TECHNICAL ARTICLE

SIMULATION 75:3, ###-###
©2000, Simulation Councils Inc.
ISSN 0037-5497/00
Printed in the United States of America

Analysis of Dual-Bus Metropolitan Area Networks Using Distributed Quantitative Stochastic Simulation

Victor Yau
German National Research Center for
Information Technology
GMD FOKUS
Berlin, Germany
E-mail: yau@cse.uta.edu

Krzysztof Pawlikowski
Department of Computer Science
University of Canterbury
Christchurch, New Zealand

The Distributed Queue Dual Bus protocol (DQDB) has been adopted by IEEE and by the International Standard Organisation as a metropolitan area network standard. DQDB is also a European Telecommunication Standard. Using distributed simulation, this paper analyzes the performance of standard DQDB networks, DQDB networks with slot bandwidth reuse, as well as DQDB networks with slot preuse and reuse. The results suggest that although slot preuse is a conceptual mirror image of slot reuse, the maximum improvement in throughput of the networks, the causes of unfairness, and the stations which are unfairly favoured differ depending on the choice of bandwidth single-use, reuse, preuse, or pre-and-reuse. All quantitative performance measures were estimated to a specific precision using automated distributed stochastic simulation. The design of the DQDB simulators and practical implications of the results are also discussed.

Keywords: High-speed networks, distributed simulation, simulator design, bandwidth multi-use, distributed queue dual bus, delay

1. Introduction

DQDB (Distributed Queue Dual Bus) is the Medium Access Control (MAC) protocol adopted by the IEEE, as well as by the International Standard Organisation (ISO) as a Metropolitan Area Network (MAN) standard [1–10]. DQDB has also been adopted as a European Telecommunications Standard (ETS 300). In the United States, DQDB is already being used as the access protocol in Switched Multi-Megabit Data Service (SMDS) networks [11]. Many telecommunications operators offering MAN-based services in Europe follow the SMDS specifications. The Connectionless Broadband Data Service (CBDS) is also provided in Europe using DQDB networks. In Australia, DQDB is also known as QPSX (Queued Packet Synchronous eXchange).

A main function of MANs is the interconnection of LANs. In addition, DQDB has been proposed as the solution for the interconnection of Personal Communication Networks (PCNs) [7, 8], flow control in ATM [7], and media access control in multi-hop lightwave networks [12–15]. The ever increasing demand for bandwidth per user has led many researchers to consider methods for upgrading standard DQDB in the future.

To enable a standard DQDB network to be upgraded while using existing transmitters and receivers installed in standard DQDB stations (i.e., without increasing the transceiving rate that has to be supported by the network interface of stations), and without increasing the network's bandwidth requirement, several protocols have been proposed that allow the spatial reuse of channel bandwidth. According to the MAC of standard DQDB, a slot can be used by at most one node during its lifetime, i.e., during the time interval between the "birth" of a given slot at a Head-of-Bus (HOB) and its "death" when it reaches the corresponding End-of-Bus (EOB); see Figure 1. Thus, in some cases, slots are effectively used only during a small fraction of their lifetimes, if they are used at all.

Let nodes observing the rules of standard DQDB for accessing its busses be called standard users of slots. Spatial slot preuse allows a slot to be used prior to serving its standard user, whereas spatial slot reuse allows a slot to be reused after serving its standard user. Thus with spatial preuse or reuse, a slot can potentially be used several times, allowing the maximum network throughput to be increased above unity. Past research has focused primarily on specific slot reuse protocols, e.g., [1, 2, 6, 13, 16, 17, 18, 19]. Several efforts, however, have proposed specific slot preuse DQDB Protocols [20, 21].

The aim of this paper differs from previous work in two main ways. First, instead of providing a design and analysis of a specific slot reuse or preuse protocol, we study two generic protocols, one that employs slot reuse, and one that uses both slot reuse and preuse, in addition to the standard DQDB protocol. The focus is on how each approach alters the fundamental causes of unfairness, and on the potential improvement in performance that could be achieved through the slot preuse, slot reuse, or slot pre-and-reuse strategies (the case of slot preuse only will not be explicitly considered since it is subsumed by the difference in performance between slot preuse-and-reuse and slot reuse).

Second, all quantitative performance measures of DQDB networks in their steady state were estimated to a specific level of accuracy using distributed stochastic simulation. A common pitfall in the performance evaluation of telecommunication systems is to overlook the fact that results from stochastic simulations are simply estimates [22–26], and a proper evaluation of their accuracy is necessary before they can be used at an appropriate level of confidence by other researchers. Of course, if a system has to meet specific tolerances, then it becomes critical to obtain results with specific accuracy.

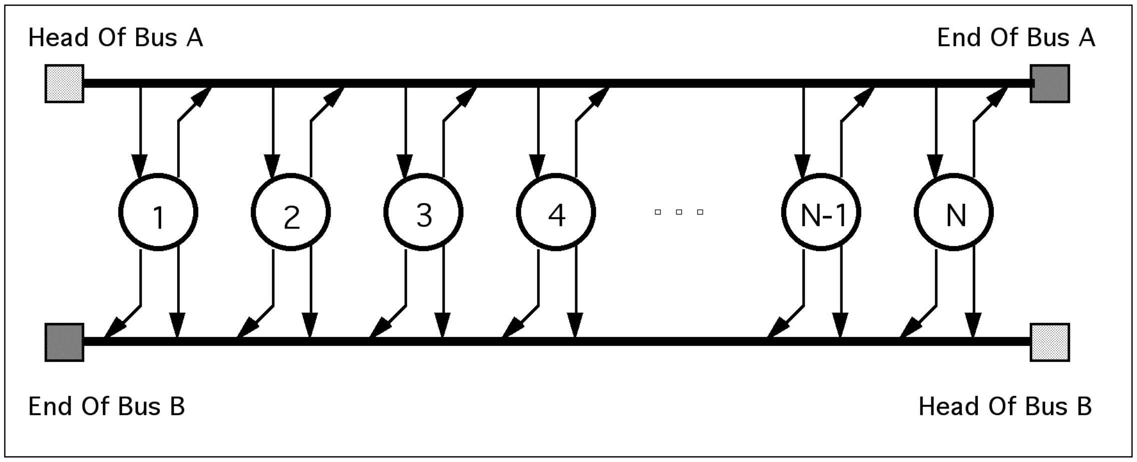
Recognising the need for obtaining reliable results within a practical time, all numerical results were produced using AKAROA [26–31], an object-oriented simulation package developed by us for automated precision control of steady-state estimates and automated parallel execution of quantitative simulations. Slots in a DQDB network can be queue-arbitrated slots (allocated dynamically to stations using the distributed queuing mechanism [7, 32]) or pre-arbitrated slots (reserved for specific connections by a bandwidth manager [8]); we assume that all slots are queue-arbitrated.

The reference protocols are described in Section 2. The structure of the DQDB simulators is explained in Section 3. The numerical results are summarised in Section 4. Section 5 focuses on some practical implications.

