Dynamic recognition of the configuration of bus networks

(As appeared in Computer Communications)

Włodek Dobosiewicz* Paweł Gburzyński† Piotr Rudnicki‡

present and analyse algorithms for recognising the network configuration by nodes in bus based LANs

December 22, 1994

Abstract

The configuration of most Local Area Networks changes in time, either as a result of adding or removing stations or as a result of stations failing for one reason or another. Many situations require that all the stations connected to the network must possess a certain amount of knowledge about its configuration. As the network configuration changes, such information has to be updated, preferably automatically, i.e., by the stations themselves without external intervention.

This paper presents a family of algorithms for automatic recognition of the current configuration for a number of bus networks. The algorithms presented here are useful in medium access protocols based on partial (or total) knowledge about the configuration.

*Supported in part by NSERC Grant No. OGP9110. Department of Computing Science, The University of Alberta, Edmonton, Alberta, Canada T6G 2H1. email: dobo@cs.ualberta.ca.

†Supported in part by NSERC Grant No. OGP9183. Department of Computing Science, The University of Alberta, Edmonton, Alberta, Canada T6G 2H1. email: pawel@cs.ualberta.ca.

‡Supported in part by NSERC Grant No. OGP9207. Department of Computing Science, The University of Alberta, Edmonton, Alberta, Canada T6G 2H1. email: piotr@cs.ualberta.ca.

Key words: Local area networks, Network configuration, Configuration recognition.

1 Introduction

A **bus network** consists of a linear bus, made of a single cable or a pair of cables, and of a number of stations attached to this bus. There are several variations, as the cables may be unidirectional or bidirectional, stations may have more than one tap to the bus, etc. The term **network architecture** denotes the number of cables, their characteristics, and the way that stations are attached to them. The term **network configuration** denotes the ordering of the stations on the bus and, when relevant, also the positions of the stations on the bus.

Numerous protocols for bus networks require some knowledge about the distribution of stations along the bus. To name a few: *SOSAM* [4], *BID* [10], *L-Expressnet* [1], *Fassnet* [3, 6], *Piggyback* ETHERNET[2].

For example, some protocols need the information about the ordering of all the stations along the bus. This is achieved by assigning to stations *soft addresses* numbered $0, \ldots, N-1$ in the order of the stations' occurrence on the bus.

Some other protocols need a full knowledge of the network's geometry, i.e. the knowledge about the *propagation delay* (distance) between every pair of stations. A similar knowledge is needed when a number of stations want to synchronise their private clocks (this is needed in many situations).

Moreover, many protocols give additional tasks to certain stations, e.g. the extreme stations on the cable(s). These stations must therefore be aware that they occupy the positions that were singled out for these tasks.

If the protocol requires some knowledge about the network, extra time is needed for network initialisation and reconfiguration. The same holds if some stations play a special role. Moreover, this knowledge must be updated periodically, due to changes in the network configuration. This issue is largely ignored in the literature, where it is assumed that the needed information is somehow hardwired into the network controllers before the

network is started. Alas, as the configuration of every network is dynamic, this assumption is unrealistic. As a result, many researchers and implementors hold a negative attitude towards protocols that require any knowledge that may change dynamically.

In this paper, we present a collection of distributed algorithms that recognise the current configuration of a bus network. These algorithms take into account the various architectures of bus networks. Ramarao [7] discusses algorithms for recognition of topology of an arbitrary network.

This paper does not address the very important issue of recognising that the configuration changed and that a suitable algorithm should be executed (note that the algorithms are distributed). We assume that the network protocols that use any of our algorithms are capable of doing so by recognising a start-up signal sent by one of the hosts.

2 Bus network architectures

A bus network consists of a number of stations. These stations are connected by a bus that offers a broadcast-type medium. From the point of view of these connections, we distinguish three basic shapes that such a bus can take:

- A single bidirectional bus.
- A single unidirectional bus, shaped like a U or like an S.
- A dual unidirectional bus.

These may, in turn, be combined into more exotic shapes, such as a dual U-shaped bus, etc.

