An Investigation on the Inaccessibility Characteristics of Standard Token-Based LANs

INESC Technical Report RT 26-92

J. Rufino, P. Veríssimo

April 1992

LIMITED DISTRIBUTION NOTICE

This report may have been submitted for publication outside Inesc. In view of copyright protection in case it is accepted for publication, its distribution is limited to peer communications and specific requests.

An Investigation on the Inaccessibility Characteristics of Standard Token-Based LANs

José Rufino, Paulo Veríssimo
Technical University of Lisboa
IST/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 assumed only a normal LAN operation in the derivation of worst-case access/transmission delays.

However, LANs are subject to failures, namely partitions. Since most applications can live with temporary glitches, reliable real-time operation is possible on non-replicated LANs, provided that these temporary partitions are time-bounded. We call these periods of inaccessibility, to differentiate from classical partitions.

This work investigates the inaccessibility characteristics of a given set of standardised token-based local area networks, namely ISO 8802/4 Token-Bus, ISO 8802/5 Token-Ring and ISO 9314 FDDI. This study allows, for each one of the mentioned LANs, the derivation of a worst-case inaccessibility figure, to be added to the worst-case transmission delay in the absence of faults. The study is interesting on the grounds that it provides a basis to predict LAN performance in the presence of failures, and provides guidance on how to improve the inaccessibility time figures.

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 imperfectly fulfilled by these LANs, if used without special measures. A number of authors have studied problems such as priority inversion [37], probability of meeting estimated access times [23, 29], extensions for medium failure resiliency through redundancy [51], potential lack of determinism [33].

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 [30] or multiple LANs [5].

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

*Instituto de Engenharia de Sistemas e Computadores, R. Alves Redol, 9 - 6º - 1000 Lisboa - Portugal, Tel.+351-1-3100000. This work has been supported in part by Junta Nacional de Investigação Científica e Tecnológica (JNICT) through Programa Ciência.

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 [23, 16, 6, 22]. 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. 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[17]. Point 3, when regarding non-replicated networks, and to the best of authors' knowledge, only recently has deserved some attention.

All three points have been addressed in [50] for LANs in general, while point 3 has been specifically addressed in [40] for Token-Bus LANs. Token-bus, given its connection with MAP, is very relevant in control and automation[3], where all these points assume a particular importance.

In this paper we review our original study on token-bus error handling performance and extend that study to a set of standardised token-ring networks, namely the ISO 8802/5 Token-Ring and the ISO 9314 *Fiber Distributed Data Interface* (FDDI). This study contributes therefore to a better understanding of these LANs operation with regard to their real-time, reliability and accessibility attributes.

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 [50]. 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 [49] 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.*

[1] LAN level information packet.
[2] User level information packet.
[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 [50]. To achieve it one must first assure that all conditions leading to partition are recovered from. For example, tolerance to one medium failure is directly assured in the dual FDDI ring [15]. A similar approach has been envisaged in [1] for the IBM Token-Ring, which is similar to the ISO 8802/5 Token-Ring. 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 buses of a dual-media token-bus [51].

Then, one needs to show that all the inaccessibility periods are time-bounded and determine the upper bound. Three relevant case studies, comprehending the aforementioned standardised token-based networks, are presented next. The scenarios described are essentially the inaccessibility periods foreseen in the corresponding standard specifications. The figures presented illustrate the intervals in the network 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 some periods much longer than average; strategies to reduce the longest periods are possible.

3 Case Study: Token-Bus

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 [34].

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 [3] or analytical models [23]. The performance of the token bus priority scheme [4, 24, 25] has also been studied. Analytical approximations for the mean frame delay, under different service disciplines, are proposed in [52].

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[38]. However, it is only qualitative. A comprehensive quantitative analysis of the 8802/4 error handling mechanisms is the central issue of this case study.

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 [19].
- ◇ **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 [19]. In this work, for simplicity, we use its value expressed in plain time, as given by equation:

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

  - 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 [46]. A typical scenario is considered: the TBC, in addition to token passing, is also performing data transmissions mixed with ring management actions.

MAC Frame	Symbol	Duration (μs)	
		$5Mbps$	$10Mbps$
frame header/trailing	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 ($l_{add} = 48$)

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 length 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 [19], is considered at both data rates.

Token-Bus 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* [19] 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

[5]The *inter_solicit_counter* [19] 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* [19]. 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.

given by equation [7]:

$$t_{ina \leftarrow join1} = t_{SD} + t_{SS1} + t_{Slot} + t_{rcp} \tag{2}$$

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]:

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

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:

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

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.

[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 [19].

[8] This is the way the standard specification is written. However, a conformant station may use, instead, any correctly received *set_successor* frame [19].

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

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:

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

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:

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

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* [19].

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).

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

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_{ST} stations, the time spent for the addition of all the remaining $(N_{ST} - N_{st})$ stations, in the same response window, is

upper-bounded by equation (8). This expression accounts for the duration of the first $(N_{ST}-N_{st}-1)$ 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 $(N_{ST} - N_{st})$ independent joins[10].

