ELSEVIER

Computer Communications 20 (1997) 449–461

computer communications

The spiral ring

Włodek Dobosiewicz[a], Paweł Gburzyński[b]

[a]Department of Computer Science, Monmouth University, West Long Branch, NJ 07764-1898, USA
[b]Department of Computing Science, University of Alberta, Edmonton, AB, Canada T6G 2H1

Received 2 September 1994; accepted 8 February 1996

Abstract

We present a new topology for high-speed ring networks: the spiral ring. We also describe a MAC-level protocol, called Distributed Spiral Multiple Access (DSMA), suitable for the spiral ring topology. The DSMA protocol is based on token passing and gives a fair and efficient network behaviour regardless of the transmission rate and network size. Under light load, the medium access delay approaches zero. When the load is heavy, the medium access delay is bounded, guaranteeing that starvation will not occur. The proposed protocol is able to handle synchronous traffic of varying intensity. Unlike other network protocols, it does not require any bandwidth pre-allocation for the synchronous part of the traffic. Another property of DSMA protocols is a provision for a graceful dynamic reconfiguration (station insertion/deletion) and costless protocol recovery after a lost token. © 1997 Elsevier Science B.V.

Keywords: High-bandwidth networks; Media access control protocols

1. Introduction

The aim of this paper is to introduce a network topology and its companion MAC-layer protocol. The proposed network falls into the same category as FDDI [1], DQDB [2], METARING [3], and the many other shared-medium networks. On the other hand, it is not intended as a competitor to ATM [4], but perhaps as a complementing private MAN or even LAN network.

The topology was designed with the following goals in mind:

1. full scalability, both in terms of transmission rates as well as of the length of the network;
2. negligible message queuing delay when the load is light;
3. reliable network management;
4. a dynamic service for synchronous traffic; this service should not require reserving any part of the total bandwidth for synchronous traffic, nor should it require any control activities, such as connection setup, renegotiation of network parameters, etc.;
5. fairness to all the stations connected to the network.

Additionally, we consider *multicast* and *no packet loss* in the Data Link Layer as essential properties of any shared-medium network.

Our MAC-layer protocol uses a token for synchronisation, but allows stations to transmit also when not in possession of the token. This makes its performance superior to FDDI's in two respects: the queuing delay is much less and the maximum throughput is higher— both these are caused by not having to wait for the token.

The main role of the token in our protocol is to provide guaranteed (but not reserved) bandwidth for each station. This, in turn, gives stations the right to set up sessions of synchronous traffic. These sessions are set up and terminated without any interference from other stations. The resulting synchronous service is superior to that of FDDI, DQDB, METARING, and even ATM, since it leaves total control to each station and does not require any negotiations involving other stations, nor does it require the station to commit itself to a certain maximum rate (or risking packet loss).

A substantial amount of literature on ring networks exists (see [5]), including more recent work in [6] and [7]. Much less was written on multiple ring networks (e.g. [8,9]).

There is no related work on the topology chosen for Distributed Spiral Multiple Access (DSMA). On the other hand, numerous works have been published that include some form of token-passing mechanism for the purpose of network synchronisation, e.g. protocols using a token to organise traffic in cycles, such as BUZZNET [10], FASNET [11], SIMPLE [12], s++ [13], etc. To our knowledge, none of them is similar to the DSMA protocol.

PII S0140-3664(97)00044-3

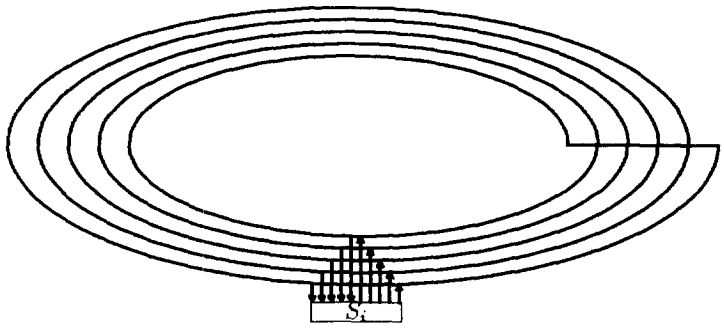

Fig. 1. A five-loop spiral.

2. The spiral ring

Fibre-optic cables are typically laid in batches containing many (e.g. 16) independent cables. Thus, it is natural to consider using more than one cable as the physical basis for a high-speed network. Practically any MAC-layer protocol can be adapted to handle several physical cables in parallel by separating the stream of submitted messages into several independent streams, one for each physical cable. We propose a network topology that handles any number of cables without such a separation.

The spiral ring is a fibre-optic cable looping in the form of a spiral. The ends of the spiral are connected and the fibre constitutes a closed, multiple, unidirectional loop which visits each station a number of times. The number of segments (circles) in the spiral is a configuration parameter. The protocol discussed in this paper can operate on a single-loop spiral (which is simply a regular ring network) retaining most of its advantageous properties. Fig. 1 shows the layout of a five-loop spiral. Only one station is shown for clarity; all the other stations have identical connections.

In a K-loop spiral network, each station has a tap connecting it to every loop of the spiral; this tap consists of an input connection and an output connection. The input and output connections forming a tap are adjacent to each other, with the input connection preceding the output connection. We will denote the tap connecting station S to loop l by S_l.

The order in which stations are connected to each loop is the same; thus, for every station, the path between its connection to loop l and its connection to loop $l+1$ includes one tap of each remaining station. A station should expect packets addressed to it on every tap; therefore, the number of receivers per station must be equal to K, each of them permanently attached to a different tap. There is one transmitter per station. This transmitter can be switched (by the protocol) among the multiple taps. Thus, a single station can only transmit one packet at a time (although it can receive multiple packets simultaneously), which automatically preserves the packet ordering at the receivers.[1] Also, each station has one receive buffer area common to all the input connections of the stations.

A single-loop spiral ring looks like a regular ring network and is similar to the topology used in a single-ring FDDI. The Pretzel ring [14,15] is an example of a spiral ring with two loops.

There are three possible packet formats for DSMA (and most other protocols), which imply three possible versions of the protocol.

- *Slotted version.* This version yields a better bandwidth utilisation (combined, however, with a greater header/payload overhead). It assumes a network filled with slots; the slot format can be borrowed from ATM or DQDB.
- *(Unslotted) version*, which seems to have limited merits for any configuration other than a two-loop spiral. The performance of a protocol similar to an unslotted DSMA, albeit inferior (the Pretzel ring), is presented in [15].
- *Semi-slotted version.* The semi-slotted variant can be viewed as a compromise between the other two versions: no explicit slot markers are inserted into the ring, but the packet length is fixed. The rationale for this compromise comes from the "packet-reuse" feature of DSMA.In this paper, we only consider the slotted version, although the other two versions can easily be envisioned. Since we focus on the slotted version, all the network events occur at slot boundaries; thus, we measure time in slots.

