An Effective Erasure Node Algorithm for Slot Reuse in DQDB

H. S. Hassanein

Department of Mathematics & Computer Science
Kuwait University
P.O.Box: 5969 Safat, Kuwait 13060

J. W. Wong

Department of Computer Science
University of Waterloo
Waterloo, Canada, N2L 3G1.

J. W. Mark

Department of Electrical & Computer Engineering
University of Waterloo
Waterloo, Canada, N2L 3G1.

ABSTRACT

The Distributed Queue Dual Bus (DQDB) protocol, which has been specified by the IEEE 802.6 as the Metropolitan Area Networks (MANs) standard, has an undesirable feature that busy slots that has been already received (i.e. read) continue to propagate downstream unnecessarily. The use of special nodes, known as erasure nodes, allows conversion of read slots to empty, so that downstream nodes may reuse these slots. This should result in an increase of the total throughput of the network. In this paper, we introduce and evaluate the performance of an erasure node algorithm. The algorithm keeps track of past activities on both busses to effectively balance the slot erasure and the request cancellation functions. We show that this algorithm possesses a number of features which make it superior to existing algorithms.

1. Introduction

The Distributed Queue Dual bus (DQDB) protocol [1] is the IEEE 802.6 standard for Metropolitan Area Networks (MANs). DQDB supports both isochronous and asynchronous traffic with up to 3 priority levels The topology of DQDB consists of two slotted unidirectional busses allowing communication in opposite directions similar to that originally proposed for Fasnet [2] (see Figure 1). However, unlike Fasnet, in DQDB each bus serves as a reservation channel for the other bus. A node desiring to transmit on one channel must reserve an empt slot by sending a request on the opposite channel.

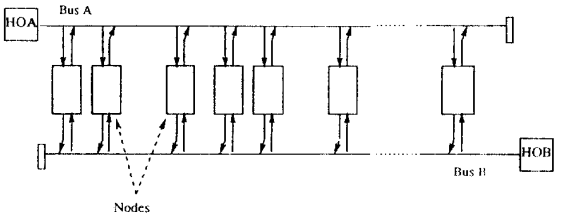

Figure 1: The DQDB dual bus topology

The DQDB slot is 53 bytes long, the first of which is the Access Control Field (ACF). The ACF consists of one status bit which indicates whether a slot is empty or busy, one slot type bit which indicates whether the slot is for isochronous or asynchronous traffic, 3 request bits (one for each priority level), a Previous Slot Read (PSR) bit which indicates if the previous slot has been read by the destination node or not, and 2 reserved bits. Slots are generated by Headend nodes as empty with no request bits set.

Since both busses are identical, one can, without loss of generality, consider Bus A to be the forward bus on which data is transmitted and Bus B to be the reverse bus on which slot reservations are made. The distributed queue is managed by requiring each node to keep track of requests from downstream nodes on Bus A. Each node can be in one of two states: IDLE (i.e., have no data to transmit), or COUNTDOWN (i.e., have a data segment queued). In the IDLE state a node keeps track of downstream requests by incrementing a *Request_Count* counter (req_ctr) for every request seen on Bus B. req_ctr is decremented for every empty slot seen on Bus A, to a minimum of zero. The value of req_ctr would, therefore, represents the number

of segments queued for transmission on Bus A from downstream nodes as seen by this node.

When a node has a segment to transmit on Bus A, it inserts this segment in the distributed queue by entering the COUNTDOWN state. The node copies the contents of req_ctr to the *Count_Down* counter (cd_ctr) and clears the req_ctr. A request is submitted for transmission on Bus B. The value of cd_ctr now represents the number of segments queued ahead of the node's own transmission. cd_ctr is decremented for every empty slot observed on Bus A. When cd_ctr becomes zero, the node uses the next empty slot for transmitting its segment, and returns to the IDLE state. While in the COUNTDOWN state, downstream requests are recorded by incrementing req_ctr.

It should be noted that when the node has more than one segment to transmit, it is not allowed to submit the next request until its current segment is transmitted. It should also be noted that segment and request transmissions are independent. That is, once a request is queued for transmission on Bus B it is transmitted regardless of whether the corresponding segment has been transmitted or not. For more details on the operation of the DQDB protocol, the reader is referred to [1].

