3	16	10
4	26	5
5	-	-
6	31	5
7	36	10
8	-	-

Counters for BACKWARD BUS

Station number	*CycleStart Counter*	*XMit Counter*
1	-	-
2	1	10
3	-	-
4	11	10
5	21	5
6	26	10
7	-	-
8	36	10

BALANCING HALF-CYCLE

At this moment, the reservation slot passes the fold and begins to propagate towards the headend station, thereby commencing the balancing half-cycle. If the number of unused slots is greater than or equal to the number of extra slot requests for a bus, there is no problem since all of them can be met. Otherwise each station computes its fair share of the unused slots and compares it with the difference between the unused slot count and the remaining extra slot request if this value is positive. Trying to maximize its gain the station chooses the maximum of these two values and updates the reservation slot accordingly:

- Station 8: This station knows that it has asked for 10 more slots and now realizes only 35 slots are available for 4 stations (including itself) that are asking for 45. Following the algorithm it computes its fair share first: $\mathcal{F} = \lfloor 35/4 \rfloor = 8$. It also computes that the remaining 3 stations are asking for a total of 35 slots. Since $(35 - 35 = 0)$, it chooses to settle with its fair share and claims 8 of the unused slots.

  (Station 8 updates the reservation slot contents to $<i,45,3,35,35,53,3,35,27>$)

- Station 7: This station computes that its extra slot request can be met entirely.

  (Station 7 updates the reservation slot contents to $<i,60,2,20,20,53,3,35,27>$)

- Station 6: This station has no outstanding slot requests for forward bus while it needs 20 extra slots on the reverse direction. Its fair share is $\mathcal{F} = \lfloor 27/3 \rfloor = 9$. On the other hand, it also computes that the remaining two stations are asking for only $(35 - 20 = 15)$ slots. If it chooses to claim only 9 slots that will cause $(27 - 9 = 18 \; and \; 18 - 15 = 3)$ 3 slots to be wasted. So, claiming 12 instead of 9, 6 updates the reservation slot contents as follows:

  $<i,60,2,20,20,65,2,15,15>$)

  Following the same principles the other stations make their reservations as follows:

- Station 5: Having no extra slot request, it does not update the slot.

- Station 4: $<i,60,2,20,20,75,1,5,5>$

- Station 3: $<i,75,1,5,5,75,1,5,5>$

- Station 2: $<i,75,1,5,5,80,0,0,0>$

- Station 1: $<i,80,0,0,0,80,0,0,0>$

During the balancing cycle, the values of counters are finalized as follows;

Counters for FORWARD BUS

Station number	*CycleStart Counter*	*XMit Counter*
8	-	-
7	56	25
6	51	5
5	-	-
4	46	5
3	21	25
2	16	5
1	1	15

Counters for BACKWARD BUS

Station number	*CycleStart Counter*	*XMit Counter*
8	63	18
7	-	-
6	41	22
5	36	5
4	16	20
3	-	-
2	1	15
1	-	-