A study on the Inaccessibility Characteristics of ISO 8802/4 Token-Bus LANs*

José Rufino, Paulo Veríssimo
Technical University of Lisboa
INESC†
e-mail:...ruf@inesc.pt, paulov@inesc.pt

Abstract

Local area networks have long been established as the basis for distributed systems. Continuity of service and bounded and known message delivery latency are requirements of a number of applications, which are imperfectly fulfilled by standard LANs. Most previous studies have addressed this issue by computing worst-case access/transmission delays only for normal LAN operation.

However, LANs are subject to failures, namely partitions. Since most applications can live with temporary glitches in LAN operation, an alternative approach is to quantify all these glitches or temporary partitions, that we named inaccessibility, and derive a worst-case figure, to be added to the worst-case transmission delay in absence of faults. In these conditions, reliable real-time operation is possible on non-replicated LANs. This paper does an exhaustive study of the inaccessibility characteristics of the ISO 8802/4 token-bus LAN.

1 Introduction

Local area networks have long been established as the basis for distributed systems. The several variants of standardised LANs (ISO 8802 and FDDI) have different mechanisms to control access to the medium and recover from errors.

Continuity of service and determinism in transmission delay are requirements of a number of applications, specially in the fault-tolerance and real-time area, which are unperfectly fulfilled by these LANs, if used without special measures. A number of authors have studied problems such as priority inversion [1], probability of meeting estimated access times [2, 3], extensions for medium failure resiliency through redundancy [4], potential lack of determinism [5].

In reliable real-time systems, the fundamental requirement of communications is that there be a bounded and known message delivery latency, in the presence of disturbing factors such as overload or faults. When the requirement is very strict (eg. for life-critical applications), specialized space-redundant architectures are the solution: point-to-point graphs [6] or multiple LANs [7]. These solutions are however costly and complex.

In spite of their limitations, standard LANs are a very important design component. It is worthwhile investigating if the real-time requirement can be reliably met in local area networks that are not replicated, except for eventual medium redundancy — at the electrical signalling level. To achieve reliable real-time communication, three fundamental conditions must be validated:

1. bounded delay from request to transmission of a frame[1], given the worst case load conditions assumed;
2. message[2] delivery despite the occurrence of omission failures (eg. lost frames);
3. control of partitions.

Most of the existing studies with this regard have addressed point 1 [2, 8, 9, 10]. However, they are helpless at representing the LAN behaviour, when faults occur. In that case, it is necessary to study the patterns for omission failures (eg. number of consecutive omission failures) and for partitions.

*Paper presented at *IEEE INFOCOM'92, the Conference on Computer Communications*, Florence, Italy, May 6-8 1992, CH3133-6/92-0958 ©1992 IEEE

†Instituto de Engenharia de Sistemas e Computadores, R. Alves Redol, 9 - 6º - 1000 Lisboa - Portugal, Tel.+351-1-3155150. This work has been supported in part by the CEC, through Esprit Project 1226 - DELTA-4 and by JNICT through Programa Ciência.

[1] LAN level information packet.
[2] User level information packet.

Uncontrolled omissions and partitions are a source of asynchrony and inconsistency. This is unacceptable for most systems, let alone real-time ones. Point 2 is addressed under the scope of the LLC type 3 service for point to point interactions. However, it has been practically disregarded for broadcast or multicast interactions. One exception is a modified token-ring mechanism described in[11]. Point 3, to the authors' knowledge, has not been treated for non-replicated networks. All three points have been addressed in [12] for LANs in general.

A study of the behaviour of an ISO 8802/4 token-bus with regard to partitions is the central issue of this paper. Token-bus, given its connection with MAP is very relevant in control and automation[3], where real-time, reliability and accessibility are a must.

This study contributes to a better understanding of the ISO 8802/4 Token-Bus LAN operation having these attributes in mind.

2 Controlling Partitions

A network is partitioned when there are subsets of the nodes which cannot communicate with each other[4]. In this sense, a single LAN displays a number of causes for partition, not all of them of physical nature, like bus failure (cable or tap defect): bus contention, ring disruption, transmitter or receiver defects; token loss; etc. Some LANs have means of recovering from some of these situations, and can/should be enhanced to recover from the others, if reliable real-time operation is desired.

However, the recovery process takes time, so in the meantime the LAN is partitioned. A solution to the problem of controlling partitions was presented in [12]. It is based on a very simple idea: if one knows for how long a network is partitioned, and if those periods are acceptably short, real-time operation of the system is possible.

Let us call them periods of *inaccessibility*, to differentiate from classical partitions. The definition of inaccessibility in [13] is summarised here:

> *Certain kinds of components may temporarily* refrain *from providing service, without that having to be necessarily considered a failure. That state is called* **inaccessibility**. *It can be made known to the users of the component; limits are specified (duration, rate); violation of those limits implies permanent failure of the component.*

[3]Manufacturing Automation Protocol.

[4]The subsets may have a single element. When the network is completely down, *all* partitions have a single element, since each node can communicate with no one.

This is not hard to implement, as shown in [12]. To achieve it one must first assure that all conditions leading to partition are recovered from. For example, tolerance to one medium failure is assured in the dual FDDI ring [14]. In a token-bus network some extensions have to be implemented, since it has no standardised redundancy. For example, we devised a glitch-free method for real-time switch-over between busses of a dual-media token-bus [4].

Then, one needs to show that all the inaccessibility periods are time-bounded and determine the upper bound. The study of ISO 8802/4 token-bus inaccessibility is presented next. The scenarios described in the following sections are the inaccessibility periods foreseen in the standard specification. The figures presented illustrate the intervals in the token-bus operation when the LAN does not provide service, although not being failed.