DQDB has the property that under overload conditions, the node that transmits first has an unfair throughput advantage over the other nodes [13]. This has led to the development of the Bandwidth Balancing (BWB) scheme [3] which achieves throughput fairness at the expense of some wasted capacity. Specifically, each node has a bandwidth balancing counter (BWB_ctr) for each bus, which is incremented for every segment transmitted by the node. When this counter reaches a certain modulus (M), it is cleared and the req_ctr is artificially incremented by 1. This causes one empty slot to pass to downstream nodes. Therefore, a node uses a maximum of M/(M+1) of the available bandwidth. For the case of N active nodes, the steady state throughput for each node is M/(MN+1) of the bus capacity [3].

An undesirable feature of a dual bus protocol like DQDB is that once a slot is used, it propagates to the end of the bus even though the information might have already been read by the destination node. One approach to releasing busy slots is the use of erasure nodes. Under this approach, a node is required to mark a slot as read by setting the PSR (Previous Slot Read) bit in the access control field of the following slot. Special nodes called erasure nodes, selectively placed on the network, are responsible for recognizing these "read" slots and converting them to empty.

For the erasure node approach to be effective, an algorithm is needed to cancel requests that have been satisfied by erased slots. The merit of an erasure node algorithm can be measured by its effectiveness in handling both local and nonlocal traffic. By *local* traffic, we mean traffic that is confined to a bus section between successive erasure nodes.*Nonlocal* traffic, on the other hand, must pass through at least one erasure node.

In this paper, we define the criteria for a "good" erasure node algorithm and introduce a new algorithm that is capable of meeting such criteria. The proposed algorithm keeps track of past activities on both busses to balance the segment clearling and the request balancing functions. The algorithm does not require changing the format of the DQDB ACF field, as it does not require the use of any additional control bits, nor any modification to the functions of the DQDB nodes. This algorithm was submitted to the IEEE 802.6 committee as a proposed erasure node standard [4]. Our proposal was accepted by the committee as the basis for standard IEEE 802.6e [5]. A first draft has been prepared and sent for ballot.

This paper is organized as follows. In section 2, we review and study the performance of existing erasure node algorithms [6-11] and benchmark them against the performance of the "good" algorithm. In section 3, we describe our new erasure node algorithm. The performance characteristics of the proposed algorithm are provided in section 4. Finally, section 5 provides some concluding remarks.

2. Performance Comparison of Existing Algorithms

Several erasure node algorithms have been introduced [6-11]. In this section, we review these algorithms and evaluate their performance. We also compare their performance to that of a "good" erasure node algorithm. We show that existing algorithms fail the criteria of a "good" erasure node algorithm.

A "good" erasure node algorithm defined as one that clears all local segments and is transparent to nonlocal segments, i.e., does not delay nonlocal slots. The algorithm must be fully compatible with the 802.6 standard, i.e., does not require the use of any additional ACF bits, nor any changes to the functions of regular DQDB nodes. A "good" algorithm should not cause any redistribution of bandwidth. Upstream nodes should not suffer from reduced channel access as a result of the use of the erasure node algorithm. As well, a "good"

algorithm should not unduly increase channel access delays. Only the one slot latency is permitted.

A brief description of existing erasure node algorithms follows.

Algorithm NM *(No Modification)* [6]:

Under this algorithm, an erasure node simply detects and erases slots that have been read. No attempt is made to adjust the flow of requests on the reverse bus.

Algorithm ENC *(Erasure Node Counter)*[7-9]:

Under this algorithm, an erasure node counter, *en_ctr*, is used at each erasure node to keep track of the number of erased slots. *en_ctr* is incremented for each slot erased. When *en_ctr* is greater than zero, any request bit seen on the reverse bus is cleared, and en_ctr is decremented. An erasure node thus takes into account the empty slots generated for downstream nodes.

Algorithm NR *(Negative Requests)* [10]:

Under this algorithm, a new bit, *NREQ*, in the ACF is used to indicate a negative request. For each slot that is erased, the erasure node sets this *NREQ* bit on the reverse bus to indicate to upstream nodes that a slot has been made free. Upon seeing this bit, a node decrements its estimate of empty slots required by downstream nodes by one.

Algorithm EMP *(EMPty slot counter)* [11]:

This algorithm is a modification of algorithm *ENC*. It uses an additional counter, *emp_ctr*, which is incremented for every empty slot observed on Bus A, and if positive, is decremented for each passing request on Bus B. Under *EMP*, a request bit is cleared if either *en_ctr* or *emp_ctr* is positive. Therefore, algorithm *EMP* keeps track of the empty slots unused by upstream nodes.

