On two Collision Protocols for High Speed Bus LANs

Włodek Dobosiewicz [*] Paweł Gburzyński [†] Piotr Rudnicki [‡]

December 22, 1994

Reprinted from Computer Networks and ISDN Systems 25(1993) 1205-1225

Abstract

It is commonly believed that collision protocols are impractical for very fast bus networks, owing to the constraints on the minimum packet length. In this paper, we present two collision protocols that contradict this opinion. These protocols are two representatives of the so-called PIGGYBACK ETHERNET family. They incur no synchronisation overhead under very light load, and under heavy traffic conditions they perform no worse than collision-free protocols. PIGGYBACK ETHERNETS are implementable on the basis of a simple, unsegmented, bidirectional, broadcast-type medium. Thus, they can be viewed as refinements of commercial ETHERNET aimed at providing high-performance service in the environments when the packet length is small in comparison to the propagation length of the bus.

[*]Supported in part by NSERC Grant No. OGP9110. Department of Computing Science, The University of Alberta, Edmonton, Alberta, Canada T6G 2H1. email: dobo@alberta.UUCP.

[†]Supported in part by NSERC Grant No. OGP9183. Department of Computing Science, The University of Alberta, Edmonton, Alberta, Canada T6G 2H1. email: pawel@alberta.UUCP.

[‡]Supported in part by NSERC Grant No. OGP9207. Department of Computing Science, The University of Alberta, Edmonton, Alberta, Canada T6G 2H1. email: piotr@alberta.UUCP.

Key words: High-speed Local area networks, CSMA/CD, Ethernet, Network performance, Fairness.

1 Introduction

This paper presents two collision protocols for bidirectional bus networks. These protocols are simple and inexpensive; yet, they offer good performance for all loads. Moreover, they are well-suited for long and fast networks.

Bus networks based on collision protocols (CSMA/CD family) are very popular means of interconnecting distributed systems spanning small geographic areas. The most successful representative of this class is commercial ETHERNET whose simplicity and good performance for light and moderate load are commonly known. Unlike numerous protocols based on complex bus reservation schemes, the ETHERNET protocol incurs no synchronisation overhead for a lightly loaded network, i.e. when there is only one station willing to transmit a packet. As most networks typically operate far below their maximum capacity, this property of ETHERNET gives it a serious advantage over more complicated (and thus more expensive) solutions.

However, ETHERNET is not suitable for all networks. One important parameter characterising a bus-type network is the ratio of the maximum distance between stations (L) to the size of a typical packet (l_p) which is denoted by a ($a = \frac{L}{l_p}$). If a is larger than 1/2, most messages are shorter than the round-trip propagation delay ($2L$), which causes performance degradation of many protocols (cf. [20]).

The commercial ETHERNET (and many other bus networks based on collision protocols) requires that the minimum packet length is not shorter than the round-trip propagation delay of the bus expressed in bit-slots. This is necessary to make all collisions detectable while the packets involved are still being transmitted. If a packet is shorter than the required minimum, it

must be filled with dummy bits (*inflated*). In consequence, the performance of the network degrades, as the useful traffic consists only of a certain percentage of all transmitted bits. This percentage becomes smaller and smaller with growing a.

Even in the commercial network, packet inflation becomes a serious problem for certain types of traffic (e.g. character-oriented communication, signal passing). For a network substantially faster than ETHERNET, the performance degradation due to stuffing the network with dummy data would be intolerable (cf. [31]). If collisions are to be recognised properly by the transmitter, the network must obey the minimum packet length; therefore, collision protocols are applicable for fast or long bus networks only if the frequency of collisions can be made very small.

Similar arguments can be voiced against various collision-free protocols for broadcast bus networks. Many such protocols are based on demand assignment schemes and the amount of time required to resolve contention among multiple stations willing to transmit their packets at the same time is dependent on $2L$. This time interval constitutes the so-called *slot* during which multiple backlogged stations are able to execute one step of the contention resolution algorithm. Thus, it is quite easy to see that in fact these protocols offer no real improvement over CSMA/CD-based solutions–at least as far as fast or long channels are concerned.

To many people, the above-mentioned obstacles appear as unavoidable drawbacks of two-way broadcast-type channels. Therefore, the main direction of search for bus protocols applicable to fast networks has been aimed at exploring the potential of unidirectional, segmented, channels (cf. [8, 12, 13, 21, 25, 30, 31, 32]), especially as optical fibres provide a natural

basis for implementing such communication media. A notable exception is [20].

The goal of this paper is to demonstrate that bidirectional channels controlled by CSMA/CD protocols can be used in circumstances when the packet length is substantially smaller than the length of the channel. The proposed protocols are hybrids combining most of the desirable properties of ETHERNET, e.g. the lack of synchronisation overhead for very low traffic, with the properties of certain schemes using the *scheduling-delay access mechanism* (cf. [7]), i.e. good performance for heavy load.

The quantitative results presented in this paper have been obtained by modeling the investigated networks and protocols in LANSF ([11]), which is a simulation package[1] oriented towards modeling distributed systems.

2 A brief overview of fast bus-type LANs

A bus network consists of a number of stations distributed along a linear channel—the bus (see figure 1). In the simplest case, the bus consists of single, bidirectional, broadcast-type medium. For some networks, the structure of the bus is more complex, i.e. it may be composed of a number of separate channels which, additionally, may be unidirectional.

For convenience, the two ends of the bus will be referred to as the *left* end and the *right* end. The length of the bus is expressed as the *end-to-end propagation delay* (L) in bit-slots (e.g. for a 10Mb/s ETHERNET, 1 bit-slot = $100ns$). Each station is assigned a specific location on the bus.

[1] The system runnable under UNIX is available from the authors upon request. UNIX is a trademark of AT&T Bell Labs.

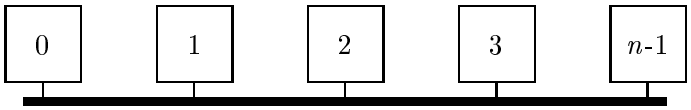

Figure 1: A bus network

The location of a station S will be represented as its distance (in bit-slot time units) from the left end of the cable and denoted by $l(S)$. For every two stations S_1 and S_2, the propagation distance between them is given as $d(S_1, S_2) = |l(S_1) - l(S_2)|$.

2.1 Ethernet

In the case of ETHERNET ([22, 26, 28]), the bus consists of a single, uniform, broadcast type channel, e.g. a coaxial cable. Stations may access the channel passively (e.g. to check whether it is *idle* or *busy*), or actively (e.g. to transmit a packet). Each station is able to detect a collision of its packet caused by interference from another station transmitting at the same time. All collisions must be detectable while the colliding packets are still being transmitted. A station detecting a collision of its packet aborts the transfer and sends a *jamming signal* to force the so-called *collision consensus*, i.e. to make sure that the collision is properly recognised by all the parties involved. Then the station waits for a random amount of time (determined by the protocol) before attempting to retransmit the packet. The commercial network uses the so-called *Binary Exponential Backoff* algorithm to determine the waiting time before retransmission.

Packets in ETHERNET must be spaced, i.e. a period of silence (the so-

called *inter-packet space*) must precede the beginning of a valid packet. Before a station decides to transmit a packet, it checks whether the channel has been idle for the amount of time corresponding to the inter-packet space. If not, the station waits until the packet space has been obeyed and then starts the transfer.

The relevant numerical parameters of commercial Ethernet (cf. [28]) are as follows. The inter-packet space – 96, the jam length – 32-48, the minimum packet length – 576, the maximum packet length – 12208. The minimum and maximum packet length includes 208 bits of frame information, i.e. the packet header and trailer.

Several attempts were made to improve various performance characteristics of ETHERNET, retaining the collision nature of the protocol (e.g. see [2, 4, 5, 9, 10, 16, 19, 23, 24]). Some of these attempts ([4, 16, 23, 24]) were aimed at reducing the impact of collisions. Some other modifications ([2, 9, 10, 19]) were directed towards adapting the protocol for real-time applications by making the maximum packet delay bounded.

To the best of our knowledge, only one attempt has been made so far to devise a version of CSMA/CD that would be applicable to a very fast (or a very long) network ([20]). This version places a number of strong requirements on the network: all the neighbouring stations must be equidistant and the maximum length of a packet must be less than or equal to the distance between neighbouring stations. These requirements make this network difficult to compare with other networks, which place no such requirements.

2.2 Scheduling delay access methods

A few bus protocols applicable to bidirectional broadcast channels with small a have been proposed in [1, 14, 15, 17, 33]. All these protocols fit into the category of schemes in which a station senses a synchronising event in the bus and schedules its transmission after a certain time delay following this event. The two new protocols proposed in this paper are based on the same general concept. Our approach is different from the above-mentioned solutions in the lack of synchronisation overhead for a lightly loaded network.

In *SOSAM* (cf. [15]), stations are assigned virtual numbers determining the order in which they are allowed to transmit. All stations have complete knowledge of the propagation delay between every pair of stations. The synchronising event is EOC (*end of carrier*) indicating the end of a packet transmission. Having detected an EOC triggered by a packet transmitted by a station S_i, a station S_j uses its knowledge of the network geometry to determine the amount of time required to learn whether stations $S_{i+1}, ..., S_{j-1}$ have packets to transmit.

In *BID* (cf. [33]), a similar approach is taken; however, the synchronisation overhead is reduced by organising the network in such a way that the station numbers correspond to the order of their occurrence along the bus. The end stations are responsible for starting *rounds*. A round is initiated by sending either a regular packet, if the corresponding extreme station has a packet to transmit, or a special dummy packet, if the station has nothing to send. In one round, all stations are allowed to transmit their packets in the order determined by their numbers. If the round has been started by the left-end station, the transmission order is from left to right. Otherwise,

the transmission order is from right to left.

Silentnet (cf. [17]), is a refinement of *BID* resulting from the observation that rounds can be started in a distributed way. Similarly as *BID*, the protocol operates in alternating rounds; however, a round can potentially be started by any station. Under low traffic, stations generate dummy packets to sustain the synchronised operation of the network.

