Report Number: WUCS-91-41

1992-06-01

The 3-Tier Structured Access Protocol to Control Unfairness in DQDB MANs

Lakshmana N. Kumar and Andreas D. Bovopoulos

This paper addresses the unfairness problem appearing in 802.6-based DQDB MANs. Traffic load demand is characterized as low (below 0.4 of the channel capacity), normal (from 0.4 to 0.9 of the channel capacity) or heavy (greater than 0.9 of the channel capacity). At low loads the 802.6 protocol is acceptably fair. At normal loads, however, the protocol performance is markedly unfair. The unfairness is related to the latency in transporting a request. At heavy loads the unfairness is both latency-related and flooding-related. In this paper, both types of unfairness are carefully analyzed. As a control measure, a 3-Tier Structured... **Read complete abstract on page 2.**

The 3-Tier Structured Access Protocol to Control Unfairness in DQDB MANs

Lakshmana N. Kumar and Andreas D. Bovopoulos

Complete Abstract:

This paper addresses the unfairness problem appearing in 802.6-based DQDB MANs. Traffic load demand is characterized as low (below 0.4 of the channel capacity), normal (from 0.4 to 0.9 of the channel capacity) or heavy (greater than 0.9 of the channel capacity). At low loads the 802.6 protocol is acceptably fair. At normal loads, however, the protocol performance is markedly unfair. The unfairness is related to the latency in transporting a request. At heavy loads the unfairness is both latency-related and flooding-related. In this paper, both types of unfairness are carefully analyzed. As a control measure, a 3-Tier Structured Access protocol is proposed. At low loads the 802.6 performance is retained. For normal loads, extra slots are allowed based on predicted demand. At heavy loads access protection is applied. A Dynamic Assessment of Network Topology (DANT) protocol is also presented. The DANT dynamically maintains the additional information required for the implementation of the 3-tier structure. The proposed fair access protocol is studied under different load types and traffic demand. A tuning scheme is proposed to optimized the performance for a particular load environment in real time. The proposed protocol has the potential for dynamic bandwidth allocation and yields satisfactory performance.

The 3-Tier Structured Access Protocol to Control Unfairness in DQDB MANs

Lakshmana N. Kumar and Andreas D. Bovopoulos

WUCS-91-41

June, 1992

Department of Computer Science
Washington University
Campus Box 1045
One Brookings Drive
St. Louis MO 63130-4899

This work was supported in part by an industrial consortium of Bellcore, BNR, DEC, Italtel SIT, NEC, NTT and SynOptics.

The 3-Tier Structured Access Protocol to Control Unfairness in DQDB MANs

Lakshmana N. Kumar[1] *and Andreas D. Bovopoulos*[2]
Department of Computer Science and
Computer and Communications Research Center
Washington University, St. Louis, MO 63130-4899

June 7, 1992

ABSTRACT

This paper addresses the unfairness problem appearing in 802.6-based DQDB MANs. Traffic load demand is characterized as low (below 0.4 of the channel capacity), normal (from 0.4 to 0.9 of the channel capacity) or heavy (greater than 0.9 of the channel capacity). At low loads the 802.6 protocol is acceptably fair. At normal loads, however, the protocol performance is markedly unfair. The unfairness is related to the latency in transporting a request. At heavy loads the unfairness is both latency-related and flooding-related. In this paper, both types of unfairness are carefully analyzed.

As a control measure, a 3-Tier Structured Access protocol is proposed. At low loads the 802.6 performance is retained. For normal loads, extra slots are allowed based on predicted demand. At heavy loads access protection is applied. A Dynamic Assessment of Network Topology (DANT) protocol is also presented. The DANT dynamically maintains the additional information required for the implementation of the 3-tier structure.

The proposed fair access protocol is studied under different load types and traffic demand. A tuning scheme is proposed to optimize the performance for a particular load environment in real time. The proposed protocol has the potential for dynamic bandwidth allocation and yields satisfactory performance.

1 The DQDB Architecture

The Distributed Queue Dual Bus (DQDB) MAN is comprised of two unidirectional buses, across which individual nodes are connected. A head-end station (or frame generator) is provided for each bus. Each head-end generates the DQDB frames that carry data along its bus. The nodes are connected to each bus via a read and write connection as shown in Figure-1. The read head is located ahead of the write head. Writing is performed by a logical OR function of the data already on the bus and the data from the node (if available). Writing in this manner is reliable in the sense that nodes can fail or be removed from the bus without disturbing the correct operation of the bus.

Figure-1 illustrates the terminology used to describe the network. The two buses are named bus-A and bus-B. The head-end that controls bus-A and generates frames is known as the *Head-end Of Bus-A* (HOB-A). The bus segment between node-i and HOB-A on bus-A is considered *upstream* of node-i with respect to bus-A. Similarly the segment between node-i and HOB-B on the bus-A is considered

[1]Currently with the Department of Electrical and Computer Engineering, University of Miami, Coral Gables, FL 33124, USA.

[2]This work was supported in part by an industrial consortium of Bellcore, BNR, DEC, Italtel SIT, NEC, NTT, SynOptics.

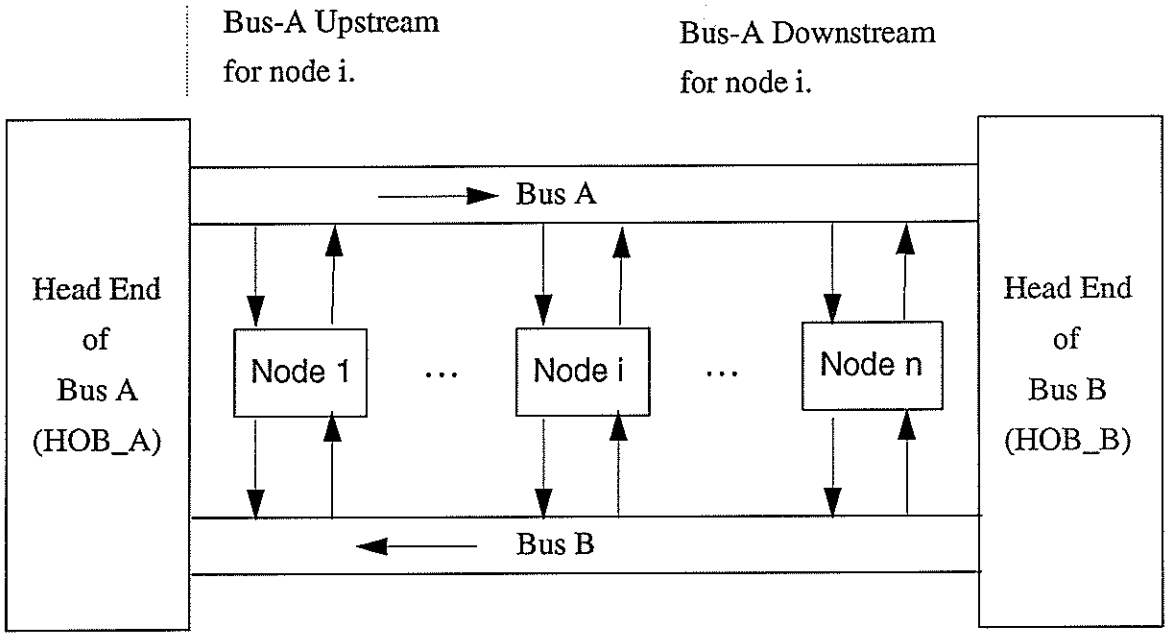


Figure 1: DQDB Dual Bus Architecture

downstream of node-i with respect to bus-A. This description of the upstream and downstream can be extended to other network entities (for example, downstream nodes, upstream bus-length, and so on). Similar terms can be used for description with respect to bus-B.

1.1 Distributed Queue Access Protocol

The Distributed Queue Access Protocol is a Medium Access Control protocol that controls access to the slots on the DQDB bus. Its access characteristics are essentially *independent* of the network size and speed. In particular it enables all of the payload bandwidth to be utilized, with the average slot access delay approximately that of a perfect scheduler. It provides this performance by making use of explicit information regarding the state of the network, specifically information acquired and stored by every node regarding the network load (*i.e.*, traffic demand).

If the distributed queue is empty, slot access is immediate. If the queue is not empty, preference is given to those segments already queued. The 802.6 slot format defines a request bit for each priority class and a busy bit for every slot. In this paper a single priority class of asynchronous packets is assumed.

A node can be in either *Idle* state or *Count Down* state. To access Bus-A, a node files a *REQUEST* by setting the *REQ* bit of a bus-B slot. The requests are filed on the bus opposite to the bus to which the node wishes to gain access. Each node maintains two sets of counters – the *RQ* (request) counter and the *CD* (count down) counter. The counters are updated as per the operational schema of the 802.6 protocol, which is presented below with regard to bus-A access.

A node that does not contest for access is said to be in the *idle* state. When a node is *idle*, every incoming request (REQ_bit set) on bus-B increments its *RQ* counter and every empty slot passing downstream on bus-A decrements its *RQ* counter, provided it is non-zero.

When a packet arrives at a node to be put on bus-A, the following actions are executed by the node:

1. The node moves the RQ counter value into the CD counter.
   $CD \leftarrow RQ$

2. It resets the RQ counter to zero.
   $RQ \leftarrow 0$

3. It sends a *REQUEST* to all upstream nodes by writing its request on the next available bus-B slot with its REQ_bit not set.

4. The node enters into the *countdown* state.

While in the *countdown* state, the node seeking access to bus-A does the following :

1. If the CD counter is zero, the node transmits its packet in the next available empty slot on bus-A. Also it marks the BUSY_bit of the slot to 1.

2. Otherwise the CD counter is decremented once for every empty slot encountered on bus-A.

3. With every new *REQUEST* arriving on bus-B, the RQ counter is incremented.

4. Once the transmission is completed, the node re-enters the *idle* state.

When the node has more than one packet to send, the packets are maintained in an internal queue. The 802.6 protocol specifies an *exhaustive scheme* of filing requests, under which a node can have only one pending request, immaterial of the length of the internal queue. Thus a node can file a request only when its previous request has been served. Each node may have at most one pending *REQUEST* for each priority level of each bus. This implies that a node can have no more than eight pending requests altogether.

2 Unfairness and its Origin

Access unfairness can be defined as the inability of a subset of nodes to gain access to a bus as quickly as the other nodes under the given access protocol mechanism.

At low loads, the observed demand for service is less than the bandwidth available. The low load range may extend up to roughly 0.4 of the channel capacity, depending on the actual traffic pattern. Because the demand is less than the unused bandwidth, unfairness is not an issue at low loads. At higher loads three types of persistent unfairness may arise.

1. **Latency-Related Unfairness :** The nodes that are closer to the frame generating head-end have no knowledge of the *REQUEST*s that are still in transition. As a result the distributed queue maintained by the 802.6 protocol is not perfect. Therefore, the latency results in a type of unfairness which is called latency-related unfairness. The latency-related unfairness is predominant when the load offered to a single bus is less than the bus capacity. When the DQDB network is overloaded two different persistent types of unfairness may appear.

2. **Access-Related Unfairness Due to Request Flooding :** If the overload situation is caused by a surge of traffic demand from downstream nodes, then upstream nodes may end up being heavily blocked for access.

3. **Access-Related Unfairness Due to Message Flooding :** If the overload situation is caused by a surge of traffic from the nodes closer to the head-end, then the downstream nodes are at a disadvantage compared to upstream nodes. Even if downstream nodes generate heavy traffic relative to upstream nodes, the upstream nodes always have easier traffic access. Unfairness due to flooding requires *access protection.*

At heavy loads the actual unfairness pattern is highly dependent on the load distribution pattern (state of the network) along the bus, at the time when heavy load conditions set in. This is especially true with large inter-node distances. In overload situations the unfairness phenomenon is also persistent. Because of the latency and flooding phenomena, the high load situation can be further subdivided into the following categories, depending on the overall load presented to a bus.

- **Normal Load :** The high load situation, in which the unfairness is predominantly caused by the propagation latency of the downstream *REQUEST*s, but in which the flooding phenomenon is still insignificant. This situation arises when the overall load presented to the network is less than the channel capacity.