2. The Reference Protocols

This study is based on generic reference protocols for slot single use, slot reuse, and slot preuse-and-reuse

(a)



(b)

Access Control Field	Segment
1 byte	52 bytes

Figure 1. DQDB network: (a) dual bus topology; (b) the slot format

in DQDB networks. The reference protocols were chosen for their suitability as representatives for the best case in their class with respect to performance, so that the performance potential and fairness characteristics of each approach could be compared with one another. Simultaneously, the reference protocols must preserve the basic protocol constraints of standard DQDB; i.e., a slot may be preused only if it could be released (emptied) for use by its standard DQDB user (the station which would use this slot if slot preuse were not applied), and a slot may be reused only after it has delivered the payload for its standard user. This guarantees that nodes of the reference networks are served in a manner not worse than they would be in standard DQDB.

The reference protocols are:

1. Standard DQDB

Given that the objective is to investigate the changes in the fairness and performance characteristics with respect to standard DQDB, an analysis of a standard DQDB network operating under the same traffic models and loads as that of the DQDB networks using slot reuse and slot preuse-and-reuse is a necessary first step. The standard DQDB architecture and a full protocol definition are given elsewhere [5], and will only be outlined here.

The DQDB network is composed of two high-speed (approximately 150 Mbps) unidirectional slotted busses[1], carrying fixed-length packets of data (called segments) in opposite directions. The MAC procedures for segment transmission in each bus (direction) are symmetric. Without loss of generality, this paper will consider the transmission of segments "from left to right" over Bus A (transmission channel), and the associated requests transmitted "from right to left" over Bus B (reservation channel); see Figure 1(a). Channels are time-slotted; see Figure 1(b) for the format of a slot. Each node is either in the IDLE or COUNTDOWN state. An IDLE node increments its request counter (RQ _CTR) for each request-carrying-slot observed on the reservation channel, and decrements RQ _CTR for each empty slot observed on the transmission channel.

An IDLE node changes to the COUNTDOWN state when it generates a segment for transmission. It uses the first slot on the reservation channel with an unset request bit for sending a request to upstream nodes, informing them that an empty slot is needed. Each slot can carry at most one request. Without waiting, it sets its countdown counter (CD_CTR) to the value of RQ _CTR, and resets RQ _CTR. A COUNTDOWN node increments its request counter (RQ _CTR) for each request-carrying-slot observed on the reservation channel, and decrements CD_CTR for each empty slot observed on the transmission channel. When CD_CTR reaches zero, the node transmits its segment in the next empty slot on the transmission channel.

Ideally[2], the standard DQDB mechanism creates two distributed FIFO queues of requests (one for each bus) that order access of nodes to empty slots traversing each bus. However, slots can be engaged into useless transport of already delivered data segments.

2. DQDB with Slot Reuse (DQDB/SRU)

Each node follows the MAC procedure as defined for standard DQDB. In addition, each node is responsible for erasing a slot that delivered the segment destined for it.

Each released slot represents extra capacity for downstream nodes, but this extra capacity is not *visible* to nodes upstream. Thus each node also maintains an Erased Slots Counter (ERC) for counting locally released slots. This information is used for balancing the number of outstanding requests recorded at upstream nodes as follows:

When a slot is released at a station, say S_i, the number of requests in the request counter (RQ _CTR) at each upstream node will be balanced by:

a. Decrementing by one the RQ _CTR at S_i, if the outstanding request counter OR_CTR of S_i is greater than zero. The value of the OR_CTR of S_i equals the number of requests that S_i needs to send on the reservation channel; otherwise,

b. If the reserved bit on the current slot on the reservation channel had been set, then reset it instead of sending it down the reservation channel; otherwise,

c. Increment ESC by one.

Whenever S_i observes a request on the reservation channel, if its ESC > 0, then it decrements its ESC by one and cancels the request (i.e., resets the request bit).

3. DQDB with Slot Preuse-and-Reuse (DQDB/SMU)

Each node follows the MAC procedure as defined for standard DQDB / SRU. In addition, whenever S_i observes an empty slot on the transmission channel that had been reserved for S_j, S_i transmits one of its own segments in that slot, provided that the slot would arrive at the destination of that segment before reaching S_j. Adding slot preuse to slot reuse has the advantage that the facility and procedure used by stations

[1] The use of dual busses was intentional. DQDB was created for high-speed networks using optical fiber. Use of unidirectional busses means nodes can interface each bus using passive unidirectional fiber taps, which is more easily achieved and incurs lower attenuation. Higher numbers of nodes can be supported before active taps (receive and regenerate) are needed. Also the effective data rate is doubled for a given transceiver rate. However, DQDB has also been used on different media at other rates. Additionally, a logical dual bus topology can be created using a different physical topology.

[2] Under the assumptions of zero propagation delay and infinite reservation channel bandwidth.

for erasing slots under DQDB/SRU can also be used for erasing slots that were preused, thereby avoiding some increases of complexity and hardware demand.

However, first, a slot reserved for standard user S9 (for example) that is being preused by S3 (for example) to send its segment to S7 (for example) would be seen by S4, S5, S6, and S7 as a busy slot, instead of an empty one. Thus the bandwidth of this slot that would be available for S9 would not be *visible* to them.

Second, S7 would release the slot after receiving the segment in it and erroneously treat the released slot as one which has already delivered a segment for a standard user, and therefore represents extra capacity for downstream nodes. It would therefore (erroneously) cancel a request as in DQDB/SRU. Both effects are balanced by requiring the preuser of a slot (S3 in this example) to send a credit request for an empty slot for the destination of the pre-using segment (S7 in this example). This is because the credit request would not be seen by S4, S5, S6 and S7 (as it normally would under DQDB). Thus the first effect is balanced. Obviously it also reserves a slot for S7. This balances the request erroneously erased by S7.

3. Performance Evaluation

The performance of standard DQDB, DQDB/SRU, and DQDB/SMU were evaluated assuming a network with N nodes. The stream of segments arriving at node i from its local data sources is modelled as a Poisson process with arrival rate λ_i segments per slot time (for i = 1, 2, .., N). Traffic generated by each node is uniformly distributed over all possible destinations. Thus, in the case of bus A, where nodes from HOB to EOB are numbered from l to N, the traffic generated by node i and addressed to its downstream neighbours has the rate:

$$\lambda_{i,A} = \frac{N-i}{N-1}\lambda_i$$

The rate of traffic generated by the same node i to its downstream neighbours on bus B is:

$$\lambda_{i,B} = \frac{i-1}{N-1}\lambda_i; \qquad \text{for } i = 1, 2, .., N$$

Each new segment arriving at a node is stored in an input buffer. Two such input buffers are assumed per node, one per each bus, and each of a capacity of M segments. The assumed internode propagation delay equals one slot time. This gives a bus length of approximately 4 km, for N = 10 nodes and transmission speed of 155 Mbps.

3.1 Structure of Sequential DQDB Simulators

The sequential DQDB simulator was constructed using AKAROA's Build module. Build is an object-oriented toolkit for fast construction of discrete event simulation models in C/C++. AKAROA Build provides three basic classes of objects: entities, queues, and event scheduler, as well as statistical support functions for representing stochastic processes and computing integrals and quantiles of common families of statistical distributions.

Modelling of Stations, Slots and Segment Entities

Three classes of entities can be identified in a DQDB network (see Figure 1(a)): the DQDB nodes, slots on Bus A and B, and the data segments. A *slot* class was defined to model standard DQDB slots. This class is used in all simulators, since the slot structure of the three protocols is identical to standard DQDB slots. To model station entities, a base class *dqdbnode* was defined, from which the descendants *dqdbCnode* (modelling a standard DQDB station), *dqdbrunode* (modelling a DQDB/SRU station), and *dqdbmunode* (modelling a DQDB/SMU station) were derived.