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

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:

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

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).

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

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:

[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* [19] 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.

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

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).

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

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:

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

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* [19]. The duration of the whole process is given by:

$$t_{ina \leftarrow tkloss} = t_{BIdle} + t_{tcp} \tag{15}$$

- 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].

$$t_{Bidle} = \begin{cases} 6 \,.\, t_{Slot} & \text{lowest station present} \\ \\ 7 \,.\, t_{Slot} & \text{lowest station failed} \end{cases} \tag{16}$$

- 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.

[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 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.

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

– $t_{CF}(i)$, 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:

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

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:

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

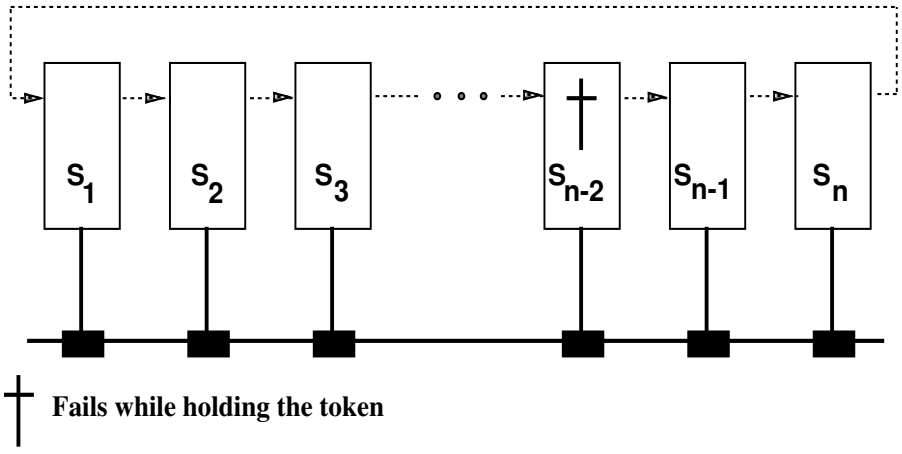

Figure 1: Side-effects of Token Loss Recovery

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. 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.

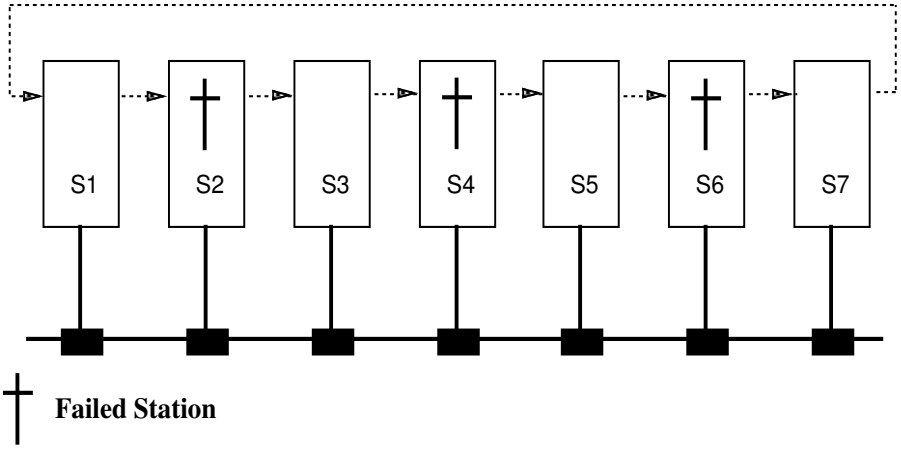

Figure 2: Example of a multi-failure scenario

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.

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

[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 [19].

failed stations and is given by equation (20).

$$t_{ina \leftarrow mfail} = N_{st \rightarrow fail} \, . \, t_{ina \leftarrow nosuc} \tag{20}$$

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:

$$t^{wc}_{ina \leftarrow mfail} = \lfloor \frac{N_{st}}{2} \rfloor \, . \, t_{ina \leftarrow nosuc} \tag{21}$$

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 *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.

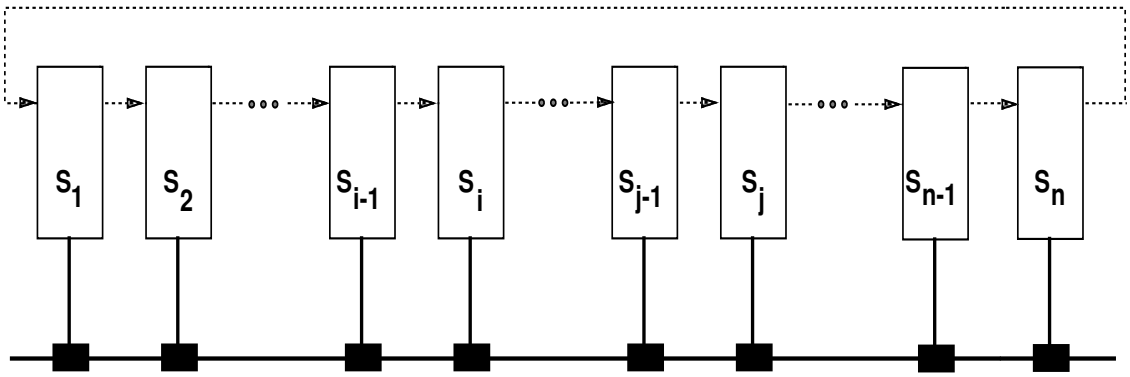

Figure 3: Token-Bus network with identified groups

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.