In discussing our proposed protocol we use the terminology of FDDI. The protocol can be viewed as a refinement of FDDI towards very high transmission rates and, in fact, it was inspired by the FDDI protocol. It is a generalisation of our previous work on overcoming the limitations of FDDI [15]. It should be emphasised, however, that the DSMA protocol is not really meant to compete with FDDI. Unlike in FDDI, the performance of our protocol does not deteriorate with the increasing ratio of the channel length (expressed in bits) to the packet length.[2] Moreover, in contrast to FDDI, the DSMA protocol incurs zero access delay when the traffic is low. Therefore, even without any

[1] If the preservation of packet ordering is not necessary, multiple transmitters per station can be used. From the description of the protocol in the following sections it is obvious how to generalise it onto multiple transmitters.

[2] Protocols with this property are called *capacity-1* protocols.

explicit comparisons, it is clear that it outperforms FDDI for a sufficiently long (or fast) network.

For the purpose of making the operation of the DSMA protocol efficient, we impose a number of hardware properties on the network.

- The stations' taps are constructed in such a way that they introduce a constant delay. The protocol does not require buffering more than the first bit of an incoming slot.
- When measured in slots, the propagation delay (latency) between two taps of the same station is a fixed integer: all that is needed is a count of complete slots passing through a station. We denote this propagation delay by L; it is measured as follows: at time τ_0, station S sends slot σ from its tap S_0. This slot will reach tap S_1 at time τ_1. The number of complete slots seen by S_1 in the time interval $\langle \tau_0, \tau_1 \rangle$ is equal to L. The true latency, measured in time units, varies with some small and known tolerance and the actual loop latency is a value in the interval $\langle L, L+\varepsilon \rangle$. Note that we never require that the values of L and ε be known to any station.
- Each tap of every station is capable of operating independently, possibly in parallel with other taps of the same station. However, we rule out the possibility of simultaneous transmission from more than one tap of a station (as stated above).
- A station is able to seize an empty slot *on the fly*, in the same manner as in all other slotted networks. e.g. as in DQDB. The slot header contains a *full* bit. Each station willing to transmit sets to 1 the full bit of every incoming slot on the fly and checks its previous value; if it was 0, the slot is considered seized.
- There is a token travelling along the spiral. A natural way to implement the token is to use a reserved bit in the slot header. Thus, the total number of special bits in the segment header required by the protocol is two. The *token* bit is initially set to 1 when the slot is generated, which implies that the slot does not carry a token.In the sequel, we denote by n the number of stations in the network. The distances between pairs of stations can be arbitrary and are assumed to be unknown; however, we postulate that a number between 0 and $n-1$ is assigned to each station in the order of their positions in the loops.

3. DSMA

DSMA is a token-passing protocol for the spiral topology. It is a slotted protocol designed as an inexpensive MAC protocol for gigabit networks.

Two methods of emptying slots are used simultaneously: *dynamically disconnecting the spiral* at one station and *source reuse*. Instead of source reuse, *destination reuse* (e.g. [3]) is also possible; although it would double the maximum throughput of the network, it would also increase the network's cost and reduce its robustness (note, however, [16,17]). Similarly, throughput enhancement through the slot reuse mechanism presented in [18] could be achieved, again requiring more costly laps and decreasing the predictability of the network.

Therefore, DSMA needs three sets of rules:

- to determine how the spiral is logically disconnected;
- to determine how stations recognise they should empty (and possibly reuse) slots they filled;
- to determine when a station is allowed to transmit and—since there are several possible taps—from which tap.

3.1. Disconnecting the spiral

The spiral is disconnected logically at a station; all the stations take turns in fulfilling this role. A token-passing mechanism similar to FDDI's is used to single out the station that disconnects the spiral and the tap at which the spiral is disconnected. There is one token which is passed according to the following rules:

- Each receiver (one per tap) of every station sets to 1 the *token* bit of each incoming slot (similarly to the *full* bit, if it has a packet to transmit). If it finds out that the previous value of the *token* bit was 0, the station assumes the role of the token-holding station when it detects the end of the slot.
- While holding the token, the station disconnects the spiral at the tap where the token was received by not forwarding the next incoming slot (the slot is thus removed from the network). In essence, this makes the spiral temporarily equivalent to a bus.
- The station holding the token is responsible for generating a new slot in the place of the one it removed (inserting it into the network). This new slot can immediately be filled by the station; the full/empty status of a slot is indicated by the *full* bit in the slot header. The station passes the token in the newly generated slot by setting its *token* bit to 0 and reconnects the loop.[3]
- Priorities may be implemented by allowing certain stations to hold the token for several slots—but always a *fixed* number.[4] Unlike in FDDI, stations do not release the token early if they have nothing to transmit; they must hold it regardless.

The rules presented above can be illustrated by an example shown in Fig. 2, which shows a fragment of one of the loops of a spiral network.

At time t_0, station A, the token-holding station, generates a new slot; this slot has its *token* bit set to 0. At time t_2, the

[3] Instead of generating a new slot, a token-holding station can simply modify the *full* bit of the incoming slot, and no slot generation is further needed. This way, the network can be filled with slots during the initialisation phase. This approach is logically identical to the one described in the text.

[4] This protocol can easily be modified to handle such priorities; we omit further mention of it for brevity.

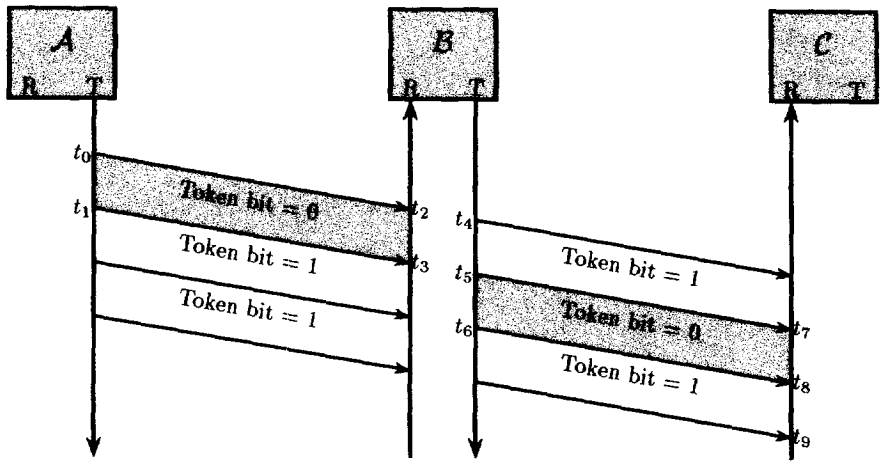

Fig. 2. Example of token movement.

next station downstream, station B, receives the first bit of that slot. Subsequently B performs the following:

- It sets the *token* bit in the slot to 1, while at the same time recording its old value.
- If B has a packet to transmit, it sets the *full* bit to 1, recording its old value.
- It inspects the old value of the *full* bit and possibly fills the slot with a new payload.
- It retransmits the slot (due to internal delay, retransmission starts at time t_4).
- At time t_3, B receives the last bit of the slot. At this point, it inspects the old value of the *token* bit; if it was 0, B assumes the role of the token-holding station at time t_5.
- Having finished relaying the slot at time t_5, B generates a new slot (with its *token* bit set to 0) and starts transmitting it. The slot reaching the receiver tap of B is read, but not forwarded. Thus, A is the token-holding station between t_0 and t_1, B holds the token between t_5 and t_6, and C holds it between t_8 and t_9.