- **Heavy Load :** The high load situation in which the unfairness is caused predominantly by the flooding phenomenon, with propagation latency playing a secondary role. Such a scenario is possible when the overall load presented to a bus is more than the channel capacity.

A protocol aimed at addressing the access unfairness problem must satisfy a number of requirements: (i) It should work for both small and large networks. (ii) It should work for any number of active nodes and for arbitrary internode distances. (iii) It should resolve both the latency and the flooding related unfairness. (iv) Wastage of bandwidth should be at the very minimum.

At low loads the 802.6 protocol is fair enough. Therefore any suggested protocol should preserve the behavior of the 802.6 protocol at low loads. At high loads any proposed protocol should address the latency related unfairness by allowing extra slots to meet the unseen (yet registered) demand from the downstream. The unfairness due to flooding should be resolved by controlling the number of *REQEUSTs* that can be honored in such a way that a heavy influx of *REQUEST*s does not penalize the users closest to the head-end of the bus. The 3-Tier Fairness protocol proposed in this paper achieves all these objectives.

As is evident from the previous discussion, one of the factors affecting the unfairness problem is the actual topology of the network – the inter-nodal distance between the *active*[3] nodes along the bus, the node population at any given time, and so on. The 802.6 protocol's unfairness can be linked to such parameters and yet no such information is explicitly made available to the individual nodes. In Section 3 a protocol called *the Dynamic Assessment of Network Topology (DANT)* is described. The DANT protocol is designed to provide dynamic updates of network dependent parameters. This protocol provides *in real time* to every active node information about the active node population in the network, the length of the bus segment in its downstream and the distance between successive nodes in the network. Such information is useful in a dynamic network environment, in which new nodes may join and some existing nodes may leave the network. In Section 4, using the information provided to each active node by the DANT protocol, the *3-Tier Fairness Protocol* is presented. This protocol is designed to retain the performance of the 802.6 protocol at low loads while improving its performance at normal and heavy loads by effectively addressing the different aspects of the unfairness problem. In Section 5, the performance of the 3-Tier Fairness Protocol is studied through a number of detailed simulations.

3 The DANT Protocol

Accurate knowledge of the population in the network and its position along the bus enable a node to claim its optimum share of the bandwidth. In this paper a protocol for the Dynamic Assessment of Network Topology (DANT) is proposed — with an additional overhead of 5 bits in the DQDB slot format. The operation of the DANT protocol is presented in detail. The performance of 802.6

[3]The word *active* denotes the nodes that are part of the network and are potential candidates to contest for access.

protocol can be tuned to different levels by changing a node's perception of the node population in the network.

Need For A Dedicated Scheme : In a given interval, some nodes may go down or several new nodes may join the network. Hence with *static preset procedures,* a node may apply inappropriate limits, and the bandwidth allocation at high loads may not be fair to every node. Every node should adapt to such network parameter changes immediately and redefine access limits. Else the extent of improvement may be very much time-dependent.

Head-ends may employ *management control functions* to monitor the network population. However they may prove expensive for several reasons. Special control slots must be employed by the head-ends to gather and communicate population information. Every node must be informed individually (through dedicated slots) of its position along the bus and this leads to considerable overhead. The validity of the information communicated depends on how frequently these assessments are made. The more frequently, the greater the overhead. Gathering information such as the length of the bus or the segments of a bus in between nodes requires complex functions, and the overhead may be enormous. Thus a dedicated protocol is necessary.

The Dynamic Assessment of Network Topology (DANT) proposed here carries out dynamic updates frequently to provide to each node accurate information which can be used to control unfairness and access control.

Objectives of DANT : The word *topology* refers to the layout or configuration of the network, including the distances between successive nodes and between a node and the head-ends of a bus. This information can be measured in terms of slots. The goals of the proposed scheme are as follows : (1) to enable every node to know its position along a particular bus correctly, (2) to enable the individual nodes to estimate their upstream and downstream bus lengths in terms of integer slots, so that they can collect history information about incoming requests and use it to estimate future traffic, (3) to update this information dynamically and to make it available at every node, (4) to detect errors and restart a new assessment cycle, if some nodes go down in the middle of a cycle, (5) to ensure that new nodes joining the network do not interfere with the ongoing assessment cycle (new nodes should join the next assessment cycle), and (6) to enable every node to compute the distance (in terms of integer slots) between every *active adjacent pair* of nodes along a particular bus. Such information can be used by new access mechanisms to better the 802.6 performance.

Structure of the DANT Protocol : The structure of the proposed DANT scheme is summarized briefly. The following discussion relates to an individual assessment cycle.

- Each head-end runs an assessment cycle on its own bus. The nodes use the information collected by the cycle on a particular bus to access that bus.
- At the end of each cycle, the network-dependent parameters (node population, the position of a node and its downstream bus-length) are updated by each node. The updates are done regularly. The scheme is thus *dynamic* because it responds to changing network conditions at the end of every assessment cycle.
- In the proposed DANT scheme, the assessment cycles are continuous *i.e.* (immediately after the end of a cycle, the next one begins) as long as the network is functioning.
- The scheme operates on the contents of the newly defined *five additional bits*. The scheme progresses in *four different phases* defined as part of each assessment cycle. The initiation, execution and termination of each phase depend on the contents of these *five* bits.
- In case of inconsistency (or on the detection of an error) the current cycle can be aborted and a new assessment cycle can begin.
- During the *abort operation* every regular node will reset all the counters associated with the assessment cycle and estimates made so far. Previous estimates that are available will be used until a fresh update is effected.
- Any inconsistencies are short lived and last only through the length of the next assessment cycle.

• All four phases come in succession, and a final update is done at the end of the cycle. The information collected represents the topology of the network as it existed at the beginning of the cycle. New nodes which may have joined the network when the cycle was in progress shall not interfere with an ongoing cycle.

3.1 Definition of Additional Bits

In order to put the proposed DANT protocol to work, a slight modification in the slot format specified in the 802.6 standards is necessary. Five more bits[4] are needed to enable the operation of the assessment protocol. With the inclusion of these bits the slot format takes the following form :

Phase Header					
A	B	C	D	E	802.6 Format
1 bit	1 bit	1 bit	1 bit	1 bit	53 bytes

The five additional bits are named A, B, C, D and E.

Phase Header Bits: Bits A, B, and C are collectively termed as *phase headers*. These bits together indicate the ongoing phase of an assessment cycle. Regular nodes identify the phase by examining these bits. Head-ends issue these phase header bits, which are occasionally modified by the regular nodes in accordance with the operational scheme or algorithm.

Node Response Bit: The D-bit is used by the regular nodes in the network for writing their responses (either by setting or resetting it) as part of the operational schema.

Independent assessment cycles run on each bus. HOB_A runs its own cycle on bus-A, and HOB_B runs another cycle on bus-B. HOB_A and HOB_B maintain their own cycles separately. A regular node uses the C-bit of the opposite bus slot to respond to an ongoing cycle on a particular bus. Any node that must write its response to the ongoing cycle on bus-A shall use the C-bit of slots arriving on bus-B and vice versa.

Head-end Response Bit: The E-bit is used by a head-end to write its response to the assessment cycle controlled by its peer. As before, the response goes in the opposite bus. Suppose HOB_B needs to respond to a cycle controlled by HOB_A on bus-A. Then HOB_B writes its response in the E-bit of its own slot and puts the slot on bus-B. This is done by HOB_B issuing a slot on bus-B with $E = 1$. Otherwise, all HOB_B slots have $E = 0$.

The *phase header* values are related to the different phases of the protocol and are presented in Table-1. There are eight different values for the phase header. Every value carries a specific meaning associated with the assessment cycle. The *phase header* information passing on a particular bus is related to the ongoing cycle of the same bus.

The 'x' value for a bit indicates that it can be either a '1' or '0'. This implies that the actual value does not relate to the context of discussion and hence is inconsequential. For example, a regular node might alter the response bit (D-bit) in a bus-A slot (as a response to the ongoing cycle in bus-B) and not affect the discussion of the cycle on bus-A.

3.2 Operation of the DANT Protocol

The following discussion is restricted to the assessment cycle run and controlled by HOB_A on bus-A. The following facts may be noted about the assessment cycle :

[4]The data unit for the Physical Layer Service is an octet. This may imply that an additional overhead of one octet is needed to support the proposed protocol. The extra bits may be used to support additional functions for future enhancements.

Table 1: Meaning of Phase Header Bits

Phase Headers (A, B, C bits)	Meaning associated within an assessment cycle
1 1 1 xx	Beginning of a new assessment cycle.
0 0 0 xx	Abort current cycle. (Issued only by a head-end.)
0 0 1 xx	End-the-phase slot : End the current phase of an ongoing cycle and begin the next phase.
1 1 0 xx	Phase-I Slot
0 1 1 xx	Phase-II Slot
0 1 0 xx	Phase-III Slot
1 0 x xx	Phase-IV Slot

- The cycle is run on bus-A, and responses to this cycle are written or observed on bus-B slots. Thus only phase headers of bus-A slots and the D–E bits of bus-B slots are important for this cycle.
- The responses to this cycle by regular nodes are written on the D-bit of bus-B slots by a regular node.
- HOB_B responds to the ongoing cycle of bus-A by issuing a slot on bus-B with its E-bit set.
- An 'x' shall indicate that the corresponding bit can either be a '1' or a '0' and is irrelevant for that part of discussion.
- Responses to the cycle on bus-B are written by regular nodes on the D-bit of bus-A slots. This does not interfere with the operation of the cycle on bus-A.
- HOB_A may still issue a slot with its E-bit set, if necessary, as a response to the cycle maintained by HOB_B. This does not interfere with the functioning of the bus-A cycle.
- The head-end and regular nodes maintain a few counters as well. Specific information regarding the need and usage of the counters is presented during the discussion on individual phases.

- The operation can be extended to the assessment cycle maintained by HOB_B on bus-B on similar lines.

3.2.1 Initiating an Assessment Cycle

HOB_A issues a 1 1 1 0x slot on bus-A. Only one such slot is issued. Every regular node then prepares for a new assessment cycle which proceeds in a phased manner. In short, the 1 1 1 0x slot serves as a declaration of an impending new cycle to every node in the network.

A regular node that sees a 1 1 1 1x slot understands that a new cycle is impending. It is possible that even though HOB_A issues a 1 1 1 0x slot, some node may still modify the D-bit of this particular slot in response to the cycle in bus-B.

Phase-I of the assessment cycle begins immediately after HOB_A issues its 1 1 1 0x slot. Once HOB_A starts issuing 1 1 0 0x slots, it continues to do so throughout this phase.

3.2.2 Operation of Phase-I

The goals of the first phase of the assessment cycle are :

- To enable an estimate by HOB_A of the bus-length in terms of integer slots.
- To enable an estimate by every regular node of the downstream bus-length[5] in terms of integer slots.

Role of Head-ends : Immediately after its 1 1 1 0x slot, HOB_A starts issuing 1 1 0 0x slots and continues to do so throughout this phase.

HOB_A also looks for incoming bus-B slots with the E-bit set to 1, indicating that HOB_B acknowledges the receipt of new cycle declaration on bus-A. Also HOB_A counts the phase-I slots it issues until it receives a response from HOB_B.

When the 1 1 1 xx slot is seen by HOB_B, it should respond by setting the E-bit of the next immediate slot on bus-B to 1.

When the acknowledgment from HOB_B is received by HOB_A, phase-I ends, and HOB_A immediately stops counting the phase-I slots issued thus far. Half the counter value is an estimate of the bus length in terms of integer slots.

$$\text{Length of bus-A} = \left\lceil \frac{\text{phase-I slot counter value}}{2} \right\rceil$$

After HOB_B's response is received in phase-I HOB_A issues a 0 0 1 0x slot that declares to every recipient node the end of phase-I.

Role of Regular Nodes : The receipt of a 1 1 1 xx slot indicates to a regular node the beginning of Phase-I on bus-A. Immediately after receipt, the regular node starts counting the Phase-I slots sent on bus-A.