Yet another variation on the theme is *L-Expressnet* (cf. [1]), in which all rounds are in the same direction, e.g. from left to right. Each round is initiated by the leftmost station that remains alive: there is no need for a distinguished station synchronising the network operation. All stations use the concept of *idle time counting* (a similar idea is employed in *Z-Net*—see below) to determine when a next round is to be started. If the station starting a new round has no regular packet ready for transmission, an explicit start-round dummy packet is used instead.

The scheduling delay access methods are handicapped by the need to sustain the synchronised operation of all stations—by sending dummy packets that force round-initiation, even if the network is silent. Irrespective of the circumstances, a station getting a packet to transmit must wait for its turn. If the network is idle, this turn may come reasonably soon; however, it seldom comes exactly at the moment when the station becomes ready. Therefore, the performance of such a protocol under very light load may be substantially worse than the performance of ETHERNET (see figure 8).

2.3 Protocols based on unidirectional channels

Unidirectional channels, naturally implementable on the basis of optical fibres, provide a convenient backbone for implementation of several bus pro-

tocols for fast LANs.

Expressnet ([7, 30, 31]) is a simple representative of this protocol class and it illustrates well the special properties of a unidirectional channel. In *Expressnet*, each station has two connections to a single unidirectional bus (see figure 2). A station wiling to send its packet awaits the *EOC* (end of carrier) event on the *outbound* segment of the bus. Immediately upon sensing such event the station starts transmitting a short preamble—a sequence of bits typically preceding a packet. While transmitting the preamble, the station can be preempted by another transfer arriving from the upstream of the cable. In such case, the upstream transmission wins and the preempted station aborts its transfer.

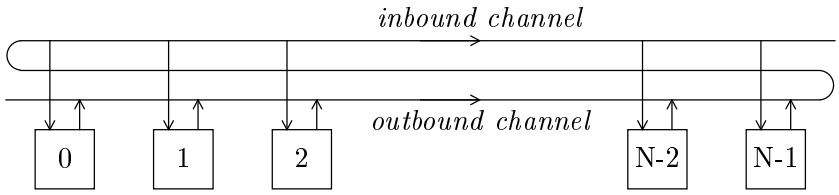

Figure 2: Expressnet.

Having completed a successful packet transmission, the station waits until the next *EOC* event before it attempts to transmit its next packet. This way all stations get an equal share of the network bandwidth.

A station detecting a period of silence following an *EOC* event on the *inbound* segment emits a short burst of activity on the *outbound* segment. This activity starts the next *cycle* of the protocol: the *EOC* event triggered by this burst will allow the backlogged stations to start their new transfer attempts.

Packets in *Expressnet* form *trains*, each train consisting of packets transmitted by all stations that were backlogged at the beginning of the current cycle. The end of such a train is detected by all stations on the *inbound* segment of the channel and indicates the beginning of the next cycle.

Several refinements of the *Expressnet* concept have been proposed, e.g. *C-Net* and *D-Net* ([32]). Their purpose is to start the packet trains in such a way as to reduce of the network bandwidth that is wasted.

The backbone of *Fasnet* (cf. [7, 21, 31]) consists of two unidirectional cables (see figure 3), each cable being used for transfers in one direction. The two extreme stations S_0 and S_{n-1} are responsible for inserting slots into the LR cable (used for transfers from left to right) and the RL cable (used for transfers from right to left), respectively. A slot is a framed period of silence in the channel where a station can insert a packet. Three binary flags are associated with each slot; these flags are used to combine slots into cycles. Within each cycle, every station that has a ready packet going in the appropriate direction can insert that packet into one of the slots.

Cycles on the LR channel are started by S_0. The end of such a cycle is detected by S_{n-1} when it notices an empty slot. Then, by setting a special flag in its next slot (going to the left), S_{n-1} informs S_0 that new cycle should be started. The two busses (and the behaviour of the extreme stations) are symmetric.

Similarly to the scheduling delay access methods, *Fasnet* must sustain the synchronised operation of all stations by generating slots and cycles, even in the absence of traffic in the network. The performance of such a protocol under very light load is worse than the performance of ETHERNET (see figure 8).

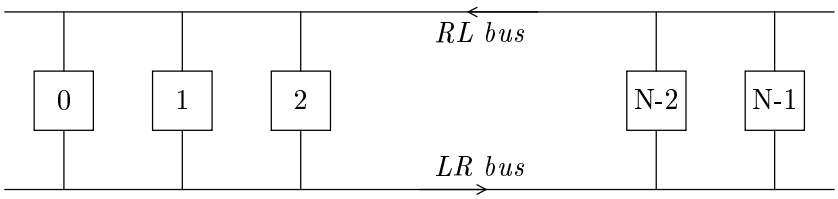

Figure 3: Fasnet.

Fasnet suffers from yet another disadvantage: the network operates in a slotted manner and the slot length must remain fixed. Although the actual maximum throughput achieved by *Fasnet* may be higher than for the other networks, this is due to the fact that *Fasnet* uses two separate channels operating independently. Despite the advantage of the two independent channels, the throughput of *Fasnet* may deteriorate quite drastically when the packet length is much shorter than the slot size.

Similar two-cable architectures have been used in two more refined protocols that attempt to eliminate certain drawbacks of the simple protocols mentioned above. *Buzz-Net* (cf. [13]) and the more recent *Z-Net* (cf. [18]) are unslotted protocols with dual mode of operation. Under very light load, the network remains in an *uncontrolled* mode in which a packet can be sent immediately, as soon as it arrives at a station. Upon detecting a contention, the stations switch into a controlled mode of operation in which they are synchronised by certain events in the channels.

Z-Net does it particularly nicely by avoiding any special synchronising signals: the *controlled* mode of operation is achieved in a purely distributed fashion. In *Z-Net*, there is absolutely no bus access delay when the traffic is very low. The penalty that is paid for the good performance character-

istics of *Z-Net* is the somewhat complex architecture of the network. It is based on two unidirectional busses, one bus being segmented by active taps. These active taps allow the stations to disconnect the channel and force their transfers despite possible interference from the upstream.

2.4 A critique

ETHERNET is a very simple, yet a very efficient protocol. Its simplicity is hard to match; therefore, it will be used as a reference point for all the more sophisticated protocols[2]. All the protocols that outperform it have an additional complexity, achieved by using one (or more) additions:

- Additional hardware: more than one tap per station, more than one channel, etc.
- Added knowledge of the topology (each station knows its relative position on the bus), partial geometry (each station knows the distances between itself and all other stations) or even full geometry (each station knows the distances between every pair of stations) of the network.
- A dual mode of operation, which makes the behaviour of the protocol depend on either the traffic intensity, or a similar parameter.
- Additional responsibilities given to certain stations.

The objective of adding extra hardware varies from protocol to protocol and is inherently related to its basic design.

[2]This paper discusses a selected subset of all bus protocols. Blanket statements refer to the protocols discussed only.

The added knowledge of the network topology and/or geometry is used to achieve one or more of the following goals:

1. introduce an implicit *token* (or *train*) controlling bus acquisition.

2. sending packets only in the proper direction.

3. reduce packet spacing and other delays.

All protocols in which stations are synchronised by explicit or implicit *tokens* (or *trains*) require a certain bus access delay, even when there is no contention. Considering that ETHERNET needs absolutely no bus access delay under very light traffic conditions, some protocols use a dual mode of operation, switching to ETHERNET mode when traffic is light.

Some protocols give additional tasks to certain stations, e.g. the extreme stations on the channel(s). This way, it becomes easier to synchronise token movement. On the other hand, such an approach makes the network vulnerable in the case of malfunction of one of the head stations.

Added complexity has its price. This price can be paid in additional cost (hardware), extra time for network initialisation and reconfiguration, reliability (distinguished stations and, sometimes, dual mode of operation).

In *SOSAM*, every station must know the distances between all pairs of stations. This requires a quadratic storage (with respect to the number of stations in the network) and renders the scheme practically useless for a network consisting of a non-trivial number of stations. This problem is eliminated in *BID*, *Silentnet*, and *L-Expressnet*, where it is assumed that the stations are numbered in the order of their occurrence along the bus. This constraint is much less serious: in most LANs, also in commercial

Protocol	Added hardware	Knowledge	Reliability
SOSAM	—	full geometry	—
BID	—	topology	two end stations
Silentnet	—	topology	—
L-Expressnet	—	topology	—
Expressnet	unidirectional carrier of length $3L$, extra receiver taps	—	—
Fasnet	two unidirectional carriers of length L, dual taps	topology	two end stations
Buzz-Net	two unidirectional carriers of length L, dual taps	—	buzz signal
Z-Net	two unidirectional carriers of length L, dual taps, disconnecting switches	—	switches
Piggyback 1	—	partial geometry	—
Piggyback 2	—	topology	—

Table 1: Cost comparison of protocols.

ETHERNET, station addresses can be defined by software operations. In *Fasnet*, a transmitting station must know the direction of the transfer, i.e. whether the receiver is located on the left or on the right side of the bus. No "special" knowledge about the network structure is required for *Buzz-Net* and *Z-Net* with exception of some commonly known estimate on the maximum propagation delay L.

While comparing the protocol performance, one should keep in mind that the hardware requirements of different protocols vary (see table 2). For instance, a single-channel network cannot be compared directly with a two-channel network—after all, any single-channel network can be turned into a double-channel one by placing two channels side by side. Thus, we should use the following guidelines when comparing the throughput of various networks:

- software costs (initialisation, reconfiguration, protocol running time) are ignored.
- if a network uses more than one channel, the throughput is divided by the number of channels.
- in *Expressnet*, there is only one channel (albeit thrice longer) and an extra set of receiver taps. We will treat it as one half of an additional channel; thus, the throughput of *Expressnet* will be divided by 1.5.