Since each station holds the token for one slot, the time that elapses between two consecutive token captures by the same station is constant and the same for all stations. Preserving the terminology of FDDI, we will denote this time by TTRT which, in our case, stands for Total Token Rotation Time. The value of TTRT is given by the formula:

$$TTRT = L + n$$

where L is the propagation delay in slots and n is the number of stations. Since the network is slotted, the value of TTRT is an integer.

Each station has a counter called the Token Rotation Counter (TRC) which counts the slots that left the station since the moment the station last acquired the token. TRC is not used for synchronous traffic (in fact, no explicit notion of synchronous traffic is needed in DSMA), but as a pointer to estimate the token position. Whenever a station acquires the token, it resets this counter to 0. The token arrives at a station exactly when its TRC equals TTRT. This observation also leads to an extremely simple recovery procedure in the case of a lost token: if the token is lost, the first station that expects it, but does not receive it on time, simply generates another token, claims it and informs the remaining stations about the incident.

The role of the token is different in DSMA and in FDDI: in FDDI it is used to inform about the bandwidth left available to the station receiving it, while in DSMA, the token is used solely for synchronising long-term clocks.[5]

3.2. Source reuse

A full slot that makes a complete circle through the network is either emptied or reused by its transmitter. The timing for this operation is based exclusively on counting slots and poses no implementation problems. Although at first sight the operation of emptying a slot (reversing the contents of the *full* bit from 1 to 0) seems more tricky than the operation of reserving a slot (changing 0 to 1), it is in fact simpler: a station that filled a slot at time t knows that the slot will reach the next tap of the station at time $t + TTRT$ (measured in slots). Thus, it can reset the *full* bit of the slot without checking its previous contents.

3.3. Transmission rules

At any given time, one tap of each station is logically differentiated from all the others: we will call it the "yellow" tap while all the other, taps will be called "green".[6] The "colour" of each tap changes dynamically with the movement of the token: when a station releases the

[5] In both FDDI and DSMA, the arrival of a token signals to the station that it should disconnect the medium. In DSMA, however, the station also makes this decision based on the value of its TRC, when an expected token does not materialise.

[6] Using the traffic-light signalling colours, "green" means the right to proceed (i.e. transmit) unconditionally, while "yellow" gives the right to proceed only if a certain condition is met.

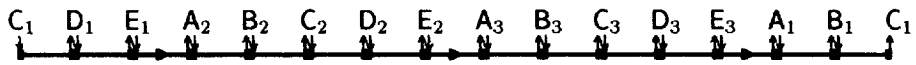

Fig. 3. A three-loop spiral.

token, it labels the tap that held it as yellow; all the other taps are labelled green.

The rules for transmitting are as follows:

1. The token-holding station is responsible for generating one new slot. It may fill this slot with a packet.
2. Stations willing to transmit continuously attempt to seize slots reaching their input taps. Possession of the token is not relevant. Note, however, that the station holding the token is also the first station to *see* the new slots inserted into the network. Thus it has a temporary priority in using the slots over all the other stations.
3. Whenever a station willing to transmit a packet seizes an empty slot in a green tap, it fills the slot header, and then fills the payload with the packet.
4. Whenever a station willing to transmit cannot seize an empty slot in its green taps, it checks whether the condition $TBC \geq n-1$ holds;[7] if it does, the station attemps to seize an empty slot reaching its yellow tap. When successful, the station transmits the packet from its yellow tap.
5. Having completed a successful packet transmission, the station immediately resumes the checking loop 1.

The receiver is guaranteed to have seen the packet before it is removed, as every station has a receiver located between the sender's two consecutive taps (this is sufficient for green tap transmissions) and the condition $TRC \geq n-1$ guarantees that the slot will make a full loop and reach its sender before it reaches the token-holding station (for yellow tap transmissions).

In our virtual implementation of DSMA we assumed the slot format of DQDB. In fact, the protocol can be viewed as a fair cousin of DQDB. It is slightly reminiscent of some ideas aimed at eliminating the unfairness of DQDB by periodically moving the slot generator [19].

3.4. Example

Consider a three-loop spiral with five stations (labelled A, B, C, D, E). The token is held by station C in tap C_1. In this situation, the spiral may be visualised as a bus to which each station is connected three times, as shown in Fig. 3. In the figure, the subscripts in the station identifiers represent the spiral loops: C_1 represents the tap connecting station C and the first loop of the spiral.[8]

In the situation shown in the figure, C_3, D_3, E_3, A_1, and B_1 are yellow taps, while all the other taps are green. The token is held at tap C_1. The value of TRC of station C is 0. When station C releases the token in the next slot, it will label C_1 as a yellow tap and increment its TRC.

The station holding the token is guaranteed a successful transmission in its token-holding tap. As the token is passed around, each station gets its chance to transmit without pre-emption.

A station seizing an empty slot reaching one of its green taps is allowed to transmit even if it does not possess the token. The transmission is always successful: the station "knows" that its packet will make it to the destination, since the path from the station's transmitting tap to its next tap is never disconnected by the token-holding tap. When the traffic is low, it is easy to seize a slot; thus, the network avoids a high token acquisition delay.

The protocol permits stations to transmit from their yellow taps, if a specific condition holds. The rationale for this condition can be explained using Fig. 3. Assume that at time t_0, station B wants to transmit a packet and sees an empty slot reaching its yellow tap B_1 (but the other taps see full slots). If B transmits its packet at this moment, two outcomes are possible:

1. The packet will reach its destination and will be received.
2. Before reaching its destination, the packet will reach the token-holding tap and will be destroyed.The second case is to be avoided. It is most likely to occur when the receiver is the upstream neighbour of B, i.e. A (note that all the other receivers of packets sent by B are upstream from this tap). A will receive the packet only if the token reaches its tap A_2 no later than the packet. The token will reach A_2 at time $t_0 + TTRT - 1 - t - distance(B_1, A_2)$, where t is the current reading of the TRC of station B. If a packet is sent by B_1 at time t_0, it will reach A_2 at time $t_0 + L - distance(B_1, A_2)$, hence the condition for success:

$$t \geq n-1$$

Note that although the notion of distance between stations is used in the above argument, there is no need to know what this distance is.