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

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 successor 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).

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

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

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

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 $t_{rcp} = t_{SSF}$.

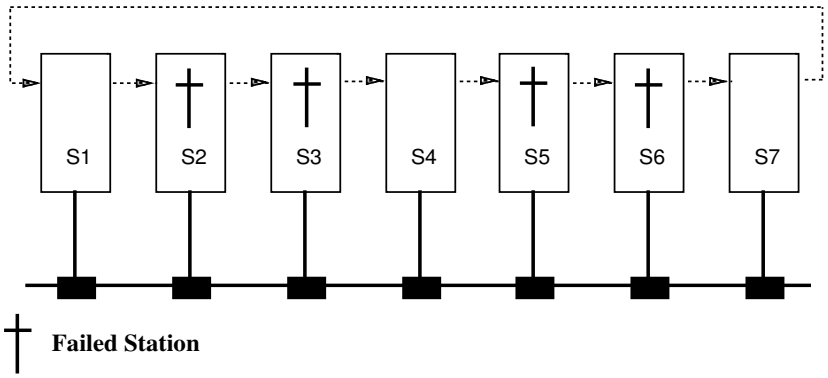

Figure 4: Example of a multi-group failure scenario

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}):

$$t_{ina \leftarrow mgfail} = N_{gfail} \,.\, t_{ina \leftarrow gfail} \tag{24}$$

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:

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

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. $Gf_{dim} = 2$.

$$t^{wc}_{ina \leftarrow mgfail} = \left\lfloor \frac{N_{st}}{3} \right\rfloor . \ t^{wc}_{ina \leftarrow gfail} \tag{26}$$

Token-Bus 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 $N_{ST} = 32$. The results of the evaluation, for each one of the studied scenarios, are summarized in Table 3.

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 ($N_{ST} = 32$)		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
		Multiple 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 ($N_{ST} = 32$)		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
		Multiple Group Fails ($N_{st} = 32$)		6.900	62.886

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

The worst case figures are quite 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; station configuration; or (in certain settings) allow more restrictive assumptions about multiple failures. This brings the highest figures drastically down, as shown in [41].

4 Case Study: Token-Ring

The ISO 8802/5 Token-Ring is a local area network that uses a token-passing access method over a ring architecture. Individual stations attach to the network through the use of *concentrators*, imposing a *star* topology over the bare ring architecture [44].

Access control uses a token passing protocol. Access to the shared broadcast medium for data transmission is only granted to the station which currently holds the token. A token priority reservation scheme is used to schedule message transmissions accordingly their urgency.

The performance of token-ring access method has been studied along the last decade by a set of authors [2, 35, 36, 45]. Although these studies neglect the influence of inaccessibility, by considering that the network always operate normally, an exploratory analysis on the performance of token-ring error handling mechanisms has been provided in [48]. However, it is essentially biased by an experimental evaluation of key error scenarios in a specific network configuration, being hard to infer absolute performance figures in other network configurations.

Our approach is substantially different: we made an exhaustive quantitative analysis of the 8802/5 error handling mechanisms, deriving a set of easy-to-use formulas that can be used to predict the inaccessibility times on any network configuration.

Data Rate ($Mbps$)	Bit Time t_{bit} (μs)	Octect Time t_{oct} (μs)
4	0.25	2.0
16	0.0625	0.5

Table 4: Parameters for an ISO 8802/5 Token-Ring LAN

Network Characterization

For our purposes, two important parameters characterize the basic operation of any ISO 8802/5 Token-Ring network:

- ◇ **Data Rate** - The rate of data signalling, in the ring. It gives a meaning to the *bit* and *octect* times.
- ◇ **Network Ring Latency** - This parameter accounts the time required for a data unit to propagate once around the ring. It is an aggregate variable accounting for station delays and network size, as expressed by equation:

$$t_{rlat} = t_{PD} + t_{bit} \,.\, (N_{ST} \,.\, lat_{st} + lat_{AM}) \tag{27}$$

  - t_{PD} – End-to-end ring cable propagation delay. It is a ring length-dependent variable with a typical value of 5 $\mu s/km$.
  - lat_{st} – Station latency, i.e. the delay expressed in bits that each individual station introduces in the circulation of data units.
  - lat_{AM} – Depth of the Active Monitor latency buffer, in bits. The extra delay, introduced by this station, aims two goals: assure a minimum latency on the ring that allows token circulation and provide phase jitter compensation [47].
  - N_{ST} – Maximum number of stations in the network. Equation (27) defines therefore a worst-case bound for the ring latency instead of their current value. However, notice that its variation can usually be neglected for small variations on the number of ring active stations.