We will not use these guidelines explicitly (with the exception of figure 5). This approach is consistent with most analyses in the literature—this is the main rationale for not scaling our graphs. The reader should bear in mind that when two protocols are compared, one of them may have an unfair advantage because it uses more hardware resources.

3 Piggyback Ethernet Version 1

This section presents the more complex of the two versions of the proposed protocol. Section section 4 shows how to reduce the protocol complexity without affecting its most essential properties. We assume the following prerequisites:

- Stations are numbered in the order of their occurrence along the bus. For convenience, we assume that the leftmost station has number 0 and the rightmost station is numbered $N - 1$.
- Every station knows the bus location of all stations in the network. The bus location of a station is given as the propagation distance from the leftmost station to the station (this requires N entries per station).

The above requirements may seem quite strong: whenever a new station is added to the network, the stations must be re-addressed and the entire network must be re-initialised. A similar problem occurs when some existing stations are moved to different locations. Let us note, however, that irrespective of the protocol, addition of a new station to a local network requires that the address tables at all stations be updated. The requirement that the address of a station reflects directly its location on the bus can be easily satisfied as station addresses are set by software operations.

The second requirement boils down to augmenting the format of the address table by one additional entry for each station in the network. Of course, we do not postulate that the contents of this entry be filled manually by the network manager. In section 6, we sketch an automated procedure of network (re)configuration. This procedure is described in detail in [6].

3.1 The operation of the protocol

The protocol can be viewed as a bimodal variant of a scheduling delay access scheme (cf. [7]). The packet header is augmented by one bit denoted by P[3] whose contents represent the so-called *piggyback direction* (left or right). In the normal (*uncontrolled*) mode of operation, a station is allowed to transmit immediately, provided that the channel is sensed idle. A station transmitting a packet in the uncontrolled mode sets its P bit at random and **inflates** the packet so that its length is not less than $2L$. This way the station is able to detect a possible collision while the packet is being transmitted.

The impact of packet inflation is negligible under very light low, i.e. when only one station at a time is willing to transmit a packet. If a collision occurs, it is handled in the same way as in ETHERNET, i.e. the participating stations abort their transfer attempts and reschedule them at a later time determined by the randomised binary exponential backoff function. After a collision, the stations **remain in the uncontrolled mode**.

A successful packet transmission (which is recognised by all stations in due time) switches the network to the *controlled* mode. When the network is in controlled mode, the station that was the last to transmit a packet is called the *leader*. Other stations are allowed to append (*piggyback*) their packets at the end of the leader's packet, with priorities derived from their distances to the leader and the value of bit P. When a station appends a packet to the leader's packet, it takes over the role of leader.

Each end of a successful transmission triggers a synchronising event which is perceived by all stations in the network. The stations use this event

[3]The IEEE 802.3 frame format includes a 2-byte type field which is unused by the MAC layer. One bit of this field can be used to represent P.

to trace the location of a *virtual token* that makes a *full circle* through the bus, visiting each station twice. The token is launched at the leader—at the moment when the leader completes its (successful) transmission.

The controlled mode is exited after the token has made the full circle through the stations and no station had a packet ready to transmit. Note that each transmission starts a new token; therefore, as long as there is a continuous supply of traffic in the network, the controlled mode is never exited.

When a station S is in the controlled mode, it performs the following steps:

1. (A transmission is in progress). Wait until the end of the transmission.

2. (EOT was just heard). Set A to be the position of the new leader. Determine the value of bit P in A's packet. Compute three time delays $d_1^l(A,S)$, $d_2^l(A,S)$ and d_e.

3. Wait $d_1^l(A,S)$ and if a transmission is sensed during this wait, go back to state (1).

4. ($d_1^l(A,S)$ elapsed since the last EOT). If a packet is ready for transmission, send the packet and then go back to state (2).

5. (S did not have a ready packet) Wait $d_2^l(A,S) - d_1^l(A,S)$ and if a transmission is sensed during this wait, go back to state (1).

6. ($d_2^l(A,S)$ elapsed since the last EOT). If a packet is ready for transmission, send the packet and then go back to state (2).

7. (S did not have a ready packet). Wait $d_e - d_2^l(A,S)$ and if a transmission is sensed during this wait, go back to state (1).

8. (d_e elapsed since the last EOT). Switch to uncontrolled mode.

The idea behind the three delays can be explained as follows. Let t_0 be the moment when S perceives the end of the packet transmitted by A. At $t_0 + d_1^l(A, S)$, the token will visit S for the first time, at $t_0 + d_2^l(A, S)$, the token will arrive at S for the second time, and at $t_0 + d_e$, the token will disappear from the network.

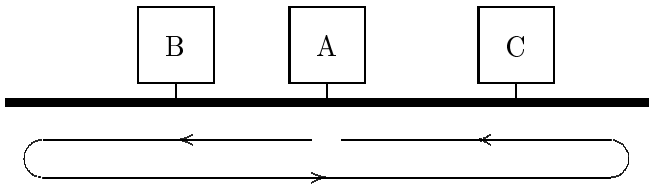

Figure 4: The order of turns in the controlled mode ($P = 0$)

Let us illustrate the operation of the protocol with an example. Assume that station A (see figure 4) transmits its packet successfully in the uncontrolled mode and that the P bit of the packet contains 0 which indicates the piggyback direction to the left. A becomes the leader station. Consider the actions of two stations B and C. Upon sensing the A's packet, B and C recognise that it originated from A and that the P bit was set to 0. When B and C sense an EOT, they switch to controlled mode and treat A as the current leader.

According to the piggyback direction determined by P, the virtual token will visit B before C. In fact, it will visit B twice (similarly as all stations located to the left of A), then it will pass through the stations between A and C (including A), next it will arrive at C, and so on, until it reaches A again. Logically, as soon as B perceives the end of the A's packet, it could

transmit its own packet. Note, however, that the stations located between A and B were visited by the token before B. If any of them transmitted a packet, B must be able to learn about this fact before B initiates its own transmission. Note that we cannot use the approach of *Expressnet* in which a transmission in progress can be preempted by another transmission coming from the upstream. Unlike in *Expressnet*, our notion of *upstream* is logical and it changes when the virtual token makes a turn. All activities, including destroyed packet preambles that would result from this approach, travel in both directions and can damage packets received at "upstream" destinations.

Therefore, another approach is taken. Having heard the end of the A's packet, B delays its transmission to make sure that the A's packet is not followed by another packet transmitted by one of the stations located between A and B. By sensing the carrier during this delay, B avoids interfering with such a transmission. To make this idea work, a station D located between A and B should use a shorter delay than B. In general, the delay used by B should depend on the number of stations separating B from the leader. It is reasonable to postulate that this delay be quantised. Thus, the immediate left neighbour of A uses 1 unit of delay,[4] the second station to the left uses 2 units, and so on. Assuming that the station identifiers reflect the ordering of stations along the bus, the delay used by B is equal to $(A - B) \times \delta$, where δ is the delay quantum.

In fact, B computes three delays: $d_1^l(A, B)$, $d_2^l(A, B)$ and d_e. The first delay is determined by the above formula. The second delay, simulates the

[4] Just to provide an inter-packet space. If packet spacing is not needed by the physical layer, the immediate left neighbour of A doesn't have to delay its transmission at all.

path of the token to the left end of the bus and then back to B. It also includes additional delays for detection of a possible transmission started by a station located to the left of B. Note that each of these stations has two chances to transmit. Finally, the third delay simulates the path of the token to the right end of the bus and back to A, including one unit of additional delay per each station occurring on that path.

Likewise, C computes the three delays $d_1^l(A,C)$, $d_2^l(A,C)$ and d_e. The values of these delays reflect the fact that all the stations located to the left of C have a higher priority (and some of them are counted twice).

If B senses a packet while waiting, it waits for the EOT, recomputes the delays with respect to the new leader and restarts the wait.

Assume that the wait period $d_l^1(A,B)$ elapses without a transmission being sensed. At this moment, station B has its *first turn*, i.e. the first chance to transmit a packet. If B has a packet awaiting transmission, it transmits it (this makes B a new leader). The packet **is not inflated**. Its P bit is copied from the A's packet; in this case it is set to 0.

If B has no packet to transmit at the moment when its first turn comes, it continues waiting, extending the delay period to $d_l^2(A,B)$. If no transmission is sensed during the second wait period, B is given a second chance to transmit a packet, if one became ready during the wait (this is B's *second turn*). If B now has a packet awaiting transmission, it transmits it (this makes B a new leader). The packet is not inflated. Its P bit is the complement of the P bit from the A's packet; in this case it is set to 1. Note that now the virtual token travels to the right.

If it happens that none of the stations located to the left of A has a packet to transmit, the stations located on the right of A are allowed to join

the party, also having two turns each.

The order in which these stations are given a chance to transmit is symmetric to the order in which the stations on the left of A were getting their turns. In particular, station C (figure 4) will piggyback its packet to the packet transmitted by A, only if no station on the left side of C is ready for transmission.

If no station has a packet to transmit, the network switches back into the uncontrolled mode of operation.

In summary, the transmission of A puts the network into the controlled mode and initiates a cycle in which all stations are given two chances (turns) to transmit their packets **without collision**. One can think of it in terms of a *virtual token* which is generated by the end of a successfully transmitted packet and makes a circular trip through all the stations. The trip starts at the transmitter and visits each station twice. The direction of the cycle traveled by the virtual token is determined at random. A packet transmitted in controlled mode may be arbitrarily short: it need not be inflated as its transmission is guaranteed to be successful.