3.5. The knowledge of receiver location

If we assume that stations are allowed to know the other stations' locations on the spiral, the protocol can be improved a little. Namely, the condition for transmitting from a yellow tap can be weakened and, consequently, the yellow taps can be eligible for transmission more often. Note that the original condition for a transmission from a yellow tap:

$$TRC \geq n-1$$

[7] The rationale for this condition is explained below.

[8] Of course, first is just an abstract concept.

guarantees that the packet will make a full loop through the network before it hits the token-holding station. What is sufficient for the success of a yellow-tap transmission is that the packet reaches its destination; it does not have to arrive back at the sender.

Assume that the stations are assigned sequential numbers from 0 to $n-1$ reflecting the stations' order on the spiral. Assume that a station S_i is willing to transmit a packet to station S_j. This transmission can be done on a yellow tap, if the following condition holds:

$$TRC \geq \begin{cases} j-i & \text{if } j > i \\ n-i+j & \text{if } j < i \end{cases}$$

The weakening of the yellow tap condition affects the yellow taps only; therefore, its impact is more visible when the number of loops in the spiral is not very big. The most interesting case from this point of view is a single-loop spiral, although for two or three loops the improvement is also visible. Considering this, we adopted this variation as the standard version of DSMA.

If a transmitting station has a backlog of packets to transmit, it could reorganise its packet queue to transmit on the yellow tap the packets addressed to the nearest destinations. This way the availability of the yellow taps for transmission will be increased, especially for heavier traffic conditions. Although, at first sight, this solution has the unpleasant taste of unfairness and postponement, there is no starvation: the station can compensate by ordering transmissions from green taps (and the transmissions done while holding the token) in a reverse manner. Moreover, some people would argue that favouring packets addressed to close neighbours is a good idea, especially in a MAN environment (e.g. "communities of interest" [20]). None the less, we chose not to incorporate this variation in the version of DSMA that we analyse below.

3.6. Synchronous traffic

DSMA does not explicitly differentiate between isochronous and asynchronous traffic. With each turn of the token, every station gets a guaranteed share of the bandwidth at fixed time intervals, which it can use as it pleases. Likewise, when a station absorbs its own packet, it can reuse it with a guaranteed success. We consider this property to be superior to the ways synchronous traffic was integrated into DQDB, FDDI, and METARING [21].

Essentially, there are three possible transmission strategies as far as synchronous traffic is concerned:

- *Transmit only during token possession.* This strategy offers the lowest possible jitter (bounded by the variation of the repeater delays) and should be used if low delay jitter is the primary concern. Each station S_i has a guaranteed isochronous bandwidth equal to 1/TTRT. Bandwidth guaranteed for isochronous traffic, but unused, may be used freely for transmitting other packets. Because no slot reuse takes place for synchronous traffic, the maximum synchronous throughput is limited to

$$\left(1 - \frac{L}{TTRT}\right) \times \frac{p}{p+h}$$

  where h represents the length of the slot header, and p is the length of the slot payload. Consequently, adding more loops to the spiral does not increase synchronous throughput.
- *Transmit during token possession and slot reuse.* Synchronous transmissions are allowed during token possession and at predictable transmission windows by stations reusing their own slots. Note that if a station transmits m slots at time t from a green tap, it is guaranteed to get a transmission window of m slots at time $t+L$. Note that with this approach it is possible for synchronous traffic to completely pre-empt asynchronous messages. It can be shown,[9] that if n is a divisor of $K \times L$, these transmission windows occur at a constant frequency, thus guaranteeing a negligible delay jitter, even if the whole throughput consists of synchronous traffic.
- *Transmit at all opportunities.* If the volume and structure of synchronous traffic are not static and exceed the guaranteed upper bound of $1/TTRT$,[10] a higher *maximum* bandwidth for synchronous traffic may be obtained by allowing the transmission of synchronous packets from every tap—at the expense of a non-zero delay jitter. There still will be no packet loss if the synchronous load of any station does not exceed the value of K/n. With this strategy, synchronous traffic is given priority over asynchronous traffic, but treated in the same way in all other respects. This approach results in maximum bandwidth availability for synchronous traffic, but the jitter is also high—due to possibly irregular access intervals.

The second strategy seems to be particularly well suited for synchronous traffic organised into sessions of more or less constant intensity. A station willing to sustain the regularity of its transmission windows should make sure that it reuses all its slots. Sometimes it might be reasonable to transmit dummy data—just to keep the slot reserved.

Regardless of the network load, the third strategy offers each station a transmission window at least every time the second strategy would give it one, thus maintaining some reasonable quality of service. Note that such a strategy would not perform properly for FDDI; in FDDI, the bandwidth reserved for synchronous traffic cannot be adjusted dynamically and when excess synchronous packets are sent as asynchronous ones they may suffer a very high delay jitter if the total load is high.

[9] See Section 5.1.4 for more details.

[10] E.g. when VBR video is present or when load patterns change with time.

3.7. Network management

All the protocols for the spiral topology discussed above are subjected to a startup phase which is similar to the initialisation of FDDI. During this phase, stations determine which station is going to generate the token. During normal operation, the protocol can be potentially confused by several exceptional events which include:

- the insertion of a station (i.e. a dormant station waking up);
- the disappearance of a station;
- the disappearance of the token (usually, but not always, combined with the disappearance of a station);
- the failure of a slot-emitting station.

Some MAC-level protocols handle these events better than others; the ability to handle them gracefully and fast is recognised as an important aspect in the overall quality of a protocol. We claim that DSMA handles all these events gracefully.

- *Station insertion.* When a station goes from bypass mode into regular operation, it waits for the token to reach it; the station does not capture the token, but resets its TRC while letting the token pass. From this moment on, the station can transmit using the standard transmission rules of DSMA (but not the packet removal rule or the token capture rule). The first packet it sends should be a broadcast control packet[11] informing all the other stations about the insertion; then, the station starts normal operation. All the other stations simply increment their TRC values upon reception of this packet. Normal operation is not disturbed at any time, although the inserted station may have to wait for some time if the network is saturated.
- *Station deletion.* When a station receives the token ahead of schedule, it knows that its predecessor(s) has disappeared. The station sends a broadcast packet informing all the other stations about the disappearance (also specifying how early the token was) and continues its normal operation. Upon reception of this packet, all the other stations update their TTRT values accordingly. The adjustment of counters at the stations remaining in the network is simple since their values are expressed in slots.
- *Token regeneration.* Each station assumes that it receives the token at a specific time based on the value of its TRC. Thus, even if the token was lost, the station that was supposed to get it captures it implicitly. Note that having "captured", a lost token, the station immediately starts to remove whatever traffic arrives at its token-holding tap. This causes no loss, since the removed traffic should have been received by its destination station beforehand; this is guaranteed by the transmission rules.
- *Slot generation.* In DSMA, all the generated slots initially carry the token. Thus, this case reduces to token regeneration.