In evaluating the performance of their respective algorithms, the authors [6-10] studied the delay-throughput characteristics of DQDB with erasure nodes. Performance advantages over standard DQDB in terms of throughput gains or reduced access delay have been demonstrated. However, and was pointed in [11] and [12], the performance under temporary overload conditions should also be considered. We, therefore, only show the performance of the above algorithms under temporary overload conditions. We show that none of these existing algorithms satisfy the criteria for a "good" erasure node algorithm.

Under overload, the nodal throughput of basic DQDB is dependent on the initial conditions [13]. With two active nodes, say nodes 1 and 2 with node 1 being closer to Headend A, there are three initial conditions of interest:

Initial Condition 1: Node 1 transmits first and fills bus (A) with Busy slots before node 2 becomes active.

Initial Condition 2: Node 2 transmits first and fills bus (B) with request bits before node 1 becomes active.

Initial Condition 3: Both nodes start transmitting at the same time.

We consider two scenarios each involving only two active users under overload (see Figure 2). In each scenario, the network is divided into two sections with an active user in each section. Node 3 is the only erasure node. In the first scenario all traffic on the network is local (see Figure 2.a). The active nodes are 1 and 4, with destinations 2 and 5, respectively. Throughput results assuming an internode separation of 10 slots, for the three initial conditions above, are shown in Table 1. Under this scenario, the throughput of each node should ideally be 100% regardless of the initial condition.

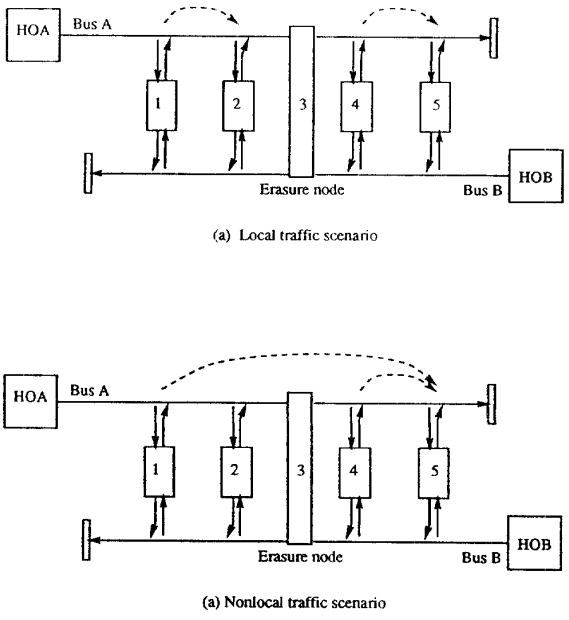

Figure 2: DQDB with two active nodes

THROUGHPUT (% of bus capacity)							
		Initial Cond. 1 : Node 1 first		Initial Cond. 2 : Node 4 first		Initial Cond. 3 : Both simult.	
Traffic	Erasure Node Algorithm	Node 1	Node 4	Node 1	Node 4	Node 1	Node 4
Local	NM	0.05	100.0	0.05	100.0	0.05	100.0
	ENC	100.0	100.0	33.3	100.0	74.4	100.0
	NR	100.0	100.0	33.3	100.0	74.4	100.0
	EMP	100.0	100.0	100.0	100.0	100.0	100.0
	good	*100.0*	*100.0*	*100.0*	*100.0*	*100.0*	*100.0*
Nonlocal	NM	98.4	1.6	11.3	88.7	50.0	50.0
	ENC	98.4	1.6	11.3	88.7	50.0	50.0
	NR	98.4	1.6	11.3	88.7	50.0	50.0
	EMP	98.4	1.6	81.3	18.8	81.2	18.7
	good	*98.4*	*1.6*	*11.3*	*88.7*	*50.0*	*50.0*

Table 1: Throughput distribution for scenarios in Figure 2

Algorithm *NM* performs very poorly as node 1 is starved under all initial conditions. This is due to the fact that node 4 does not sense any busy slots on Bus A and generates a continuous stream of request bits on Bus B that is not cleared by the erasure node. Algorithm *ENC* adjusts these request bits with the use of the *en_ctr*. However, under initial condition 2, request bits from node 4 arrive at node 3 before the busy slots used by node 1. These request bits are left untouched since node 3's *en_ctr* is still zero, thereby preventing node 1 from using all the available capacity. Algorithm *NR* performs similar to *ENC* as it suffers from the same problem. Under algorithm *EMP*, node 4's requests are cleared because of the presence of emp_ctr. EMP therefore exhibits ideal throughput.