The three delays are given by the following formulas (assuming station A is the leader and $P = 0$):

$$d_1^l(A,B) = \begin{cases} (A-B)\times\delta & \text{if } A > B \\ 2l(A) + (A+B+1)\times\delta & \text{if } A \le B \end{cases} \tag{1}$$

$$d_2^l(A,B) = \begin{cases} 2l(B) + (A+B+1)\times\delta & \text{if } A > B \\ 2(L + l(A) - l(B)) + (2N + A - B)\times\delta & \text{if } A \le B \end{cases} \tag{2}$$

$$d_e = 2L + 2N \times \delta \quad , \tag{3}$$

where N is the number of stations in the network, L is the propagation

length of the bus, and $l(S)$ is the location of S, i.e. the propagation distance of S from the left end of the cable, and δ is a small interval which we call the *delay quantum*.

The formula for $d_1^l(A,B)$ when $B < A$ was already derived. To illustrate how to derive the remaining formulas, let us calculate $d_1^l(A,B)$ when $A \leq B$. If we ignore the additional delays for yielding to "upstream" transmissions, the token arrives at B $2l(A)$ time units after B perceives the end of tha A's packet. Indeed, if we imagine that the end of this packet bounces from the left end of the bus, it will reach B $2l(A)+l(B)-l(A)$ time units since it leaves A. However, B counts its waiting time since the moment it perceives the end of the A's packet; hence, the imaginary "bounced" copy of this end will reach B $2l(A)$ time units after the (real) original. This waiting time must be augmented by the additional delay needed by B to yield to a possible transmission of a station preceding B in the virtual ring. One quantum of the additional delay must be added per each station visited by the token. Each of the stations located to the left of A (including A) is visited twice and the are $A+1$ such stations.[5] Then B has to yield to $B-A-1$ stations between B and A (excluding A). This way we get $2(A+1)+B-A-1$ which combined with $2l(A)$ gives the value of $d_1^l(A,B)$ when $A \leq B$.

The calculation of d_e is simple: the complete its cycle, the virtual token must cross the entire bus twice ($2L$ time units) and visit each station twice. Each visit at a station incurs an additional delay of δ.

The case when $P = 1$ is symmetric. The corresponding delays when the piggyback direction of the A's packet is *right*, denoted by d_1^r and d_2^r, can be obtained from the above formulas by replacing A and B (in the right hand

[5]We assume that stations are numbered from 0.

side) by $N - A$ and $N - B$, respectively. Note that d_e does not depend on the piggyback direction.

A station B piggybacking its packet (transmitting the packet in the controlled mode initiated by A), sets the P bit of the packet according to the following rules (we assume that the piggyback direction of the A's packet is *left*, i.e. 0):

$P = 0$ if $B < A$ and the packet is piggybacked in the first turn, or $B \geq A$ and the packet is piggybacked in the second turn;

$P = 1$ if $B < A$ and the packet is piggybacked in the second turn, or $B \geq A$ and the packet is piggybacked in the first turn.

The rules of setting the P bit when the piggyback direction of the A's packet is *right* can be obtained from the above rules by replacing 0 with 1 and reversing the inequalities.

Note that the piggyback direction of the packet transmitted by B corresponds to the direction of the path traveled by the virtual token (see figure 4). This way, the new cycle started by the B's packet can be viewed as a continuation of the previous one. This way the protocol is fair: as long as the controlled mode is sustained, all stations are serviced in a circular manner. In some sense, packets in PIGGYBACK ETHERNET form trains, similarly as packets in *Expressnet*. Unlike in *Expressnet*, packet trains in PIGGYBACK ETHERNET are virtual: the (intended) direction of the packet, i.e. the location of its recipient, has nothing to do with the direction of the train which is determined solely by the contents of the P bits. By the rules of the protocol, each station can transmit twice in a cycle understood as a full round trip of the virtual train.

Theoretically, the delay δ can be arbitrarily short. In practice, it depends on the tolerance of clocks at particular stations and their response time to events in the bus. The inaccuracy of clocks seems to be a much more important factor. Let us assume that the tolerance of clocks is τ. It means that a time interval of length Δ to be measured by a station's clock may come up as any interval of length between $\Delta(1-\tau)$ and $\Delta(1+\tau)$. The maximum interval to be measured by a station in PIGGYBACK ETHERNET is d_e and the difference between the measurements of this interval for any two stations must be less than δ. To be on the safe side, we should postulate that the error be not bigger than a certain fraction of δ, e.g. $\delta/2$. With this assumption, even at the biggest possible error, the station still has $\delta/2$ time units to properly interpret the bus status. The above postulate, boils down to the following inequality:

$$d_e \times (1+\tau) - d_e \times (1-\tau) \leq \frac{\delta}{2}$$

which gives:

$$(2L + 2N \times \delta) \times 2\tau \leq \frac{\delta}{2}$$

and in consequence:

$$\delta \geq \frac{8L\tau}{1-8N\tau} \quad . \tag{4}$$

As expected, δ depends on the propagation length of the bus and the number of stations in the network. One can observe that the denominator of the right side of inequality 4 must be strictly positive. In other words, the condition:

$$1 - 8N\tau > 0 \tag{5}$$

restricts the maximum number of stations in the network for a given clock tolerance τ. To get an impression how serious this restriction is, let us

assume that τ is equal to 0.0001, which corresponds to the accuracy of clocks in commercial ETHERNET[6] (cf. [27]). With this tolerance, the maximum number of stations determined by the denominator from inequality 4 must be less than 1250. Conversely: if 50 stations are distributed along a 2000 bit cable (e.g. 400 meters of a 1Gb/s carrier), the minimum value of δ is less than 2 bits.

4 Piggyback Ethernet Version 2

Again, we assume that all stations in the network are assigned addresses from 0 to $N-1$, where N is the total number of stations, in the increasing order of the station's distance from the left end of the cable. We no longer postulate that each station be aware of the exact location of all other stations on the channel.

Version 2 is quite similar to version 1; however, the concept of a *cycle* has been simplified substantially. Bit P no longer exists explicitly; its role is now played by one predetermined bit of the packet. As the contents of this bit must be more or less random, the least significant bit of the packet's CRC code is a natural candidate.

The protocol operates in two modes, similar to version 1. In the *uncontrolled mode*, it behaves precisely in the same way as PIGGYBACK ETHERNET version 1.

When a station senses a successful transmission, it switches to the *controlled mode*. In the controlled mode, some stations are allowed to append

[6]The tolerance of clocks in *Ethernet* is rather rough: the accuracy of clocks is not a serious issue in the commercial network.

their packets to other packets, in an order based on priorities. These priorities are derived from their positions with respect to the sender of that packet.

Assume that a packet has been transmitted successfully by station A and its end is sensed by B at time t_0. The contents of the P bit determine the *piggyback direction* of the A's packet. If $P = 0$ and $B < A$ or $P = 1$ and $B > A$, we say that B is *eligible* for *piggybacking* its packet to the packet transmitted by A. Only eligible stations are allowed to piggyback their packets; all the other stations have to wait until the network returns to uncontrolled mode or until another transmission changes their eligibility status.

An eligible station gets only one chance to piggyback its packet after the packet transmitted by A. The virtual token travels only one section of the bus—the segment on the piggyback direction of station A—and it does it only once. As the stations don't have to simulate the return sweep of the token (the token doesn't bounce off the bus end), they don't have to know the propagation length of the segment travelled by the token. Thus, an eligible station B is only obliged to yield to the stations located between A and B. According to the rules described in the previous section, it waits for:

$$d_p(A, B) = |A - B| \times \delta \tag{6}$$

time units, where δ plays the same part as in version 1. If the bus has been idle through this time, B is allowed to start its transmission at $t_0 + d_p(A, B)$. The protocol guarantees that the transmission of B will be successful and therefore, B does not inflate its packet.

If the channel has become busy in the meantime, it means that B has

been preempted by another eligible station, say C, located closer to A. In such case, B waits for the end of the (necessarily successful) transmission of C and uses the P bit of the C's packet to determine its further actions. The P bit being random, non-eligible stations may become eligible and vice versa.

If, after a successful transmission by A, there are no transmissions during the amount of time equal to

$$d_e(A) = 2L + \delta \times \begin{cases} (A+1) & \text{if } P = 0 \\ (N-A) & \text{otherwise} \end{cases} \tag{7}$$

all the stations return to uncontrolled mode. Note that the value of d_e corresponds to the moment when the virtual token falls off the end of the bus along the piggyback direction.

The P bit of a packet transmitted by one of the two extreme stations should be set deterministically to indicate the piggyback direction towards the populated end of the cable. This postulate is difficult to satisfy if the P bit is not explicit. Note however, that a station detecting the end of a successfully transmitted packet is able to recognise its transmitter. Thus it may ignore the contents of P if the packet happens to have been transmitted by an extreme station, and assume that the piggyback direction of such packet is always the same.

The requirements of version 2 as to the accuracy of local clocks are slightly less strict than for version 1. The maximum length of a time interval to be measured by a station (the upper bound on d_e) is equal to $2L + N\delta$. Thus, under assumptions similar to those made in the previous section we

obtain the following inequality for δ and τ:

$$\delta \geq \frac{8L\tau}{1-4N\tau} \quad . \tag{8}$$

In particular, for a given clock tolerance τ, the maximum number of stations determined by the denominator of inequality 8 is two times bigger for version 2 than for version 1.

5 Performance

In this section we compare the performance of eight protocols: the two versions of PIGGYBACK ETHERNET introduced in the previous two sections (denoted by *p1* and *p2* on the figures), *Z-Net* (denoted by *zn*), *BID* (*bi*), *L-Expressnet* (*le*), *Expressnet* (*ex*), *Fasnet* (*fs*), and ETHERNET (*et*). To make the comparison meaningful, all the protocols were implemented (in LANSF) on the basis of the same (or similar) "hardware" with parameters as the maximum packet length, the length of the packet header and trailer, etc. borrowed from commercial ETHERNET. The stations were equally spaced along the bus, the distance between two immediate neighbours being $L/(N-1)$, where N was the number of stations in the network. The experimental results presented in this section were obtained for a network consisting of 32 stations. Networks with more and fewer stations were also investigated, the results were qualitatively consistent with these presented here.