Each regular node also keeps a watch on bus-B. When it sees a slot on bus-B with the E-bit set to 1, it stops the counter. Half the value of the counter provides the node with the length of bus-A, downstream from the node.

$$\text{Bus-A downstream length} = \left\lceil \frac{\text{phase-I counter value}}{2} \right\rceil$$

Thus each regular node is able to estimate its downstream bus-length (in terms of integer slots) by itself.

When the *end-the-phase slot* 0 0 1 xx arrives on bus-A, every regular node updates its knowledge of the downstream bus-length by saving the estimate made in this phase.

Also it is possible for every regular node to estimate the length of bus-A, upstream from the node by counting the phase-I slots arriving on bus-A during the interval between the time a slot with E-bit = 1 is seen on bus-B to the time the end-the-phase slot on bus-A is seen. Half the counter value (upper ceiling applied) gives the length of bus-A upstream from the node.

3.2.3 Operation of Phase-II

During this phase the head-end estimates the *active population* in the network. The word *active* refers to the nodes that are participating in the network. Unlike the bus-length measurements of phase-I, the

[5] For any node the portion of bus-A extending from the node itself to HOB_B corresponds to the downstream bus-length on bus-A. This length is expressed in terms of integer slots.

active population may change often as nodes go down or become active and join the network. During this phase the regular nodes do nothing for themselves except answering the call of attendance issued by HOB_A on bus-A under the rules of the DANT protocol as discussed below.

Commencement of Phase-II : Immediately after issuing an end-the-phase slot to conclude phase-I, HOB_A begins the next phase by issuing a `0 1 1 0x` slot. It continues to issue `0 1 1 0x` slots throughout phase-II.

Role of Regular Nodes : On receipt of the very first phase-II slot `0 1 1 xx` , every regular node must write a response on the D-bit of the next available bus-B slot.

Contention for writing such a response is possible because of the concentration of nodes. Especially when the bus-length is smaller than the active population of nodes, (e.g., 100 nodes active along a bus of length 30 slots), special attention is required. In this example, at the end of 30 slots, HOB_B sends an $E = 1$ response to HOB_A authorizing it to end its phase-II. This response slot reaches HOB_A 30 slots later along bus-B. By this time only about 60 nodes have sent their responses. HOB_A can have no knowledge of impending responses.

To overcome this difficulty, a restriction is imposed. A regular node will not allow a slot on bus-B whose E-bit is set if it has not answered the call of attendance in phase-II. Instead it will reset [6] the E-bit to 0 and wait its turn to write its response. At the time of its own D-bit response, it will also set the E-bit in the same slot. In this way every regular node should be able write its response.

Role of Head-ends : After initiating phase-II HOB_A keeps a watch on the slots arriving on bus-B, looking for responses on either D-bit or E-bit. D-bit responses are the answers from the regular nodes to the call of attendance. E-bit response informs HOB_A to conclude phase-II.

During this phase HOB_A counts the D-bits responses received on bus-B and terminates the count when it sees a bus-B slot with E-bit set. If the bus-B slot with the E-bit set also has its D-bit set, then it too is counted as a response.

The counter value indicates the *active* population in the network. The E-bit response should be written by HOB_B in the next slot on bus-B, immediately following the receipt of the first slot of phase-II `0 1 1 xx` on bus-A.

Immediately after evaluating the active population, HOB_A concludes phase-II by issuing an *end-the-phase* slot as `0 0 1 0x` on bus-B.

Error and Abort : It is possible that a regular node may go down after resetting the E-bit but before writing it back. This can be detected by the protocol as follows. HOB_A has the estimate of bus-length from phase-I, say B_s. If the E-bit response is not seen by HOB_A on bus-B in $2(B_s + 1)$ slots, then either there are more active nodes than B_s or a regular node may have gone down after resetting the E-bit to 0 but before setting back to 1. HOB_A should check every incoming bus-B slot beyond the $2(B_s + 1)^{th}$ slot. HOB_A should observe a sequence of slots having the D-bit set and the E-bit 0, followed by a slot having both the D-bit and E-bit set to 1. If this rule is violated then HOB_A will abort the cycle by issuing an *abort* slot as `0 0 0 0x` on bus-A.

Thus during phase-II, the HOB_A is able to assess the active network population. During phase-II, regular nodes make no assessments of their own.

3.2.4 Operation of Phase-III

The goals of phase-III are to enable the head-ends (in this discussion, HOB_A) to map the topology information of the network and allow the regular nodes to estimate the number of active nodes in the upstream of the bus on which this phase runs (in this discussion, the number of upstream nodes with respect to bus-A).

[6] A node writes into a slot via a logical OR. Resetting of the E-bit might pose a problem and needs specific attention.

Topology refers to the inter-nodal distances between successive nodes along the bus in terms of integer slots, taking upper ceiling wherever necessary. Because of the use of the upper ceiling criteria, the sum of individual inter-nodal distances (in integer slots) need not sum up to the bus-length (in integer slots).

The knowledge of the active population in the network acquired in phase-II is made use of by the head-end during this phase. In order to store the inter-nodal distance information, the head-end maintains an array of $(N+1)$ elements for a network of N active nodes. It also employs a counter to count the phase-III slots issued.

Role of Head-ends : HOB_A makes use of the information acquired in previous two phases, namely the active node population N and the bus-length B_s.

Throughout phase-III HOB_A issues | 0 1 0 0x | slots on bus-A and keeps a count of the number of such slots issued. It ends phase-III by issuing an end-the-phase slot | 0 0 1 0x | , after getting a bus-B slot with its E-bit set to 1 by HOB_B.

Every regular node that has answered the call of attendance will send a response through the D-bit of bus-B slots. When N such responses and the E-bit response from bus-B are received, HOB_A concludes the phase by issuing an *end-the-phase* | 0 0 1 0x | slot on bus-A. The evaluation of the topology and other details are presented as a separate item below.

HOB_B sends its E-bit response only after receiving the first phase-III slot on bus-A as | 0 1 0 xx | , and the response is sent in the very next slot on bus-B.

Unlike phase-II wherein the regular node responses may arrive in succession with every bus-B slot, the rules specified for the regular nodes in phase-III force responses to be sent in a way that has a direct relationship to the inter-nodal distances in slots. As a result, the manner in which responses arrive at HOB_A helps map the topology of the network.

Topology Mapping by Head-end : Based on the way HOB_A receives the D-bit responses on bus-B, HOB_A can evaluate the inter-nodal distances (or map the topology) between successive nodes along bus-A as follows :

- As soon as phase-III starts, HOB_A starts a counter, say *S_CTR*, which is incremented with every incoming bus-B slot since the commencement of phase-III.
- HOB_A initializes a response counter, say *R_CTR*, which is incremented with every incoming bus-B slot with its D-bit response set.
- Since the active node population is estimated to be N, HOB_A must open an array of $(N+1)$ elements to store the inter-node distances. Including the two head-ends, there are $(N+2)$ nodes in the network and $(N+1)$ inter-nodal distances measured in terms of slot lengths.
- Let the array name be *lap-distance*. Then the individual elements have the following meaning.

lap-distance [1]	$\Rightarrow$	From HOB_A to node-1
lap-distance [2]	$\Rightarrow$	From node-1 to node-2
$\vdots$	$\vdots$	$\vdots$
lap-distance [N]	$\Rightarrow$	From node-$N-1$ to node-N
lap-distance [N+1]	$\Rightarrow$	From node-N to HOB_B

- Another counter called *L_CTR* (lap counter) is reset and started. This counter helps keep track of slots in between successive D-bit responses received on bus-B. The counter is incremented with every incoming bus-B slot. When an incoming bus-B slot has its D-bit set, the counter is incremented, its contents are stored in the *lap-distance* array, and the counter is reset to zero.
- To summarize the evaluation done by the DANT protocol, HOB_A does the following.

1. At the beginning of phase-III, HOB_A initializes all the three counters, namely *S_CTR, R_CTR, L_CTR,* to zero.

2. When HOB_A receives a slot on bus-B with the D-bit not set, it does the following:

S_CTR ++;
if ($S\_CTR \leq (2(B_s + 1) + N)$) {
    L_CTR ++;
} else {
    ABORT the current cycle.
}

3. When HOB_A receives the first bus-B slot with its D-bit set, it does the following:

S_CTR ++;
R_CTR ++;
L_CTR ++;
lap-distance [R_CTR] $= \left\lceil \frac{L\_CTR}{2} \right\rceil$;
$L\_CTR = 0$;

4. For every subsequent bus-B slot received by HOB_A with the D-bit set and the E-bit not set HOB_A does the following:

S_CTR ++;
R_CTR ++;
L_CTR ++;
lap-distance [R_CTR] $= \lceil L\_CTR \rceil$;
$L\_CTR = 0$;

It is possible that both the D-bit and the E-bit of a bus-B slot may be set. This can happen only when the distance between the last node along bus-A and HOB_B is less than one slot. This slot is handled according to the actions specified for a slot with the E-bit set.

5. Phase-III comes to an end when HOB_A receives a bus-B slot with its E-bit set. The following actions occur at HOB_A.

S_CTR ++;
R_CTR ++
L_CTR ++;
lap-distance [R_CTR] $= \lceil L\_CTR \rceil$;
if (D-bit == 1) {
    R_CTR ++
    lap-distance [R_CTR] $= 1$;
}
$L\_CTR = 0$;
if ($R\_CTR == N$) {
    issue *end-the-phase* slot
} else {
    ABORT the current cycle.
}

Role of Regular Nodes : Every regular node that has answered the call of attendance will send a response through the D-bit of bus-B slots.

After seeing the *end-the-phase* slot [0 0 1 0x] during phase-II, every regular node starts phase-III and waits for the first [0 1 0 xx] slot on bus-A.

After the commencement of phase-III, it is possible that a node may continue to receive [0 1 1 xx] , for reasons to be explained below. Every regular node initializes a counter as soon as phase-III commences and counts the number of such [0 1 1 xx] slots received before the arrival of the first phase-III slot.

On seeing the first phase-III slot, every regular node sets the C-bit of that slot before allowing it to pass on bus-A. Each node also writes a response in the D-bit of the *immediate next* slot on bus-B. Second and subsequent phase-III slots require no action on the part of regular nodes.

Because the C-bit of phase-III slot is modified by successive nodes along bus-A, different slots are seen as the very first [0 1 0 xx] slot of phase-III by different nodes along bus-A. Hence regular nodes encounter no contention when writing their D-bit responses in next slot on bus-B.

The setting of the C-bit makes some phase-III slots on bus-A appear like phase-II slots (with phase headers transformed into [0 1 1 xx]). This is not problematic because the phases are ordered and the end of phase-II has already been declared as over (by the specific *end-the-phase* slot).

Only in phase-III may a regular node modify a phase header bit (C-bit), and this is permitted only once. Every regular node keeps a count of the number of [0 1 1 xx] slots received before the first phase-III slot. At the end of the phase *this counter value yields the number of nodes in the upstream of bus-A.*

After the commencement of phase-III, the upstream nodes modify and send [0 1 1 xx] slots. However once the first phase-III slot [0 1 0 xx] is seen, then a regular node will see only [0 1 0 xx] slots till the end of phase-III.

The participation of regular nodes in this phase is confined to writing their responses.

Error and Abort : It is possible that a regular node that answered the call of attendance might have gone down and hence no longer be active. This eventuality can be traced or detected as follows:

Even in the worst possible scenario of the distribution along the bus, all regular nodes should be able to write D-bit responses within $(2(B_s + 1) + N)$ slots on bus-B from the time phase-III started. Thus if HOB_A fails to read N (the active node population estimated in phase-II) D-bit responses within the said number of bus-B slots since the commencement of phase-III, then possibly a node may have gone down. In such a situation HOB_A *aborts* the cycle by issuing a [0 0 0 0x] slot on bus-A.

3.2.5 Operation of Phase-IV

During phase-III, HOB_A has collected information on network topology. Phase-IV is aimed at transmitting this topology information to the regular nodes. The regular nodes collect the information from bus-A.

Role of Head-ends : Throughout phase-IV, HOB_A sends two types of slots, [1 0 1 0x] and [1 0 0 0x] on the bus-A.