We next consider an instance of nonlocal traffic (Figure 2.b), where both nodes 1 and 4 are transmitting to node 5. Simulation results are shown in Table 1. Note that the throughputs obtained for all algorithms agree well with that of the "good" erasure node algorithm, except for *EMP*. Under *EMP*, node 1 has an unfair advantage over node 4 in most cases. This is because the non-zero *emp_ctr* in node 3 causes clearing of request bits from node 4.

The simple performance comparison above shows existing algorithms to perform well under either local or nonlocal traffic, but not both. A "good" erasure node algorithm must be able to handle both local and nonlocal traffic effectively. Moreover, it can be easily shown that existing algorithms also fail some or all of the other criteria as well.

3. Proposed Algorithm

Our proposed algorithm keeps track of past activities on both busses to effectively balance the slot resue and request cancellation functions. The algorithm has three states, namely: IDLE, ERASING and NON-ERASING.

Each node is equipped with an occupancy counter (*oc_ctr*), and a Request Shift Register (Req_SR) of length D (the value of D will be discussed later). *oc_ctr* represents the remaining time in the current state under the condition that no busy slot is observed on Bus A. Req_SR keeps track of request bit activities during the past D slots.A state diagram description of our algorithm is shown in Figure 3. In that description, the notation X<Y> is interpreted as follows. X indicates the condition of a state transition and Y the actions taken in response to the transition.

All erasure nodes are initialized IDLE. A local slot on bus A (i.e., a slot that has been read) causes a transition to the ERASING state, and a nonlocal slot (i.e., not read) causes a transition to the NON-ERASING state.

Upon entering the ERASING state *oc_ctr* is set to D. If a request bit is observed on Bus B, a "1" is shifted into Req_SR and the request bit is cleared; otherwise, a "0" is shifted into Req_SR. *oc_ctr* is decremented for each empty slot on bus A. Whenever a local slot is sensed, *oc_ctr* is reset to D. When *oc_ctr* reaches zero, the node returns to the IDLE state. If a nonlocal slot is sensed on bus A, the node enters the NON-ERASING state.

Upon entering the NON-ERASING state, *oc_ctr* is set to D. As well, a request is queued for transmission on the request bus, if the lead bit in the Req_SR is a "1". *oc_ctr* is decremented for each empty slot on bus A. Whenever a nonlocal slot is sensed, oc_ctr is reset to D. When *oc_ctr* reaches zero, the node returns to the IDLE state. Finally, a local slot causes a transition to the ERASING state.

D represents the length of request bit activities kept by an erasure node. In general, a value of D equal to the propagation delay between the current and the next downstream erasure nodes would provide sufficient history of past activities for our algorithm. The use of Req_SR removes any unfair request cancellation in the ERASING state. That is the algorithm alleviates the problems of algorithm *EMP*.

The need for a state minimum occupancy time can be justified by considering the local traffic scenario in Figure 2.a under initial condition 2. (This initial condition resulted in the severest deviation from the ideal throughput for existing algorithms.) Under initial condition 2, the reverse bus is filled with requests from node 4. As node 1 starts its transmission, the arrival of the first busy slot would cause the erasure node to enter the

ERASING state for at least a period of D. During this period, request bits originated from node 4 are cleared. Subsequent busy slots will keep the erasure node in the ERASING state. (Since these subsequent slots are not consecutive at first, the need for a minimum occupancy of D slots in the ERASING state should be clear.) Therefore, both nodes 1 and 4 can achieve 100% throughput.