For the protocols that impose no restriction on the minimum packet length, this length was assumed to be 8 bits, i.e. unrestricted. In situations when a packet had to be inflated, its total length (including the header and trailer) was forced to be $2L + 64$ bits[7]. The length of the interval δ for

[7] The *safety margin* of 64 bits was the same as in the commercial network

PIGGYBACK ETHERNET was determined by formula 4, then incremented by 10%, and finally rounded up to the nearest integer number of bits. The accuracy of clocks was assumed to be 0.01%, i.e. the same as for commercial ETHERNET.

BID and *L-Expressnet* use delays similar to δ which, however, are independent of the cable length. Thus, the length of the corresponding delays for the two protocols was set (optimistically) to one bit[8]. Another parameter specific to *BID* and *L-Expressnet* is the length of the dummy packet sent to initialise the next round. Such a dummy packet is transmitted if the extreme station responsible for starting the round has no ready information packet to transfer. The total length of this special packet was assumed to be 96 bits. Note that a dummy packet need not have any special structure. Its exclusive purpose is to provide an EOC signal which is perceived by all stations.

Expressnet uses the concept of a packet preamble which can be destroyed partially without affecting the integrity of a packet that possibly follows the preamble. The purpose of the preamble is to initialise packet trains and to preempt downstream transmissions. Packet headers in ETHERNET already include a preamble, so no extra bits were added to the packet header in *Expressnet*.

The slot length in *Fasnet* was optimistically set up to correspond to the mean packet length as defined by the investigated traffic pattern augmented by the standard ETHERNET header and trailer. Eight extra bits were added to represent flags associated with the slot.

[8]The impact of δ on the performance of *Piggyback Ethernet*, *BID*, and *L-Expressnet* is negligible for practically any reasonable bus length.

The frame bits (i.e. packet headers, trailers, and other "special" items including dummy bits used to inflate short packets) were not included when calculating the throughput of the investigated protocols. The throughput presented in figures 6 and 7 is the so-called *effective throughput* determined as the ratio of the total number of information bits successfully transmitted and received at their destinations to the network operation time expressed in bits.

We start with comparing Version 1 of PIGGYBACK ETHERNET with the other protocols. In subsection 5.3, we compare the performance of the two versions of PIGGYBACK ETHERNET.

5.1 Maximum throughput

This section starts with the most representative of the figures: figure 5, which shows the degradation rates of the networks as $a = L/l_p$ increases (the distribution of packet length is exponential with mean l_p). On this figure only, throughput is normalised to take into account the added hardware of some networks (cf. section 2.4). Moreover, the maximum "fair and stable" throughput is plotted, that is, lengths of message queues at stations were bounded by the same constant.

BID and PIGGYBACK ETHERNET1 are the clear winners, unless a is extremely large. Note, however, that packets suffer much greater delays in *BID* than in PIGGYBACK ETHERNET1 (cf. section 5.2).

Figures 6 and 7 illustrate the degradation of the maximum effective throughput achievable by the investigated protocols for the increasing length of the bus. Traffic was uniform with exponentially distributed packet length. For the relatively short mean packet length of 512 bits, ETHERNET degrades

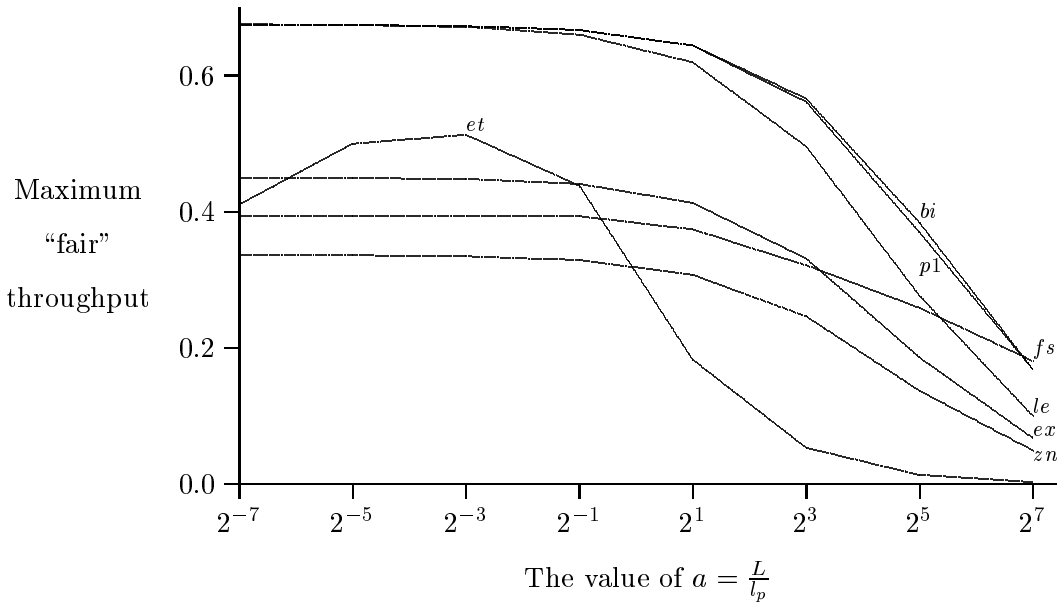

Figure 5: Throughput degradation as a function of a

more severely than other networks[9]: when a becomes substantially greater than 1, the throughput of ETHERNET falls down to 0. These results are consistent with those presented in [31]. The situation does not improve much for substantially longer packets.

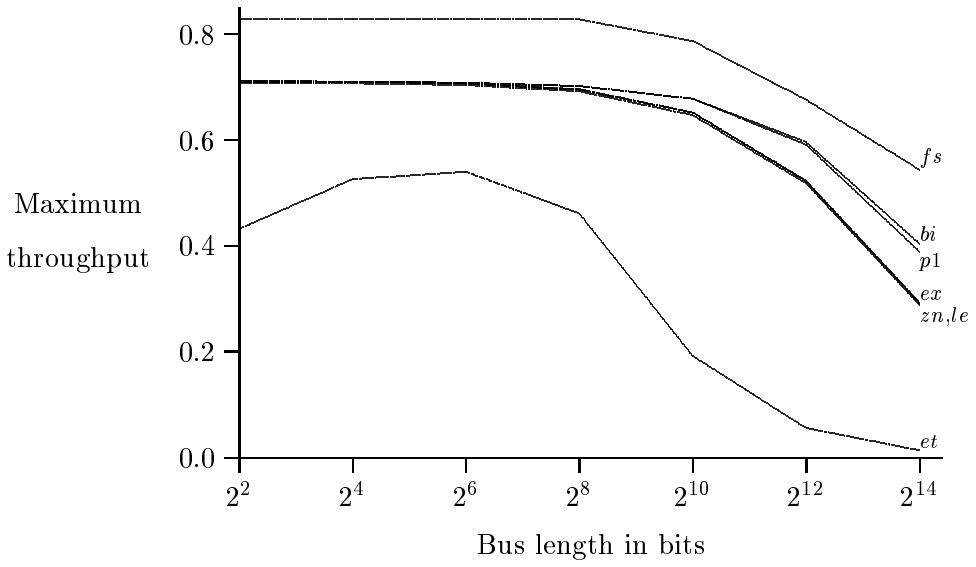

Figure 6: Maximum effective throughput for 512 bit messages.

For all the other networks, the two figures differ very little: the second (7) can be derived from the first (6) by shifting the horizontal axis by 2^4 (8192/512) and shifting the vertical axis by about 40 per cent (the frame bits occupy a smaller fraction of a packet). This observation leads to the conclusion that the figures are representative for all messages sizes in the

[9]The apparent anomaly in the network performance for a very short bus is due to the properties of the backoff function.

same range.

The maximum throughput achieved by Version 1 of PIGGYBACK ETHERNET puts it into the same class as *BID*. The spectacular performance of *Fasnet* is due to the fact that the network is equipped with two busses transmitting data independently. Moreover, the mean packet length coincides with the slot size which is also a factor favourable for the network. In fact, the throughput achieved by *Fasnet* should be divided by two[10] before it is compared to the throughput of a single-bus network. One should postulate the same about *Z-Net*, which uses the two non-symmetric busses as if they constituted a single bus with certain "special" properties. Likewise, the throughput of *Expressnet* should be divided by 1.5.

The throughput of Version 1 can be determined formally. Let l_p denote the packet length and l_h the length of the packet frame information. Let us assume that n out of N stations in the network are constantly backlogged. Thus n/N can be viewed as a measure of the traffic intensity. Imagine that a station has just completed a successful transmission. The virtual token generated by this transmission will make a circular "sweep" through the network and visit each station twice. The number of packets successfully transmitted during the sweep will be $2n$ and the time needed to complete the sweep is given by $2n \times l_p + 2L + 2N \times \delta$. The throughput of PIGGYBACK ETHERNET(Version 1), denoted by $T_{pb}(n, N)$, is equal to the number of information bits successfully transmitted divided by the time required to transmit them, i.e.:

$$T_{pb}(n, N) = \frac{n \times l_p}{n \times l_p + L + N \times \delta} \quad .$$

[10] As it is on figure 5.