Base class *dqdbnode* comprises member data used by all (standard, SRU, and SMU) DQDB nodes. They include variables for recording the node's unique identifier, its current state (IDLE or COUNTDOWN), data structures corresponding to the various counters, its buffer size, as well as a *segq* queue object. *segq* models the node's input buffer, and is a FIFO queue of *seg* (segment) entities. *dqdbnode* also defines an:

*ent * arrival(ent * seg); // Event of segment arrival to node*

member function, where *seg* is an object modelling a new segment. This member is invoked whenever a segment is generated at the DQDB station for transmission. It is also used by all DQDB stations, since all three protocols follow the same procedure for processing generations of new segments. Each of *dqdbCnode*, *dqdbrunode*, and *dqdbmunode* comprises their derived constructor for initialising the station's private data, and an additional member function:

*double processnewslots(slot * tchannel[], slot * rchannel[]);*

The *processnewslots* member of each node is invoked at the start of each time slot, i.e., whenever new slots arrive at the node. The definition of *processnewslots* differs in *dqdbCnode*, *dqdbrunode*, and *dqdbmunode*, each defining a procedure for executing the MAC rules of their respective protocols (polymorphic response), as specified in Section 2.

Events

Two scheduled events may occur:

1. *segarrival* which occurs whenever a new segment is generated at a node for transmission
2. *slotarrival* which occurs at the start of every time slot, when slots arrive at nodes.

Simulation Execution Sequence

An N node standard DQDB network with a bus length of L slots is created by instantiating N *dqdbCnode* objects and 2L slot objects. The network is initialised to an empty-and-idle state. The simulation is launched by scheduling a *segarrival* event for each station, and by scheduling one *slotarrival* event.

Whenever *segarrival* occurs the

*ent: * arrival(ent * seg);*

member of the *dqdbCnode* object (station) associated with the event is invoked. This member adds *seg* to the tail of the object's *segq* (modelling its transmission buffer) if it is not full. If *segq* is full, *arrival* returns a pointer to *seg* to indicate that the segment cannot be buffered. In both cases, *arrival* also schedules the next *segarrival* event at the station.

Whenever *slotarrival* occurs, the

*double processnewslots(slot * tchannel[], slot * rchannel[]);*

member function of every *dqdbCnode* object is invoked. Each node then processes each new data and reservation slot that arrived following the standard DQDB protocol. Whenever a segment first reaches the head of *segq*, the current simulated time is stored in the *seg*'s begin_contention_time attribute. If a station transmitted a segment on the arriving data slot, then it returns the access delay of that segment. The access delay of a segment is calculated as:

access_delay = (current simulated time) –
(begin_contention_time attribute of segment)

Access delay values of node i are passed to an AKAROA output analysis object such as:

stopsimulation = sav.processnewobs(access_delay,i);

where *sav* is a SPECTRALANALYSIS object provided by AKAROA. *sav* returns true if all parameters have been estimated to a desired precision at a stated level of confidence. If *sav* returns true, the simulation can stop. Otherwise another *slotarrival* event is scheduled exactly one time-slot from the current simulated time.

The main body of the DQDB simulator is a loop which repeatedly asks AKAROA Build's scheduler for the next_to_occur event, and invokes *slotarrival* or *segarrival* accordingly, until *sav.processnewobs* returns true.

Figure 2 illustrates the execution path of the standard DQDB simulator constructed using AKAROA Build. Simulators for DQDB/SRU and DQDB/SMU networks are implemented in a similar way.

Transformation of Simulators for Parallel Execution

The results reported here characterise performance of the networks in their steady state. All simulation experiments were run under AKAROA, an object-oriented simulation package developed by us for fully automated distributed quantitative steady-state simulation with full control of the final statistical error of all estimates [26, 27, 30, 32, 33]. The methodology used by AKAROA for transforming sequential DQDB simulators into ones suited for parallel execution, for run length and precision control during simulation experiments, as well as the speedup achieved and the intermachine communication and "warmup" costs for each of the parameters estimated during DQDB simulation experiments, are reported in a companion paper [34]. (Observations generated during the warmup period of each process were discarded to reduce initialization bias of the estimates.) In all experiments, runs were stopped when all estimates of performance measures obtained the relative precision (defined as the width of the confidence interval relative to the absolute value of the point estimate) of less than 5%, at the confidence level 0.95.

The primary measures considered here are:

- *Mean access delay at node S_i on the transmission bus (bus A)*, defined as the average time interval between the instant when a given segment buffered for transmission over bus A reaches the head of queue in the input buffer *at S_i* and the instant of time when it captures a slot for its transmission; and
- *Throughput*, defined as the average number of data segments transmitted in the network per slot time.

4. Results

4.1 Fairness Comparisons

The first set of comparisons of standard DQDB, DQDB/SRU, and DQDB/SMU involved the analysis of the impact of stations' relative locations along the busses on their mean access delays. Under the standard traffic model, $\lambda_i = \lambda$ for all $1 \leq i \leq N$. Thus:

$$\lambda_{i,A} = \frac{N-i}{N-1}\lambda$$

and the total normalised load (ρ) on bus A equals $\rho = 0.5\ N\lambda$ segments per slot time.

Point estimates of the mean access delay at S_i of segments transmitted on bus A are plotted in Figures 3, 4, and 5 as a function of station index, for N = 10, M = 20, and ρ = 0.40, 0.90, and 500, respectively. The value of 500 represents the overload situation when the transmission buffers of all stations are almost always full. A number of observations can be made based on these results:

R1. None of the protocols treats stations equally. The unfair behaviour of DQDB, especially under heavy traffic, has been well investigated previously [10, 17, 35, 36, 37, 38]. It is studied here by us so that it can be compared with DQDB/SRU and DQDB/SMU under the same assumptions, and the same operating parameters.

R2. 2.1) Under standard DQDB, the set of stations which are (unfairly) favoured differs depending on the load, ρ.

2.2) DQDB (unfairly) favours stations near the HOB under low traffic (ρ = 0.40).

2.3) Under high traffic (ρ = 500) DQDB favours nodes near the EOB most of all, followed by stations near the HOB. Nodes near the mid-