The study yields interesting conclusions, like: a figure for the worst case duration of inaccessibility (to be added to a worst-case access time...); the existence of only one or two periods much longer than average; strategies to reduce the longest periods are possible.

3 The Token-Bus Network

The ISO 8802/4 Token-Passing Bus is a local area network (LAN) which has gathered a growing attention after it has been selected as the communication infra-structure of MAP, the Manufacturing Automation Protocol, becoming then a standard for interconnection and interworking in the factory floor [15].

Access control is performed by a token passing protocol, which establishes a logical ring over the physical bus. Access to the shared broadcast medium for data transmission is only granted to the station which currently holds the token. A timed-token mechanism is used to guarantee an upper bound on access delay to the network. This bound is unfolded with successively lower guarantees, in four classes, through priorities.

The efficiency of this scheme has been studied in the last few years, either through simulation work [16] or analytical models [2]. The performance of the token bus priority scheme [17, 18, 19] has also been studied. Analytical approximations for the mean frame delay, under different service disciplines, are proposed in [20].

Most of these studies assume that the network always operates normally, neglecting the influence of inaccessibility. However, in the presence of faults corrective actions must be performed and in consequence the operation of the logical ring is affected, sometimes severely, in the sense that it can no longer ensure the calculated frame transmission delay bound.

An analysis of the Token-Bus error handling mechanisms is provided in [21]. However, it is only qualitative. This paper makes a quantitative analysis of the 8802/4 error handling mechanisms in a comprehensive way.

Data Rate ($Mbps$)	Bit Time t_{bit} (μs)	Octect Time t_{oct} (μs)	Station Delay t_{SD} (μs)
5	0.2	1.6	11.3
10	0.1	0.8	20.9

Table 1: Station Delay for a MC68824 Token-Bus LAN controller

Network Characterization

For our purposes, two important parameters characterize the basic operation of any ISO 8802/4 Token-Bus network:

- ◇ **Data Rate** - The rate of data signalling, in the bus. It gives a meaning to the *bit* and *octect* times. A wide set of MAC protocol timers are scaled by these timing references [22].
- ◇ **Slot Time** - This parameter is set up by station management entities. It is an aggregate variable accounting for medium access control (MAC) sub-layer intrinsic performance and network size. The slot time is usually expressed in octect times, due to its formal definition, provided in [22]. In this work, for simplicity, we use its value expressed in plain time, as given by equation:

$$t_{Slot} = 2 \,.\, (t_{PD} + t_{SD}) \quad (1)$$

  - t_{PD} is the worst case end-to-end propagation delay of the physical layer. This variable accounts for the network cable propagation delay, plus the modem delays (at both transmit and receiver ends) and the regenerative repeater delay, whenever used. For the length-dependent cable propagation delay a typical value of 5 $\mu s/km$ is assumed.
  - t_{SD} is the station delay. This variable accounts for the intrinsic performance of each particular MAC VLSI implementation. The values presented in Table 1 refer to the Motorola MC68824 Token Bus Controller [23]. A typical scenario is considered: the TBC, in addition to token passing, is also performing data transmissions mixed with ring management actions.

MAC Protocol Data Units

Several types of protocol data frames are exchanged between peer MAC entities, in order to provide logical ring housekeeping functions, essential for normal ring membership management, and for the preservation of ring integrity in the presence of faults. These functions are supported through the following set of actions:

- Solicit new stations for logical ring entry.
- Find out the successor of a failed station (*who follows*).
- Solicit any station to respond at successor querying.
- Resolve contentions.
- Establish a new successor.
- Bid for token generation.
- Token passing.

The duration of all MAC protocol frames which take part in these actions, exception made to token claiming, are presented in Table 2, for an address length (l_{add}) of 48 bits. The values were computed based on frame lenght and data rate, assuming that fast synchronising modems are used. A three octect preamble, required to fulfill the minimum 2 μs preamble duration at 10 Mbps [22], is considered at both data rates.

MAC Frame	Symbol	Duration (μs)	
		$5Mbps$	$10Mbps$
header/trailer	t_{HrTr}	35.2	17.6
solicit_successor_1	t_{SS1}	35.2	17.6
solicit_successor_2	t_{SS2}	35.2	17.6
set_successor	t_{SSF}	44.8	22.4
resolve_contention	t_{RC}	35.2	17.6
token	t_{TK}	35.2	17.6
who_follows	t_{WF}	44.8	22.4

Table 2: Duration of MAC Token-Bus Protocol Data Units ($l_{add} = 48$)

4 Accessibility Constraints

In this section we will cover a set of scenarios leading to network inaccessibility. For many of them we start with very simple situations that then evolve to less restrictive – and thus more realistic – operating conditions/fault assumptions. For most of the cases, the main analysis is completely general, being particularized for best and worst cases, afterwards.

Addition Of New Stations

Each station on the logical ring periodically offers other stations the opportunity to become ring members. This operation is performed through a controlled contention process called *solicit successor procedure*

which allows non-participant stations to declare their wish of being admitted into the logical ring. The value of a special counter within the MAC sub-layer controls how often a ring member can perform management operations[5]. The process is initiated by the then-current token holding station and its operation varies slightly depending on whether or not this station is the one having the lowest address.

Let us consider in first place the case where the current token holder does not have the lowest address, in the network. In such a case, the current token holder issues a *solicit_successor_1* frame with the destination address set to its successor, waiting afterwards for possible responses during a window of one slot time.