Token recovery is especially painless, as it involves no overhead whatsoever. The other events are recognised by the network after some delay, but the operation of the network is not disrupted.

3.8. Fairness and absence of starvation

Since DSMA operates on a symmetric network, it is clear that no station is in any way privileged at the expense of another. Additionally, each station receives the token with the same frequency and holds it for the same amount of time. Since slots are reused by the source station, a station that acquires a token can take over a slot and reuse it during the whole life of the slot, if there is such a need. Thus, each station can, if it wants to, monopolise every nth slot in an n-station network.

From the above it is clear that starvation is not possible, since the maximum access delay is n slots. Also, each station can claim its fair share of the bandwidth ($1/n$ of the total bandwidth) even when traffic is heavy. When traffic is light, an active station can use almost all of the total bandwidth (except for the empty slots seen by the yellow tap when the send condition is not met).

Note that since in DSMA, each station decides individually how to divide the bandwidth available to it into synchronous and asynchronous traffic, DSMA is fair for both types of traffic.

4. Simulated environment

The performance of DSMA was investigated by simulation. The spirals and the protocols were modelled in SMURPH [22,23].

Typically, each point in a performance curve was produced as an average of four independent experiments. The number of messages transmitted in one experiment depended on several criteria aimed at ensuring the stability of the results. The network would start from an idle state, and the observed message delay was monitored after every reception of 40,000 messages. When this delay ceased to grow (meaning that the network has reached an equilibrium state), the experiment was continued for a further 300,000–1,000,000 messages, depending on the propagation length of the spiral. During this phase the proper measurements were taken.

4.1. Network geometry

The results presented here have been obtained for spirals with one, three, and five loops. A one-loop spiral is the simplest possible spiral network. All the protocols discussed

[11] A control packet is recognised by the network controller of each station and processed immediately.

in this paper work in a single-loop environment and this environment is an important special case. With any number of loops greater than one, DSMA protocols offer zero queuing delay under light load. But in the single-loop network, the light load queuing delay is generally non-zero (all taps are always yellow).

The results for the other two cases (three and five spirals), and also for other configurations not discussed here, demonstrate that when the number of loops is greater than one, the protocol behaviour is practically independent of the number, if the observed throughput is scaled by a factor equal to the number of loops.

Three lengths of a ring (one loop) were considered: 10^5 bits, 10^6 bits, and 10^7 bits. Assuming a 200 km ring, the three propagation lengths represent transmission rates of 100 Mb/s, 1 Gb/s, and 10 Gb/s; conversely, assuming a transmission rate of 100 Mb/s, they represent distances of 200 km, 2000 km, and 20,000 km (the latter being more suitable for Jupiter than for Earth).

In all networks, the number of stations was 32. The 32 stations were equally spaced along the spiral.

4.2. Packet format

The slot format of DQDB was assumed: it consists of an 8-bit header (containing the control bits), a 32-bit address field, and the 384-bit payload part. Two bits from the 8-bit header are used by our protocol: one to indicate the full/empty status of the slot, the other to pass the token.

4.3. Traffic patterns

Except for one experiment (Fig. 7), the traffic pattern was uniform with the likelihood of every pair *sender, receiver* being the same. The packet length was fixed and assumed to match exactly the payload length of the slot.[12] The results obtained do not account for fragmentation from splitting a message into multiple slots.

A simple model of packetised voice traffic was used to investigate the network behaviour under synchronous load. In the synchronous traffic model, we assumed two-slot voice messages, produced every 10 ms. The traffic was organised into multiple streams (each stream involving one sender and one receiver) randomly assigned to stations. The traffic intensity was adjusted by changing the number of streams.

4.4. Performance measures

The performance of the protocols described here is compared by relating their throughput-versus-delay graphs. In determining the throughput, only the slot payload bits are counted, i.e. headers do not count as transmitted bits, although they use bandwidth. The throughput tells the number of useful bits received by all stations within a time unit. The time is measured in bits; thus, the throughput is normalised.

The delay measure is the so-called *packet queuing delay* understood as the amount of time (expressed in bits) elapsing since a packet is queued for transmission, until the packet has been completely transmitted by the sender. It includes all the waiting time at the sender, but excludes the propagation time. This approach seems reasonable for comparing networks with different propagation lengths. Additionally, the reader can easily re-scale the curves, if the complete packet delay happens to be of interest.

[12] Issues related to segmentation and reassembly are not considered in this study.

5. Comparison

There is little published work on multiple rings, other than dealing with fault tolerance. On the other hand, many protocols for dual rings have been proposed; several have also been implemented. They can be classified as falling into one of two categories: co-rotating dual rings and counter-rotating dual rings.

The counter-rotating dual ring was originally introduced in [24] under the name of *Distributed Double Loop Computer Network*; METARING adopted significant properties from DDLCN.

5.1. Throughput

5.1.1. FDDI

It is difficult to compare "true" FDDI with the other protocols, since it is not a slotted protocol. To make the comparison meaningful, we used a slotted variation, in which there are no preambles and the pockets have the format of DQDB slots. Additionally, since we are comparing FDDI with a K-loop spiral, we simulated K independent FDDI networks.

The performance of FDDi depends on TTRT. The maximum effective throughput of FDDI is equal to:

$$K \times \frac{TTRT - L}{TTRT} \times \frac{p}{p+h}$$

where L is the propagation length of the ring, h stands for the length of the packet header, and p is the length of the packet payload. Jain [5] gives a slightly different formula, which, converted to the symbols used here, is:

$$K \times \frac{TTRT - L}{TTRT + L/n} \times \frac{p}{p+h}$$

which is similar, but slightly pessimistic.