For convenience, the two ends of the bus will be referred to as the *left* end and the *right* end[1]. The distances on the bus are expressed as *propagation*

[1] Throughout this paper, the notion of left and right is only a convention.

delays in bit-slots (e.g. for a 10Mb/s ETHERNET, 1 bit-slot = $100ns$). Each station is assigned a specific location on the bus. The location of a station S represents its distance (in bit-slot time units) from the left end (the leftmost station) of the bus and is denoted by $l(S)$. Stations are attached to the bus at arbitrary locations. In particular, several stations may be attached to the bus in almost exactly the same location.

2.1 Bidirectional bus

The bidirectional bus consists of a single, bidirectional, broadcast-type medium and of a number of stations distributed along the bus (see figure 1) (cf. ETHERNET[8], *L-Expressnet* [1], *Piggyback* [2]).

——————— figure 1 should be placed around here

In a bidirectional bus, the propagation distance between two stations S_1 and S_2 equals $d(S_1, S_2) = |l(S_2) - l(S_1)|$.

Every station has an internal clock, which ticks independently from the clocks of other stations. As the accuracy of this clock is finite, a station may have problems in sensing the time difference between two events that are almost simultaneous. Let the minimum time interval that can be perceived by a station S be denoted by δ_S. Note that different stations have slightly different values of δ.

If two stations, S_1 and S_2, are very close to each other, the propagation distance between them can be less than $\min(\delta_{S_1}, \delta_{S_2})$. This implies, among other things, that both these stations hear an incoming packet at the same time (taking the accuracy of their clocks into account), or, at least, cannot determine which of them heard it first.

2.2 Single unidirectional bus

Unidirectional channels are naturally implementable on the basis of optical fibres and, as such, are commonly used in fast bus networks.

The bus is logically divided into two segments: the *outbound* segment and the *inbound* segment. By convention, transmissions are assumed to propagate on the *outbound* segment in the direction from the *left* to the *right*. Each station has two connections to the bus:

- A transmitter/sensor tap connected to the outbound segment. The sensor tap is capable of recognising the presence of a transmission from upstream, but is not capable of receiving a packet.
- A receiver tap connected to the inbound segment.

2.2.1 S-shaped bus

——————— figure 2 should be placed around here

In an S-shaped bus (see figure 2, cf. *Expressnet* [9]), the propagation distance between two stations S_1 and S_2 equals $d(S_1, S_2) = 2L + l(S_2) - l(S_1)$, where L is the length of the *outbound* segment.

2.2.2 U-shaped bus

——————— figure 3 should be placed around here

In a U-shaped bus (see figure 3), the propagation distance between two stations S_1 and S_2 equals $d(S_1, S_2) = 2L - l(S_2) - l(S_1)$, where L is the length of the *outbound* segment.

2.3 Dual unidirectional bus

A dual unidirectional bus consists of two unidirectional cables (see figure 4), each cable being used for transfers in one direction, cf. *Fasnet* [6], *Z-net* [5]. One of the cables is used for *left-to-right* transmissions and is called the LR cable; the other cable, called the RL cable, is used for *right-to-left* transfers. The cables are symmetric; thus, the labelling is arbitrary. By convention,

we assume that the leftmost station on the LR cable should be assigned label 0.

——————— figure 4 should be placed around here

The propagation distance between two stations S_1 and S_2 equals $d(S_1, S_2) = |l(S_2) - l(S_1)|$, assuming that the two cables are identical (see the next subsection).

2.4 Symmetry in busses

Computing the distances between pairs of stations in a distributed fashion requires that the distances between stations are symmetric, i.e., $d(S_1, S_2) = d(S_2, S_1)$ for all pairs S_1, S_2. This assumption must be true for a bidirectional bus. It may be true for the *U-shaped* and the *H-shaped* busses if the cable segments are identical in length. However, it cannot be true for the *S-shaped* bus. This implies that no distributed algorithm computing distances between stations of an *S-shaped* bus exists—one can only compute the values of $d(S_1, S_2) + d(S_2, S_1)$, see Section 5.2.1.