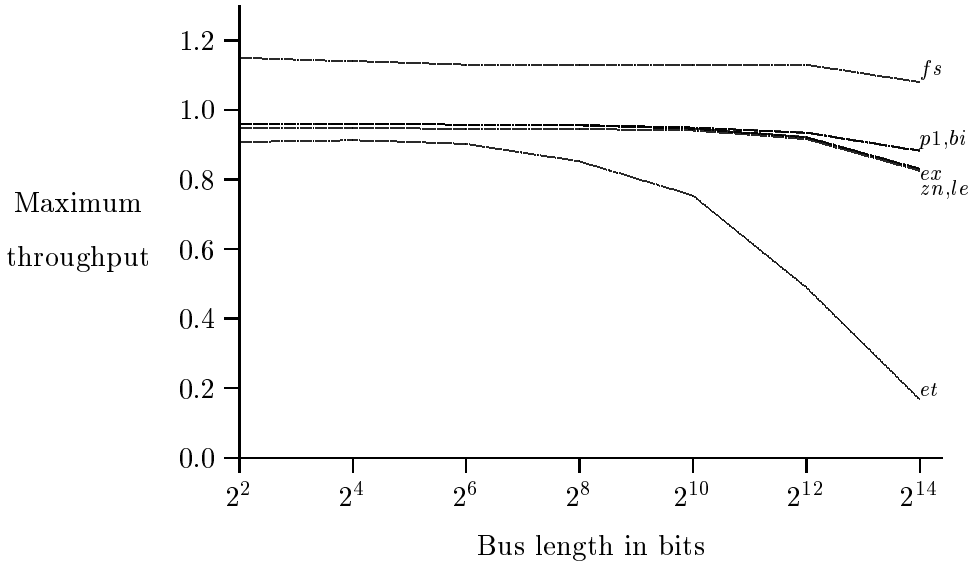

Figure 7: Maximum effective throughput for 8192 bit messages.

By rearranging slightly the above equation and substituting a for L/l_p we get:

$$T_{pb}(n,N) = \frac{1}{1 + \frac{a}{n} + \frac{N \times \delta}{n \times l_p}} \quad . \tag{9}$$

Moreover, we can eliminate δ by replacing it with the minimum value determined by inequality 4, i.e. assuming the optimum implementation of the network. This way we finally obtain:

$$T_{pb}(n,N) = \frac{1}{1 + \frac{a}{n} + \frac{8Na\tau}{n(1-8N\tau)}} \quad , \tag{10}$$

where τ is the clock tolerance.

The throughput of $Z - Net$ (cf. [18]) is given by:

$$T_z(n,N) = \frac{1}{1 + \frac{2a}{n}} \quad . \tag{11}$$

Thus we can conclude that as long as:

$$16N\tau < 1 \tag{12}$$

version 1 of PIGGYBACK ETHERNET achieves higher throughput than *Z-Net*. A comparison of inequality 12 with condition 5 shows that the proposed protocol has a higher maximum throughput than *Z-Net* if the number of stations is not more than half of the maximum permitted by 5. Note that equation 12 is equivalent to $N \times \delta < L$, which is easier for intuitive interpretation.

The throughput achieved by *BID* was reported in [7] and is practically identical to that of PIGGYBACK ETHERNET (cf. equation 9). This is clearly visible in figures 6 and 7. The difference in favour of *BID* is the shorter delay δ which is independent of the number of stations. One disadvantage of *BID* is the occasional presence of dummy packets which degrade the throughput slightly. Similarly, the maximum throughput of *L-Expressnet* (cf. [7]) is

almost identical to the throughput of *Z-Net*. From the point of view of the resulting behaviour of stations for very heavy traffic, the two protocols are essentially identical. Namely, the stations are scanned along one direction of the bus. After the last station is given a chance to transmit, the scanning starts from the opposite end.

The throughput of *Expressnet* (cf. [7]) is given as:

$$T_{ex}(n,N) = \frac{1}{1 + \frac{a(L+t)}{nL} + \frac{2\Delta}{n \times l_p}} \quad , \tag{13}$$

where t is the length of the turning segment of the bus and Δ is the length of the packet preamble (the *preemption delay*) which for a typical network should be of the order of δ. Normally, t is equal to L and therefore, the maximum throughput achieved by *Expressnet* is practically identical to that of *Z-net*, as $\Delta \ll l_p$.

The following formula describes the throughput of *Fasnet* (cf. [31]):

$$T_{fs}(n,N) = \frac{2}{\frac{S}{l_p} + \frac{2a}{n}} \quad , \tag{14}$$

where S is the slot length. Although in the ideal situation, when the message length coincides with the slot length, the maximum throughput achieved by *Fasnet* is roughly twice as high as for the other networks, the performance of the protocol may deteriorate substantially when the two lengths are incompatible. In our experiments (figures 6 and 7), we assumed that the message length is exponentially distributed with the mean value equal to the slot size.

5.2 Packet access time for light load

The main purpose of the proposed protocol is to provide fast access to the network under light traffic conditions. Formally, this performance measure can be expressed as the so-called *mean packet access time*, i.e. the expected amount of time elapsing since a packet becomes ready for transmission (appears on top of the packet queue) until the moment when the successful transmission of that packet is commenced. Note that the *mean packet access time* does not include the time spent by the packet in the message queue (under very light traffic this time is irrelevant) and its propagation time to the recipient (which depends on the bus length).

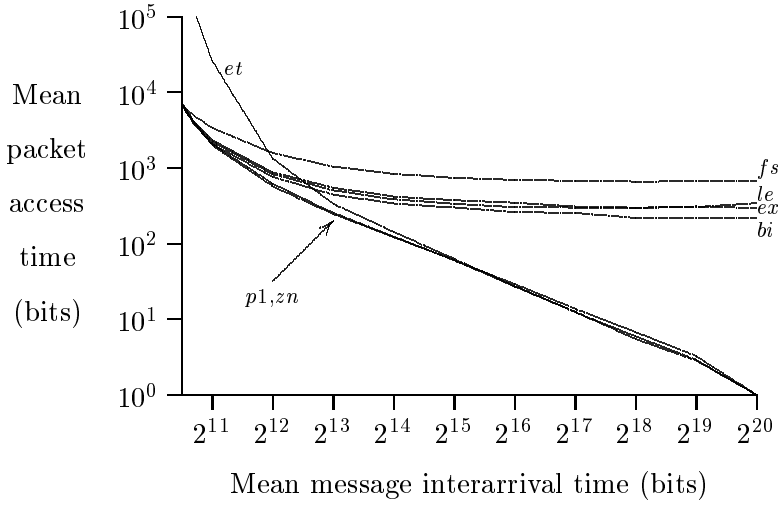

Figure 8: Mean packet access time for 256-bit bus.

When the network is silent, the proposed protocol operates without any synchronisation overhead and a packet arriving to a station is transmitted

immediately. Thus, the mean packet access time converges to zero with the increasing mean message interarrival time. The same property holds for *Z-Net* and *Ethernet*. All the remaining protocols investigated in this paper incur a certain overhead, even if the traffic is very low. For example, in *BID*, the expected access time of a single packet in the absence of other contenders is about $L/2$. For *L-Expressnet* and for *Expressnet*, the mean packet access time is of the order of L. In the case of *Fasnet*, the situation is somewhat better, as the average waiting time for a ready packet is equal to half of the slot length, i.e. $S/2$.

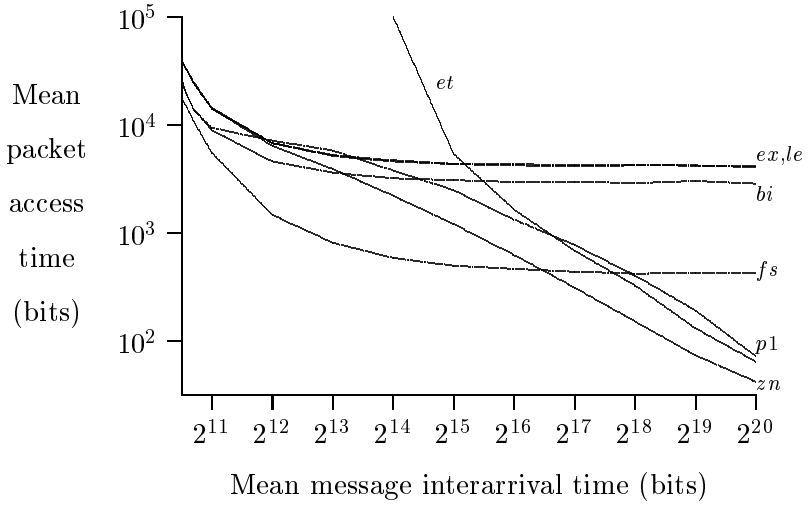

Figure 9: Mean packet access time for 4096-bit bus.

Figures 8 and 9 illustrate these phenomena quite well. They were produced for uniform traffic with 1024 bit mean packet length (exponentially distributed). For a long bus, in the medium range of traffic conditions,

Version 1 of PIGGYBACK ETHERNET performs worse than *Z-Net*. This difference disappears for a shorter bus. (On a long bus the cost of a collision for PIGGYBACK ETHERNET is high.)

Note that the relative performance of *Fasnet* is better for the 4096-bit bus than for the shorter one, as the impact of the slot length is less visible when the bus is longer.

To determine whether the good observed performance of our protocol is retained when the traffic pattern is non-uniform, we carried out a number of experiments involving biased traffic. According to this pattern, 95% of all traffic in the network was either originated at or addressed to (with equal probability) one station located close to the middle of the bus. The remaining parameters of the traffic were the same as before, i.e. the message length was exponentially distributed with the mean of 1024 bits.

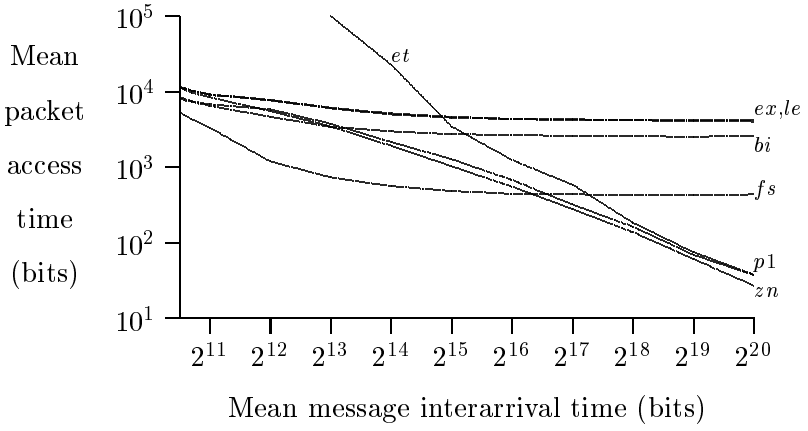

Figure 10: Mean packet access time for biased traffic (4096-bit bus).