By increasing TTRT, the maximum throughput achieved by FDDI can be pushed theoretically arbitrarily close to 1. In reality, this method has obvious limitations: it assumes that each station always has sufficiently many packets to

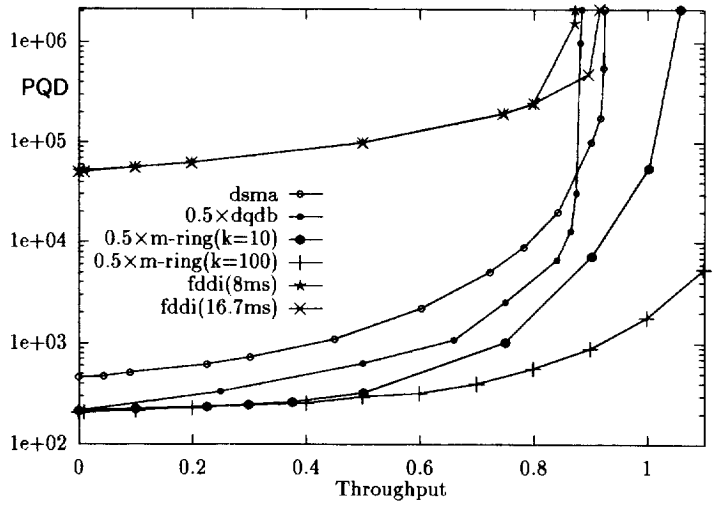

Fig. 4. One segment 100K.

use the entire token-holding time of the network. Moreover, it drastically increases the packet queuing time for asynchronous traffic, which is poorly bounded even for moderate values of TTRT. In Fig. 4, the values of TTRT used for FDDI are 8 ms and 16.7 ms; in Fig. 5, the values used for FDDI are 16.7 ms and 167 ms.

Note that when the network is very long (Fig. 5 and Fig. 6), the maximum throughput of FDDI is very low, even if TTRT is large. Moreover, the impact of the packet queuing time for light load (L/K on average) becomes very pronounced. For this reason, FDDI is not present in Fig. 6 (it would not fit in the figure box).

5.1.2. METARING

In principle, METARING is a *capacity-1* protocol, but only if the value of the k parameter [3] is unlimited. But then the network is starvation prone. It was suggested in [3] that small values of k should be used to reduce the probability of starvation (cf. [25]). The maximum throughput of METARING (under uniform load) is:

$$\frac{p}{p+h} \times \min\left(\frac{2nk}{L}, 8\right)$$

where n is the number of stations. The role of k can be compared to the role of the token-holding time in FDDI. In contrast to FDDI, METARING offers zero queuing time under light load, irrespective of L, but in terms of achieved throughput, exhibits similar characteristics to FDDI. By increasing the value of k, the throughput of METARING can be pushed arbitrarily close to its theoretical maximum; however, when k is much bigger than 1, METARING becomes starvation prone. Note that k refers to the number of packets, thus $k = 10$ (Figs 1, 2, 3, 4 and Fig. 5) corresponds to 4240 bits, etc. Note that for the longest network (Fig. 6) the maximum throughput achieved by METARING is quite low, even though k (referring to slots) is by no

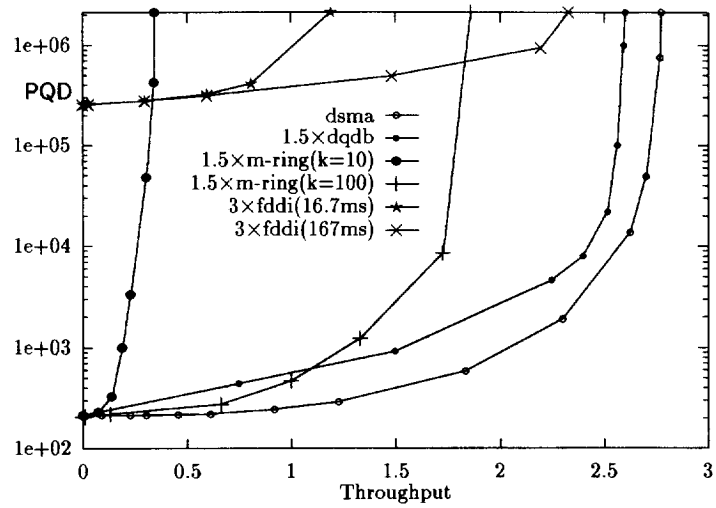

Fig. 5. Three segments 1M.

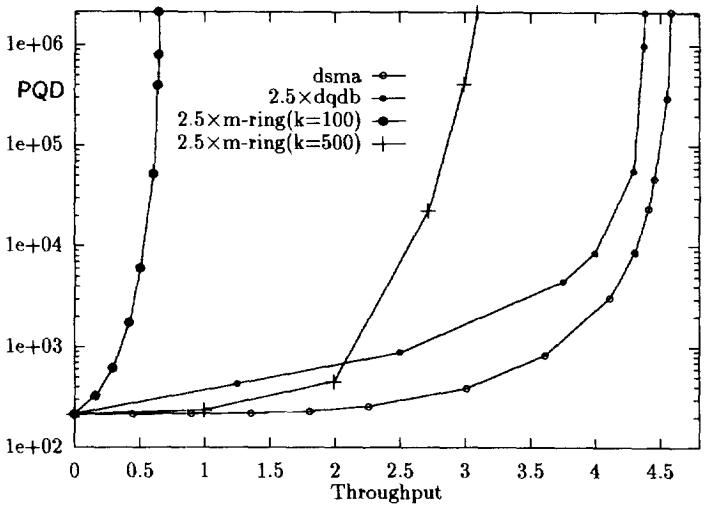

Fig. 6. Five segments 10M.

means modest ($k = 500$ corresponds to 212,000 bits, which, in the case of 32 stations, gives each station a guarantee of being able to transmit after a waiting period not exceeding 6,572,000 bits of time).

5.1.3. DQDB

The average performance of DQDB (in terms of the queuing-delay-versus-throughput characteristics) is quite independent of the network size and roughly equals $2 \times p/(p+h)$. For a K-cable DQDB (feasible only for even values of K), the maximum throughput would be equal to $K \times p/(p+h)$. Note that when $K > 2$, each station must assign submitted messages to one of the $K/2$ possible cables before sending the first slot containing part of the message (otherwise, the packet could arrive out of order).

5.1.4. DSMA

Here, we derive a formula for the maximum throughput of DSMA with an optimised transmission rule (i.e. stations use their knowledge of the relative positions of the other stations). The derivation of a formula for the maximum throughput of DSMA consists of three stages. In the first stage, we calculate the duration of the life of a slot. The second stage results in a formula for the average number of times a slot is used during its life, and the final stage yields a formula for the maximum throughput of DSMA operating on a K-loop spiral.

Assume a K-loop spiral which is saturated, i.e. in which each station has an unbounded supply of packets to be sent, let a slot σ be generated by station S_0 at time t_0. By definition, σ carries the token—it will be removed by station S_1 when σ reaches that station. From then on, σ will propagate along the spiral until it reaches a token-holding station. Let this station be S_m; by definition, S_m will destroy σ and replace it with a new slot.