Any station waiting to become a ring member, compares the addresses of the received *solicit_successor* frame with its own address. Only those stations whose individual address falls in the range between the addresses of the current token holder and its successor will respond to the query[6] through the issuing of a *set_successor* frame. Three distinct situations may then occur:

- **No station answers** - In this case no *set_successor* frame is received and the station simply passes the token to its former successor, after a one-slot time delay.
- **Exactly one station answers** - Upon the receipt of the *set successor* frame the station updates its *next_station* [22] variable and passes the token to this new successor. After receiving the token, the new ring member can start using it.
- **More than one station answer** - Contention arises when multiple stations simultaneously answer to the *solicit_successor* query and, in consequence, only unrecognizable noise may be heard during the response period. Contention is detected by the *solicit successor sender* and resolved through an arbitration algorithm which we will describe ahead. Once contention is resolved, ring management operations proceed, with the actions described in the previous case.

The first inaccessibility situation thus occurs when the MAC sub-layer, performs the actions required for the addition of a new station. The duration of the resulting inaccessibility period is given by equation [7]:

$$t_{ina \leftarrow join1} = t_{SD} + t_{SS1} + t_{Slot} + t_{rcp} \qquad (2)$$

The exact duration of the *resolve contention process* – t_{rcp} – depends on whether or not a contention between stations occurs and on how quickly this contention is resolved. Contention is detected by the *solicit successor sender* and its resolution is started by issuing a *resolve_contention* frame. This action opens four response windows. The contending stations start to use the first two bits of their station addresses, in order to establish which of the four response windows they will use to respond, with a *set_successor* frame. The *resolve responder algorithm* schedules stations with higher $TwoBit$ values to respond sooner. Any station that hears bus activity, before issuing its response, withdraws from the contention process. If a station hears no activity until the moment when it should issue the response, it transmits the *set_successor* frame. The process ends when the token holder, at the end of a contending round, detects a valid *set_successor* frame. A contending station detects that it has won the contention when it receives the token. The *resolve responder algorithm* is won by the first station heard without errors[8], usually the contending station with the highest address, and the average time spent in its resolution can be estimated by the corresponding line in equation (3), where we have considered that responses are only received near the end of the fourth slot time during 1/4 of the rounds[9]:

$$t_{rcp} = \begin{cases} 0 & \text{no response} \\ t_{SSF} & \text{no contention} \\ \sum_{nb_{rounds}} (t_{RC} + 4\,.\,t_{Slot} + \frac{t_{SSF}}{4}) & \text{contention} \end{cases} \qquad (3)$$

An upper bound can be placed on t_{rcp}. This bound, that we signal with superscript ^{wc}, is obtained by considering that competing stations, with addresses differing only in the last two bits, systematically respond in the fourth window to the resolve contention request:

[5]The *inter_solicit_counter* [22] is decremented one unit in each token rotation; when it reaches zero, ring management operations can begin, if there is available bandwidth, i.e. provided that the time elapsed since the last token visit is lower than the associated *target token rotation time* [22]. This can inhibit a large number of stations from performing ring management during a single token rotation.

[6]This restriction aims at preserving the descending order organization of the logical ring. It also serves to reduce the number of possible contenders.

[7]The value of t_{SD}, accounting the overhead needed by a station to enter in ring maintenance, may represent a slightly pessimistic upper bound. Its straight utilization however, does not significantly influence overall computations. It simply aims to avoid the introduction of network parameters not defined in [22].

[8]This is the way the standard specification is written. However, a conformant station may use, instead, any correctly received *set_successor* frame [22].

[9]Thus spreading out from the four window period.

$$t_{rcp}^{wc} = \sum_{i=1}^{(l_{add}/2)} (t_{RC} + 4\,.\,t_{Slot} + t_{SSF}) \qquad (4)$$

Let us now consider the scenario where the station which triggers the *solicit successor procedure* is the one having the lowest address in the network. It is the only station where its own address is lower than its successor's. In consequence, a slightly different ring management procedure is required, so that both stations with addresses below its own and stations with addresses above the highest one may enter. The differences are only in the way the process is initiated:

— The station with the lowest address begins the *solicit successor procedure* by transmitting a *solicit_successor_2* frame addressed to its successor, i.e. the station with the highest address in the network, and waits for possible responses during two consecutive windows.

— Stations with addresses below the token holder respond in the first window.

— For stations with addresses above the token holder successor the issuing of possible responses is delayed by one slot time. If these stations hear no activity in the bus during this period, they issue their responses during the second window. Otherwise, they abandon the process.

The remaining aspects of the process, including the resolution of possible contentions, do not differ from the previous case and therefore the expression for the inaccessibility time is given by:

$$t_{ina \leftarrow join2} = t_{SD} + t_{SS2} + 2\,.\,t_{Slot} + t_{rcp} \qquad (5)$$

The upper bound for the inaccessibility time, due to the addition of a single station into the logical ring, can be easily obtained from the previous equations:

$$t_{ina \leftarrow join}^{wc} = t_{SD} + t_{SS2} + 2\,.\,t_{Slot} + \frac{l_{add}}{2}\,(t_{RC} + 4\,.\,t_{Slot} + t_{SSF}) \qquad (6)$$

The *solicit successor procedure* always adds new stations to the logical ring in a one by one basis. Requests for logical ring entry, issued by several stations in the same response window, beyond leading to a contention process, also originate a compound set of actions where the various ring entries may or may not be sequenced in the same token rotation. This is a complex process whose thorough analysis will be performed next.

Multiple Joins

When a station joins the logical ring through the process described in the previous scenario two important actions, decisive for subsequent management operations, are performed upon ring entry:

- The *inter_solicit_counter* of the new ring member is set to zero.
- Its *token rotation timer for ring maintenance* is set to a special value, furnished by station management entities, known as *ring maintenance timer initial value* [22].