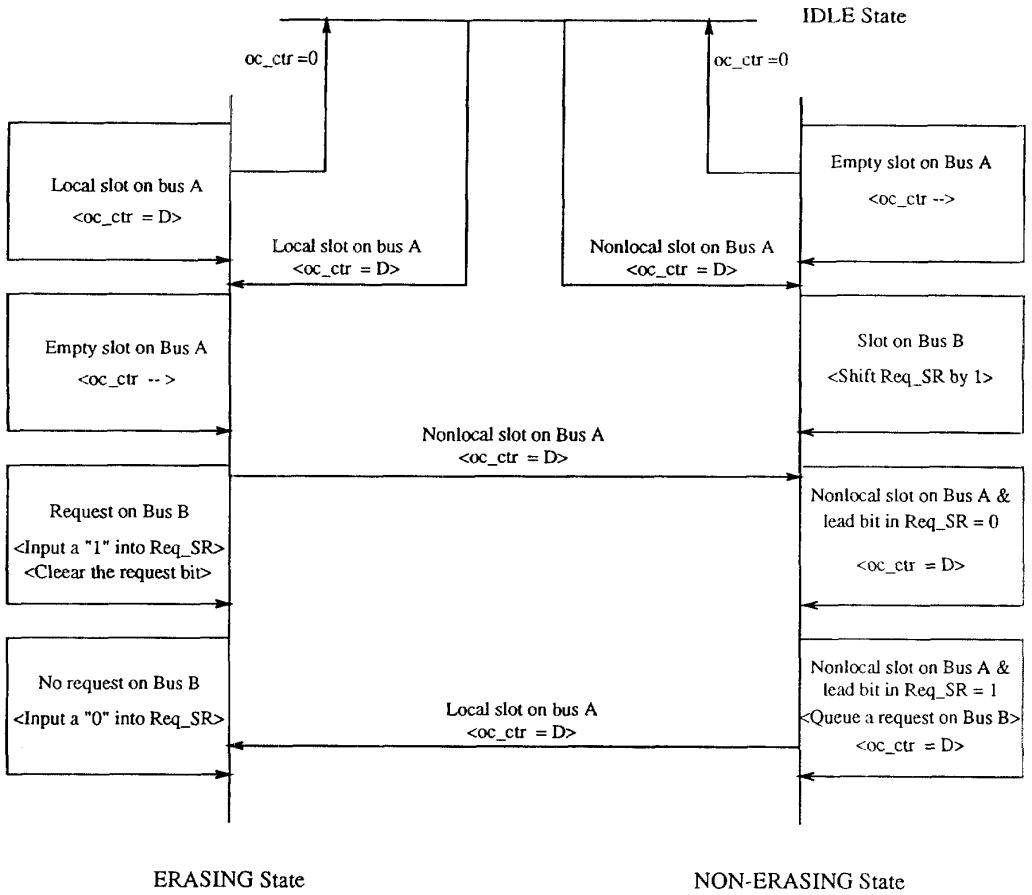

Figure 3: State diagram for proposed algorithm

4. Performance Evaluation

It can be verified that the proposed erasure node algorithm satisfies the functional and the compatibility criteria for a "good" erasure node algorithm. Here, we study the performance of the proposed algorithm.

Table 2 shows throughput results for the proposed algorithm for the two traffic scenarios mentioned in section 3. It can be easily noted that the results conform to those of the "good" algorithm, shown in italics, regardless of the traffic pattern, with and without bandwidth balancing.

THROUGHPUT (% of bus capacity)

Traffic	BWB	EN Algorithm	Initial Cond. 1 : Node 1 first		Initial Cond. 2 : Node 4 first		Initial Cond. 3 : Both simult.	
			Node 1	Node 4	Node 1	Node 4	Node 1	Node 4
Local	OFF	proposed	100.0	100.0	100.0	100.0	100.0	100.0
		good	*100.0*	*100.0*	*100.0*	*100.0*	*100.0*	*100.0*
	ON	proposed	90.0	90.0	90.0	90.0	90.0	90.0
		good	*90.0*	*90.0*	*90.0*	*90.0*	*90.0*	*90.0*
Nonlocal	OFF	proposed	98.4	1.6	11.3	88.7	50.0	50.0
		good	*98.4*	*1.6*	*11.3*	*88.7*	*50.0*	*50.0*
	ON	proposed	47.4	47.4	47.4	47.4	47.4	47.4
		good	*47.4*	*47.4*	*47.4*	*47.4*	*47.4*	*47.4*

Table 2: Throughput distribution for two active users

We have studied the performance of the proposed algorithm under different number of nodes, nodal separation and erasure node(s) locations. For the scenarios that we considered, throughput results have shown that the proposed algorithm has a behavior similar to that of a "good" erasure node algorithm.

Next, we study the effect of the proposed erasure node algorithm on the delay-throughput characteristics of DQDB with erasure nodes.

