1 High Speed Protocols

High speed Metropolitan Area networks, must provide a standard means of interconnecting a large number of users (few hundreds to a thousand) spread over a large area (50 km to 100 km) using high speed fiber optic links. The ideal high speed protocol will allow users to transmit information at rates close to the raw bit rate that the media can sustain. In practice, additional limitations are imposed, usually by those aspects of the protocol that effect distributed control. The mechanics of carrier sensing, which is key to distributed control in Ethernet, imposes a strong limitation on the span of Ethernet networks. Similarly, the per station overhead of token handling, imposes limitations on the total number of nodes that can be networked satisfactorily using token passing schemes.

The IEEE 802.6 protocol, better known as DQDB (Distributed Queue Dual Bus), has received much attention because it was designed for the fiber optic environment and like ATM, segments data into 53 byte cells. Since DQDB does not involve token passing, the delay induced by the protocol is not directly related to the maximum propagation time on the network. As a result, DQDB is more suitable for Metropolitan Area networks than FDDI which is a token passing protocol. The access scheme in DQDB makes it possible for a node to write data on the network via an OR interface. This eliminates the need to extract and reinsert packets from the network. Thus the addition of a new node does not result in additional delays for every other node in the network. Consequently, DQDB copes better with very large numbers of users than token passing schemes.

An individual user or node in a Metropolitan Area Network could be a single user or a gateway to a local area network. Such a mixed set of users is better served by a network that employs short packets or cells, like DQDB, than by a network like FDDI, which employs much larger frames. It is also desirable to be able to assign a grade of service or type of privilege that is appropriate to the user. One of the drawbacks of DQDB is the inability to do so. The shape of the delay curve as a function of the user’s spatial location is non-uniform and unalterable. This implies that the grade of service received by a user will depend significantly on its location regardless of its needs. Furthermore there are no parameters that can be used to customize the access privileges of nodes that have special needs. A complete and accurate delay analysis of DQDB has so far not been presented.

Although the availability of high speed processors makes it possible to implement complex protocols, there is value in analyzing protocols. A careful analysis may reveal hidden flaws in a protocol, or lead to insight that can be used to refine the architecture of the network.

Analytical tractability was a key criterion in the development of this protocol. Our ultimate goal is to study end-to-end performance of a network that involves access protocols, like DQDB, connected to wide area networks, like Frame Relay, through gateways. For the most part, such networks are impossible to study analytically and approximations have to be employed. This leads to some uncertainties. As an alternative, we could formulate protocols that are easier to analyze even if they are not quite as efficient. We are attempting to trade rigor of analysis for protocol efficiency. This is an unorthodox approach. Such an approach might be deemed unacceptable, especially if the application environment is well understood. But, if the application environment is complex and ill understood, there may be some
value to using protocols whose performance is not quite as uncertain.

In this paper we present a new protocol, henceforth referred to as CPCQ (Counteracting Priority Coupled Queue). It may be viewed as a simpler alternative to DQDB. It can operate just as effectively in the high speed fiber optic environment, in addition it is analytically more tractable and allows for assignable privileges.

2 Description of CPCQ

Just as DQDB, the CPCQ system is composed of two uni-directional buses, that carry data in opposite directions. Each user is connected to both buses. Each bus has a control unit at the head of the bus, that generates fixed length slots and performs other housekeeping functions. The length of the slot is 53 bytes just as in DQDB. These slots have access control fields that are used to communicate control information. Of particular interest are the busy bits and the reservation bits.

When a user needs to transmit to one of its neighbors, it first identifies the appropriate bus to transmit on. Say this is bus A. The user must first send a reservation on bus B in the opposite direction. Subsequently, the user may transmit on the upper bus. More precisely, the reservation and transmission process evolves as follows.

Each user has a pair of queues connected in tandem. Arriving packets initially queue up in the external queue. They can move from the external queue to the internal queue once they successfully make a reservation. User $i$ attempts a reservation on the lower bus, with probability $p_i$ whenever it has a packet to transmit. The reservation is successful only if the slot was not previously reserved by another downstream user. This can be determined by examining the request bit, which gets set to one by the user that succeeds in making the reservation. Because of the unidirectional nature of the bus, this implies that user $i+1$ has a higher priority than user $i$ on the reservation bus.

Whenever packets are queued up in its internal queue, user $i$ attempts to transmit on the upper bus, with probability $r_i$. The transmission is successful only if the slot is not already occupied by an upstream user's packet. This can be determined by examining the busy bit, which gets set to one by the user that succeeds in occupying the slot. It can be seen that user $i+1$ has a lower priority than user $i$ on the transmission bus.

There are thus two stages in the interaction between the users. First the interaction with the downstream users to make a reservation, and then the interaction with the upstream users to transmit. It is intuitively clear that a user's location on the bus will determine which of the two is likely to be the bottleneck. Users at the far end of the bus can make reservations quite quickly but wait much longer thereafter to transmit. In contrast, the users on the near end may have to wait for long to make a reservation but the time to transmit thereafter will be much less. These two counterbalancing forces lie at the heart of both the CPCQ protocol.

2.1 Relationship with DQDB

As in DQDB, the users in the CPCQ system can be split into those downstream and those upstream of any given test user. The influence of users downstream of a test user is projected over the reservation channel and the influence of the upstream users is propagated over the transmission channel. If all the downstream users are idle then the test user can make a reservation whenever it chooses to. If all upstream users are idle then free transmission slots will become available immediately.

There are obvious differences. In CPCQ, the internal queue does not track the requests made by the downstream users and nor is there any effort to let exactly as many transmission slots go by. This is the primary reason for the model's tractability.