Every [1 0 0 0x] slot from HOB_A corresponds to one unit of inter-nodal distance, and every [1 0 1 0x] slot from HOB_A indicates the presence of an active node. Together these slots are sent in a fashion that represents the topology information of the network, collected by HOB_A during previous phases. For example, suppose *lap-distance* [1] = 3 and *lap-distance* [2] = 5. During phase-IV, HOB_A will send the following sequence of slots to carry the information.

First 3 slots as	1 0 0 0x	⇒	HOB_A to node-1 slot distance
Next slot as	1 0 1 0x	⇒	Node-1 present
Next 5 slots as	1 0 0 0x	⇒	Node-1 to node-2 slot distance
Next slot as	1 0 1 0x	⇒	Node-2 present
⋮	⋮	⋮	⋮
One slot as	1 0 1 0x	⇒	HOB_B present
Last slot as	0 0 1 0x	⇒	*end-the-phase* slot

Exactly $(N+1)$ slots are issued as 1 0 1 0x from HOB_A, one slot for each active node population in the network and an additional slot for the other head-end.

Immediately after issuing the 1 0 1 0x slot corresponding to the presence of HOB_B, the next slot is issued as the *end-the-phase* slot by HOB_A. The last 1 0 1 0x implies the end of the phase-IV and the current assessment cycle.

Unlike previous phases, HOB_A does not collect any information or depend on any response bits along bus-B to run or terminate this final phase. HOB_A runs this phase on its own.

Role of Regular Nodes : Each regular node initializes a counter at the beginning of phase-IV and then counts the number of 1 0 1 xx slots seen in phase-IV. Let this counter be denoted by *N_CTR*. At the end of phase-IV this *N_CTR* value decremented by one will represent the active node population in the network.

Similar to the evaluation of the topology explained in phase-III, every regular node can read the topology information from the incoming slots. Every regular node initializes a new counter, say *LA_CTR* at the beginning of phase-IV. Throughout phase-IV, *LA_CTR* counts the 1 0 0 xx slots (arriving on bus-A) until a 1 0 1 xx slot is seen. At this point, the *LA_CTR* value is moved to *lap_distance* [*N_CTR*]. Also at this point *LA_CTR* is again reset to zero and counting begins again. When *end-the-phase* slot is seen, every node can evaluate its position along bus-B from available information as follows. The number of active upstream nodes along bus-A is known from phase-III. Adding one to this value gives the actual position of a regular node along the bus. Also, the active node population in the network is known from phase-IV. The *N_CTR* used in phase-IV contains this value, as explained previously.

Error and Abort : In this phase the failure of a node can not be traced, since the head-end receives no information or response from the regular nodes. However this is the last phase, and the next assessment cycle should be able to remove any discrepancy.

3.3 Length of an Assessment Cycle and Slot Format

For each assessment cycle, five slots are needed to declare the commencement of a new cycle and declare the end of each of the four phases. The length of each phase can also be approximately estimated. Consider a network of N nodes with bus-length B_s slots. It is not very difficult to see that the length of a DANT cycle runs up to $(8B_s + 3N + 5)$ slots. As an example, consider a network operating at 44.7 Mbps with 100 nodes located on a bus of length 50 Kms. This corresponds to a slot interval of 8.59 μsecs and a bus-length of about 30 slots. Hence the DANT cycle can have a maximum length of 545 slots. This translates to about 4.68 msecs. Thus, for this example, a dynamic update is possible every 4.68 msecs. It may be possible to trade-off overhead for the cycle length by retaining the DANT algorithm but using control slots, instead of introducing the bits in the slot format of 802.6.

3.4 Safeguards for the DANT Protocol

A regular node that newly joins a network should not participate half way through the ongoing assessment cycle on a particular bus. Instead it should wait for the new cycle declaration 1 1 1 xx slot to be seen on that bus.

When a node goes down in the middle of an assessment cycle, then the cycle may be aborted and a new cycle started. The failure of the node that caused the abort will also be detected in the next assessment cycle.

If a regular node finds the phases of an ongoing cycle out of sequence (this may happen if there is a bit error due to the noise in the channel), then the node can *ignore* the ongoing cycle and wait for the commencement of the next assessment cycle. Update in this case will be delayed until the end of the next cycle. Only head-ends can issue the abort slot. Regular nodes can not abort on their own.

The proposed DANT protocol thus provides the head-ends and each active node in a DQDB network with real time information about the active node population, the internode distance, the position of each node along the bus and the length of a node's downstream bus segment. DANT's current implementation introduces an overhead of 5 bits per slot, but alternative implementations which retain the current slot structure of 802.6 and implement the DANT protocol through the use of periodically issued control slots is possible.

4 The 3-Tier Fairness Protocol

4.1 Remedy for the Heavy Load Unfairness

The dominant cause of heavy load access unfairness is the *flooding phenomenon*. In order to avoid this type of unfairness, limits must be applied to the *REQUEST* counter values at individual nodes. To ensure a fair bandwidth share for every node, an *access protection limit* for a node must be a function of the node population in the network and its position along the bus.

Similarly to Filipiak [9] access protection scheme proposed in this paper involves the introduction of an *upper protection limit*. Let the upper protection limit of a node-i with respect to accessing Bus-A be denoted by $\hat{P}_i^A$. Assume that a packet arrives for access and the node's $RQ > 0$. Under the upper protection scheme,

$$CD = \min\left\{RQ, \hat{P}_i^A\right\} \quad .$$

4.1.1 Source-Destination Pair Concept

The traffic offered to the network is assumed to be unbiased in the sense that whenever a node has a packet, it is destined for any one of the remaining nodes with equal probability. This kind of network activity is natural and is referred to as *symmetric traffic*.

The *Access Protection Scheme* (referred to as APS [7]) presented in this paper uses the source-destination pair criteria to estimate the upper protection limits. The logic behind the source-destination concept is that the traffic in the network is assumed to be symmetric and thus the bandwidth claimed by a particular node is proportional to the number of potential destination nodes along the bus. As explained in the previous section, the DANT protocol provides each node with its position along a bus as well as the number of its downstream nodes even under changing network conditions. Thus the protection limits of the APS are valid even in a dynamic network environment of varying node population.

[7] The APS protocol is part of the 3-Tier fairness protocol and handles the heavy load situations. From now on, the term APS protocol performance shall refer to the heavy load performance of the 3-Tier fairness protocol also.

Let node-i be the i^{th} node from the frame generating head-end along the direction of a bus in a network with a population of N nodes at a given instant. Node-i then has $(N-i)$ possible destinations. Thus the total possible source-destination pairs along a bus is :

$$\sum_{i=1}^{N}(N-i) = \frac{N(N-1)}{2}$$

For node-i, let the bandwidth that is to be guaranteed by the APS be denoted as BW_i, where $\mathrm{BW}_i$is a fraction of the overall channel capacity[8] at heavy loads. Under the access protection scheme, BW_i depends on the particular node's downstream nodes and the total possible source-destination pairs. Accordingly,

$$\begin{aligned} \mathrm{BW}_i &= \frac{\text{No. of potential destinations for node-}i}{\text{Total no. of source-destination pairs}} \\ &= \frac{(N-i)}{\frac{N(N-1)}{2}} = \frac{2(N-i)}{N(N-1)} \end{aligned} \tag{1}$$

Let $\hat{P}_i$ be the upper protection limit applied by node-i. Node-i is guaranteed at least $\frac{1}{\hat{P}_i+1}$ of the bandwidth left unused by its upstream nodes. The bandwidth that is used by all upstream nodes is the sum of the bandwidth guaranteed to them individually by the protocol. At heavy loads this can be written analytically as :

$$\mathrm{BW}_i = \frac{1}{\hat{P}_i+1}\left[1 - \sum_{k=1}^{i-1}\mathrm{BW}_k\right] \qquad \text{for } 1 \le i \le N \tag{2}$$

Using Equation 1 and Equation 2,

$$\hat{P}_i + 1 = \frac{N(N-1)}{2(N-i)}\left[1 - \frac{2}{N(N-1)}\sum_{k=1}^{i-1}(N-k)\right] \quad .$$

Therefore,

$$\hat{P}_i = \frac{N-(i+1)}{2} \qquad \text{for } 1 \le i \le N \quad . \tag{3}$$

Table-2 shows the bandwidth and the individual protection limits for every node in a 10-node network.

Under APS, a node operates with two different limits. For example, Node-i uses the following protection limits to access each of the buses :

$$\hat{P}_i^A = \frac{N-(i+1)}{2} \quad \text{and} \quad \hat{P}_i^B = \frac{(i-2)}{2}$$

Notice that both the limits are expressed in terms of i and N alone, in the above equations.

The original DQDB protocol can be considered as having a limiting behavior with $\hat{P}_i^A = \infty$, especially with large internode distances. The maximum value of the RQ counter at a node should be a function of the initial conditions and the internode distance [26]. With the *exhaustive scheme* of filing a *REQUEST* in 802.6, there is a limit to the value that an RQ counter can assume, since a new request is filed only when transmission is completed for the previously-filed request. Thus the RQ counter of node-i can have a maximum value of $(N-i)$ only. Values greater than $(N-i)$ may not be realized under the *exhaustive scheme* of filing requests. This implies that the 802.6 protocol

[8] Hence $\sum_{i=1}^{N}\mathrm{BW}_i = 1$ is true.

Table 2: Illustration of protection limits in a small network

Protection Limits in a NW of 10 Nodes				
	For Bus-A		For Bus-B	
Node-id	$\hat{P}_i^A$		$\hat{P}_i^B$	
i	$\frac{N-(i+1)}{2}$	BW_i^A	$\frac{i-2}{2}$	BW_i^B
1	4.0	18/90	-0.5	0
2	3.5	16/90	0.0	2/90
3	3.0	14/90	0.5	4/90
4	2.5	12/90	1.0	6/90
5	2.0	10/90	1.5	8/90
6	1.5	8/90	2.0	10/90
7	1.0	6/90	2.5	12/90
8	0.5	4/90	3.0	14/90
9	0.0	2/90	3.5	16/90
10	-0.5	0	4.0	18/90

can be thought of as an access protection scheme with an upper protection limit defined as :

$$\hat{P}_i^A = (N - i) \ .$$

The above upper protection limit produces the same performance as the 802.6 protocol for several simulated network configurations.

Compared with the 802.6 protocol, the APS protocol achieves a performance improvement by changing the definition of the access protection limits from $(N-i)$ to $(N-(i+1))/2$. In general, the access protection limits can be defined in terms of *an access weight parameter* , as follows :

$$\hat{P}_i^A = \frac{(N-i)(1+\alpha) - (1-\alpha)}{2}$$

subject to the condition[9] $0 \leq \alpha \leq 1$. With the above definition, the 802.6 protocol [10] corresponds to $\alpha = 1$ and the APS protocol corresponds to $\alpha = 0$. The performance characteristics optimum for each type of load can be achieved by varying the value of α, for $0 \leq \alpha \leq 1$. A successful characterization of the optimal value of the *weight parameter* as a function of the workload type would result in a family of access protection schemes, each tuned for a particular load type. This mechanism is referred to as *AlphaTuning*.

4.2 Remedy for the Latency-Related Unfairness

Any strategy aimed at addressing latency-related unfairness should provide each node with an accurate estimate of the number of downstream *REQUEST*s. In this subsection the *Anticipatory Demand Scheme (ADS)* is proposed. The ADS uses a history-based estimate of the number of downstream *REQUEST*s. In Table 3, the terminology and variables needed to describe the ADS are given. Table 3 and the rest of this subsection discuss parameters with respect to port-A (*i.e.*, with regard to access to the bus-A). Similar variables are used with respect to port-B. Again, a *single priority level* is assumed. In case of multiple priority levels, individual counters must be maintained for every priority level.

The additional functionalities needed to address latency-related unfairness are as follows:

- Every node (more specifically, every bus-port[11]) collects a record of the demand (*i.e.*, requests

[9] α can not take values more than 1, for that would exceed 802.6 protocol under which a node can have utmost one pending *REQUEST* only. Thus using $\alpha > 1$ in effect will have no impact and the basic 802.6 protocol performance is retained.

[10] Of 802.6 protocol and the APS protocol protection limit definitions, 802.6 is the upper bound. $P_{802.6} - P_{\text{APS}} = \frac{(N-i+1)}{2}$ and since $i \leq N$ this quantity is always positive.