Figure 10 compares the mean packet access times of the seven protocols

for biased traffic and 4096-bit bus. Again, the performance of our protocol is only marginally worse than the performance of *Z-Net*. The observed behaviour of the networks is very similar to that presented in figure 9.

5.3 Comparison of the two versions of Piggyback Ethernet

One could expect the performance of the Version 2 of PIGGYBACK ETHERNET is significantly worse than the performance of Version 1. As it turns out, the difference is quite small. Figure 11 compares the maximum throughputs achieved by the two versions for 512 and 8192-bit messages and uniform traffic pattern.

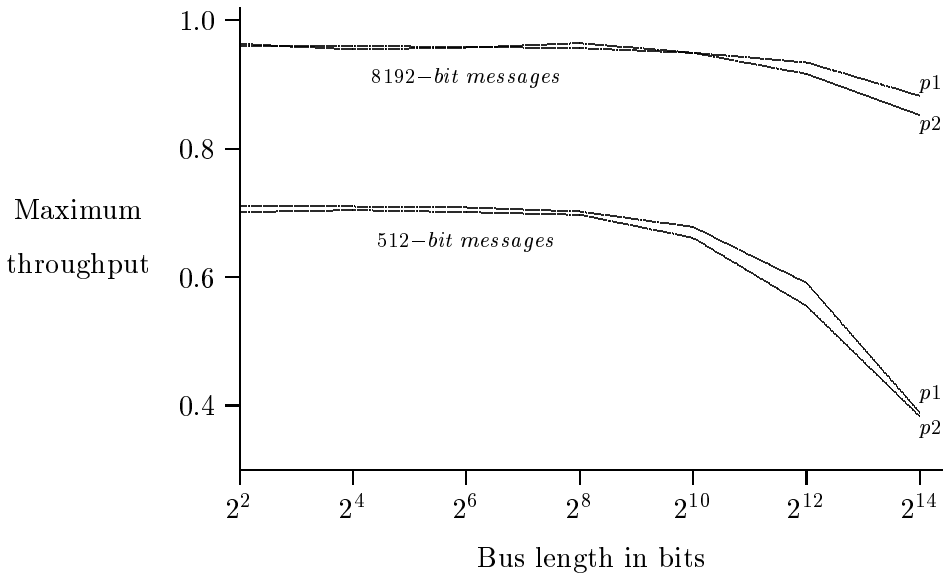

Figure 11: Maximum throughput achieved by Version 1 and Version 2.

In terms of the average packet access time, the the two versions are also very close. Figure 12 shows the difference in the performance of Version 1 and Version 2 as far as the packet access time is concerned. The traffic pattern used was uniform with the message length exponentially distributed around a mean of 1024 bits.

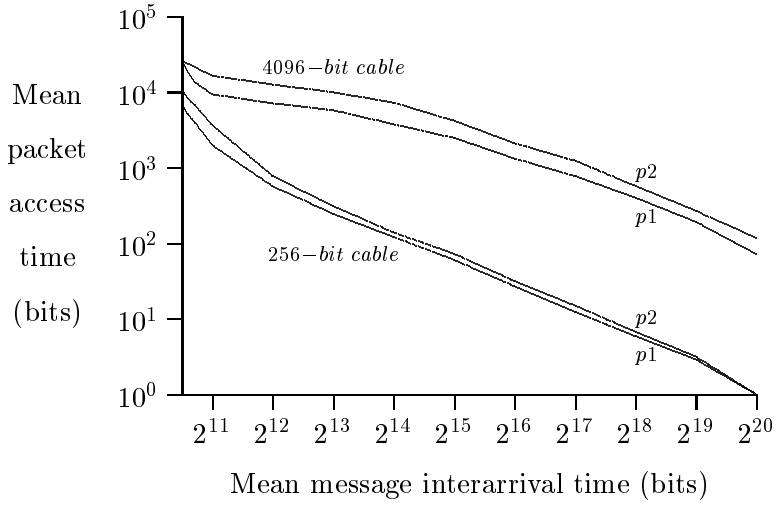

Figure 12: Mean packet access time for Version 1 and Version 2.

One problem with Version 2 is its *fairness*. Note that in Version 1, once a round is started, the network remains in the synchronised mode of operation as long as there are stations ready to transmit packets. In this mode, the medium access is controlled by a token that rotates among the stations. Thus there is no starvation and each station gets access to the bus after a bounded time. Within each complete turn of the virtual token, each station gets the token twice. The interval between two consecutive token visits within the same cycle is shorter for a station located close to the end of

the bus than for a station in the middle. The average interval between two consecutive visits of the token is, clearly, the same for all stations. The way the virtual token sweeps through the stations is directly related to a method of fulfilling disk requests called SCAN. This method has been extensively investigated by Denning ([3]) and Teorey and Pinkerton ([29]). Although, based on the results of these investigations, one could expect that the delays measured at stations located closer to the ends of the bus will be slightly higher than for stations situated near the middle, no visible effect of this kind was observed.

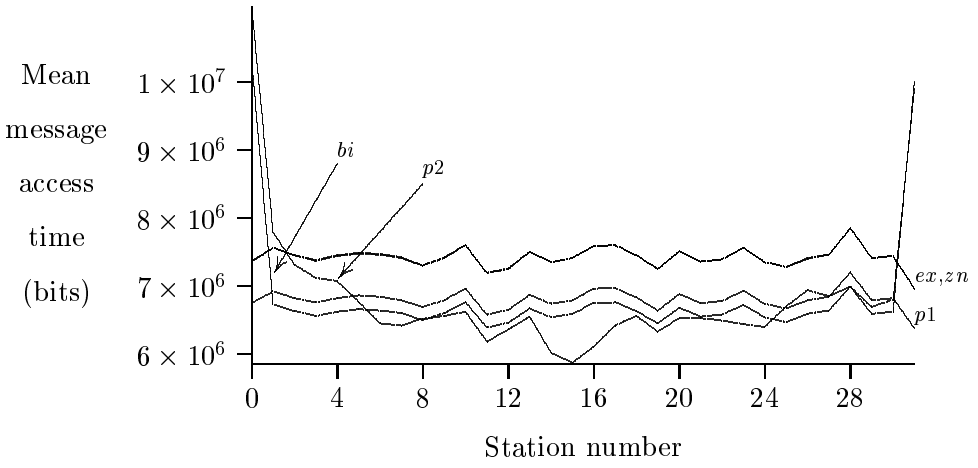

Figure 13: Mean message access time measured at different stations.

In the case of Version 2, the situation is more complicated. The cycle triggered by a successful transmission does not necessarily give every station a chance to send a packet without a collision. A certain subset of stations

is only favoured; if none of those stations is ready to transmit a packet, the cycle ends and the synchronised mode of operation is exited[11]. Similarly as commercial ETHERNET, Version 2 of the proposed protocol exhibits statistical behaviour and it is no longer clear that all stations get the same share of the channel bandwidth.

To investigate this problem, we have conducted an experiment in which we measured the average message access time at individual stations for a traffic pattern consisting of a single huge burst of 200000 messages arriving all at the same time. The allocation of messages to stations was uniform. The message length was exponentially distributed with the mean of 1024 bits. After the arrival of the burst, the traffic generator was switched off and the simulation continued until all messages were successfully received at their destinations.

By the *message access time* we understand the amount of time elapsing since a message is generated and queued for transmission (the beginning of simulation in this case) to the moment when the successful transmission of the last packet of the message is started. Note that in the experiment described above, almost all messages were transmitted as single packets.

Figure 13 illustrates the results for five protocols. In Version 2 and *BID*, stations located closer to the ends of the bus suffer slightly higher delays than those nearer the center. *Expressnet* and *Z-Net* do not exhibit this

[11] One could think of a simple improvement: an extreme station detecting an "unused" cycle with the piggyback direction towards the station's end of the bus may transmit a dummy packet and sustain the synchronised mode of the network. Note that even a failure of an extreme station would not "crash" the protocol, but merely slightly worsen its performance. However, this modification does not eliminate the inherently statistical nature of the protocol.

behaviour: in these protocols stations get their turns in one-way sweeps, which method guarantees absolute fairness (cf. [29]).

6 Configurability

The implementation of the proposed protocol is based on the assumption that all stations know their ordering along the bus. In Version 1, it is additionally assumed that every station has (almost) exact knowledge of the distances of all stations from one end of the carrier. The automated acquisition and maintenance of this information is a necessary prerequisite for a realistic implementation of the protocol.

We start with a sketch of an algorithm that orders the stations according to their occurrence along the bus. A family of initialisation algorithms for bus networks was presented in our paper [6].

1. All stations play a tournament consisting of N games. The goal of a game is to select one station to be removed from the tournament such that all other stations will know whether they are located to the left or to the right of the removed station. The removed station does not participate in subsequent games.

2. Upon detecting the initialisation signal, all stations start broadcasting packets. Possibly after a number of collisions, one of them will succeed. Let us denote this station by A (see figure 14).

3. Having transmitted successfully, A defers its further activities until it senses the end of the second successful transmission. All stations other than A continue their transfer attempts until one of them succeeds. Let us denote that second station by B.

4. Having recognised the end of the B's transmission, A immediately follows it by its own packet. All other stations wait until they hear: first, the end of the packet transmitted by B, and next, the beginning of the A's transfer.

5. Every station detecting that the two packets are (almost) adjacent, assumes that it is located on the *left* side of A. If, as perceived by a station, the two packets are separated by a clearly recognizable period of silence, the station assumes that it lies on the *right* side of A. Thus, after the first step all stations are able to tell their relative locations with respect to A. Station A is called the *reference station* of the current game of the tournament. The reference station transmits one more packet (see below) and then it is removed from the tournament—it will not contend in subsequent games. However, as long as the tournament is still in progress, all removed stations passively monitor the bus, to learn about their locations with respect to subsequent reference stations.