After receiving the token the new ring member checks all its user access classes, looking for any enqueued traffic, and finally checks whether or not it should open a window for ring management. Since the *inter_solicit_counter* has been set to zero on ring insertion, this decision will only be affected by the availability of network bandwidth.

Nevertheless, each station is provided with the aforementioned dedicated facilities for bandwidth control upon ring entry and, in consequence, it can be parameterized either to immediately start a new *solicit successor procedure* or to postpone it until a forthcoming opportunity. In this latter case multiple competing stations will only be admitted, one at a time, when there is available bandwidth.

Conversely, when a station is parameterized to start a *solicit successor procedure* upon ring entry, all the competing stations will be admitted in the same token rotation, with each new member immediately promoting the admission of a new successor, until all the stations have succeeded in entering the logical ring. The normal token rotation is thence slowed down by the amount of time given in equation (7).

$$t_{ina \leftarrow mjoin} = \sum_{N_{st \rightarrow join}+1} t_{ina \leftarrow joinx} \qquad (7)$$

In this expression, the index under the sum symbol represents the number of ring management rounds performed upon the addition of $N_{st \rightarrow join}$ stations to the logical ring. The value of $t_{ina \leftarrow joinx}$ will be given by equation (5) as long as the station leading the process is the one with the lowest address in the network. Otherwise, it will be expressed by equation (2).

In a network with N_{st} active stations, allowing a maximum number of N_{max} stations, the time spent for the addition of all the remaining $(N_{max} - N_{st})$ stations, in the same response window, is upper-bounded

by equation (8). This expression accounts for the duration of the first $(N_{max} - N_{st} - 1)$ contending rounds, the duration of the last station join and also the time spent by this station in the final opening of a window to which no station will respond. This result is thus slightly different from the one which can be obtained for $(N_{max} - N_{st})$ independent joins[10].

$$\begin{aligned} t^{ub}_{ina \leftarrow mjoin} &= (N_{max} - N_{st} - 1) \,.\, t^{wc}_{ina \leftarrow join} + \\ &\quad t_{SD} + t_{SS2} + 2 \,.\, t_{Slot} + t_{SSF} + \\ &\quad t_{SD} + t_{SS2} + 2 \,.\, t_{Slot} \\ &= (N_{max} - N_{st} - 1) \,.\, t^{wc}_{ina \leftarrow join} + \\ &\quad 2 \,.\, (t_{SD} + t_{SS2}) + 4 \,.\, t_{Slot} + t_{SSF} \end{aligned} \quad (8)$$

The function given by equation (8) decreases with increasing N_{st}. Therefore the worst-case scenario is obtained when $N_{st} = 2$ and a massive join demand occurs[11], although such a situation will only arise in the power-up of the given set of stations. The time spent in the addition of those stations to the logical ring is expressed by equation:

$$\begin{aligned} t^{wc}_{ina \leftarrow mjoin} &= (N_{max} - 3) \,.\, t^{wc}_{ina \leftarrow join} + \\ &\quad 2 \,.\, (t_{SD} + t_{SS2}) + 4 \,.\, t_{Slot} + t_{SSF} \end{aligned} \quad (9)$$

The best case scenario, for multiple station joins, is obtained when two stations enter the logical ring, in the same maintenance opportunity, without contention. Contention will only be avoided if one of the joining stations has an address smaller than the lowest station already in the network, while the other one has an address greater than the highest station. The station with the smaller address responds first and enters the logical ring without contention, promoting afterwards the entrance of the station with the higher address. The time spent in these operations is given by equation (10).

$$\begin{aligned} t^{bc}_{ina \leftarrow mjoin} &= 2 \,.\, t_{ina \leftarrow join2}(no\_contention) + \\ &\quad t_{ina \leftarrow join1}(no\_response) \end{aligned} \quad (10)$$

[10]The occurrence of a high number of independent joins, in the same token rotation, is not likely to happen. In part this is due to the limit usually placed on the available network bandwidth; in part it is because a small randomizer is used within MAC to dither the *max_inter_solicit_counter* [22] from its nominal value, once every $50ms$ or after its use, which minimizes the clustering of windows opening in the same token rotation.

[11]These results are derived in the assumption that a logical ring already exists.

Station Leave

Stations may leave the logical ring either in an abrupt or orderly way. Abrupt leaves usually result from station failure and they will be dealt with in coming scenarios.

An orderly leave from a logical ring is only possible when that station holds the token. Station withdrawal is achieved through a ring patch between its predecessor and successor stations. For that purpose, the leaving station before passing the token issues a *set_successor* frame, addressed to its predecessor, and carrying the address of its future successor. The inaccessibility time due to the leave operation is very short and it is simply given by:

$$t_{ina \leftarrow leave} = t_{SD} + t_{SSF} \quad (11)$$

Multiple Leaves

Station leaves can always be executed whether or not there is bandwidth available for ring management operations. Therefore, if several stations want to abandon the logical ring in the same token rotation they are always allowed to do it. This means that the normal token rotation time will be slowed down by the time spent in these actions, which is given by:

$$t_{ina \leftarrow mleave} = N_{st \rightarrow leave} \,.\, t_{ina \leftarrow leave} \quad (12)$$

where $N_{st \rightarrow leave}$ represents the number of stations that will abandon the logical ring in the current token rotation. Its value is upper bounded by $(N_{st} - 2)$ and therefore the worst-case value for multiple station leaves can be obtained with equation (13).

$$t^{wc}_{ina \leftarrow mleave} = (N_{st} - 2) \,.\, t_{ina \leftarrow leave} \quad (13)$$