[11] A node can be conceived as having two ports one connecting to bus-A and the other to connect to bus-B.

Table 3: Terminology of Parameters Used

Parameter	Comments
Bus-port A	The port connecting a node to bus-A.
ds_slots_A	Downstream bus length (in *integer* slots) of a node along bus-A. The portion of bus-A between the node itself and HOB_B.
ds_cycle_A	Name for *REQUEST* count cycle that counts the incoming *REQUEST*s on bus-B.
ds_cycle_count_A	Counter to assist the running of *ds_cycle_A*. Reset every time, when it reaches *ds_slots_A* value.
ds_RQ_A	Counter that counts the actual *REQUEST*s coming along on bus-B, during *ds_cycle_A*. Reset at the end of the cycle, after saving the count in *ds_add_RQ_A*.
ds_add_RQ_A	Bus-port-A of a node will allow this many extra slots, when its self-request is enqueued. Updated as some function of *ds_RQ_A* at the end of every *ds_cycle_A* slots.
up_cycle_A	Name for the cycle that counts *empty* slots passing on bus-A.
up_cycle_count_A	Counter to assist the running of *up_cycle_A*. Reset every time, when it reaches *ds_slots_A* value.
ds_emp_A	Counter that counts the *empty* incoming slots along bus-A during *up_cycle_A*. Reset at the end of the cycle, after saving the count in *ds_emp_rec_A*.
ds_emp_rec_A	Record of *empty slots* passing on bus-A during the previous *up_cycle_A*. Updated with *ds_emp_A*, at the end of every *ds_cycle_A* slots.

received) during every cycle of incoming slots.

- The duration of the estimation cycle *ds_slots* of a particular node equals its downstream bus-length in slots. The DANT protocol provides the capabilities of estimating this length. This estimate is updated with every estimation cycle.

- A new cycle begins with the resetting of the *REQUEST* counter. When one such cycle is in progress and a node wants to access the corresponding bus, a few extra slots will be allowed — based on the history record of *REQUEST*s received in its previous cycle.

- Similarly a count of *empty* slots arriving on a bus should be maintained at every bus-port. The length of such a cycle will also be *ds_slots*. This cycle and the estimation cycle are run independently of each of each other.

- Δ extra slots are allowed if and only if the demand is greater than the unused bandwidth (empty slots) observed. *I.e.* extra slots are allowed if and only if $ds\_RQ\_A > ds\_emp\_rec\_A$. Otherwise no extra slots are allowed.

- If traffic demand exceeds the observed unused bandwidth on the access bus, then Δ is estimated as follows :

  if $(ds\_RQ\_A > ds\_emp\_rec\_A)$ {
  $ds\_add\_RQ\_A = \left\lceil \frac{ds\_RQ\_A}{ds\_slots\_A} * ds\_RQ\_A \right\rceil$
  } else {
  $ds\_add\_RQ\_A = 0$
  }

  The estimated value is stored in the $ds\_add\_RQ\_A$ record, which is updated at the end of every request count cycle. Δ is derived from this record whenever need arises as :

  $$\Delta = ds\_add\_RQ\_A \quad .$$

  The reasoning behind the above estimate of Δ is that the same demand is assumed to persist between successive cycles.

- The demand ds_RQ_A is also weighed in proportion to the cycle length as $\frac{ds\_RQ\_A}{ds\_slots\_A} * ds\_RQ\_A$. This is necessitated by the fact that the modified protocol tends to deviate from the 802.6 protocol, only for the duration of the extra slots. Thus the anticipated traffic is proportionately estimated.

The estimate of the downstream *REQUEST*s in progress Δ is derived from :

$$\Delta = ds\_add\_RQ\_A \quad .$$

$ds\_add\_RQ\_A$ is updated at the end of every demand history cycle (*i.e.*, ds_cycle_A). The two cycles are maintained independently of each other. Both cycles up_cycle_A and ds_cycle_A have the same length equal to ds_slots_A. $ds\_add\_RQ\_A$ is updated as a function of the ds_RQ_A counter value at the end of every ds_cycle_A cycle (which runs on bus-B). Similarly $ds\_emp\_rec\_A$ is updated with the ds_emp_A counter value at the end of every up_cycle_A cycle (which runs on bus-A).

4.3 The 3-Tier Fairness Protocol

In this section modification of the 802.6 protocol into the proposed 3-Tier Fairness protocol is presented. The proposed scheme incorporates the access mechanisms of the 802.6, the ADS and the APS. As per the 802.6 protocol, when a node changes state from *IDLE* to *COUNTDOWN*, the *RQ* counter value is loaded into *CD* counter and the *RQ* counter is reset to zero. *I.e.*,

$CD \leftarrow RQ$

$RQ = 0$

Change state from *IDLE* to *COUNTDOWN*.

This is modified as follows, to deal with both low and heavy load situations.

- The Δ is the number of extra slots to be allowed. The estimate of the downstream *REQUEST*s in progress Δ is derived from :

  $$\Delta = ds\_add\_RQ\_A \quad .$$

  The estimation policy is as explained in the previous section. The modified scheme would require then following actions.

```
if (RQ > P̂_i^A) {
   CD ← P̂_i^A
} else {
   if ((RQ + Δ) > P̂_i^A) {
      CD ← P̂_i^A
   } else {
      if ((RQ + Δ) > 0) {
         CD ← (RQ + Δ)
      } else {
         CD ← 0
      }
   }
}
RQ = RQ − CD
Change state from IDLE to COUNTDOWN.
```

The value of Δ depends on the definition of the policy that is used to arrive at an estimate of extra slots. Alternate estimate policies for Δ can be easily implemented.

The Bandwidth Balancing Mechanism (BBM) [24] introduces bandwidth wastage[12], which, especially at low loads can be avoided by using the 3-tier structured access protocol.

5 Simulation Details

In the simulations, the channel capacity is 44.7 Mbps and the 802.6 slot format of 53 bytes is used. Zero slot overhead is assumed. The messages are assumed to fit into a single slot. Further no *standby* state is employed. The nodes are assumed to have infinite buffer length and only single priority messages are considered. Multiple packet sizes are also considered–with an incoming message size being 1, 8 or 20 packets. The simulations are carried out with two networks having 100 and 25 nodes at the inter-node distances of 0.5 Kms and 2 Kms respectively. Messages generated at a node follow the

[12]Under BBM, one extra slot is allowed once in every N (N is fixed number) attempts by the individual nodes for access to a bus. A *trigger* counter is maintained to facilitate this.

Poisson arrival pattern. Two load types, symmetric load and equal probability load, are considered. The simulations are run for 2 seconds of network activity.

In the literature the term *heavy load*is typically used to mean that the queues of all active nodes are never empty. In this paper the term heavy load means a *high load* situation wherein the stochastically generated traffic approaches or exceeds the capacity of the channel. Every node generates traffic according to the same distribution. The destination selection policy forces a message to seek access to a particular access bus and leads to different load types.

5.1 Performance Characteristics

The performance characteristics considered to evaluate the merit of an access scheme are the average access delay and the success rate of a node.

Average Access Delay

The access delay of a packet is defined as the period between the time a packet enters the network and the instant at which it is put on the access bus slot. In the computation of average delay, the delay encountered by only the packets that could be successfully transmitted is considered. The delay of a packet is measured in terms of the number of slots that pass through the bus when the particular packet is waiting for access. Especially at heavy loads the packet queue length and hence the average delay also tend to be very high. As a matter of fact, the longer the simulation period the higher is the average access delay–since infinite buffer sizes are assumed.

Success Rate

The success rate is defined as the ratio between the number of successful transmissions of packets (or slots claimed) by a node to the total number of packets originally arrived at a node (bound for some node in its downstream) along a particular access bus.

The above two performance indices are plotted against the node index as performance characteristics, to investigate the performance of the access schemes.

6 Discussion of the Simulation Studies

In this section the performance of the 3-Tier Fairness protocol is presented, based on the results of simulations. The discussion is classified under the various load type headings and extends over the entire range of traffic demand. The load values studied are 0.2, 0.4, 0.75, 0.9, 1.0 and 1.2. The performance study is with respect to accessing a single bus. Additional performance curves[13] are also included with regard to multiple packet sized messages and various configurations.

Fig-A and *Fig-B* show the performance with single packet messages, under symmetric and equal probability loads respectively with a 100-node network. *Fig-C* shows the performance of a 100-node network with symmetric load and multiple packet sized message arrivals. *Fig-D* shows the performance of a 25-node network with equal probability load, having multiple packet sized message arrivals. The success rate characteristics are shown only for the heavy load single packet case. The 802.6 performance characteristics are shown with *solid* lines, with those of the 3-Tier protocol depicted by *dotted* lines.

6.1 Symmetric Load Traffic

Under this traffic the volume of traffic (the total number packets queued up for access to a bus) decreases along the direction of the access bus. The performance of the 3-Tier Fairness protocol is compared with the 802.6 performance, and the following observations are made:

[13] In [32], an entire range of simulation results are available. In this paper only a few selected performance curves are presented, for space considerations.

1. At loads of 0.2 and 0.4, the performance of the 802.6 protocol is retained.

2. At a load of 0.75, the nodes in the first-half of the bus have their access delays increased by one or two slots. Few end-nodes experience a marginal access delays decrease.

3. At a load of 0.9, the nodes in the first-half of the access bus experience an increase in their access delays of a few slots The rest of the nodes experience a decrease in access delay by a few slots.

4. At a load of 1.0, the first few nodes along the access bus have sharp peaks in their access delays under the 802.6 protocol. With the modified protocol, such peaks are considerably diminished in magnitude.

5. At a heavy load of 1.2, the modified protocol produces flat characteristics of access delay and success rate all along the bus.

Thus the 3-Tier fairness protocol produces *satisfactory* performance at all loads types. At low loads the 3-Tier protocol is exactly the same as as 802.6, as if only the 802.6 protocol were running. At normal loads the 3-Tier fairness protocol produces exactly the ADS performance, as if the ADS protocol alone were being used. At high loads the behavior of the 3-Tier protocol is very much the same as that of the APS protocol, as if it alone were running. Simulations results show that with multiple packet sized messages, the behavior of the 3-Tier protocol is more impressive. Further the performance of the 3-Tier protocol is independent of the network configuration [32].

6.2 Equal Probability Load Traffic

Under this load type an incoming message attempts to access either of the two buses with equal probability, independent of the number of downstream or upstream nodes. As a result, with both Poisson and bursty arrival patterns, each node tends to submit the same number of packets for access to a bus.

1. At a load of 0.2, the basic 802.6 performance is retained.

2. At a load of 0.4, the average access delays of many nodes along the direction of the bus are marginally increased (by less than a slot), and no visible gains in access delays are observed at the end-nodes.

3. At a load of 0.75, the nodes in the first-half of the bus have their delays increased by several slots, and the other nodes experience a decrease in access delay. However access protection also comes into play with the last few downstream nodes experiencing a rise in access delay.

4. At loads of 1.0 and 1.2, the 802.6 performance itself is fairly uniform. The 3-Tier Fairness protocol, on the other hand, results in a window of few end-nodes that suffer a sharp increase in access delay. All other nodes have fairly uniform and reduced access delays.

To summarize, 802.6 protocol performance is retained at low loads. At normal loads of up to 0.75, the performance of the 3-Tier fairness protocol produces some improvement for downstream nodes, though the upstream nodes experience a small increase in access delay. The 3-Tier protocol affects a few end-nodes slightly around the 0.75 load and controls load situations higher than that.

While it could be argued that the 802.6 protocol should be used for the significantly fair performance with equal probability load, this type of load is the only type fairly served by the 802.6 protocol, and this load type is often unrealistic. An equal probability load would imply that in a network of 100 nodes, node-99 would generate the same amount of traffic bound for node-100 alone as the total volume traffic it generates for the other 98 nodes. Unless the user-99 is biased towards accessing its

immediate neighbors only, this bias can not be justified. If every participating node makes full (or best) use of the potential extended by MANs, then the traffic offered by a node is far more likely to be symmetric than equal probability type. It is for this reason that improvement over the 802.6 performance should be explored.