MAC Protocol Data Units

Several types of protocol data frames are exchanged between peer MAC[15] entities to perform the following functions:

- Purge the ring of frame remnants.
- Elect a new active monitor.
- Delimit a failure domain.
- Report the presence of an active monitor on the ring.
- Support of the *neighbor notification process*[16].
- Detect the presence of stations with duplicated addresses.

MAC Frame	Symbol	Duration (μs)	
		$4Mbps$	$16Mbps$
Token	t_{Tk}	6	1.5
Frame header/trailing	t_{HrTr}	42	10.5
Claim Token	t_{CLM}	66	16.5
Duplicate Address Test	t_{DAT}	50	12.5
Active Monitor Alive	t_{AMA}	66	16.5
Standby Monitor Present	t_{SMP}	66	16.5
Beacon	t_{BCN}	70	17.5
Ring Purge	t_{PRG}	66	16.5

Table 5: Duration of MAC Token-Ring Protocol Data Units ($l_{add} = 48$)

MAC Timer	Symbol	Evaluating Expression	Default Values
Return to Repeat	t_{TRR}		4.1 ms
Holding Token	t_{THT}		10 ms
Queue PDU	t_{TQP}		10 ms
Valid Transmission	t_{TVX}	$t_{TRR} + t_{THT}$	10 ms
No Token	t_{TNT}	$t_{TRR} + N_{max} \,.\, t_{THT}$	1 s
Contention Monitor	t_{TCM}		1 s
Active Monitor	t_{TAM}		7 s
Standby Monitor	t_{TSM}		15 s
Unresolved Beacon	t_{TUB}		26 s

Table 6: MAC Protocol Timers

The duration of all standard MAC protocol frames which take part in these actions are presented in Table 5, for an address length (l_{add}) of 48 bits. The values were computed based on frame length and data rate.

[15] Medium Access Control.

[16] This process aims to provide each station with the address of its nearest upstream neighbor, required for failure domain delimitation. Additional details on neighbor notification can be found in [20].

MAC Protocol Timers

A set of timers are used by the standard MAC protocol either to detect the abnormal absence of given actions/events or to control the duration/interleaving of others. These timers are listed in Table 6 together with the default values used in the token-ring implementation provided by [47]. Specific timers – *contention monitor* and *unresolved beacon* – used by [47] are also included in the table, although not defined in the standard specification.

Token-Ring Accessibility Constrains

A comprehensive set of scenarios leading to network inaccessibility will be analysed in this section. As a general rule, we start with very simple single error situations that in many cases degenerate into more complex scenarios. Additionally the analysed scenarios progressively evolve to less restrictive – and thus more realistic – operating conditions/fault assumptions. For most of the cases, the main analysis is completely general, being particularised for best and worst cases, afterwards.

Stripping Errors

Before transmitting any queued frame into the network a station must wait for the reception of a *free token*. Should the priority of queued frames be equal or greater than the priority of the received token and they will be elected for transmission. Frame transmission is initiated by setting the *token bit* in the received *access control* field and by clearing the corresponding *monitor* and *priority reservation*[17] bits. The remaining frame control fields, data and ending sequence are appended afterwards.

After completing the transmission of the last queued frame or when there is not enough time remaining in *Timer:Holding Token* (THT) to finish the next transmission, the emitting station will start to send "fill symbols". A new *free token* is generated upon the reception of a frame with a source address matching its own address, but the station will continue to send "fill symbols" until the last transmitted frame is received[18] or until the expiration of *Timer:Return to Repeat* [19] (TRR). In normal operation, this action strips out from the ring all the frames that a station has transmitted. Should the station fail before removing from the ring the transmitted frames, and these would tend to persist in circulating through the ring if appropriate recovery mechanisms were not provided.

A similar error scenario occurs when a station that has issued a *priority token* fails before it is able to restore the former ring service priority, as specified in [20]. In this circumstance, priority tokens would also tend to persist on the ring.

The presence of *circulating frames* or *circulating priority tokens* on the network ring is prevented through a dedicated *active monitor* mechanism. The time required for the elimination of these packets results from the sum of two distinct contributions: the time needed for error detection plus the time required for its recovery, as described by equation (28).

$$t_{ina \leftarrow nostrip} = t_{dect \leftarrow nostrip} + t_{prp} \tag{28}$$

In order to detect the presence of persistent data units on the ring, the active monitor marks every passing frame or priority token, by setting the corresponding *monitor bit*. Any frame or token arriving at the active monitor with its *monitor bit* set is stripped from the ring, by this station.

[17] Frames are always transmitted with the priority of the received token, regardless their own priority.
[18] Identified by the non-assertion of the *intermediate bit* in the *ending delimiter* field.
[19] This timer has been re-started upon transmission of the last frame.