Consider a network with 20 active nodes distributed equally and evenly among five DQDB sections separated by four erasure nodes. Nodes are indexed in ascending order, 0 to N, from Headend A. Headend nodes for busses A and B are nodes 0 and N respectively. We thus have N=24 with nodes 5, 10, 15 and 20 being erasure nodes. Overload is not assumed, instead the message generation process at each node is Poisson. Message destinations are chosen randomly among downstream nodes. Nodes are assumed to be equally spaced with an internode separation of 10 slots. We assume that erasure nodes may be involved in active communication sessions. Erasure nodes are permitted to use the slots they erase for their own transmission.

The performance measure we use is the average segment access time defined as the elapsed time from the moment a segment reaches the top of a node queue until it gains access to the media. To simplify and speed simulation, and since we are only concerned with segment access time, we assume all generated messages to be of fixed length equal to one segment. All results provided in this section are in slot times.

In Figure 4, we show average nodal access time for a DQDB network under the proposed algorithm for a heavily loaded network (bus throughput = 96%). Results of the basic no-erasure-node DQDB are also shown for comparison. The results clearly demonstrate the performance gain obtained by use of the proposed algorithm. For instance, the average access time at node 5, an erasure node, drops from 13 to 0.7 slot times, for the case of basic DQDB and DQDB with erasure nodes, respectively, a factor of 18. Similar gains are observed for other nodes.

It should be noted that the performance gain in average access time for nodes belonging to the network section closest to Headend A, nodes 1-4. This gain is not attributed to slot reuse, but rather to the effective request cancellation scheme used in our erasure node algorithm. An algorithm that does not balance the slot release and the request cancellation functions, would let some requests propagate upstream unnecessarily, and, subsequently, have poorer performance.

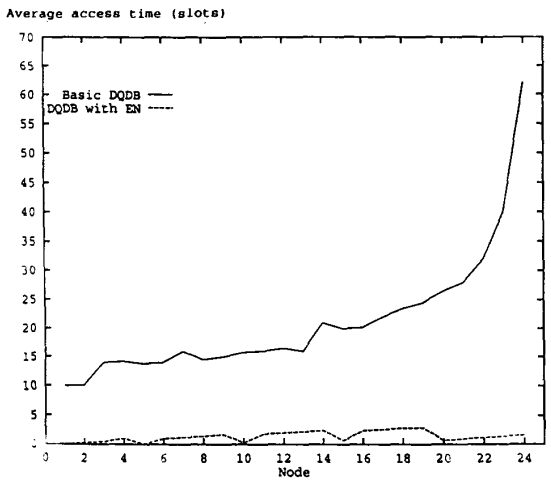

Figure 4: Average nodal access time (96% throughput)

It should be also noted that not only does incorporation of the proposed algorithm result in lower access delay, but also in a fair uniform one. As Figure 5 shows, the nodal access time for DQDB using our proposed slot reuse algorithm is almost uniform.

Figure 5 shows the delay-throughput characteristics of the network above. For comparison purposes, we also include performance results from algorithm *ENC* [7-9]. For both algorithms, under light to medium offered loads the average access time is comparable to that of the basic DQDB. However, and as expected, as the load increases, a network with a basic DQDB protocol saturates, while a network employing a slot reuse mechanism is still operating below its saturation point. Hence the huge performance gain at throughput values close to unity. Of more interset, is to note that the delay-throughput characteristics of the proposed algorithm almost conform to those of algorithm *ENC* [7-9]. That is, delaying of the request stream as suggested in our algorithm has little effect on the delay characteristics of DQDB with slot reuse.

5. Conclusions

A new erasure node algorithm has been introduced. This algorithm has three states: IDLE, ERASING and NON-ERASING. While in the ERASING state, all requests propagating upstream are cancelled. Whereas, in the NON-ERASING state, no requests are cancelled. To compensate for request overcancellation in the ERASING

state, a buffer of past activities on the reverse bus is maintained. These requests may be later injected into the request stream in the NONERASING state.