6.3 Alpha Tuning and Dynamic Bandwidth Control

Simulations are run in a 100 node network with nodes 0.5 Kms apart with symmetric and equal probability load types, with single packet messages and with a network load of 1.2. For selected values of α, the delay characteristics are shown in *Fig-E*. The results can be summarized as follows :

- *Symmetric Loads* : With $\alpha = 0$, the APS performance is reproduced. With further increase in α, the performance slowly tends to the 802.6 protocol performance with 802.6 performance when $\alpha \geq 0.65$.

- *Equal Probability Loads* : This load type is found to be very sensitive to *α-tuning*. The APS performance is found when $\alpha = 0$, and the performance slowly returns back to the 802.6 protocol performance. Unlike the previous types of loads, even when α equals 0.65, the performance still has not returned to 802.6 performance. When α equals 1.0, the 802.6 performance returns and continues beyond this load level.

Thus it is possible to tune specific access protection performance to meet the traffic demands. The *alpha tuning* has the capacity to tune the performance to levels in between the 802.6 and APS protocols. With symmetric load the tuning has significant impact. The equal probability load is found to be very sensitive to the tuning process.

6.4 Evaluation of Overall Performance

The overall performance improvements of the 3-Tier fairness protocol scheme can be summarized as follows :

- The 3-Tier fairness protocol adopts different access policies depending on network load activity.

- At low loads the 3-Tier fairness protocol performs exactly as the 802.6, as if only the 802.6 protocol were running. This domain extends up to load values of about 0.4 (0.2 with the equal probability load).

- At normal loads the 3-Tier fairness protocol produces enhancements, with the domain extending to about a load of 0.75. In the case of the equal probability load, a few end-nodes along the bus also come under the influence of access protection and experience small increases in access delays. This performance is very much similar to the ADS protocol, as if it alone were running.

- At loads of 0.9 and above, APS comes into effect. The performance is *extremely convincing* with the symmetric load – a very natural type of traffic. With equal probability load, the end-node access delays suffer sharp rises, and all other nodes experience uniform access delays. This unfairness at high loads is tolerable, given the unnaturalness of such a load type and the performance improvement experienced at lower and more realistic loads.

- By tuning the access protection limits (α-tuning), it is possible to serve even better the requirements of some eccentric traffic patterns.

The 3-Tier fairness protocol performance performs even better with multiple packet sized messages.

7 Conclusion

In this paper the 3-Tier structure access protocol is proposed. Network load is classified into three domains, and an attempt is made to counter the unfairness by adopting a suitable strategy at any given instant. At low loads the basic 802.6 performance is retained, and extra slots are allowed at higher loads to counter the latency in transportation of *REQUEST*s. At heavy loads, access protection ensures fair access to the front-end nodes along the bus.

The 3-Tier fairness protocol presents an acceptable performance over the entire range of the traffic demand. Also, it is possible to tune the performance of the access protection at heavy loads by resorting to dynamic bandwidth control. The tuning is necessary because in a realistic scenario, the network load may not follow a specific pattern. The proposed DANT scheme dynamically maintains the additional time dependent network information required to implement the proposed protocol. Future research in the direction of the dynamic bandwidth control (through alpha tuning) may pave the way for a fair performance protocol that tunes its performance to serve better all real time traffic patterns.

References

[1] R. W. Klessig. "Overview of Metropolitan Area Networks," IEEE Communications Magazine, Jan 1986.

[2] R. M. Newman, Z. L. Budrikis and J. L. Hullet. "The QPSX Man," IEEE Communications Magazine, Apr 1988, Vol-26, No.4.

[3] Editor, The IEEE 802.6 Working Group, "Project 802: Local an Metropolitan Area Networks – Draft copy of Proposed Standards," Institute of Electrical and Electronics Engineers, Inc, 1989.

[4] P. T. Gia and T. Stock, "Approximate Performance Analysis of the DQDB Access Protocol," ITC specialist Seminar, Adelaide, 1989, Paper No.16.1.

[5] M. Zukerman and P. Potter, "The DQDB Protocol and its Peformance under Overload Traffic Conditions," ITC Specialist Seminar, Adelaide, 1989, Paper No. 16.4.

[6] K. Sauer, and W. Schödl, "Performance Aspects of the DQDB Protocol," ITC Specialist Seminar, Adelaide, 1989, Paper No. 16.3.

[7] M. Conti, E.Gregori, and L.Lenzini, "DQDB Media Access Control Protocol: Performance Evaluation and Unfairness Analysis," Proceedings of the Third IEEE Workshop on MAN, Dana Point, CA, March 28-30 1989, pp. 375-408.

[8] J. W. Wong, "Throughput of DQDB Networks under heavy load," Proceedings of EFOC/LAN 89, Amsterdam, The Netherlands, Jun 11-16, 1989, pp. 146-151.

[9] J. Filipiak, "Access Protection for Fairness in a Distributed Queue Dual Bus Metropolitan Area Network," Proceedings of the International Conference on Communications, Boston, June 11-14, 1989, pp. 635-639.

[10] K. M. Khalil and M. E. Koblentz, "A Fair Distributed Queue Dual Bus Access Method," Proc. of the 14th Conf. on Local Computer Networks, Minneapolis, MN, Oct 1989, pp. 180-188.

[11] P. Martini, "The DQDB Protocol - What About Fairness ?," GLOBECOM'89 Conf. Record, 1989, pp. 298-302.

[12] P. Martini, "Fairness Issues in DQDB Protocol," Proc. of the 14th Conf. on Local Computer Networks, Minneapolis, MN, Oct 1989, pp. 160-170.

[13] Y. Yaw, Y. K. Yea, W. D. Ju and P. A. Ng, "Analysis on Access Fairness and a Technique to Extend Distance for 802.6," Proc. of the 14th Conf. on Local Computer Networks, Minneapolis, MN, Oct 1989, pp. 171-179.

[14] M. W. Garrett, and S. Q. Li, "A Study of Slot Reuse in Dual Bus Multiple Access Networks," Proceedings of IEEE INFOCOM'90, San Francisco, June 1990, pp. 617-629.

[15] H. Kaur, and G. Campbell, "DQDB - An Access Delay Analysis," Proceedings of IEEE INFOCOM 1990, pp. 630-635.

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

[17] B. Mukerjee and S. Banerjee, "Alternative Strategies For Improving The Fairness In and An Analytical Model of DQDB Networks," Proceedings of IEEE INFOCOM 1990, pp. 879-888.

[18] B. G. Kim, "Packet Delays in the IEEE 802.6 DQDB Protocol," Proceedings of the International Conference on Communications, pp. 346.4.1-346.4.5, 1990.

[19] T. Stock, "Influences of Multiple Priorities on DQDB Protocol Performance," Proceedings of the 13th ITC, Copenhagen, 1991, pp. 947-952.

[20] Martin de Prycker, "Asynchronous Transfer Mode Solution for Broadband ISDN," Published by ELLIS HORWOOD limited, Chrekestyer, England, 1991.

[21] H. R. van As, J. W. Wong and P. Zafiropulo, "Fairness, Priority and Predictability of The DQDB MAC Protocol Under Heavy Load," Zurich Seminar on Digital Communication Systems, Zurich, March 1990 pp 410-417.

[22] E. Y. Huang and L. F. Merakos, "On the Access Fairness of the DQDB MAN Protocol," 9th Annual Int'l Phoenix Conf. on Computers and Communications, March 1990, pp. 556-559.

[23] M. A. Rodrigues, "Evaluating Performance of High-Speed Multi-access Networks," IEEE Network Magazine, May 1990, pp. 36-41.

[24] H. R. van As., "Performance Evaluation of Bandwidth Balancing in the DQDB MAC Protocol," Proc. of 8th EFOC/LAN Conference, Munich, June 1990, pp 231-239.

[25] A. Baiocchi, M.Carosi, M.Listanti, G.Pacifici, A.Roveri, and R.Winkler, "The ACCI Protocol for a Twin bus ATM Metropolitan Area Network," INFOCOM'90, San Francisco, June 1990, pp. 165-174.

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

[27] H. R. Müller, 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," INFOCOM'90, San Francisco, June 1990, pp. 185-191.

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

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

[30] E. L. Hahne and N. F. Maxemchuk, "Fair Access of Multi-Priority Traffic to DQDB Networks," INFOCOM'91, Miami FL, April 1991, pp 889-900.

[31] B. Muherjee, A. C. Lantz, N. S. Matloff and S. Banerjee, "Dynamic Control and Accuracy of the p_i-Persistent Protocol Using Channel Feedback," IEEE Trans. on Communications, Vol-39, No.6, June 1991, pp 887-898.

[32] L. N. Kumar, "A 3-Tier Fairness Protocol: A Proposal for the Solution of the Unfairness Problem in DQDB MANs," M. S. Thesis, Wahington Universtity in St. Louis, MO, Aug 1991.

[33] L. N. Kumar and A. D. Bovopoulos, "An Access Protection Solution for Heavy Load Unfairness in DQDB," INFOCOM'92, Florence, Italy, May 1992.

[34] L. N. Kumar and A. D. Bovopoulos, "A Protocol For Dynamic Assessment of Network Topolgy in DQDB MANs," ISMM Int'l Conference on Computer Communications and Networks (IC^3N), San Diego, CA, USA, June 8-10, 1992.

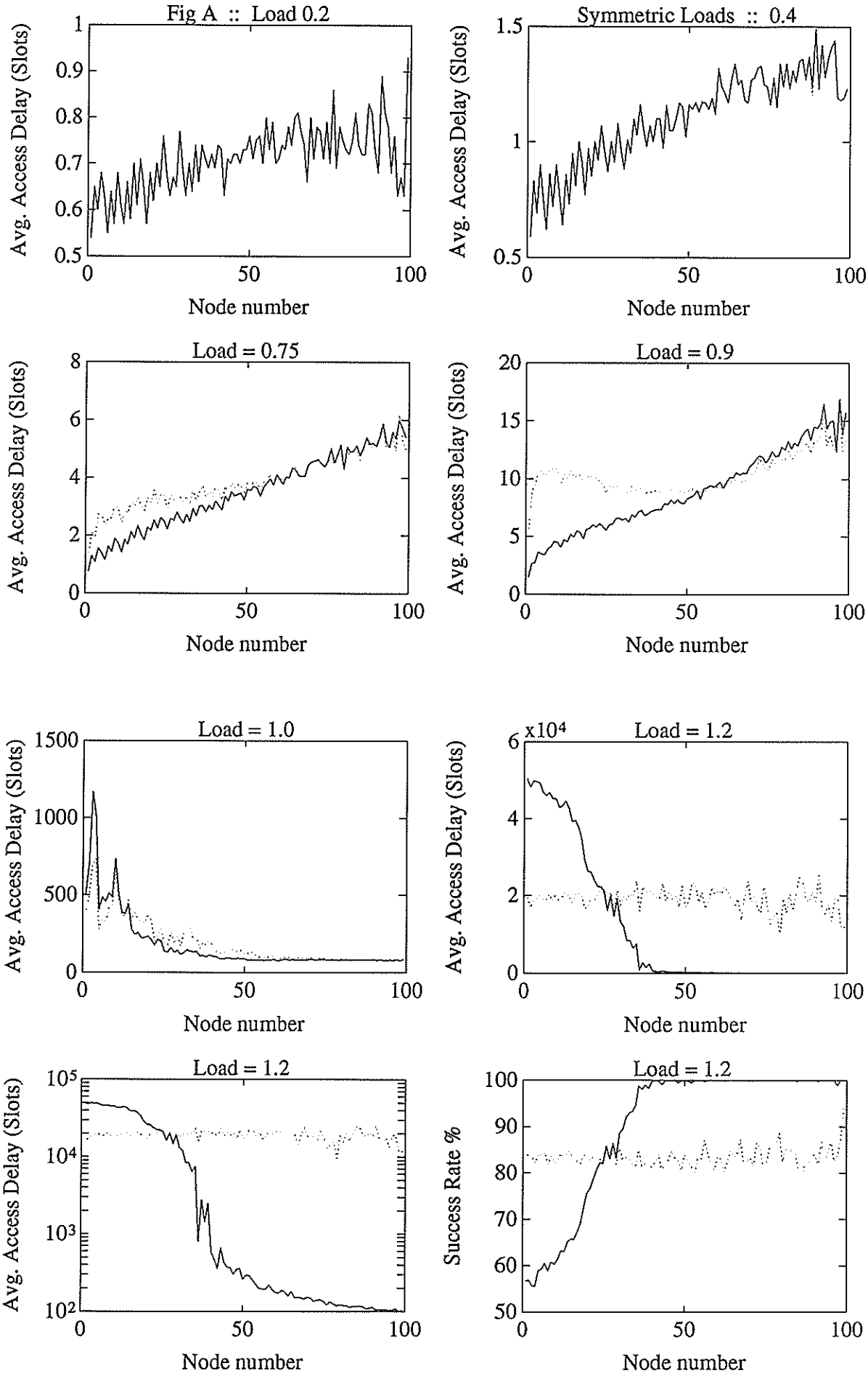


Fig A :: Load 0.2
Avg. Access Delay (Slots)
Node number
Symmetric Loads :: 0.4
Load = 0.75
Load = 0.9
Load = 1.0
x10⁴
Load = 1.2
Load = 1.2
Success Rate %