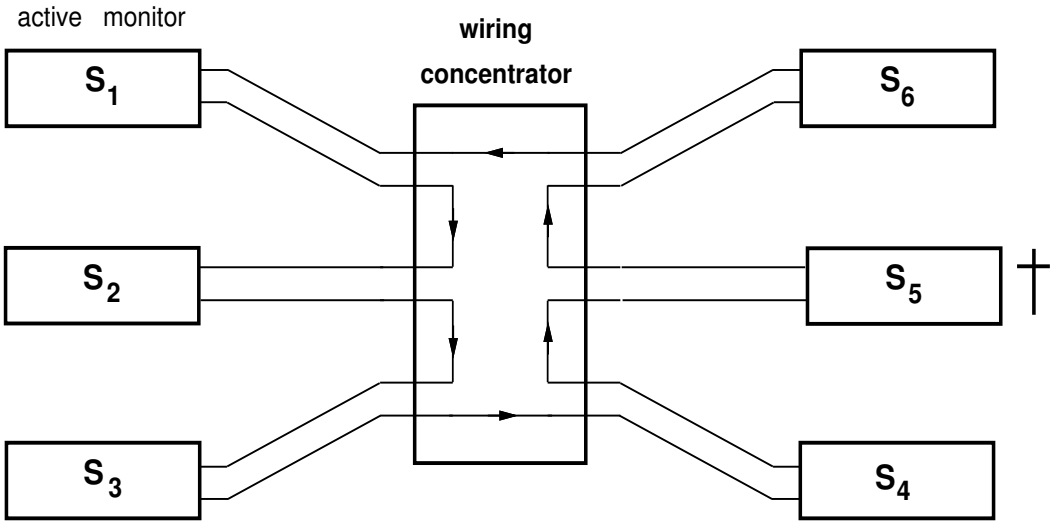

Figure 5: Example of a token-ring network

Error detection can never be performed in a time lower than the network ring latency and its exact value depends on the failed station and active monitor relative locations. In the best-case the active monitor immediately follows the station originating the error whereas in the worst-case the same station immediately precedes the active monitor. This means that both situations differ at most in one ring latency period, as described by equation (29).

$$t_{dect \leftarrow nostrip} = \begin{cases} t_{rlat} & \text{best-case scenario} \\ 2 \,.\, t_{rlat} & \text{worst-case scenario} \end{cases} \tag{29}$$

Upon error detection, the active monitor initiates a recovery procedure that is started with the continuous transmission of *ring_purge* frames. Once the active monitor detects the reception of one of its own *ring_purge* frames, it starts to transmit "fill symbols" during a period given by t_{TRR}, to ensure that all these protocol data units have been stripped from the ring and generates a "new token" afterwards. This *purge ring process* – t_{prp} – terminates timely, in the absence of other network errors, with a best-case duration given by:

$$t_{prp}^{bc} = t_{prPrg} + t_{PRG} + t_{rlat} + t_{TRR} + t_{prTk} \tag{30}$$

where t_{prPrg} and t_{prTk} represent the processing overheads required to generate the *ring_purge* frame and the *token*, respectively. The designation processing overhead, herein employed, is essentially due to a particular token-ring implementation [47] where media access protocols are executed in a special-purpose LAN communications processor. With such an architecture it is reasonable assume that generation of MAC frames and *tokens* may be affected by some latency and therefore we consider worst-case figures, in the processing overheads, for the derivation of an expression that, in the absence of other network errors, gives the worst-case duration of the *purge ring process*:

$$t_{prp}^{wc} = t_{prPrg}^{wc} + t_{PRG} + t_{rlat} + t_{TRR} + t_{prTk}^{wc} \tag{31}$$

Expressions for the best and worst inaccessibility durations, due to the absence of ring stripping can be easily derived. These durations, that we signal with superscripts ^{bc} and ^{wc}, are given by equations (32) and (33).

$$t^{bc}_{ina \leftarrow nostrip} = 2 \,.\, t_{rlat} + t_{prPrg} + t_{PRG} + t_{TRR} + t_{prTk} \tag{32}$$

$$t^{wc}_{ina \leftarrow nostrip} = 3 \,.\, t_{rlat} + t^{wc}_{prPrg} + t_{PRG} + t_{TRR} + t^{wc}_{prTk} \tag{33}$$

Let us now analyse some side effects that token regeneration, by the active monitor, may have on the normal token circulation. Consider the token-ring network pictured in Fig. 5. Station $\mathcal{S}_5$ is either the then-current token holder or the *priority token* remover when it fails, while station $\mathcal{S}_1$ is the then-current active monitor. Station $\mathcal{S}_6$ which was about to receive the token, when the failure of its upstream station, is passed over. So, in addition to the time required for token regeneration, station $\mathcal{S}_6$ 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 raw inaccessibility duration. It must be consolidated with the addition of the network access delay upper-bound.