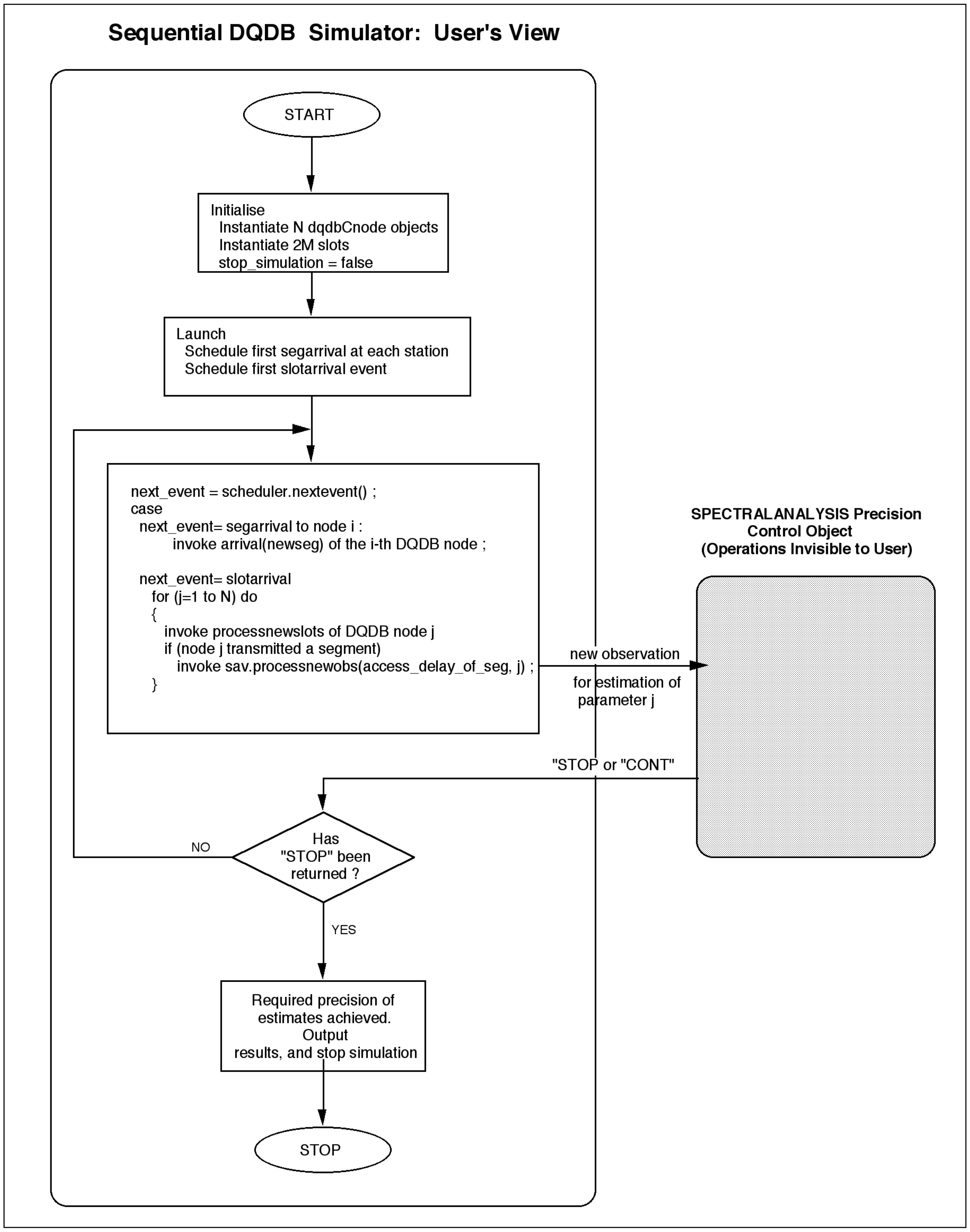

Sequential DQDB Simulator: User's View
START
Initialise
Instantiate N dqdbCnode objects
Instantiate 2M slots
stop_simulation = false
Launch
Schedule first segarrival at each station
Schedule first slotarrival event
next_event = scheduler.nextevent() ;
case
next_event= segarrival to node i :
invoke arrival(newseg) of the i-th DQDB node ;
next_event= slotarrival
for (j=1 to N) do
{
invoke processnewslots of DQDB node j
if (node j transmitted a segment)
invoke sav.processnewobs(access_delay_of_seg, j) ;
}
new observation
for estimation of parameter j
SPECTRALANALYSIS Precision Control Object
(Operations Invisible to User)
"STOP or "CONT"
Has "STOP" been returned ?
NO
YES
Required precision of estimates achieved. Output results, and stop simulation
STOP

Figure 2. User's view of DQDB simulator execution path

dle experienced the highest mean access delay.

R3. 3.1) DQDB/SRU favours stations near EOB most of all. The mean access delay at all nodes is lower than in standard DQDB.

3.2) The privileged treatment of near EOB stations increases with load. Nodes near the middle of the bus experienced the highest mean access delay.

3.3) The degree of unfairness depends on the offered load, ρ.

R4. 4.1) Adding slot preuse to DQDB/SRU gives DQDB/SMU. It reduces the access delay of stations near HOB, but does not reduce the mean access delay of stations near the EOB above that achieved using DQDB/SRU, whatever the load.

4.2) Nodes near the HOB and EOB enjoy lower mean delays that those near the middle.

4.3) Nodes which are (unfairly) favoured are the same ones (i.e., those near HOB or EOB) for all levels of offered load considered.

Intuitively, observations R1 to R4 can be due to:

L1. The limited bandwidth of the reservation channel, and

L2. The non-zero propagation delay of requests.

L1 and L2 imply that the goal of global FIFO service cannot be maintained in standard DQDB, causing unfairness. In DQDB/SMU and DQDB/SRU the causes of unfairness are complicated by:

L3. Segments preusing or reusing slots can jump queue[3], and

L4. The non-zero delay until requests could be cancelled.

Case of Standard DQDB

Stations near HOB see requests later (due to L2.). This gives them unfair advantage since they may use empty slots which they would have had to forego if the reservation channel had unlimited bandwidth and zero delay. Under light traffic, the competition for the bandwidth of the reservation channel is low, but the propagation delay of requests from downstream stations remains unchanged. This explains the higher delay experienced by downstream stations under light traffic. Under heavy traffic, competition for the bandwidth of the reservation channel is high. Downstream stations have unfair access to the reservation channel bandwidth. This explains their lower mean delay under high traffic.

[3] Packet sequence between source and destination would still always be maintained.

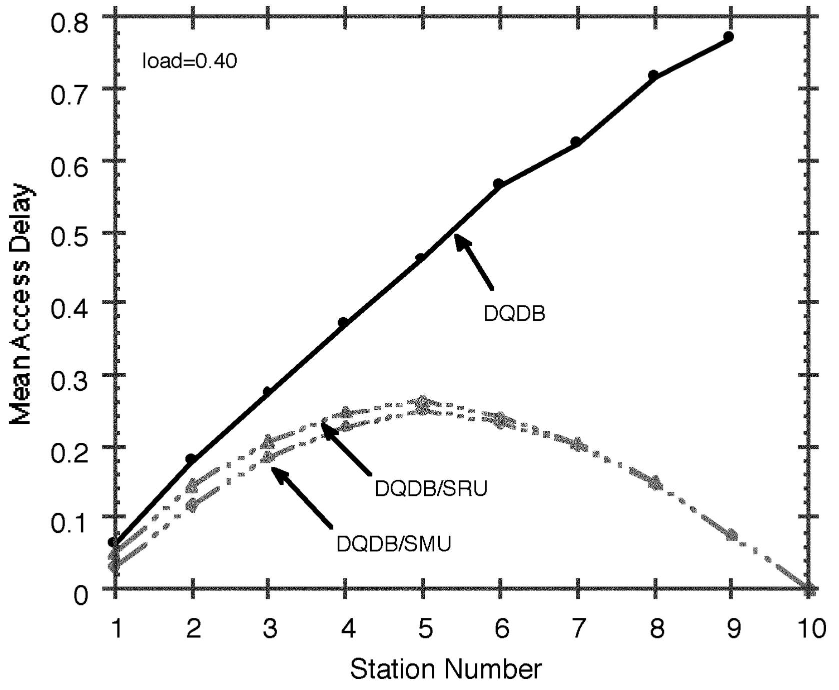


Figure 3. Mean access delay on the transmission bus as a function of station number in standard DQDB, DQDB/SRU, and DQDB/SMU networks with N = 10, M = 20 and ρ = 0.40