No Successor

A station holding the token may transmit during an allowed period of time. Afterwards it should pass the token to its successor. If this station is failed, the token passing operation will not succeed. After the token pass checking period (one slot time) has elapsed a recovery strategy is tried. This includes the repetition of token transmission (checked during one more slot time) followed by a variant of the *solicit successor procedure*. During this *who follows* query the current token holder waits for a *set_successor* frame, until the end of a three-slot-time period. Under a single station failure assumption, the current token holder will

receive, within this period, a response from the successor of the failed station. This establishes a new successor for the current token holder and ends the recovery procedures. The time spent in performing these actions is given by:

$$t_{ina \leftarrow nosuc} = t_{SD} + 2 \, . \, t_{TK} + t_{WF} + 5 \, . \, t_{Slot} + t_{SSF} \quad (14)$$

Token Loss

In the previous scenario we have considered that a station is not holding the token when it fails. We now consider that a station fails while it is holding the token.

Token losses are detected by monitoring the absence of bus activity and recovered through a *claim token process* [22]. The duration of the whole process is given by:

$$t_{ina \leftarrow tkloss} = t_{BIdle} + t_{tcp} \quad (15)$$

- The first term on this expression represents the time elapsed between the loss of the token and the beginning of the token claim process. It corresponds to the value loaded into the *Bus Idle Timer*. Usually this timer assumes the value of seven slot times. The exception is the station having the lowest address in the network, where the *Bus Idle Timer* is loaded with only six slot times[12].

$$t_{BIdle} = \begin{cases} 6 \, . \, t_{Slot} & \text{lowest station present} \\ 7 \, . \, t_{Slot} & \text{lowest station failed} \end{cases} \quad (16)$$

- The second term represents the duration of the token claim process. Any station where the *Bus Idle Timer* expires, initiates a recovery procedure by issuing a *claim_token* frame whose length depends on the first two bits of the station address. After issuing its request the sending station waits during one slot time. If at the end of this period the station detects activity in the bus, then it drops out from the *token claim process*. If no activity is detected and there are unused bits in the address string, the station repeats the process using the next two address bits. The process ends after the winning station has used all the bits of its address, plus two randomly chosen bits.

  The time spent in these actions is given by equation (17), for stations having an address length of l_{add} bits. The first $l_{add}/2$ iterations use the station address bits, to define the length of the *claim_token* frames. The last iteration employs a random two-bit value.

$$t_{tcp} = \sum_{i=1}^{(l_{add}/2)+1} (t_{CF}(i) + t_{Slot}) \quad (17)$$

  - $t_{CF}(i)$, represents the duration of a *claim_token* frame issued in round i, and is given by:

$$t_{CF}(i) = t_{HdTr} + 2 \times TwoBit_{add}(i) \times t_{Slot}$$

    where t_{HdTr} stands for the duration of the MAC header/trailing sequence. The $TwoBit_{add}$ value can range from zero to three.

When the lowest address station is not present in the network, probably due to its failure, the *address sort algorithm* used by the *token claim process* always leads to the selection of the station with the highest address as the token user. The worst-case value can then be obtained by considering that the station which wins the contention has the highest possible address:

$$t^{wc}_{ina \leftarrow tkloss} = 7 . t_{Slot} + (\frac{l_{add}}{2} + 1) \, (t_{HrTr} + 7 . t_{Slot}) \quad (18)$$

The best-case value can also be obtained, considering that a station with all its address bits set to zero is present and active. The recovery time, from a token loss, will then be given by:

$$t^{bc}_{ina \leftarrow tkloss} = 6 \, . \, t_{Slot} + (\frac{l_{add}}{2} + 1) \, (t_{HrTr} + t_{Slot}) \quad (19)$$

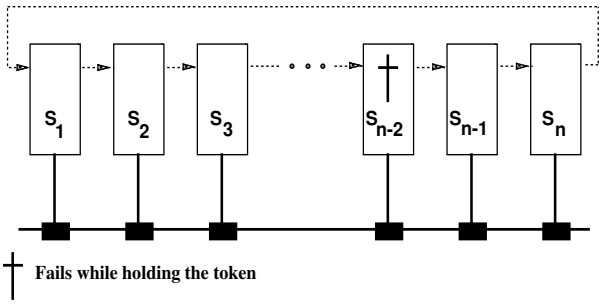

Figure 1: Side-effects of Token Loss Recovery

Let us now analyze some side effects that token recovery may have on the normal token circulation. Consider the token bus network, whose logical ring is pictured in Fig. 1. Station $\mathcal{S}_1$ has the highest address, while station $\mathcal{S}_n$ has the lowest one. Additionally, consider that station $\mathcal{S}_{n-2}$ fails while holding the token.

[12] This procedure aims at avoiding collisions during the *token claim process* by allowing one station to start it in isolation. Since the station processing the event is the one having the lowest address, the generation of a new token will be performed more quickly, as we will see ahead.

The claiming of a new token is started after the bus idle timer has expired in some station, and since the station with the lowest address on the network is not failed, the token claim process will be initiated and won by this station.

Station $\mathcal{S}_{n-1}$, which was about to receive the token when the failure of its predecessor occurred, is passed over. So, in addition to the token recovery time, station $\mathcal{S}_{n-1}$ must wait an extra token rotation to get access to the network. This means that different stations in the network may have a different view of inaccessibility and that, as a general rule, the inaccessibility time may not be merely represented by the time required to recover the token. On the other hand its worst-case value, as given by equation (18), must be consolidated with the addition of the network access delay upper-bound.