We note that σ can meet the token only at a station. Consequently, σ must make exactly K more loops than the

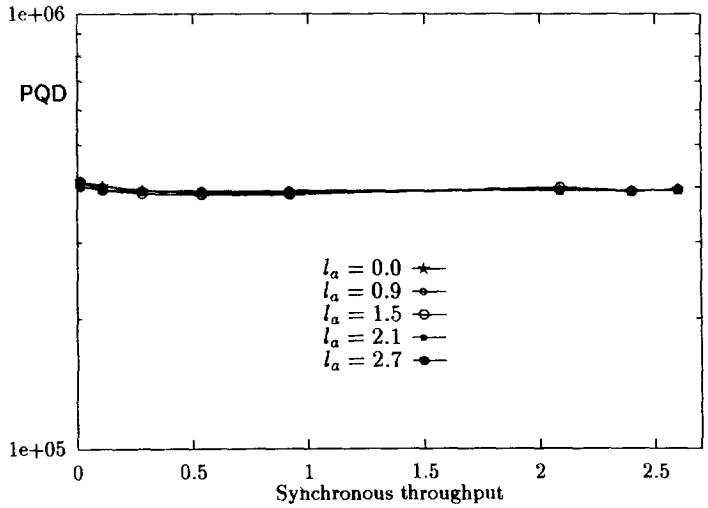

Fig. 7. Three segments 1M, DSMA synchronous traffic against uniform background.

token (since both of them started together from S_0). Let the number of full loops (i.e. visits to taps of station S_0) traversed by σ be R.

Thus, σ will reach the token at S_m at time:

$$t_\sigma = t_0 + R \times L + distance(S_0, S_m)$$

Likewise, the token reaches S_m at time:

$$t_t = t_0 + (R-K)(L+n) + distance(S_0, S_m) + m - 1$$

The $m-1$ factor comes from the fact that stations $S_1, ..., S_{m-1}$ held the token during the last (partial) loop.

Since we assumed that S_m holds the token when σ reaches it, $t_\sigma = t_t + 1$. Hence,

$$(R-K)(L+n) + m = R \times L$$

and

$$R = K + \frac{KL-m}{n}$$

Since R, K, L are all integer values, $m = KL \bmod n$, and

$$R = K + \frac{KL}{n}$$

Since the network is saturated (by assumption), σ was used at time t_0 and subsequently reused by S_0 during each of its $R-1$ first visits to the station. During its last visit, σ was reused only if the oldest packet waiting in the output queue of S_0 was destined to a station not beyond S_m. Otherwise, S_0 has to empty σ and let it go empty to the next station, where a next attempt to fill will be made (etc.). Assuming uniform traffic, the probability that none of the stations $S_0, ..., S_{m-1}$ has—at the front of its input queue—a packet destined to a station not beyond S_m is:

$$\wp = \frac{n-1-m}{n-1} \times ... \times \frac{n-2}{n-1} = \frac{(n-2)!}{(n-2-m)!(n-1)^m}$$

The total number of times σ was filled with a packet averages out to $F = R\wp + (R+1)(1-\wp)$ during a lifetime lasting $R \pm m/n$.

Consequently, the maximum throughput achieved by DSMA on a K-loop network is

$$\frac{F}{R+m/n} \times K \times \frac{p}{p+h}$$

which yields the formula for maximum throughput:

$$\left(K + \frac{1}{L+n} \times \left(n - m - \frac{n!}{(n-m-2)!(n-1)^{m+1}}\right)\right) \times \frac{p}{p+h}$$

where $m = KL \bmod n$. When n is a divisor of $K \times L$, it amounts to $K \times p/(p+h)$. If n is not a divisor of $K \times L$, the maximum throughput of DSMA marginally exceeds $K \times p/(p+h)$, since

$$(n-1)^{m+1} > (n-m-1) \times \prod_{i=1}^{m} n - m + i$$

for $1 \le m < n$. This can be explained intuitively by the fact that some packets do not complete a whole rotation back to their sender (they are removed by the token-holding station before that).

To achieve its maximum throughput, the spiral ring must be tuned, i.e. its length adjusted, possibly by inserting a short fibre loop into the ring. In Figs 4, 5, 6, and Fig. 7, arbitrary, and thus non-optimal values of L were used, so the maximum throughput shown there for DSMA is less than the theoretical maximum.

5.2. Simulation results

Figs 4, 5, 6, and Fig. 7 show the performance of DSMA for several network configurations. In the figures, the y-axis (PQD) represents the packet queuing delay, while the x-axis represents normalised throughput, if needed, multiplied by a suitable value to compensate for a difference in the number of rings (buses). For comparison, we include performance curves for FDDI, DQDB, and METARING. The curves have been scaled properly—to account for the difference in the number of loops (rings, buses). For example, when comparing DQDB and METARING with single-loop DSMA (Fig. 4), the throughputs of DQDB and METARING have been divided by two (they are dual ring or bus networks). Similarly, the throughputs of DQDB and METARING in Fig. 5 (three-loop spiral) have been multiplied by 3/2. The scaling of DQDB and METARING results is the correct approach,[13] since, in the se of these networks, increasing the number of cables is equivalent to increasing the transmission rate (i.e. two 100 Mb/s DQDB networks behave like one 200 Mb/s DQDB). On the other hand, this is not the case with FDDI, because of the token-passing mechanism. Thus, a three-ring and a five-ring FDDI were used to derive the curves in Fig. 5 and Fig. 6, since the true FDDI operates on a single ring. For all the networks, the slot format of DQDB was assumed.

Fig. 7 shows the packet queuing delay in DSMA for synchronous traffic. The measurements were performed against the background of uniform asynchronous traffic of varying intensity (l_a). The transmission strategy for synchronous traffic is the "compromise" strategy in which synchronous packets are transmitted during token possession and at the "predictable" moments when previously transmitted synchronous packets are reused. The observed indifference of the network's behaviour for synchronous traffic with respect to the uniform load is pronounced.

5.3. Confidence of the results

The confidence of the maximum throughput estimates in Figs 4, 5 and Fig. 6 is high. Starting from the moment when

[13] The approach was forced by the fact that these networks require an even number of cables.

the network reaches a steady behaviour, the observed throughput ceases to grow and remains practically the same, irrespective of the number of received packets. This equilibrium value exhibits zero variance across different simulation runs (with different random seeds).