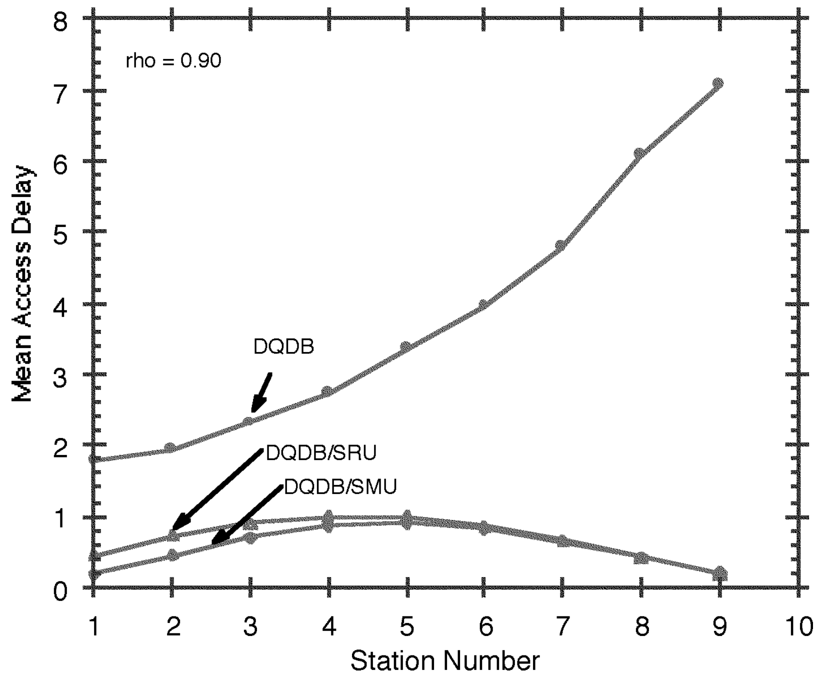


Figure 4. Mean access delay on the transmission bus as a function of station number in standard DQDB, DQDB/SRU, and DQDB/SMU networks with N = 10, M = 20 and ρ = 0.90

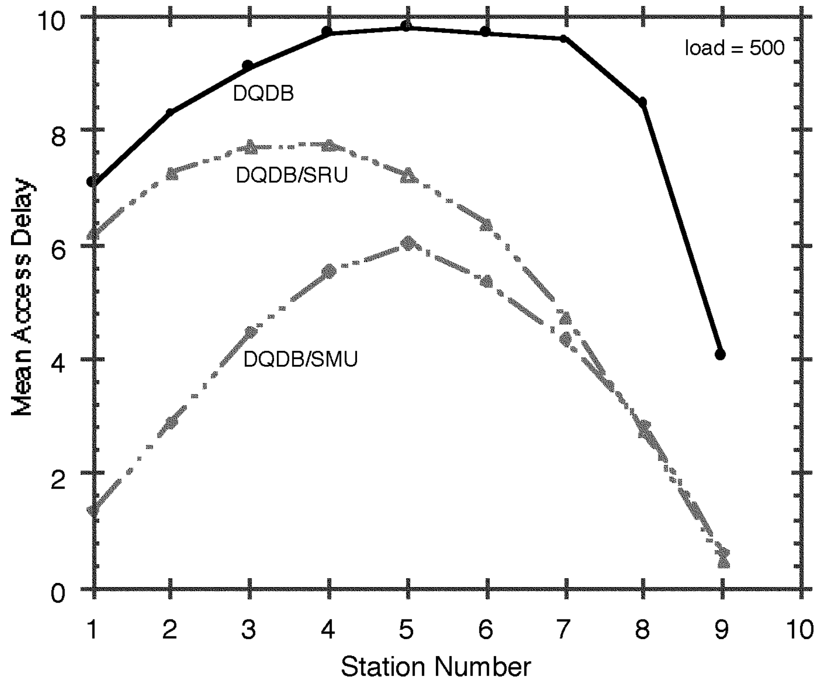


Figure 5. Mean access delay on the transmission bus as a function of station number in standard DQDB, DQDB/SRU, and DQDB/SMU networks with N = 10, M = 20 and ρ = 500.0

Case of DQDB/SRU and DQDB/SMU

Stations in DQDB/SRU and DQDB/SMU networks can benefit over standard DQDB in two ways: receive service earlier (L3), and enjoy temporal "over-reservation" during the non-zero delay until requests could be cancelled (L4). During this delay, slots may be erroneously left unused for them by upstream nodes.

With DQDB/SRU, stations near the EOB are more likely to be able to reuse slots, and hence gain an unfair advantage due to L3 and L4. This explains why SRU offers substantial delay improvements for stations near the EOB.

With DQDB/SMU, stations near the HOB are more likely to be able to preuse slots, and hence gain an unfair advantage due to L3 and L4. This explains why adding slot preuse improves performance over DQDB/SRU

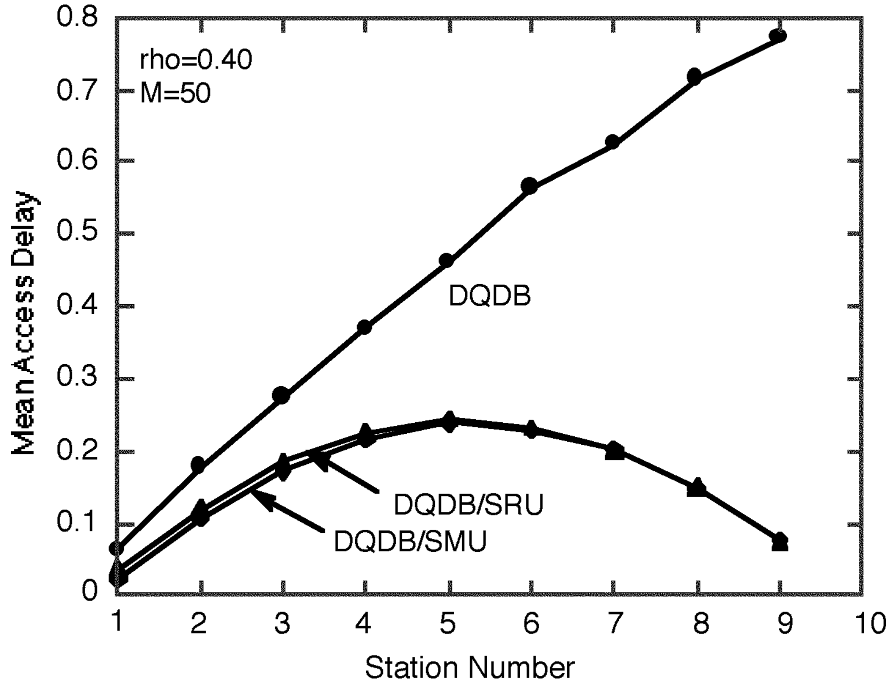


Figure 6. Mean access delay on the transmission bus when the capacity of the transmission buffer of each station has been increased to M = 50, N = 10 and ρ = 0.40