The token rotation which immediately follows the token recovery process is of fundamental importance for network operation, even if there are no skipped stations. Depending on their values, some or all of the lower priority *Target Token Rotation Timers* may have expired during the recovery process and if so, no bandwidth is available for those access classes. The first token rotation will then restart the *Target Token Rotation Timers*. The access to the network, immediately after token recovery, is thus only assured for the highest priority access class[13]. During this token rotation, a second inaccessibility period, corresponding to a *no successor* scenario, will occur. It is a direct consequence of the primary failure and arises when the predecessor of the failed station tries to pass it the token.

Multiple Failed Stations

So far, we have considered that only one station in the network fails at a given time. Let us now consider a less restrictive scenario by allowing the failure of multiple stations. To be compliant with our initial single failure assumption we will consider that all these multiple failures happen at the same time, due to some common cause. Common mode failures are realistic enough to be taken into account. Consider, for instance, a network where several stations are connected to the same power supply line. A failure or a shutdown in this line will bring all these stations down.

Let us, for the moment, additionally assume that none of the failed stations are adjacent in the token-passing order. In consequence all the failures can be recovered through successive *who follows* queries. Between each recovery action non-failed stations may transmit data, provided there is available bandwidth. This means that inaccessibility periods may alternate with the flow of data in the network. As in the previous case the view that ring members get of inaccessibility is not consistent among all the stations.

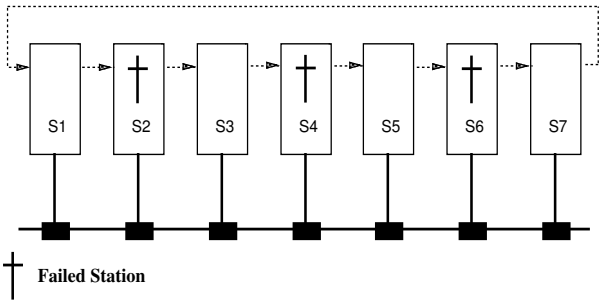

Figure 2: Example of a multi-failure scenario

The current token rotation is slowed down by the amount of time needed to perform all the *who follows* queries. The time spent in these successive queries grows linearly with the number of failed stations and is given by equation (20).

$$t_{ina \leftarrow mfail} = N_{st \rightarrow fail} \cdot t_{ina \leftarrow nosuc} \qquad (20)$$

In the worst-case, failed and non-failed stations are completely intermixed, as shown in the example of Fig. 2. The time spent in the recovery actions is given by:

$$t^{wc}_{ina \leftarrow mfail} = \lfloor \frac{N_{st}}{2} \rfloor \cdot t_{ina \leftarrow nosuc} \qquad (21)$$

where $\lfloor\ \rfloor$ represents the *floor* function[14].

Station Group Fail

We now relax our last restriction, by allowing the simultaneous failure of adjacent stations in the token-passing order. If this is the case, the previously described process of looking for a new successor fails, since there will be no answer to the *who follows* query. After one repetition of the *who follows* query, checked during a three-slot time period, another variant of the

[13]The 8802/4 MAC protocol guarantees that any station may hold the token during a fixed amount of time, for servicing the highest priority class. Service of lower priority ones, including the management of station joins, can only be performed if the time elapsed since the last token visit is lower than the *target token rotation time* associated with that access class. Details on this mechanism can be found in [22].

[14]The *floor* function $\lfloor x \rfloor$ is defined as the greatest integer not greater than x.

solicit successor procedure is started. The token holder begins the *solicit any procedure* with the transmission of a *solicit_successor_2* frame addressed to itself. This enables any active station to respond to the query, during two consecutive slot time windows.

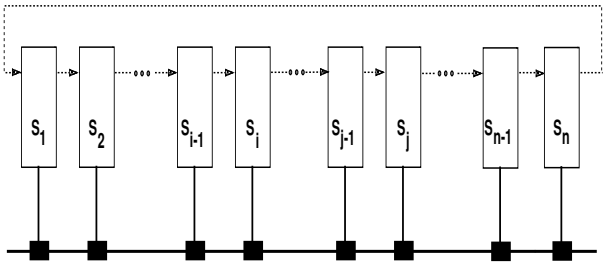

Figure 3: Token-Bus network with identified groups

To thoroughly analyze the subset of scenarios resulting from the failure of a group of stations, let us consider the network presented in Fig 3. Between stations $\mathcal{S}_2$ and $\mathcal{S}_{i-1}$ there can be any number of stations, as well as between $\mathcal{S}_i$ and $\mathcal{S}_{j-1}$, and also between $\mathcal{S}_j$ and $\mathcal{S}_{n-1}$. The following situations may then happen:

i) *The failed group of stations is located at the ring edges*, i.e. stations $\mathcal{S}_1$ to $\mathcal{S}_{i-1}$ or stations $\mathcal{S}_i$ to $\mathcal{S}_n$ have failed. In both cases the *solicit any procedure* is performed by the active station with the lowest address. This means that all the presumable responders will have addresses above the *solicit successor sender* and therefore they will delay their *set_successor* frame transmissions until the second window. The process ends with the election of the non-failed station with the highest address as the new successor, for the current token holder.

ii) *The failed group of stations is far from the ring edges*, i.e. stations $\mathcal{S}_i$ up to $\mathcal{S}_{j-1}$ have failed. The *solicit any procedure* is started when station $\mathcal{S}_{i-1}$ is trying to pass the token. In this case, stations with addresses above and below the *solicit successor sender* are simultaneously present. Stations with addresses below the *solicit successor sender* respond in the first window and the one with the highest address is chosen, i.e. station $\mathcal{S}_j$, which was the successor of the last failed station.