Finally, it should be clear that CPCQ can be physically implemented using the same hardware as DQDB. The dual bus topology and the OR write interface, two crucial features of DQDB, are preserved. This suggests that CPCQ can be used as an alternative if its performance is found to be advantageous.

3 Results

Analytical studies of CPCQ have been reported elsewhere [1,3]. In this manuscript, we discuss performance figures obtained by emulating the proposed protocol. In all of these examples the
transmission rate was assumed to be 44.7 Mbps. Each packet was assumed to have 48 bytes of data and 5 bytes of overhead. In view of the symmetry we focus on traffic flow from left to right. Identical figures can be obtained for flow in the opposite direction. The total load is always apportioned to produce equal traffic between every pair of users. The arrival process is assumed to be Poisson. The multiplexing delay figures obtained correspond to the duration of time a packet takes from the instant of generation to the instant of transmission on the upper bus. It does not include propagation time on the upper bus. The propagation delay is not protocol sensitive and hence our focus on multiplexing delay.

CPCQ Throughput. Total traffic rates can approach unity without pushing the protocol into an unstable region, for a wide range of cable lengths and user population sizes. This attests to the high speed nature of the protocol.

Figure 1,[2], documents the sensitivity of the protocol to the number of nodes. The data presented here pertains to a system in which the total load was held constant at .95 packet arrivals per slot. The users were spread uniformly over lengths of 19, 38, 76 and 121 Kms. This amounts to assuming that the maximum number of slots that can simultaneously propagate on the bus is equal to the number of nodes.

CPCQ Protocol Overhead. The total delay figures are relatively small compared to the propagation delay on the cable. Even under heavy load conditions the multiplexing delay is less than the average delay that would result from propagation delays on the cable.

It is of interest to study how initial buffer occupancies affect the performance of the protocol. We studied a set up in which 39 nodes were spread over 38 Kms. The total load was .95 packets per slot. Figure 2 documents the sensitivity to the initial buffer occupancies. The three curves correspond to all buffer occupancies equal to zero, all buffer occupancies except at node 20 equal to zero, and all buffer occupancies except at nodes 20 and 39 equal to zero. In the latter two cases the initial occupancy was 100. There is little impact on the multiplexing delay at neighboring nodes. However there is a significant impact on nodes at the head end. We think this latter effect is due to the high overall load of .95 packets per slot. We expect this effect to have a lesser impact if the load is less.

Figure 3,[2], documents the sensitivity of the protocol to the span of the network, again for a total throughput of .95 packets per slot.

© 1992 IEEE
Shape of the Delay Curve. As the total traffic increases, the shape of the delay curve changes from essentially flat to convex cup. This is inevitable in some ways. The higher order terms in the delay distribution on the individual buses become more prominent as the total traffic increases. As a result the total delay is smaller for the users in the center than the ones at either ends. This is documented in Figure 4, [2], for a system which consists of 9 users spread uniformly over 19 km.

Choice of $p_i$ and $\pi_i$. $\pi$ and $p_i$ are best set to one for some values of the parameters, as shown in Figure 5. We had not expected this outcome but it is clear that setting these parameters to other values will not reduce the average value of the traffic.

A first order analysis would also suggest that the delay will depend primarily on the average value of the transmitted traffic. This appears to imply that higher order effects may sometimes not be very prominent. In fact using non unity values will increase the variance of the service process and possibly add to the delay.

Figure 6 shows the multiplexing delay for a different choice of parameters, this time with 40 users spread over 76 km and carrying a total load of .95. Now we see that there is little that is lost by decreasing $p_i$. This may in fact be desirable, because it would prevent a heavy user from hogging the reservation channel. In this case setting $p_i = .5$ led to a very large delay for user1. There is of course no need to set $p_i$ to anything other than 1, since there are no other stations that can exploit unused slots. The consequences of making an assymetric assignment of $p_i$ has not yet been fully explored.

4 Refinements

The dependence of the multiplexing delay on $p_i$ and $\pi_i$ lead us to the realization that the shape of the delay curve is best controlled by directly effecting the the total average traffic. The total traffic on the transmission bus can not be altered by making changes to the protocol. It is after all a function of the arrival rate. On the other hand the total traffic on the reservation bus can be altered. For example, we could allow up to $K_i$ packets from user $i$'s external queue to move to the user's internal queue for every successful reservation that user $i$ makes. This modification could cut down the reservation traffic by some amount. The actual reduction will be an increasing function of the total traffic. This phenomenon was indeed observed and is exhibited in Figure 7 [2]. One may note that there is a reduction in delay for the most upstream nodes. These are the nodes that suffer most due to contention on the reservation bus. The most downstream nodes suffer on account of contention on the transmit bus,
and this component is not altered significantly by a reduction of contention on the lower bus.

![Fig 7: Sensitivity to the Reservation Traffic](image)

Figure 8 is similar to figure 6 except that we have set all $K_i$ to 10. A comparison of figures 6 and 8 will reveal that this choice of $K_i$ demonstrates the value of this feature.

![Fig 8: Sensitivity to access probability](image)

5 Conclusions

CPCQ is qualitatively similar to DQDB in that the delay curve is not uniform. CPCQ is simpler to implement because the protocol requires only limited sensing. Nodes need not constantly monitor the channel to track the number of outstanding requests from downstream nodes. In this manuscript we have explored the multiplexing delay induced by the protocol. We have shown that it possesses many desirable properties. These include, robustness and lack of sensitivity to a number of system parameters. Further exploration of the multiplexing delays and analytical studies of the delay induced by the protocol is underway.

6 References