Figure A

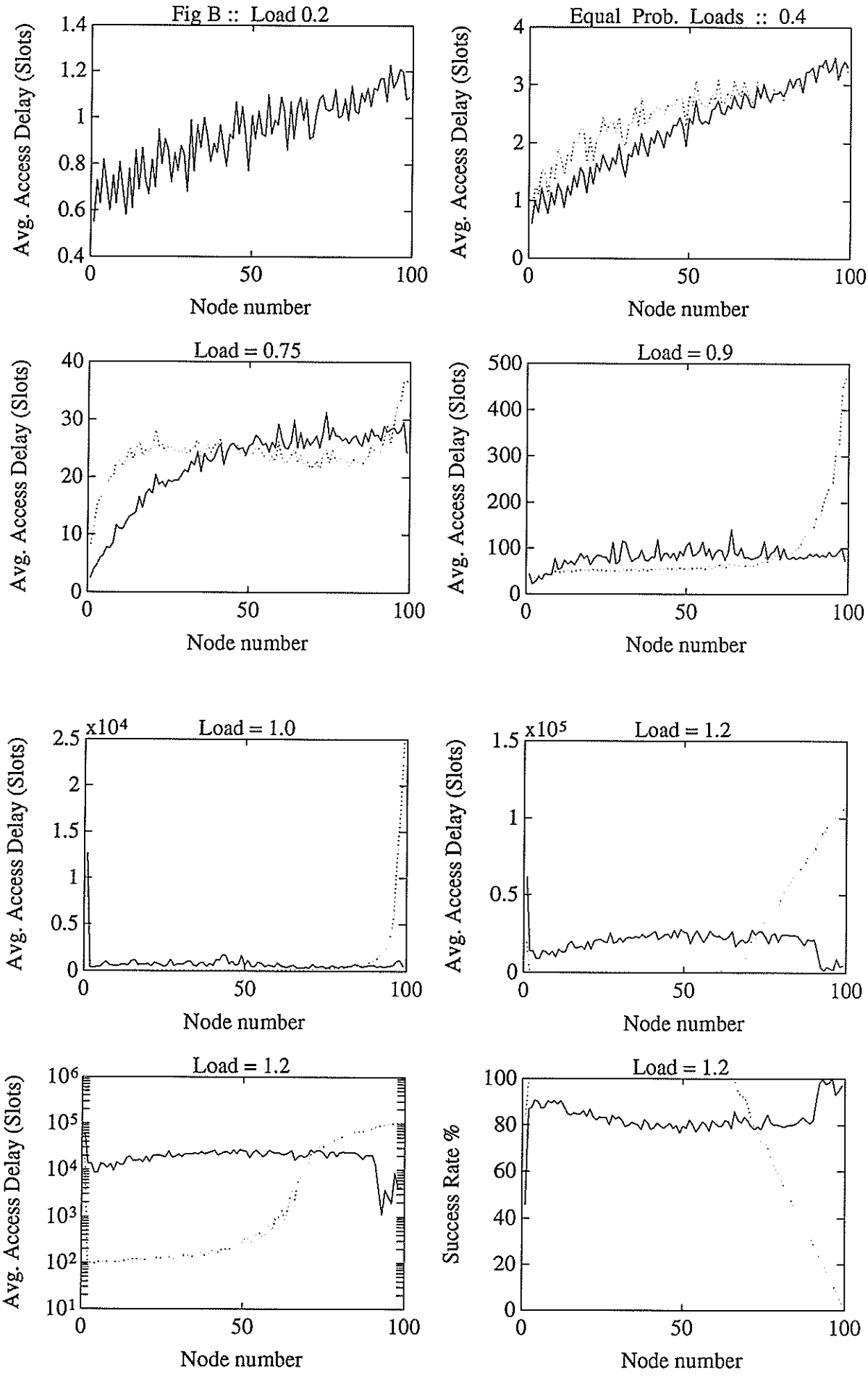


Fig B :: Load 0.2
Equal Prob. Loads :: 0.4
Load = 0.75
Load = 0.9
x10^4
Load = 1.0
x10^5
Load = 1.2
Load = 1.2
Load = 1.2
Avg. Access Delay (Slots)
Success Rate %
Node number

Figure B

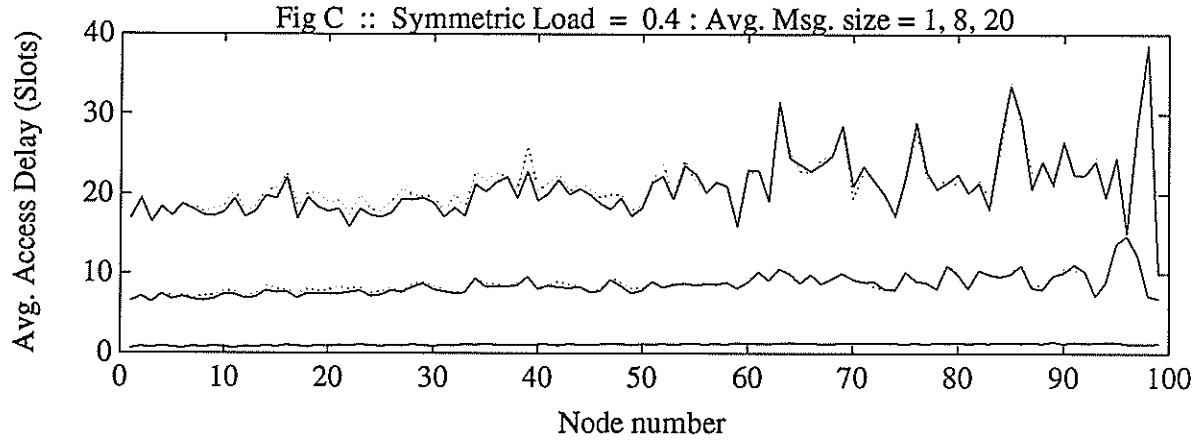
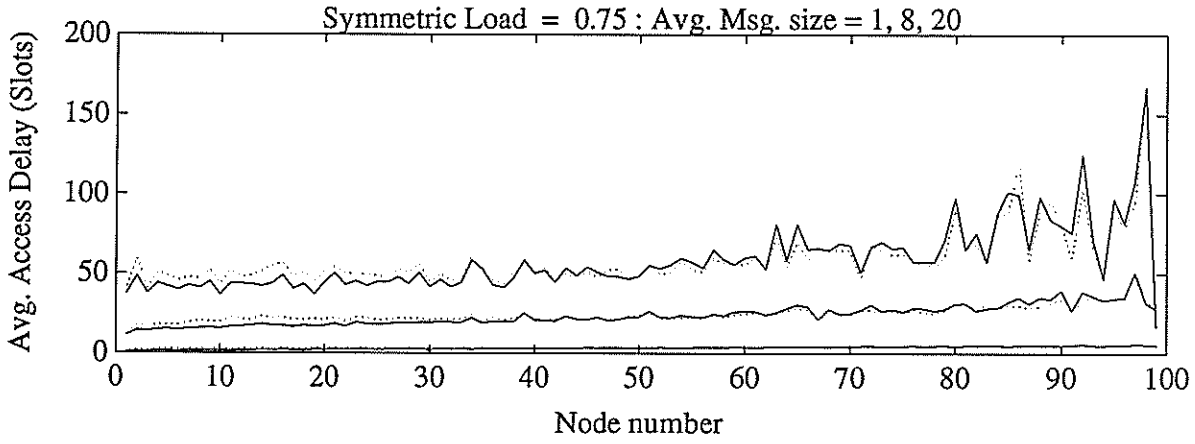
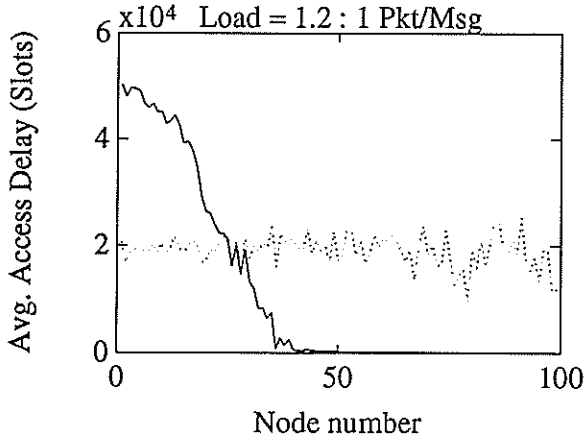
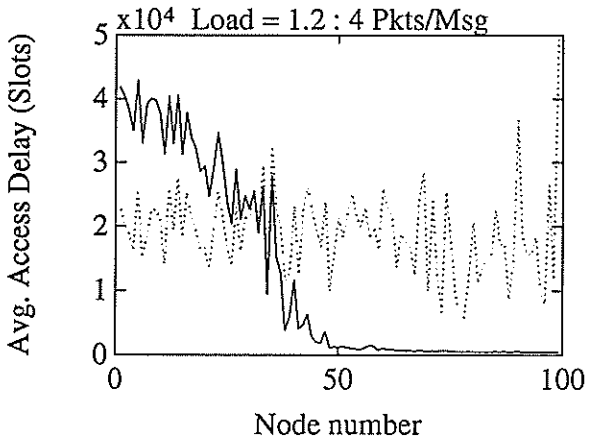


Fig C :: Symmetric Load = 0.4 : Avg. Msg. size = 1, 8, 20
Avg. Access Delay (Slots)
Node number