3 Some properties of bus networks

3.1 Hardwired knowledge

We assume that when the configuration recognition algorithm is executed, the stations that are attached to the network know very little about its configuration. They know the protocol and the architecture of the network, i.e. the number of cables and their properties. Although they do not know the actual length of the bus, the stations know some upper limit L on that length. The value of L may be quite pessimistic[2]. Each station knows its

[2] E.g., in commercial *Ethernet* the value of L used by the backoff function (256 bits) is a rather pessimistic upper limit.

hardwired address which is guaranteed to be different from the *hardwired* address of any other station. Of course, the hardwired addresses are not expected to obey any specific ordering. Moreover, each station knows only its own address and does not know the number of other stations on the bus, nor their hardwired addresses.

We assume that nothing more is known to the stations when the configuration recognition algorithm starts.

3.2 Automatic configuration recognition

Upon detection of a startup signal, which may come as a special packet transmitted by one (or more) stations upon request from a host, the stations start executing the configuration recognition procedure. We assume that all the stations that are alive receive the startup signal at approximately the same time, so that they start executing their recognition procedures within L time from each other.

The purpose of the configuration recognition procedure is to determine the number of stations connected to the bus (N) and to assign to every station a "soft" address, i.e., a number from 0 to $N-1$ reflecting the station's position relative to one end of the bus (called **left** end by convention). The soft addresses of stations are determined by the non-decreasing order of their distances from the left end of the bus. If two stations happen to occupy the same location (or two undistinguishable locations) on the cable, their order cannot be determined and can be assigned arbitrarily[3]. If needed, the configuration recognition algorithm also computes the propagation delays between every pair of stations.

[3]This case is relevant only for the bidirectional bus.

3.3 Configuration protocol

When the stations are executing their configuration recognition algorithm, they obey a special protocol, which is different from the "normal" protocol that is used by the network. This configuration-time protocol depends on the architecture of the network.

The proposed algorithm for a bidirectional bus uses the contention resolution approach of the ETHERNET protocol (CSMA/CD). This approach is useful in the situation when one station among k eligible stations must be singled out. It is assumed that the reader is familiar with CSMA/CD; thus, the details of contention resolution will not be presented here.

Algorithms for the unidirectional bus take advantage of the packet preemption rule which allows stations to sequence their packets. A station willing to send its packet awaits a period of silence on the *outbound* segment of the bus. Upon sensing a sufficiently long period of silence, the station starts transmitting a short preamble—a predetermined sequence of bits. While transmitting the preamble, the station listens for other transmissions on the *outbound* segment of the bus. If it senses another transmission (which can only be coming from upstream, as the bus is unidirectional), the station considers itself **preempted** and stops transmitting. In such a case, the preempted station resumes its wait for a period of silence and restarts its transmission all over again.

The same rule applies to a dual unidirectional bus, with transmitting and sensing for other packets performed on the same cable.

When a packet arrives to a transmitting station, the station is preempted and stops transmitting. However, before it recognises this situation, its transmitting corrupts the beginning of the incoming packet. This is one of the reasons why every packet starts with a preamble—which should be long enough so that only its beginning is corrupted, leaving enough bits for the synchronisation of the receiver.

4 Bidirectional bus

4.1 Soft addresses and propagation delays

This section presents an algorithm for calculating soft addresses of the stations without computing the propagation delays. The soft addresses reflect the order of the stations along the bus. It is possible that some stations will occupy almost identical locations; thus, the algorithm cannot rely on some minimum distance between adjacent stations[4]. This makes station numbering somewhat arbitrary, if two stations occupy the same location. Thus, the algorithm below guarantees a correct numbering for any pair of stations S_i and S_j such that $d(S_i, S_j) > \max_{0 \leq k < N} \delta_{S_k}$.

The algorithm consists of three phases.

All stations start by performing N contention rounds using the contention resolution rule of CSMA/CD. The station that wins the first contention round is given a temporary label #0. The next to win gets a temporary label #1, etc. up to the last station, which gets a label $\#N - 1$. The station labelled #0 will play the role of leader in phase two. The purpose of phase two is to determine one extreme station (called *leftmost* by convention). This station takes the role of leader in the last phase, which determines the final ordering of all the stations. As a byproduct, distances between stations are also computed.

The algorithm takes into account the possibility that the station that is expected to transmit fails unexpectedly: if a silence of length $4L$ is sensed, all the stations that are still alive restart the algorithm from the very beginning.