Bit Corruption Errors

Let us proceed our analysis of token-ring inaccessibility by considering a scenario where some selected bits, inside the frame stream, see their values modified due to the occurrence of faults. Although corruption of individual bits is not likely to happen, this analysis is important because it evidences relevant aspects of protocol robustness.

For example, in 8802/5 frames a set of very important control bits are transmitted outside the FCS coverage domain, due to the need of their "on-fly" modification by the MAC protocol, and it is important to understand how the corruption of these and other equally relevant bits can affect ring operation. This scenario provides a systematic analysis of the possible bit corruption errors.

- **Access Control Field** – The following set of bits are defined within this field:
  - ◇ *Priority and reservation bits* – Corruption of priority and reservation bits can affect both frames and tokens, and they can eventually lead to the issuing of a token with a priority either higher or lower than the correct. In Table 7 we provide a systematic view of the possible error conditions. Errors due to incorrect *priority bits* are more severe than those resulting from the corruption of *reservation bits* since they escape from the control of the *priority reservation* scheme[20]. Its recovery is performed either through an *active monitor stripping* action or through a dedicated *error recovery procedure* that clears the priority stacks [20]. Errors resulting from reservation bits corruption are recovered strictly by the intervention of the *priority reservation* mechanisms.
  - ◇ *Monitor bit* – This bit is always originally transmitted with a zero value. Its eventual corruption may lead to the premature removal of the frame or priority token from the ring, through the aforementioned *active monitor stripping* action.
  - ◇ *Token bit* – The setting of this bit, by corruption, can transform a *free token* into a *busy token* that will be later eliminated through an *active monitor stripping* action. On the other hand, the incorrect clearing of the *token bit* converts a frame into an incorrectly formed token. Such a scenario will be studied ahead.
- **Frame Body** – The frame body is protected by FCS checking and consequently, a corruption in this stream will be detected with the coverage provided by the associated CRC mechanism. Frames with CRC errors will be usually discarded by the MAC sub-layer and, exception made for the eventual corruption of the *source address* field, they do not normally lead to network inaccessibility. Source address recognition is a mandatory condition for token release, by the then-current holding station, and the lack of such event leads to a token loss scenario, to be

[20] Additional details on priority reservation can be found in [20].

Token Priority Incorrectness	Corrupted bits	
	Priority	Reservation
Higher	• Incorrect issuing of a priority token. • Elimination by the active monitor.	• Circulation of a priority token. • Reservation of correct priority value. • Correct priority restored on next token release.
Lower	• Incorrect issuing of a low priority token. • **Error Treatment:** clearing of priority stacks.	• Circulation of a low priority token. • Normal priority management and ring utilization operations. • Correct priority restored upon circulation of the low priority token.

Table 7: Consequences of priority/reservation bits corruption

described ahead. Notice that errors in other frame body fields do not need to be considered since they do not directly affect MAC protocol operation.

- **Ending Delimiter** – A particular bit in the end delimiter — *intermediate bit* — indicates whether or not a given frame is the last transmission issued by the then-current token holding station. Corruption of the *intermediate bit* can lead a station either to the premature or to the late abandonment of *frame stripping* operations. In the first case, frames continue to be repeated by downstream stations, until its elimination by the next token holding station or by the active monitor. On the other hand, the absence of a frame received with its *intermediate bit* cleared maintains the station in "stripping". This situation can be quite complex and calls for a more thorough analysis, to be performed next.

Persistent Stripping

A station that does not recognise its last transmission, due to a corrupted or incorrectly formed ending delimiter, continues to send "fill symbols" and to strip any arriving data, until the expiration of *Timer:Return to Repeat*. Assuming that no other errors have occurred, the token should be released by the stripping station and it may be used then by another station. In this scenario, three distinct situations may happen:

- No station uses the token, that will be eliminated by the *stripping station*, thus leading to a token loss scenario.
- The next station using the token transmits during a period – t_{tx} – obeying to relation $t_{tx} \leq t_{TRR}$. Due to the presence of the *stripping station*, no frames will be repeated downstream until detection of the then-current token holder last transmission[21] or until a period given by t_{TRR} has elapsed. In either cases, the token holding station does not hear any of its own transmissions and consequently, it will not release the token. Such a situation degenerates therefore in a token loss scenario.
- The station using the token transmits during a period given by relation $t_{tx} = t_{TRR} + t_{ltx} \leq t_{THT}$, where t_{ltx} is the duration of the last $n \geq 1$ transmissions. In the absence of other errors, these n transmissions are correctly repeated by the former *stripping station* and received by the then-current token holding station. This allows the straight restoration of normal ring operation.

[21] Detection of a frame with the *intermediate bit* cleared is a sufficient condition for abandon stripping operations.