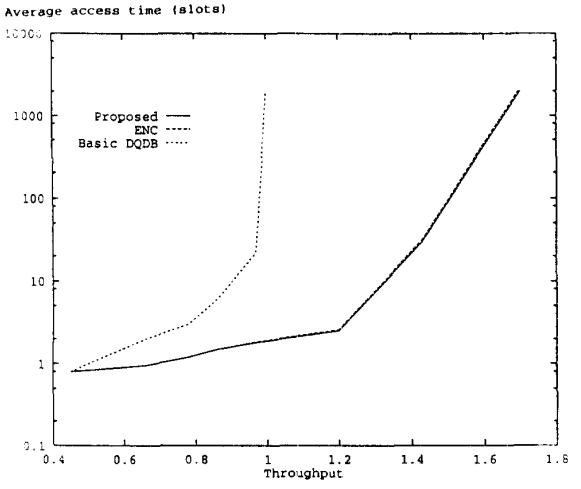

Figure 5: Delay-throughput characteristics

We have studied the behavior of the proposed algorithm under different number of nodes, nodal separation and erasure node(s) locations. For the scenarios that we considered, throughput results have shown that the proposed algorithm has a behavior similar to that of a "good" erasure node algorithm.

Stationary delay-throughput characteristics of DQDB with erasure nodes has demonstrated that the performance under the proposed algorithms is at least as good as existing algorithms in terms of throughput gains and average delay reduction. Since, we have shown our algorithm to outperform existing ones under overload, we can, therefore, claim that our algorithm is superior to existing ones.

The proposed algorithm does not require changing the format of the DQDB ACF field, as it does not require the use of any additional control bits, nor any modification to the functions of DQDB nodes. Our algorithm, therefore, guarantees backward compatibility with the IEEE 802.6 standard.

We have submitted our algorithm to the IEEE 802.6 committee as a proposed standard for erasure nodes [4]. Our proposal was accepted by the committee as the basis for IEEE standard 802.6e [5], which we were asked to prepare. A first draft of the standard was presented at the last IEEE 802 meeting in July 1993 and has been sent for ballot.

The authors would like to draw the readers attention that a more comprehensive version of this paper will appear in the June, 1994 issue of the IEE proceedings I.

References

[1] IEEE Standard 802.6, "Distributed Queue Dual Bus (DQDB) Metropolitan Area Network (MAN)," Dec. 1990.

[2] J.Limb and C. Flores, "Description of Fasnet-Unidirectional Local Area Communications Network," *Bell sys. Tech. J.*, Vol. 61, No. 7, Sept. 1982, pp. 1413-1440.

[3] E.L. Hahne, A. K. Choudhury, N. F. Maxemchuck, "Improving the Fairness of the Distributed Queue Dual Bus Networks," *INFOCOM 90*, San Francisco CA, June 1990.

[4] H. S. Hassanein, J. W. Wong and J.W. Mark, "An Effective Erasure Node Algorithm for Enhancing Slot Reuse in DQDB," Contribution 802.6-1992/43, IEEE 802.6 working group, Nov. 1992.

[5] IEEE Working Document, "Erasure Node Algorithm for Slot Reuse in Distributed Queue Dual Bus (DQDB) Subnetwork of A Metropolitan Area Network (MAN)", IEEE 802.6e Unapproved Draft, July 1993, Rapporteurs: H. S. Hassanein, J. W. Wong and J.W. Mark.

[6] M. A. Rodrigues, "Erasure Node: Performance Improvements for the IEEE 802.6 MAN," *INFOCOM 90*, San Francisco CA, June 1990, pp. 636-643.

[7] R. Breault and V. Phung, "DQDB Performance Improvement with Erasure Nodes," Contribution to the IEEE 802.6 working group, March 1990.

[8] M. Zukerman and P. Potter, "A Protocol for Erasure Nodes Implementation Within The DQDB Framework," *GLOBECOM 90*, San Diego CA, Dec. 1990, pp. 1400-1404.

[9] D. Luciani, R. Pignatelli and L. Susanna, "The Balanced Erasure Node: A Mechanism for Slot Reuse in DQDB Protocol," *ICC 91*, Denver CO, June 1991, pp. 1350-1354.

[10] M. W. Garret and S-Q. Li, "A Study of Slot Reuse in Dual Bus Multiple Access Networks," *INFOCOM 90*, San Francisco CA, June 1990, pp. 617-629.

[11] Lee, Performance of the DQDB Protocol under Heavy Load, Univ. of Waterloo, Dept. of Computer Sci., Master thesis, 1991.

[12] K. Sauser and W. Schodl, "Performance Aspects of the DQDB Protocol," *Computer Networks and ISDN Systems*, 20 (1990), pp. 253-260.

[13] J. W. Wong, "Throughput of DQDB Networks under Heavy Load," *EFOC/LAN-89*, Amsterdam, pp. 146-151.