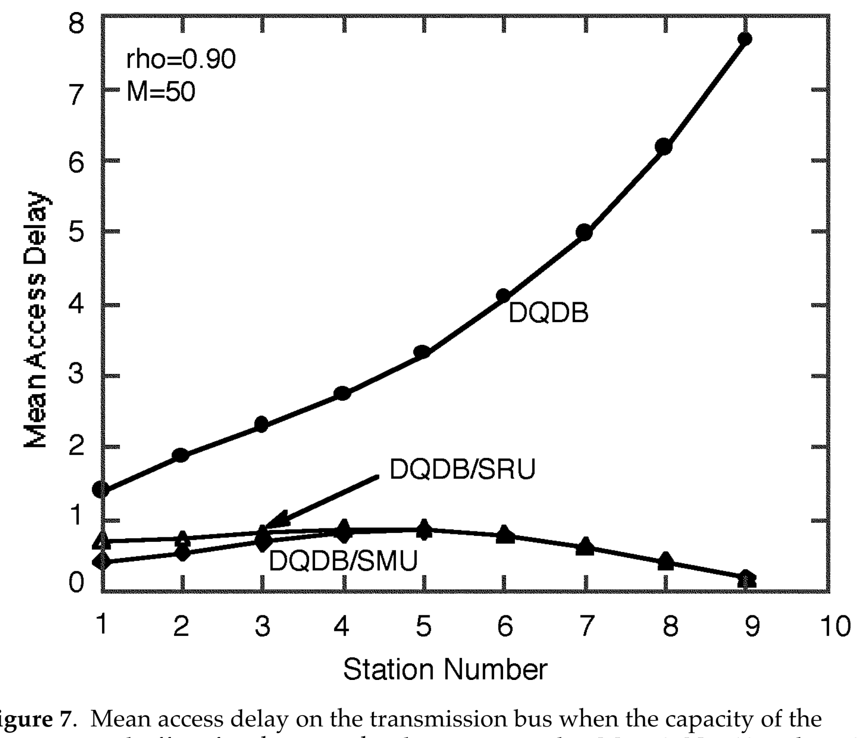


Figure 7. Mean access delay on the transmission bus when the capacity of the transmission buffer of each station has been increased to M = 50, N = 10 and ρ = 0.90

only for stations near the HOB. Under low to medium traffic, slot preuse is rarely needed since stations near the HOB already have an advantage in access to slots due to L1 and L2, as in standard DQDB. This explains the delay performance congruency between DQDB/SMU and DQDB/SRU under low traffic. Slot preuse takes places more often under high load due to the higher number of requests logged at upstream stations, and the higher number of packets waiting at those stations.

4.3 Effect of Buffer Size and Network Growth

Consider the same networks with N = 10 stations, but with the transmission buffer capacity of each station increased from M = 20 to M = 50. Point estimates of the mean access delay at S_i of segments transmitted on bus A are plotted in Figures 6 and 7 as a function of node index for p = 0.40 and 0.90, respectively. Results for networks with N = 20 stations and a transmission buffer size of 50 segments per station are graphed in Figures 8 and 9.

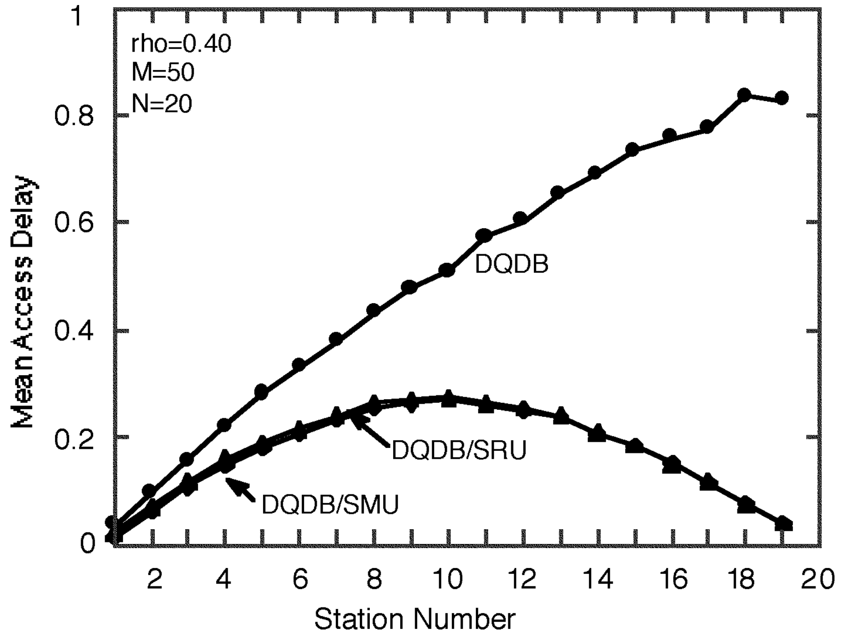


Figure 8. Mean access delay on the transmission bus when the number of stations has been increased to N = 20, M = 50 and ρ = 0.40

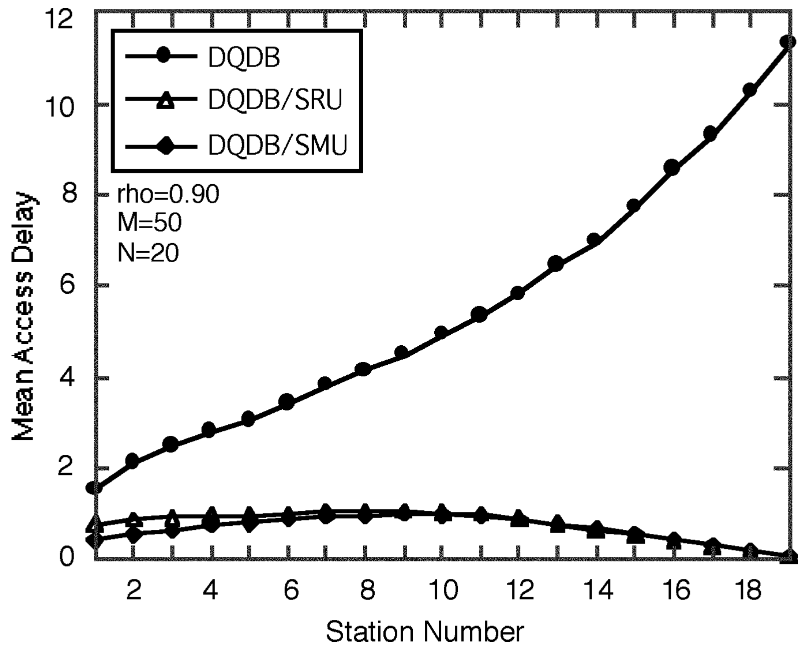


Figure 9. Mean access delay on the transmission bus when the number of stations has been increased to N = 20, M = 50 and ρ = 0.90

As can be seen, the same unfairness characteristics are evident when the number of stations is doubled, or when the buffer capacity is increased.