6. All stations that appeared to be located on the right side of the last reference station wait for $2L$ while the other (non-removed) stations contend to select a winner—the first station that broadcasts its packet successfully. If the contention does not start within the nearest $2L$ time units, the right-side stations will take over. Let us assume that there is a left-side station, say C, that wins.

7. Having detected the end of the C's packet, A responds with its own packet. Then C immediately appends another packet to the end of the packet transmitted by A. This way C becomes the new reference

station and the algorithm continues at 5 with station C replacing A. A station that sees the packet sent by A immediately followed by the C's packet is located to the left of C, and if the packets were separated by a recognizable gap, then the station is to the right of C.

8. If there is no station on the left side of the reference station (other than possibly some removed stations), the stations on the right side detect that fact after $2L$ time units of silence following the last packet sent by the reference station. Then, they will start their contention to determine a winner. That winner becomes the next reference station. Note that the "orientation" of this new reference station will be different from the previous one. However, all stations monitoring the bus will recognise that the orientation has changed and they will reverse their notion of sides (step 5).

9. The algorithm continues until there are no contending stations on any side of the last reference station. This is recognised by all stations when they sense a $4L$ silence period following the end of the packet transmitted by the last reference station.

The obvious invariant of the algorithm is that every station knows its relative location with respect to all stations that have been removed so far. With each turn (game) of the algorithm one new station becomes removed. Thus, the algorithm terminates and then all stations are removed. In each game there are 3 successful transmissions, except the first that requires only 2. When the algorithm stops (after $3N - 1$ packets have been successfully transmitted), each station is able to tell its position on the bus.

The above algorithm suffices for PIGGYBACK ETHERNET Version 2. For

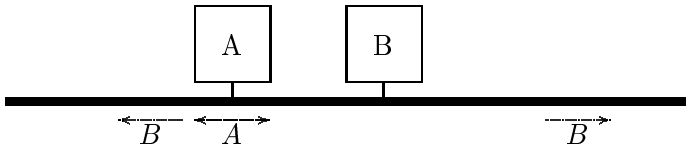

Figure 14: The first step of the configuration algorithm.

PIGGYBACK ETHERNET Version 1, the distances are also needed. After finishing the above algorithm, the leftmost station takes the role of the leader and exchanges packets with each station on the bus. By comparing the time of sending a packet to a station and the time of obtaining the answer, the leader can tell the distance to the station. This requires to send $2(N-1)$ messages. Finally, the leader broadcasts its knowledge (i.e. the distances from the leader to any other station) and each station can compute its distances to other stations.

Note that the values produced by expressions 1, 2 and 3 (expressions 6 and 7 for Version 2) can (and perhaps should) be precomputed and stored in a lookup table. This way, stations will not have to evaluate these expressions each time they hear the end of a successful transmission. The size of the lookup table at each station will be $4N-3$ for Version 1 and N for Version 2.

7 Summary

We have presented two versions of a collision protocol applicable for fast bus-type networks based on single, uniform, unsegmented, broadcast-type carriers. The protocol assumes that each station possesses certain global knowledge about the geometry of the network. Other protocols the perform

equally well as the proposed ones require a similar or bigger amount of knowledge about the geometry.

The performance of the proposed protocols for very heavy traffic is not worse than the performance of the protocols operating in a completely synchronised manner. The advantage of our solution is in a good performance for light load which is practically not worse than the performance of ETHERNET. Moreover, compared to other protocols for fast networks, PIGGYBACK ETHERNET has quite modest hardware requirements. The network is based on a straightforward, bidirectional cable, and each station has only one connection to the bus.

References

[1] F. Borgonovo, L. Fratta, F. Tarini, and P. Zini. L-Express-net: A communication protocol for local area networks. In *Proceedings of IEEE INFOCOM'83*, San Diego, CA, Apr. 1983.

[2] L. Chlamtac, W. Franta, and D. Levin. BRAM: The broadcast recognizing access method. *IEEE Transactions on Communications*, 27:1183–1190, Aug. 1979.

[3] P. Denning. Effects of scheduling on file memory operations. In *Proceedings of the AFIPS Spring Joint Computer Conference*, pages 9–21, 1967.

[4] W. Dobosiewicz and P. Gburzyński. Ethernet with segmented carrier. In *Proceedings of IEEE Computer Networking Symposium*, pages 72–78, Washington, DC, Apr. 1988.

[5] W. Dobosiewicz and P. Gburzyński. Improving fairness in CSMA/CD networks. In *Proceedings of IEEE SICON'89*, Singapore, July 1989.

[6] W. Dobosiewicz, P. Gburzyński, and P. Rudnicki. An Ethernet-like CSMA/CD protocol for high speed bus LANs. In *Proceedings of IEEE INFOCOM'90*, pages 238–245, 1990.

[7] M. Fine and F. Tobagi. Demand assignment multiple access schemes in broadcast bus local area networks. *IEEE Transactions on Computers*, 33(12):1130–1159, Dec. 1984.

[8] L. Fratta, F. Borgonovo, and F. Tobagi. The Express-net: A local area communication network integrating voice and data. In *Proceedings of International Conference on Performance of Data Communication Systems and Applications*, Paris, France, Sept. 1981.

[9] P. Gburzyński and P. Rudnicki. A better-than-Token protocol with bounded packet delay time for Ethernet-type LAN's. In *Proc. of Symposium on the Simulation of Computer Networks*, pages 110–117, Colorado Springs, Co., Aug. 1987. IEEE.

[10] P. Gburzyński and P. Rudnicki. Bounded packet delay protocols for CSMA/CD bus: Modeling in LANSF. In *Proceedings of the 19th Annual Pittsburgh Conference on Modeling and Simulation*, volume 19, pages 1185–1194. Instrumentation Society of America, 1988.

[11] P. Gburzyński and P. Rudnicki. LANSF: a protocol modelling environment and its implementation. *Software Practice and Experience*, 21(1):51–76, Jan. 1991.

[12] M. Gerla, P. Rodrigues, and C. Yeh. U-net: A unidirectional fiber bus network. In *Proceedings of FOC/LAN '84*, pages 295–299, Las Vegas, NV, Sept. 1984.

[13] M. Gerla, G. Wang, and P. Rodrigues. Buzz-net: A hybrid token/random access LAN. *IEEE Journal on Selected Areas in Communications*, 5:977–988, July 1987.

[14] Y. Gold and W. Franta. An efficient scheduling function for distributed multiplexing of a communication bus shared by a large number of users. In *Proceedings of International Conference on Communications*, Philadelphia, PA, June 1982.

[15] Y. Gold and W. Franta. An efficient collision-free protocol for prioritized access control of cable or radio channel. *Computer Networks*, 7:83–89, 1983.

[16] P. Gopal and J. Wong. Analysis of a hybrid Token-CSMA/CD protocol for bus networks. *Computer Networks and ISDN Systems*, 9:131–141, 1985.

[17] E. Jensen, M. Tokoro, and L. Sha. Bus allocation scheme for distributed real-time systems. Technical report, Carnegie-Mellon University, Pittsburgh, PA, Dec. 1980.

[18] A. Kamal and B. Abeysundara. Z-NET: A dual bus fiber-optic LAN using active and passive switches. In *Proceedings of IEEE INFOCOM'89*, 1989.

[19] W. Kiesel and P. Küehn. A new CSMA/CD protocol for local area networks with dynamic priorities and low collision probability. *IEEE Journal on Selected Areas in Communications*, 1(5):869–876, 1983.

[20] L. Kleinrock and H. Levy. On the behavior of a very fast bidirectional bus network. In *Proceedings of the 1987 IEEE International Communications Conference*, pages 1419–1426, Seattle, Washington, June 1987.

[21] J. Limb and C. Flores. Description of Fasnet, A unidirectional local area communications network. *Bell Systems Technical Journal*, Sept. 1982.

[22] R. Metcalfe and D. Boggs. Ethernet: Distributed packet switching for local computer networks. *Communications of the ACM*, 19(7):395–404, July 1976.

[23] M. Molloy. Collision resolution on the CSMA/CD bus. *Computer Networks and ISDN Systems*, 9:209–214, 1985.

[24] J. Moura et al. Simulation of collision control algorithms in Ethernets. In *Proceedings of the 6th Data Communication Symposium*, Asilomar Conference Grounds, Pacific Grove, Ca., 1979.

[25] P. Rodrigues, L. Fratta, and M. Gerla. Tokenless protocols for fiber optics local area networks. In *Proceedings of the International Communications Conference*, volume 3, pages 1150–1153, 1984.

[26] J. Shoch et al. Evolution of the Ethernet local computer network. *IEEE Computer*, 15(8):10–26, Aug. 1982.

[27] W. Stallings. *Local Network Technology*. IEEE Computer Society Press, 1985. second edition.

[28] W. Stallings. A tutorial on the IEEE 802 Local Network Standard. In R. Pickholtz, editor, *Local Area & Multiple Access Networks*, pages 1–30. Computer Science Press, 1986.

[29] T. Teorey and T. Pinkerton. A comparative analysis of disk scheduling policies. *Communications of the ACM*, 15(3):177–184, Mar. 1972.

[30] F. Tobagi, F. Borgonovo, and L. Fratta. Express-net: A high-performance integrated-services local area network. *IEEE Journal on Selected Areas in Communication*, 1(5):898–913, Nov. 1983.

[31] F. Tobagi and M. Fine. Performance of unidirectional broadcast local area networks: Expressnet and Fasnet. *IEEE Journal on Selected Areas in Communications*, 1:913–926, Nov. 1983.

[32] C. Tseng and B. Chen. D-net: A new scheme for high data rate optical local area networks. *IEEE Journal on Selected Areas in Communications*, 1(3):493–499, Apr. 1983.

[33] M. Ulug, G. White, and W. Adams. Bidirectional token flow system. In *Proceedings of the 7th Data Communication Symposium*, Mexico City, Mexico, Oct. 1981.