1. Upon detecting the startup signal, all stations perform contention rounds that establish their temporary labelling. In each round, all the stations that have not been labelled yet attempt to broadcast

[4] If we are allowed to assume that the minimum distance between stations more than a bit, a much faster algorithm can be used.

packets of length $2L$ containing their hardwired addresses. In the case of a collision, they use the contention resolution rule of ETHERNET. "Eventually", one of the stations will succeed in transmitting its packet. The winning station assigns to itself the next available number (starting with #0) and waits until the end of phase one (all the other stations note the label and the hardwired address). All other stations perform additional contention rounds, until all stations are labelled. The end of phase one is recognised as a period of silence of length $2L$ following a successful transfer. At this moment, every station has a label, which is a number between #0 and $\#(N-1)$.

2. Phase two consists of an exchange of $N-1$ pairs of very short packets. Station #0 sends a packet. Immediately upon sensing its end, station #1 sends a packet of its own. Upon sensing the end of #1's packet, station #0 sends another packet, to which station #2 responds immediately, etc. After sending a packet, every station measures the amount of time elapsing until another packet is heard. The value measured represents:

   - For station #0, the i-th value measured ($R_{\#i}$) satisfies the inequality:

     $$2d(\#0,\#i) - \delta_{\#0} < R_{\#i} < 2d(\#0,\#i) + \delta_{\#0}$$

     as the i-th packet that it hears is sent by station $\#i$.

   - For any other station S, the measured value r_S satisfies the inequality (S sends only one packet)

     $$2d(S,\#0) - \delta_S < r_S < 2d(S,\#0) + \delta_S$$

     as the next packet that it hears is a packet sent by #0. Note that this is also true for station $\#(N-1)$, see below.

3. At the end of phase two, station #0 has a table of values $R_{\#1}, \ldots, R_{\#N-1}$ equal to the times measured. Scanning the values $R_{\#}i$, station #0 finds all the indices $\#i$ such that $R_{\#}i$ is the largest of all the values. Then, station #0 sends one more packet, in which it lists all these station indices in ascending order. Station $\#(N-1)$ uses this packet to determine its value $r_{\#(N-1)}$.

4. In this phase, all the stations listed in the last packet sent by #0 determine which among them is the outermost station. They do so by sending one short packet each; this packet contains the value r as measured by the station. The stations broadcast their packets in order of their labels. The first station to broadcast the maximum value is chosen as the outermost station.

5. The chosen station considers itself to be the *leftmost* station on the bus and adopts for itself the soft address 0. Note that this station is not farther than $\delta_{\#0}$ away from the actual station occupying the same end of the bus. Station 0 repeats the steps performed in phase two (by station #0). At the end, it collects a vector of values $R'_{\#i}$. For every i, $R'_{\#i}$ satisfies the inequality:

$$2d(0,\#i) - \delta_0 < R'_{\#i} < 2d(0,\#i) + \delta_0$$

This inequality implies that:

$$R'_{\#j} < R'_{\#i} \Rightarrow \; d(0,\#j) - d(0,\#i) < \delta_0$$

6. Station 0 sorts the vector R'. Then, it assigns permanent soft addresses based on the order of stations and broadcasts the results.

When the algorithm stops, each station knows its soft address on the bus. If several stations are separated by a distance smaller than δ_0, their soft addresses may be different than their true locations on the bus (as perceived

by someone with perfect measuring instruments). If there are stations that are to the left of station 0, they will get "incorrect" soft addresses—they may be placed after stations that are up to $\delta_{\#0}$ away from them.

The algorithm requires N contention rounds (each consisting of an arbitrary number of collisions followed by one successful transfer) and then no more than $4N - 2$ packets transmitted in non-contention mode.

5 Unidirectional bus

This section presents configuration recognition algorithms for unidirectional-bus networks. All these algorithms take advantage of the preemption rule described in Section 3.3.

5.1 Soft addresses

5.1.1 Single unidirectional bus

The algorithm is the same for the U-shaped and for the S-shaped unidirectional bus networks.

- After sensing the start-up signal, every station sends an arbitrary packet of length $2L$ obeying the preemption rule. The preemption rule guarantees that the leftmost station is the only station that has not been preempted.
- The leftmost station sends a packet containing its hardwired address.
- All the other stations listen for a period of silence following a packet. When they sense it, they transmit a packet containing their own hardwired address. While transmitting, each station obeys the preemption rule.