Symmetric Load = 0.75 : Avg. Msg. size = 1, 8, 20
Avg. Access Delay (Slots)
Node number


x10^4 Load = 1.2 : 1 Pkt/Msg
Avg. Access Delay (Slots)
Node number


x10^4 Load = 1.2 : 4 Pkts/Msg
Avg. Access Delay (Slots)
Node number

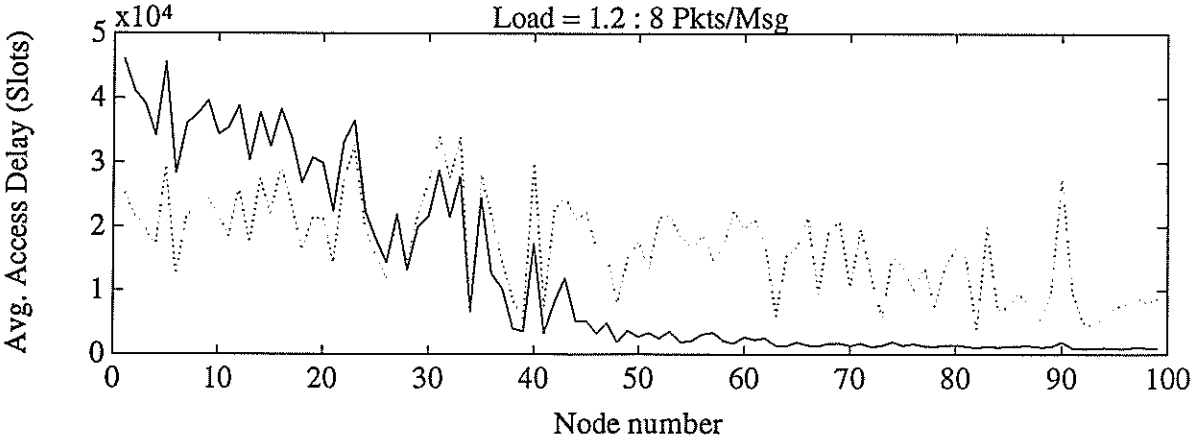

x10^4
Load = 1.2 : 8 Pkts/Msg
Avg. Access Delay (Slots)
Node number

Figure C

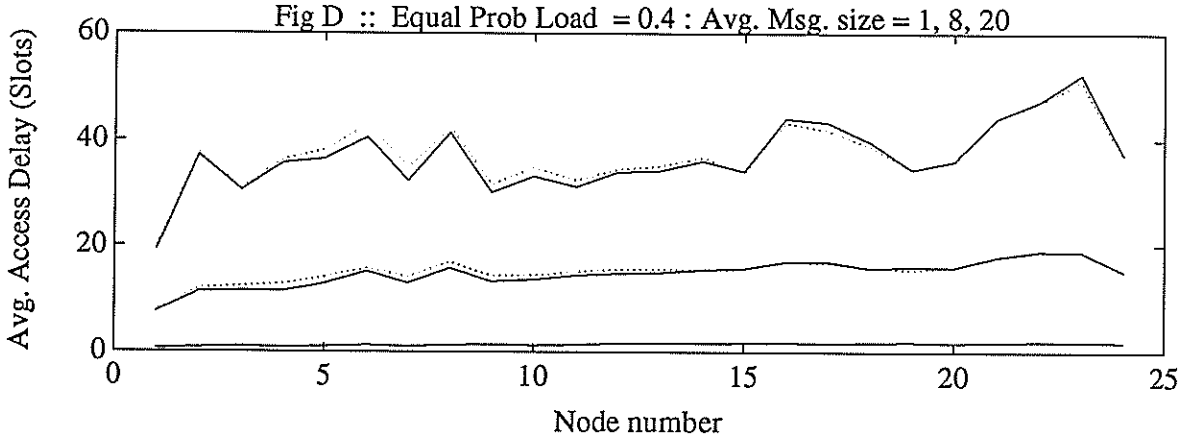
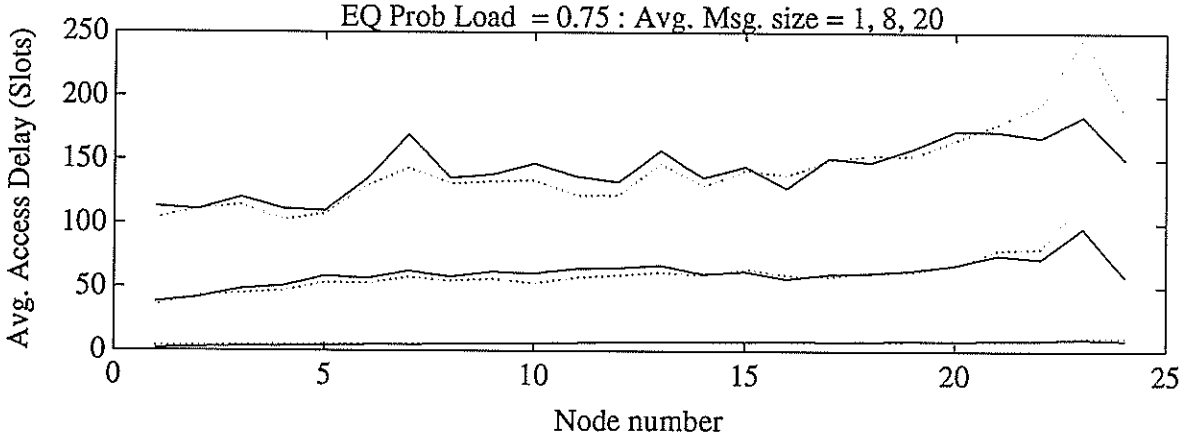
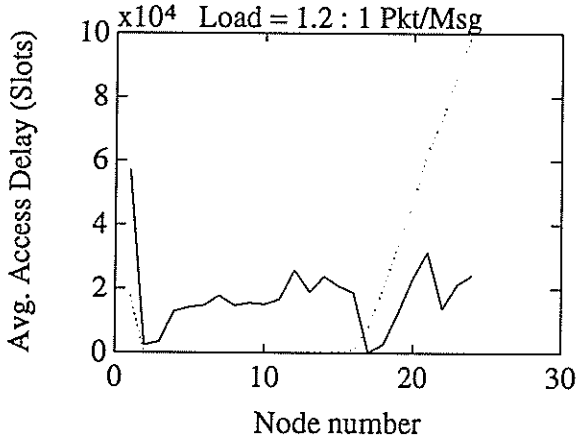
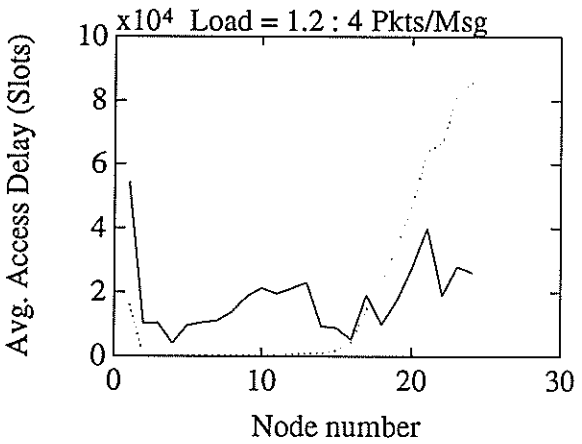


Fig D :: Equal Prob Load = 0.4 : Avg. Msg. size = 1, 8, 20
Avg. Access Delay (Slots)
Node number


EQ Prob Load = 0.75 : Avg. Msg. size = 1, 8, 20
Avg. Access Delay (Slots)
Node number


x10⁴ Load = 1.2 : 1 Pkt/Msg
Avg. Access Delay (Slots)
Node number


x10⁴ Load = 1.2 : 4 Pkts/Msg
Avg. Access Delay (Slots)
Node number

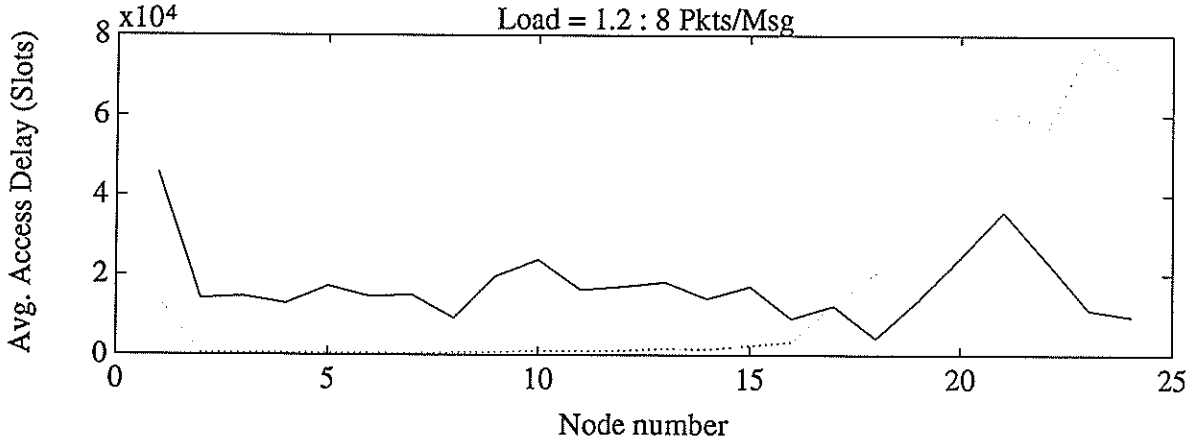

x10⁴
Load = 1.2 : 8 Pkts/Msg
Avg. Access Delay (Slots)
Node number

Figure D

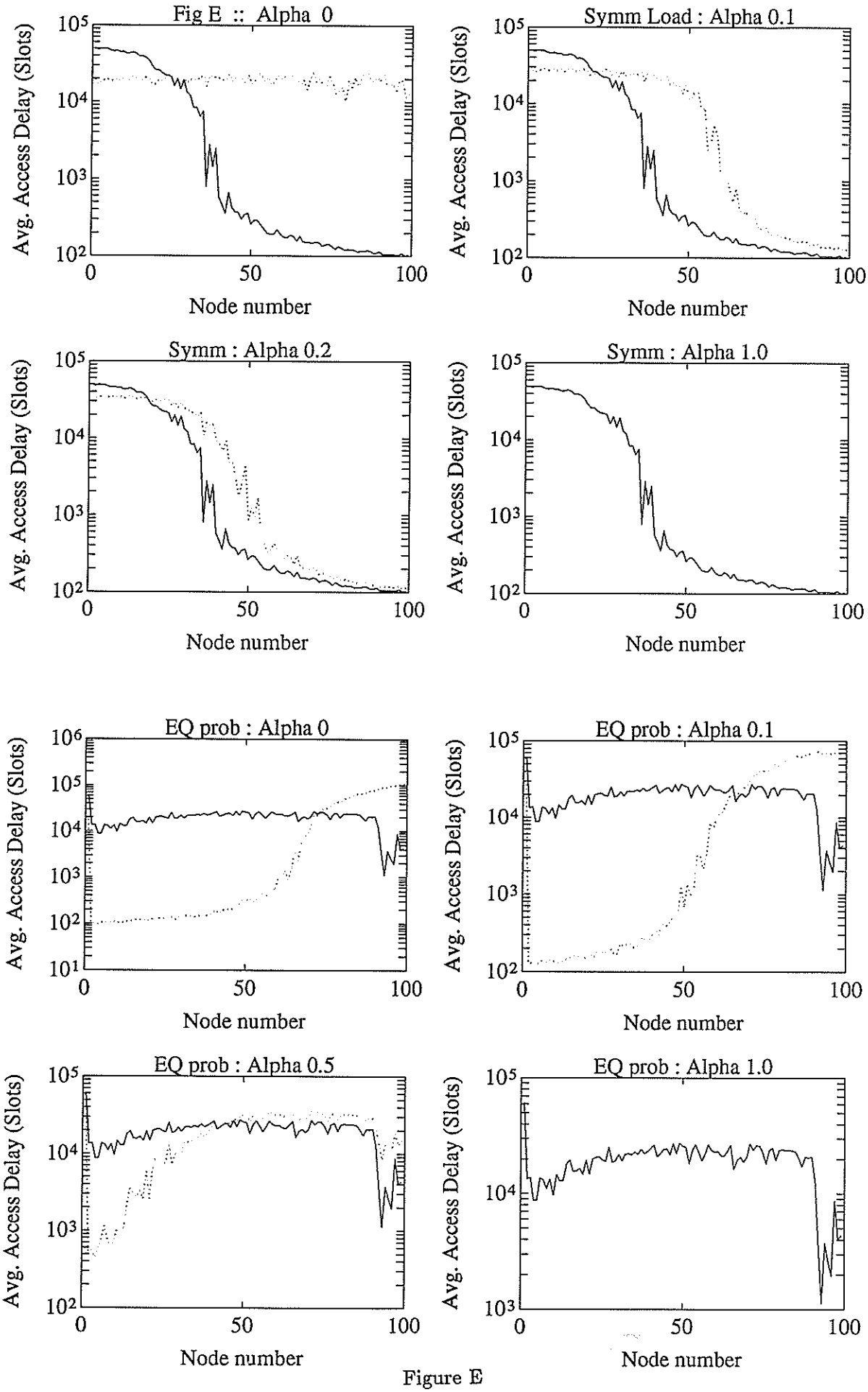


Fig E :: Alpha 0
Symm Load : Alpha 0.1
Symm : Alpha 0.2
Symm : Alpha 1.0
EQ prob : Alpha 0
EQ prob : Alpha 0.1
EQ prob : Alpha 0.5
EQ prob : Alpha 1.0
Avg. Access Delay (Slots)
Node number
0
50
100

Figure E