As far as the delay versus throughput characteristics are concerned, the situation is slightly more complicated. Generally, as the points get closer to the saturation threshold, their confidence decreases. This happens because the standard deviation of the queuing time tends to grow faster than the mean. To get some insight into the problem, let us consider Fig. 5 and the curve for DSMA. The part of the curve to the right of throughput 4.5 is steep and its confidence is not very interesting. On the other hand, all the points located to the left of this point will have shorter confidence intervals. The observed mean packet queuing time at throughput 4.5 is $m = 2 \times 10^6$ bits and the observed standard deviation $\sigma = 0.8 \times 10^6$. The total number of samples used to determine the value of this point (from four experiments) was $N_s = 3,200,000$. The length of the confidence interval for a given probability α is:

$$\Delta = 2\frac{\sigma z_\alpha}{\sqrt{N_s}}$$

where z_α is the area under the normal probability curve between 0 and $\alpha/2$. Let $\alpha = 0.95$. Substituting the numerical values for the parameters, we get $\Delta \approx 850$, which is slightly more than 4×10^{-4} of the observed mean.

Even if we divide the number of samples N_s by 4 (assuming that the queuing delay of the packets transmitted during one token-holding interval is strongly correlated) we get $\Delta = 1700$, which is slightly below 0.1% of the observed mean.

6. Summary

We presented a network topology (the spiral ring) and a MAC-level protocol for this topology. The proposed solution can be viewed as a family of networks parameterised by the number of loops forming the spiral. The maximum throughput achieved by our protocol does not degrade with the increasing propagation length of the spiral. Even in the special case of a single loop (i.e. a ring), the network is able to use a fixed fraction of the channel bandwidth irrespective of its length. Moreover, in contrast to some other *capacity-1* solutions (e.g. DQDB), our network is absolutely fair. If the spiral is comprised of more than one loop, the network incurs no access delay when the traffic is light.

Acknowledgements

P. Gburzyński was supported in part by NSERC Grant No. OGP9183.

References

[1] Fiber Distributed Data Interface (FDDI)—Token Ring Media Access Control (MAC), American National Standard for Information Systems, Doc. No. X3, New York, NY, 139-1987, 1987.
[2] Distributed Queue Dual Bus Subnetwork of a Metropolitan Area Network, IEEE Std. 802.6-1990, 1991.
[3] I. Cidon, Y. Ofek, A full-duplex ring with fairness and spatial reuse, in: Proceedings of IEEE INFOCOM'90, 1990, pp. 969–981.
[4] ATM User-Network Interface Specification, version 3.0, The ATM Forum, 1993.
[5] R. Jain, FDDI Handbook, Addison-Wesley, Reading, MA, 1994.
[6] S. Breuer, T. Meuser, Enhanced throughput in slotted rings employing spatial slot reuse, in: Proceedings of IEEE INFOCOM'94, Toronto, Canada, 1994, pp. 1120–1129.
[7] I. Cidon, L. Georgiadis, R. Guerin, Y. Shavitt, Improved fairness algorithms for rings with spatial reuse, in: Proceedings of IEEE INFOCOM'94, Toronto, Canada, 1994, pp. 1103–1111.
[8] R. Cohen, A. Segall, Multiple logical token rings in a single high-speed ring, Technion Technical Report No. 738, 1992.
[9] T. Todd, The token grid network, IEEE/ACM Transactions on Networking 2 (3) (1994) 279–287.
[10] M. Gerla, G. Wang, P. Rodrigues, Buzz-net: A hybrid token/random access LAN, IEEE Journal on Selected Areas in Communications 5 (1987) 977–988.
[11] J. Limb, C. Flores, Description of Fasnet, a unidirectional local area communications network, Bell Systems Technical Journal (Sept. 1982).
[12] J. Limb, A simple multiple access protocol for metropolitan area networks, in: SIG-COMM'90 Symposium, 1990, pp. 69–78.
[13] G. Watson, S. Tohmé, A performance analysis of S + + : A MAC protocol for high speed networks, in: IFIP Workshop on Protocols for High-Speed Networks, Stockholm, Sweden, 1992.
[14] W. Dobosiewiez, P. Gburzyński, A new topology for MANs: The pretzel ring, in: Proceedings of IEEE INFOCOM'92, Florence, Italy, 1992, pp. 2408–2414.
[15] W. Dobosiewicz, P. Gburzyński, An alternative to FDDI: DPMA and the pretzel ring, IEEE Transactions on Communications 42 (1994) 1076–1083.
[16] L. Georgiadis, W. Szpankowski, L. Tassiulas, Stability analysis of scheduling policies in ring networks with spatial reuse, in: Proceedings of 21st Annual Allerton Conference on Communication, Control, and Computing, Allerton, IL, 1993, pp. 1109–1119.
[17] L. Tassiulas, L. Georgiadis, Any work-conserving policy stabilizes the ring with spatial reuse, in: Proceedings of IEEE INFOCOM'94, Toronto, Canada, 1994, pp. 66–70.
[18] O. Sharon, A. Segall, A simple scheme for slot reuse without latency for a dual bus configuration, IEEE/ACM Transactions on Networking 1 (1) (1993) 96–104.
[19] D. Karvelas, M. Papamichail, G. Polyzos, Performance analysis of tile rotating slot generator scheme, in: Proceedings of IEEE INFOCOM'92, Florence, Italy, 1992, pp. 794–803.
[20] A. Pach, S. Palazzo, D. Panno, Improving DQDB throughput by a slot preuse technique, in: ICC'92, Chicago, IL, 1992.
[21] H. Wu, Y. Ofek, K. Sohraby, Integration of synchronous and asynchronous traffic on the Metaring architecture and its analysis, IBM Technical Report RC 17718, 1992.
[22] P. Gburzyński, P. Rudnicki, Object-oriented simulation is SMURPH: A case study of DQDB protocol, in: Proceedings of 1991 Western Multi Conference on Object-Oriented Simulation, Anaheim, CA, 1991, pp. 12–21.
[23] P. Gburzyński, P. Rudnicki, The SMURPH protocol modelling environment, University of Alberta, Department of Computing Science, Edmonton, AB, Canada, 1991.
[24] M. Liu, Distributed loop control networks, Advances in Computers 17 (1978) 163–221.

[25] J. Chen, I. Cidon, Y. Ofek, A local fairness algorithm for gigabit LANs/MANs with spatial reuse, IEEE Journal on Selected Areas in Communications 11 (8) (1993) 1183–1192.

Wlodek Dobosiewicz received the MSc and PhD degrees in computer science from the University of Warsaw, Poland. He has been an Associate Professor at Monmouth University since 1994. He is currently working on MAC-layer protocols for high-speed computer networks. He has also published on sorting.

Pawel Gburzynski received his MSc and PhD in computer science from the University of Warsaw, Poland in 1976 and 1982, respectively. Before coming to Canada in 1984, he had been a research associate, systems programmer, and consultant in the Department of Mathematics, Informatics and Mechanics at the University of Warsaw. Since 1985 he has been with the Department of Computing Science, University of Alberta, where he is a professor. His research interests are in communication networks, operating systems, simulation, and performance evaluation. He authored LANSF and SMURPH-software packages for modelling communication protocols and a book on protocol design for local and metropolitan area networks.