The presence of a *stripping station* prevents any frame repetition and consequently their reception, during a time interval bounded by t_{TRR}, in all the downstream stations till the station retaining the token (the out-partition). At least during this period the network is partitioned, meaning that only stations located between the then-current token holder and the *stripping station* (the in-partition) will be able to receive that station transmissions. This selective partition pattern, when strictly due to MAC protocol operation, is typical of networks with a ring topology. During the inaccessibility period, at most one station – defining the in-partition start boundary – can capture the token. A second network access, by this station, will only be possible after recovery. This means that inaccessibility due to the presence of a *persistent stripping* station is controlled both in terms of their number of occurrences, as explained above, and duration, with best and worst-case figures given by:

$$t^{bc}_{ina \leftarrow pstrip} = t_{TRR} \tag{34}$$

$$t^{wc}_{ina \leftarrow pstrip} = t^{wc}_{ina \leftarrow tkloss} \tag{35}$$

Bad Format Errors

In a previous scenario we have analysed the impact of individual bit corruption on MAC protocol operation. However, individual bit corruption seldom occurs; extensive adulteration of frame fields due to corruption bursts is a more realistic scenario. On the other hand bit modification is not the only source for frame errors: EMI can originate electric signaling code violations and frames can be truncated by a deficient operation of physical layer components.

Let us group these different source of errors into the unique designation of bad format errors. Two particular cases are considered: start of frame sequence and badly formed tokens. The remaining cases either were already addressed in the previous scenarios or are not relevant for our study of MAC protocol robustness.

- **Start Frame Sequence** – If the start of frame sequence, i.e. either the *start delimiter* or the *access control* fields are incorrectly formed, no frames or tokens will not be recognised by network stations. In consequence, the scenario degenerates in the token loss, to be described ahead.
- **Bad Token Format** – In this case we analyse the behaviour of token-ring stations upon reception of a token whose start sequence (SD and AC fields) is not immediately followed by a correct ending delimiter. This can result from the transformation of a frame into a token through the modification of its *token bit* or due to a corrupted/truncated ending delimiter.

  Let us assume that a station in the network captures such a token and starts using it by modifying the corresponding *token bit*. Once it detects that the token is not properly formed, it immediately terminates frame transmission with the issuing of an abort sequence and makes a transition to the *repeat state* [20]. The frame remnant, previously transmitted, will be soon eliminated through an *active monitor stripping* action. An incorrect formed *priority token* will be eliminated in the same way, even if it is not captured by any station. On the other hand, an incorrect *non-priority token* that no station tries to use will tend to persist in circulating through the ring. This error is usually also corrected by the active monitor, but the exact set of actions rather depends of each particular system architecture.

  For example, [48] mentiones that in the IBM Token-Ring, the reception of a correct start sequence is a sufficient condition for token recognition. Therefore, no recovery action will be initiated in this case and the defective *non-priority token* remains undetected, continuing to circulate on the ring, until a station tries to use it. Conversely, in other implementations a token might be only recognised when properly formed. In consequence, recovery from a token format error can be triggered either by an attempt to use the defective token or by an abnormal absence of valid packets on the ring.

In this study, we will consider a network operating accordingly with this last approach. This favours network performability in situations of low network traffic. However, the two methodologies provide nearly identical results in situations of fair to high traffic loads.

Token Loss

The loss of a token can be due either to the failure of the then-current token-holding station or to the aforementioned errors in network operation (i.e. unrecognisable start of frame sequence, source address field, last transmission indicator or bad token format).

Token losses are detected by monitoring the absence of correct ring activity and recovered through the already described *purge ring process*. The whole process is leadered by the then-current active monitor. Each time that a frame or a token passes through the active monitor a dedicated MAC protocol timer that monitors valid network transmissions is re-started. In the absence of valid formed packets *Timer:Valid Transmission* (TVX) is not re-started[22] and will expire. This event triggers the process of purging the ring from any frame remnant and the subsequent issuing of a new token. The duration of the resulting inaccessibility period is given by:

$$t_{ina \leftarrow tkloss} = t_{dect \leftarrow tkloss} + t_{prp} \tag{36}$$

where $t_{dect \leftarrow tkloss}$ is the token loss error detection latency and t_{prp} is the duration of the *purge ring process*. The time needed for error detection depends on the time elapsed between the last TVX re-start and the occurrence of the fault. In the worst-case the fault occurs just after TVX re-start, while in the best-case the token almost completes an entire round before its loss. Therefore, the corresponding error detection latency varies between the nominal value of TVX minus one ring latency period and their exact value.

$$t_{dect \leftarrow tkloss} = \begin{cases} t_{TVX} - t_{rlat} & \text{best-case scenario} \\ \\ t_{TVX} & \text{worst-case scenario} \end{cases} \tag{37}$$

The conjunction of this expression with equations (30) and (31) allow us to derive the best and worst-case bounds for this scenario.

$$t^{bc}_{ina \leftarrow tkloss} = t_{TVX} + t_{prPrg} + t_{PRG} + t_{TRR} + t_{prTk} \tag{38}$$

$$t^{wc}_{ina \leftarrow tkloss} = t_{TVX} + t^{wc}_{prPrg} + t_{PRG} + t_{rlat} + t_{TRR} + t^{wc}_{prTk} \tag{39}$$

Active Monitor Loss

In the presence of a long term ring error condition, the previous method for token regeneration may not timely terminate. The duration of the *purge ring process* is controlled, inside the active monitor, through the use of *Timer:No Token* (TNT), which is started on process initiation. Upon its timeout, due to the non-receiving of its own *ring_purge* frames, the active monitor abandons its protocol functions to enter in *standby monitor* operation. However, this is not the only cause for active monitor loss. For example, its physical layer components may not provide a properly operating *latency buffer* or a correct *master ring clock* and even the whole station may fail.