iii) *The recovery process is initiated by the highest address station*, i.e. stations $\mathcal{S}_2$ up to $\mathcal{S}_{i-1}$ have failed. In this case all the stations in the network, whether failed or not, have addresses below the *solicit successor sender*. Responses to the *solicit any* query are all issued in the first window. The process is won by the highest address responder ($\mathcal{S}_i$ in the given example), i.e. by the successor of the last failed station, in the former logical ring.

iv) *The failed group of stations immediately precedes the lowest address station*, i.e. stations $\mathcal{S}_j$ up to $\mathcal{S}_{n-1}$ have failed. The *solicit any procedure* is started when station $\mathcal{S}_{j-1}$ is about to pass the token. In this case, there will be only one station in the network with its address below the solicit sender. Contention is avoided and in consequence, logical ring restoration will be performed very fast. Therefore, this situation corresponds to the best case scenario of station group failure.

The time spent in the recovery process just described – which does not present disturbing effects, like the appearance of passed over or skipped stations – is given by equation (22). In this equation t_{rcp} represents the duration of an eventual contention process, as given by equation (3).

$$t_{ina \leftarrow gfail} = t_{SD} + 2\,.\,(t_{TK} + t_{WF}) + 10\,.\,t_{Slot} + t_{SS2} + t_{rcp} \quad (22)$$

In the worst-case, the recovery of the logical ring for a single failed group of stations can last as long as:

$$t^{wc}_{ina \leftarrow gfail} = t_{SD} + 2\,.\,(t_{TK} + t_{WF}) + 10\,.\,t_{Slot} + t_{SS2} + t^{wc}_{rcp} \quad (23)$$

The best case scenario only occurs when the lowest addressed station immediately follows the last failed station, as previously mentioned. The inaccessibility time is obtained directly from equation (22), making $t_{rcp} = t_{SSF}$.

Multiple Failed Groups

It is now time to generalize our previous results on multiple failures, by allowing the existence of several groups of failed stations. In this case, the overall inaccessibility time can be obtained directly from the previous results on the single group failure, through its multiplication by the number of failed groups (N_{gfail}):

$$t_{ina \leftarrow mgfail} = N_{gfail}\,.\,t_{ina \leftarrow gfail} \quad (24)$$

When all the failed groups have the same number of stations, i.e. the same group dimension (Gf_{dim}), equation (24) can be written in the form:

$$t_{ina \leftarrow mgfail} = \left\lfloor \frac{N_{st}}{Gf_{dim} + 1} \right\rfloor . t_{ina \leftarrow gfail} \quad (25)$$

The worst-case scenario is obtained with the highest possible number of failed groups, which for a given

set of active stations occurs when all the failed groups present the same minimum dimension, i.e. $Gf_{dim} = 2$.

$$t^{wc}_{ina \leftarrow mgfail} = \left\lfloor \frac{N_{st}}{3} \right\rfloor \cdot t^{wc}_{ina \leftarrow gfail} \qquad (26)$$

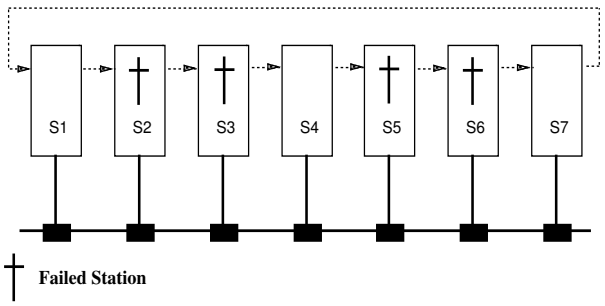

Figure 4: Example of a multi-group failure scenario

Analytic Results

In order to complete our study of networking accessibility we now evaluate the inaccessibility time bounds, for a given Token-Bus network, for example, an industrial environment with a small cell network for real-time manufacturing control. The network length $C_l = 500m$ and the total number of stations $N_{max} = 32$. The results of the evaluation, for each one of the studied scenarios, are summarized in Table 3.

The worst case figures are rather high, like for example $140ms$ for the multiple joins case in the $5Mbps$ bus. However, note that some of the situations either: occur only during start-up of the network; or can be avoided through station configuration (ex. non-opening of windows by just-arrived stations); or (in certain settings) allow more restrictive assumptions about multiple failures. This brings some of the higher figures drastically down.

The study is interesting on the grounds that it provides a basis to know what to expect from token-bus performance in the presence of failures, and provides guidance on how to improve the situation and *justifiably* achieve better performability , and thus better respect any bounded delay requirements.

Data Rate - $5Mbps$					
t_{SD} (μs)	t_{Slot} (μs)	*Scenario*		t_{ina} (ms)	
				min.	*max.*
11	27	*Station Join*	No Joins	0.073	0.100
			No Contention	0.118	0.145
			Contention	0.382	4.612
		Multiple Joins ($N_{max} = 32$)		0.363	139.999
		Station Leave		0.056	
		Multiple Leaves ($N_{st} = 32$)		0.112	1.674
		No Successor		0.306	
		Token Loss		1.717	5.794
		Multiple Fails ($N_{st} = 32$)		0.612	4.896
		Station Group Fail		0.521	5.176
		Multi-Group Fails ($N_{st} = 32$)		5.697	51.762
Data Rate - $10Mbps$					
t_{SD} (μs)	t_{Slot} (μs)	*Scenario*		t_{ina} (ms)	
				min.	*max.*
21	47	*Station Join*	*No Joins*	0.086	0.133
			No Contention	0.108	0.155
			Contention	0.508	5.605
		Multiple Joins ($N_{max} = 32$)		0.396	162.821
		Station Leave		0.043	
		Multiple Leaves ($N_{st} = 32$)		0.087	1.302
		No Successor		0.336	
		Token Loss		1.897	8.994
		Multiple Fails ($N_{st} = 32$)		0.672	5.376
		Station Group Fail		0.611	6.289
		Multi-Group Fails ($N_{st} = 32$)		6.900	62.886

Table 3: Token-Bus Inaccessibility Times ($l_{add} = 48$)

5 Conclusions

To achieve reliable real-time operation of a local area network, a bounded delay requirement must be met. Most previous studies have addressed this issue by computing worst-case access/transmission delays only for normal LAN operation.