4.4 Throughput Comparison

The mean throughputs per bus of DQDB, DQDB/SRU and DQDB/SMU are compared in Figure 10, taking N = 10, M = 20 networks. DQDB/SRU lifts maximum network throughput to approximately 190% of that of DQDB. In contrast, DQDB/SMU lifts maximum network throughput to approximately 240% of that of DQDB. Thus while DQDB/SMU moderately reduces the mean access delays of segments at stations near the HOB (compared with DQDB/SRU; e.g., see Figures 3 to 5), it significantly reduces the network-wide mean access delay and improves throughput. This is due to the fact that stations near the HOB generate more segments for transmission on that bus, so the majority of segments benefits from the addition of slot preuse to slot reuse. On the other hand, DQDB/SRU significantly reduces the mean access delay of segments transmitted from nodes near the EOB (Figures 3 to 5), but fewer segments benefit from slot reuse since nodes near the EOB generate fewer segments for transmission on that bus.

5. Conclusions

Two approaches for upgrading standard DQDB were evaluated and their performances compared with each other and with that of standard DQDB by means of automated distributed simulation.

The preceding analysis suggests that when introducing spatial multi-use of slots to upgrade the performance of standard DQDB, the causes of unfairness are complicated by:

- Segments preusing or reusing slots can jump queue, and
- The non-zero delay until requests could be cancelled.

The simulation results suggest that under the standard traffic model:

- DQDB/SRU favours stations near EOB most of all, and the (unfair) advantage enjoyed by stations near the EOB increases with load.
- In contrast, adding slot preuse to DQDB/SRU (giving DQDB/SMU) moderately improves the mean access delay of stations near the HOB, but does not improve (reduce) the mean access delay of stations near the EOB above that achieved using DQDB/SRU. Nevertheless, it significantly reduces the network-wide mean access delay and improves throughput, since stations near the HOB generate more segments for transmission on that bus.
- Nodes which suffer (unfairly) high mean access delay are the same ones (i.e., those near the middle of the bus).
- Hence the fairness characteristics of DQDB, DQDB/SRU, and DQDB/SMU differ from each other depending on the level of offered traffic.

Thus the effectiveness of existing mechanisms proposed for improving the fairness of standard DQDB such as the (optional) Bandwidth Balancing Mechanism [10, 36, 39, 40] may not be applicable to the enhanced protocols. However, it should be acknowledged that

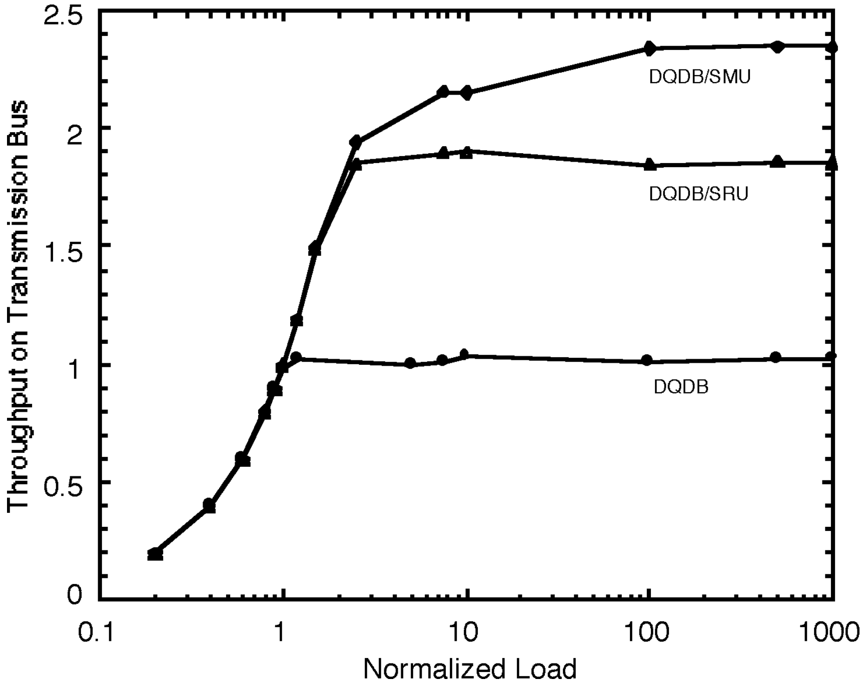


Figure 10. Comparison of throughput of DQDB, DQDB/SRU and DQDB/SMU as a function of load

both approaches to spatial slot multi-use examined here significantly improve the delay and throughput performance of all stations over standard DQDB.

5. References

[1] Huang, N-F. and Liu, H.-I. "A Study of Isochronous Channel Reuse in DQDB Metropolitan Area Networks." *IEEE/ACM Transactions on Networking*, Vol. 6, No. 4, pp 475-484, 1998.

[2] Yang, Y., Lai, T.-H., and Liu, M.-T. "Isonchronous Bandwidth Utilization Improvement in Distributed Queue Dual Bus-Based Personal Communicatin Networks." *Computer Communications*, Vol. 21, pp 1420-1433, 1998.

[3] Yau, V. 1997. **Victor: In first paragraph of Introduction, you used [YAU97] which wasn't in references. Please provide.**

[4] Burr, W.E., Wakid, S., Xiaomei, Q. and Vaman, D. "A Comparison of FDDI Asychronous Mode and DQDB Queue Arbitrated Mode Data Transmission for Metropolitan Area Network Applications." *IEEE Transactions on Communications*, Vol. 42, Issue 2-4, pp 1758-1768, 1994.

[5] "ISO/IEC ANSI/IEEE Standard 802.6 LAN/MAN Standards DQDB Access Method Packaage." IEEE, 1994.

[6] Sharon, O. "A Proof for Lack of Starvation in DQDB With and Without Slot Reuse." *IEEE/ACM Transactions on Networking*, Vol. 5, No. 3, pp 410-419, 1997.

[7] Hairong, S., Ke, H. and Lemin, L. "Performance Analysis of the Constant-Bit-Rate Services in an ATM DQDB MAN." *Computer Communications*, Vol. 21, pp 186-194, 1998.

[8] Dong, X. and Lai, T. "An Efficient Protocol for Call Setup and Path Migration in IEEE 802.6 based Personal Communication Networks." *IEEE Transactions on Computers*, Vol. 46, No. 3, pp 252-259 1997.

[9] Orozco-Barbosa, L. and Ahmed, N.U. "Dynamic Modelling of the IEEE 802.6 Medium Access Mechanism." *Proceedings of 18th Biennial Symposium on Communication*, pp 21-24, 1996.

[10] Wu, J.S. and Hsieh, Y.T. "Improving Access Delay Fairness of DQDB MANs Under Overload Condition." *Computer Communications*, Vol. 19, No. 6, pp 571-579, 1996.

[11] Atkins, J. and Norris, M. *Total Area Networking: ATM, IP, Frame Relay, and SMDS Explained*. John Wiley and Sons, 1999.

[12] Foor. 1995. **Victor: In second paragraph of Introduction, you used [FOOR95] which wasn't in references. Please provide.**

[13] Yau, V. "Lighwave Network Design and Destination Conflicts." PhD Thesis, University of Canterbury, New Zealand, Dec. 1996.

[14] Yau, V. "Optical WDMA Network Design and Function Placement." *Computer Networks Journal*, Vol. 31, No. 22, pp 2411-2427, Dec. 1999.

[15] Yau, V. and Pawlikowski, K. "CA-STAR: A Centrally-Arbitrated Passive Broadcast-and-Select Star Architecture for Lightwave Networks." *Journal of High Speed Networks*, Vol. 8, No. 3, pp 1-21, 1999.