- During this phase, all the stations listen for packets on the *inbound* segment of the bus. Every station hears precisely N packets, containing the hardwired addresses of all the stations. The order of the packets is the same as the order of stations on the bus; thus, the soft addresses can be derived directly from it.

The total number of packets successfully sent during the execution of this algorithm equals N.

5.1.2 Dual unidirectional bus

It is assumed that the stations know the architecture of the bus, i.e. they distinguish between the LR cable and the RL cable.

The algorithm for the H-shaped dual bus is similar to the algorithm for the single unidirectional bus. We assume that the soft addresses reflect the order of the stations on the LR cable and that the order on the RL cable is reverse (otherwise, there would be a pair of stations S_i and S_j such that S_i would not be able to send messages to S_j).

- After sensing the start-up signal, every station sends on the LR cable an arbitrary packet of length $2L$ obeying the preemption rule. The preemption rule guarantees that the leftmost station is the only station that has not been preempted.

- The leftmost station sends a packet containing its hardwired address.

- All the other stations listen for a period of silence following a packet. When they sense it, they transmit a packet containing their own hardwired address of length P. While transmitting, each station obeys the preemption rule.

- During this phase, all the stations read in all the packets coming from upstream. Every station hears as many packets as it has predecessors;

each of these packets contains the hardwired address of one of these stations. The order of the packets is the same as the order of stations on the LR cable. In particular, the rightmost station received precisely $N-1$ packets from other stations. Their order of arrival determines the ordering of all the stations on the LR cable, i.e. their soft addresses.

- All the stations remain silent for a period of L after finishing their transmission.
- Every station sends on the RL cable an arbitrary packet of length $L+P$ obeying the preemption rule. The preemption rule guarantees that the rightmost station is the only station that has not been preempted.
- The rightmost station sends on the RL cable one packet containing the hardwired addresses of all the stations in the order of their soft addresses.

The total number of packets successfully sent during the execution of this algorithm is $N+3$.

5.2 Propagation delays

5.2.1 S-shaped bus

The problem of finding the propagation delays in an S-shaped bus is equivalent to finding propagation delays in a unidirectional ring (see figure 5). This is not feasible under the assumption that stations have independent clocks. Consider the case when $n = 2$. A simple, knowledge-based argument indicates that station 1 cannot learn its distance from station 0. Station 1 can append the reading of its clock to a packet send by station 0 which then can be read by other stations (including 0). However, the initial setting of 1's clock can be chosen in a way depending on its location such that other stations will hear identical messages send by 1, irrespective of 1's location.

(The same reasoning applies to any other message that station 1 appends to a packet sent by station 0.)

——————— figure 5 should be placed around here

5.2.2 U-shaped bus

This algorithm is quite simple—it consists of two parts. In the first part, every station sends one packet and measures the amount of time that elapses until it hears its own packet on the *inbound* segment. Stations broadcast in the order of their soft addresses, see 5.1.1.

In the second part, every station broadcasts the result of its measurement in part one, again, in the order of their soft addresses.

Now, every station knows the result sent by every other station. The distances from S_0 can be determined from the formula:

$$l(S_i) = (R_0 - R_i)/2$$

where R_i is the result transmitted by station S_i.

This algorithm requires sending $2N$ packets.

5.2.3 H-shaped bus

This algorithm also consists of two parts. In the first part, station S_0 sends $N-1$ packets on the LR bus. Station S_i responds to packet $i-1$ by sending on the RL bus a packet of its own. Station S_0 measures the amount of time that elapses from the moment when it sends each packet and the moment when it hears the reply. This value equals twice the propagation distance between S_0 and the responding station.

In the second part, station S_0 broadcasts the results of its measurements. The total number of packets sent is $2N-1$.

6 Summary

This paper investigates the problem of recognition of the current configuration of a bus-type network. Various bus architectures are considered—a bidirectional bus and various unidirectional busses. We have presented a collection of distributed algorithms for two special cases of this problem:

- To determine the relative ordering of all the stations with respect to one end of the bus.
- To compute the exact distances between every pair of stations.