However, achieving the bounded delay requirement means, amongst other factors, ensuring continuity of service. LANs are subject to failures, namely partitions: if these are not controlled, the above mentioned requirement is not met. While LAN replication is a solution, it is costly and complex. Some applications can live with temporary glitches in LAN operation, so an alternative approach is to quantify all these glitches or temporary partitions, which we have named *inaccessibility* periods, and derive a worst-case figure, to be added to the worst-case transmission delay expected in the absence of faults.

In these conditions, reliable real-time operation is possible on non-replicated LANs, through appropriate techniques[12].

This paper does an exhaustive study of the inaccessibility characteristics of the ISO 8802/4 token-bus LAN. To the authors' knowledge, no previous study addressed the problem of temporary LAN partitioning in this comprehensive way. The figures derived are thus corrective terms for computing transmission delays in various situations.

References

[1] Jeffery H. Peden and Alfred C. Weaver. The utilization of priorities on token ring networks. In *Proceedings of the 13th Conference on Local Computer Networks*, Minneapolis, USA, October 1988.

[2] D. Janetzky and K. Watson. Token bus performance in MAP and PROWAY. In *Proceedings of the IFAC Workshop on Distributed Computer Protocol System*, 1986.

[3] Marjory J. Johnson. Proof that timing requirements of the FDDI token ring protocol are satisfied. *IEEE Transactions on Communications*, 35(6), June 1987.

[4] Paulo Veríssimo. Redundant media mechanisms for dependable communication in token-bus LANs. In *Proceedings of the 13th Local Computer Network Conference*, Minneapolis-USA, October 1988. IEEE.

[5] Gerard LeLann. Critical issues in distributed real-time computing. In *Proceedings of the Workshop on communication networks and distributed operating systems within the space environment*. European Space Research and Technology Centre, October 1989.

[6] L. Lamport, R. Shostak, and M. Pease. The Byzantine Generals Problem. *ACM Transactions on Prog. Lang. and Systems*, 4(3), July 1982.

[7] Flaviu Cristian. Synchronous atomic broadcast for redundant broadcast channels. Technical report, IBM Almaden Research Center, 1989.

[8] R.Mangala Gorur and Alfred C. Weaver. Setting target rotation times in an IEEE Token Bus network. *IEEE Transactions on Industrial Electronics*, 35(3), August 1988.

[9] D. Dykeman and W. Bux. An investigation of the FDDI media-access control protocol. In *Proceedings of the EFOC/LAN Conference*, Basel, Switzerland, June 1987.

[10] Raj Jain. Performance analysis of FDDI token ring networks: effect of parameters and guidelines for setting TTRT. In *Proceedings of the ACM-SIGCOM'90 Symposium*, Philadelphia-USA, September 1990.

[11] C. Guerin, H. Raison, and P. Martin. Procedé de diffusion sûre de messages dans un anneau et dispositif permettant la mise en oeuvre du procedé. *French Patent*, (85.002.02), January 1985.

[12] P. Veríssimo, J. Rufino, and L. Rodrigues. Enforcing real-time behaviour of LAN-based protocols. In *Proceedings of the 10th IFAC Workshop on Distributed Computer Control Systems*, Semmering, Austria, September 1991. IFAC.

[13] P. Veríssimo and José A. Marques. Reliable broadcast for fault-tolerance on local computer networks. In *Proceedings of the 9th Symposium on Reliable Distributed Systems*, Huntsville, Alabama-USA, October 1990. IEEE. Also as INESC AR/24-90.

[14] X3T9.5 FDDI. *FDDI documents: Media Access Layer, Physical and Medium Dependent Layer, Station Mgt.*, 1986.

[15] *Manufacturing Automation Protocol Specification V2.1*, March 1985.

[16] Jade Y. Chien. *Performance Analysis of the 802.4 Token Bus Media Access Control Protocol*. IEEE - 802.4 working paper, September 1984.

[17] M. Colvin and A. Weaver. Performance of single access classes on the IEEE 802.4 Token Bus. *IEEE Transactions on Communications*, 34(12):1253–1255, December 1986.

[18] A. Jayasumana. Comment on "performance of single access classes on the IEEE 802.4 Token Bus. *IEEE Transactions on Communications*, 36(2):224–225, February 1988.

[19] A. Jayasumana. Throughput analysis of the IEEE 802.4 priority scheme. *IEEE Transactions on Communications*, 37(6):565–571, June 1989.

[20] On-Ching Yue and C. Brooks. Performance of the timed token scheme in MAP. *IEEE Transactions on Communications*, 38(7):1006–1012, July 1990.

[21] Thomas L. Phinney and George D. Jelatis. Error Handling in the IEEE 802 Token-Passing Bus LAN. *IEEE Journal on Selected Areas in Communications*, 1(5), November 1983.

[22] *ISO DIS 8802/4-85, Token Passing Bus Access Method*, 1985.

[23] Motorola. *MC68824 Token Bus Controller Data Sheet*, January 1987.