[16] Potter, P.G. and Zukerman, M. "Analysis of a DQDB Subnetwork With Eraser Nodes." *Teletraffic and Datatraffic in a Period of Change, Proceedings of the ITC-13*, North-Holland, pp 941-946, 1991.

[17] Mukherjee, B. and Bisdikian, C. "A Journey Through the DQDB Network Literature." *Performance Evaluation*, Vol. 165, pp 129-158, 1992.

[18] Hassanein, H.S., Wong, J.W. and Mark, J.W., "An Effective Erasure Node Algorithm for Slot Reuse in DQDB." *Proceedings of IEEE INFOCOM'94*, pp 1302-1309, 1994.

[19] Jyh-Horng, W. "An Efficient Reuse Protocol for DQDB Networks." *Proceedings of ICCS'94*, Vol. 2, pp 469-473, 1994.

[20] Pach, A.R., Palazzo, S. and Panno, D. "Slot Pre-Using in IEEE 802.6 Metropolitan Area Networks." *IEEE Journal on Selected Areas in Communication*, Vol. 11, pp 1249-1258, 1993.

[21] Yau, V. and Pawlikowski, K. "Distributed Queue Dual Bus MAN with Multi-used Slots." Technical Report No. COSC 07/93, Dept. of Computer Science, University of Canterbury, New Zealand.

[22] Pawlikowski, K. "Steady-state Simulation of Queueing Processes: A Survey of Problems and Solutions." *ACM Computing Surveys*, No. 2, pp 124-170, June 1990.

[23] McNickle, D., Pawlikowski, K. and Ewing, G. "Experimental Evaluation of Confidence Interval Procedures in Sequential Steady-State Simulation." *Proceedings of the Winter Simulation Conference, WSC'96*, San Diego, pp 382-389, IEEE Press, Dec. 1996.

[24] Pawlikowski, K., Ewing, G. and McNickle, D. "Coverage of Confidence Intervals in Sequential Steady-State Simulation." *Journal of Simulation Practise and Theory*, Vol. 6, No. 3, pp 255-267, 1998.

[25] Pawlikowski, K. "Simulation Studies of Telecommunication Networks and Their Credibility." *Proceedings of European Simulation Multiconference (ESM'99)*, Society for Computer Simulation International, 1999.

[26] Yau, V. "Automating Parallel Simulation Using Parallel Time Streams." Accepted for publication in *ACM Transactions on Modelling and Computer Simulation.*

[27] Yau, V. "Automating Parallel and Distributed Quantitative Stochastic Simulation." Technical Report No. COSC 05/96, Sept. 1996, Dept. of Computer Science, University of Canterbury, New Zealand, pg 118.

[28] Yau, V. and Pawlikowski, K. "An Algorithm that Uses Forward Planning to Expedite Conflict-Free Traffic Assignment in Time-Multiples Switching Systems." *IEEE Transactions on Communications*, Vol. 47, No. 11, pp 1757-1765, Nov. 1999.

[29] Pawlikowski, K. and Yau, V. "Methodology for Stochastic Simulation for Performance Evaluation of Data Communication Networks." Technical Report No. COSC 03/93, Dept. of Computer Science, University of Canterbury, New Zealand.

[30] Pawlikowski, K., McNickle, D. and Yau, V. "Investigations of Experimental Control for Simulation." Technical Report No. COSC 02/93, Dept. of Computer Science, University of Canterbury, New Zealand.

[31] Pawlikowski, K. and Yau, V. "Independent Replications Versus Spectral Analysis in Steady-State Simulation of High Speed Data Networks." *Proceedings of ATRS'91*, Wollongong, Australia, pp 182-190, Nov. 1991.

[32] Yau, V. and Pawlikowski, K. "AKAROA: A Package for Automating Generation and Process Control of Parallel Stochastic Simulation." *Proceedings of the Sixteenth Australian Computer Science Conference*, G. Gopal, et al. (Eds.), Australian Computer Science Communications, pp 71 -83, 1993.

[33] Pawlikowski, K., Yau, V. and McNickle, D. "Distributed Stochastic Discrete-Event Simulation in Parallel Time Streams." *Proceedings of 1994 Winter Simulation Conference*, IEEE Press, pp 723-730.

[34] Yau, V. and Pawlikowski, K. "Experiences in Parallel Simulation of Dual Bus Metropolitan Area Networks for Speedup and for Obtaining Performance Estimates with Specific Accuracy." Accepted and in press, *SIMULATION*, Vol. 75, 2000.

[35] Popovich, S.G., Alam, M. and Bandyopadhyay, S. "Simulation of a DQDB MAC Protocol using NETWORK 11.5." *International Journal of Computer Simulation*, Vol. 6, No. 1, pp 113-135, 1996.

[36] Kumar, L. and Douligeris, C. "The Dynamic Three-Tier Protocol: An Access Remedial Scheme for DQDB Metropolitan Area Networks." *Computer Communications*, Vol. 21, pp 624-643, 1998.

[37] Kabatepe, M. and Vastola, K. "The Fair Distributed Queue (FDQ) Protocol for High-Speed Metropolitan-Area Networks." *IEEE/ACM Transactions on Networking*, Vol. 4, No. 3, pp 331-339, 1996.

[38] Chen, C.Y.R., Makhoul, G.A. and Meliksetian, D.S. "A Queueing Analysis of the Performance of DQDB." *IEEE/ACM Transactions on Networking*, Vol. 3, No. 6, pp 872-881, 1995.

[39] Conti, M., Gregori, E. and Lenzini, L. "Influence of the BWB Mechanism on Some Performance Figures of a DQDB Subnetwork." *Computer Networks and ISDN Systems*, Vol. 27, Issue 7, pp 1137-1161, 1995.

[40] Ferguson, M.J. "DQDB: An Overload Cycle Analysis of Generalized Bandwidth Balancing with Strict Priority." *Performance Evaluation*, Vol. 23, Issue 1, pp 53-63, 1995.

Victor Yau was born in Hong Kong, China. He received his BSc degree from the University of Otago, New Zealand, in 1987. In 1988 and 1989, he worked as a Systems Analyst with the Information Technology Service Center, Wellington, New Zealand. He received a PostGraduate Diploma in Applied Science from Victoria University of Wellington in 1990, and a PhD degree from the University of Canterbury, Christchurch, New Zealand, in 1996, both in Computer Science. The work that Dr. Yau reports in this paper was performed in part while he was working with the German National Research Center for Information Technology, GMD FOKUS, in Berlin, Germany. Currently he is a Visiting Associate Professor in the Department of Computer Science and Engineering at the University of Texas at Arlington. Dr. Yau's research interests include simulation, communication systems, and distributed computing.

Krzysztof Pawlikowski is an Associate Professor (Reader) in Computer Science at University of Canterbury, Christchurch, New Zealand. He received his PhD in Computer Engineering from the Technical University of Gdansk in Poland. Dr. Pawlikowski is the author of more than 90 research papers and four books. His research interests include stochastic simulation, cluster processing, performance modelling of telecommunication networks (Internet, ATM, optical and wireless technology) and teletraffic modelling. He is a Senior Member of IEEE. More information is available at *www.cosc.canterbury.ac.nz/~krys*.