These cases are not always simple. In particular, the second case cannot be solved for a unidirectional bus in which not all distances are symmetric. A different type of difficulty arises for the bidirectional bus: considering the imperfect accuracy of the internal clocks used by the stations, it is not always possible to order them correctly (i.e., solve the first case), although the presented algorithm gives a fuzzy solution that is equivalent to the correct one with the inaccuracy of clocks taken into account.

The algorithms can be used in a variety of different applications. Among other, they are of use when the network protocol requires knowledge about the current configuration; another application arises when two or more stations need to synchronise their internal clocks.

Although the algorithms presented here are quite efficient, it should be noted that the complexity of these algorithms is of marginal importance. The configuration recognition procedure must only be performed when something changes in the physical structure of the network, i.e. some stations are added, permanently removed, or moved to other locations. The physical cost of such operation is much more critical than the execution time of any reasonable algorithm.

References

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

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

[3] 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.

[4] 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.

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

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

[7] K. V. S. Ramarao. Distributed algorithms for network recognition problems. *IEEE Trans. on Computers*, 38(9):1240–1248, Sept. 1989.

[8] J. Schoch et al. Evolution of the Ethernet local computer network. *IEEE Computer*, Aug. 1982.

[9] F. Tobagi, F. Borgonovo, and L. Fratta. Express-net: A high-performance integrated-services local area network. *IEEE Journal on Selected Areas in Communication*, Nov. 1983.

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

Captions to figures

Figure 1: A bidirectional bus network.

Figure 2: An S-shaped unidirectional bus.

Figure 3: A U-shaped unidirectional bus.

Figure 4: An H-shaped dual unidirectional bus.

Figure 5: An S-shaped bus disguised as a ring.

Figure 1: A bidirectional bus network.

paper by W. Dobosiewicz et al.

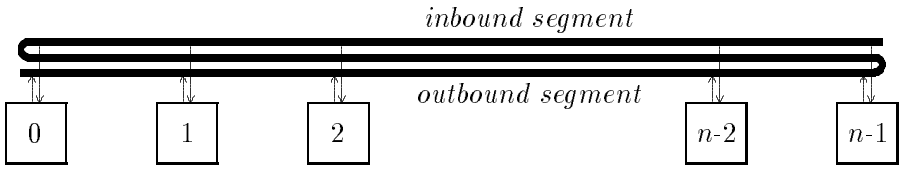

Figure 2: An S-shaped unidirectional bus.

paper by W. Dobosiewicz et al.

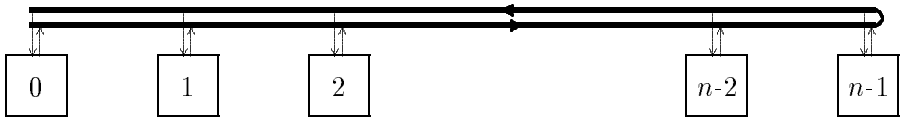

Figure 3: A U-shaped unidirectional bus.

paper by W. Dobosiewicz et al.

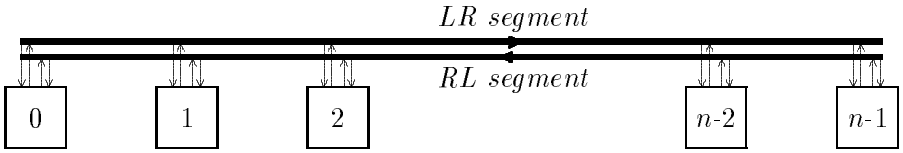

Figure 4: An H-shaped dual unidirectional bus.

paper by W. Dobosiewicz et al.

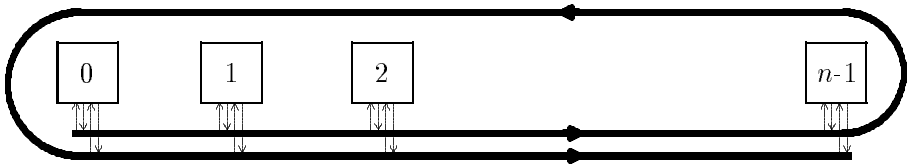

Figure 5: An S-shaped bus disguised as a ring.

paper by W. Dobosiewicz et al.