[22] As a matter of fact, only *non-priority tokens* need to be properly formed. Incorrect frames and priority tokens can always re-start TVX timer, provided that their start sequences are valid formed. This procedure aims to speed-up error recovery by allowing token regeneration to be carried out through an *active monitor stripping* action rather than by token loss detection.

In all the above situations, the structure supporting token circulation collapses. This event is detected in the *standby monitors* through the use of a dedicated mechanism based on *Timer:No Token* (TNT) which, under normal operation, is re-started upon each token passing[23]. In the absence of such data units, TNT is not re-started and will expire, triggering then the election of a new active monitor.

A more complex active monitor loss situation occurs when its MAC sub-layer, due to a failure, stops to ensure the corresponding protocol functions, i.e. circulating packets checking, purge ring control and token generation. Should these mechanisms be required for the recovery of other ring errors and the situation will be detected in some *standby monitor*, upon TNT timeout. However, in the absence of those errors, a situation where MAC active monitor functions are not assured, by any station in the network, might subsist for a very long period. In order to place an upper bound on this time, the active monitor activity is tracked by the remaining network stations. While operating correctly, the active monitor cyclely enqueues for transmission, with a period given by *Timer:Active Monitor* (TAM), an *active_monitor_alive* frame that, when received by the standby monitors, re-starts their *Timer:Standby Monitor* (TSM). In the absence of these active monitor indications, some TSM timers will eventually expire[24]. Stations where this event occurs, start the election of a new active monitor.

Therefore, the standby monitors are provided with two distinct mechanisms for the detection of active monitor failures. Since both are timer-based, the error detection latency actually depends on the time elapsed between the last timer re-start and the occurrence of the fault. The method based on the TNT aims to speed-up error detection upon the failure of physical layer components[25] while the active monitor tracking covers MAC sub-layer failures[26]. The resulting error detection latency is given by expression (40).

$$t_{dect \leftarrow AMloss} = \begin{cases} t_{TNT} - t_{rlat} & \text{best-case scenario} \\ \\ t_{TSM} & \text{worst-case scenario} \end{cases} \tag{40}$$

For the evaluation of the corresponding inaccessibility period, two distinct contributions must be added to the error detection latency: the time required for the election of a new active monitor and the time spent in the token regeneration.

$$t_{ina \leftarrow AMloss} = t_{dect \leftarrow AMloss} + t_{tcp} + t_{prp} \tag{41}$$

The exact duration of a new active monitor election, to be performed through a *token claim process* – t_{tcp} – depends on whether or not a contention between stations occurs and on how quickly this contention is resolved. The *token claim process* starts with the transition of a standby monitor from the *standby* to the *transmit claim* state, where *claim_token* frames are continuously transmitted until they are received back from the network. Upon this event a transition is made to the *active monitor state* [20], initiating then the aforementioned *purge ring process*. In the best-case only one standby monitor executes token claiming. No contention occurs and assuming that the first transmitted *claim token* frame is correctly received, the duration of the process is given by equation:

$$t_{tcp}^{bc} = t_{prClm} + t_{CLM} + t_{rlat} + t_{prAM} + t_{ctc} \tag{42}$$

[23]This mechanism is bypassed under ring recovery by allowing timer re-start to be performed also upon reception of *beacon*, *claim_token* or *ring_purge* frames.

[24]Active monitor tracking is bypassed under ring recovery, by allowing TSM re-start upon reception of *beacon* frames.

[25]This is the standard detection methodology, with best and worst cases given by $(t_{TNT} - t_{rlat})$ and (t_{TNT}). Nevertheless, a subset of severe ring faults can be detected more efficiently by a method that we will describe in the next scenario.

[26]In a time between $(t_{TSM} - t_{TAM})$ and (t_{TSM}).

where t_{prClm} and t_{prAM} are the overheads required for the processing of *transmit claim* and *active monitor* transitions, respectively. The time accounted by t_{ctc} – claiming time correction – represents a corrective term that accommodates modeling of some token-ring implementations [47] where reception of more than one *claim_token* frame is required for process termination. Such corrective term is given by equation:

$$t_{ctc} = N_{Clm} \,.\, (t_{CLM} + t_{CIFS}) \tag{43}$$

where N_{Clm} is the number of extra *claim_token* receptions and t_{CIFS} is the inter-frame spacing between successive *claim_token* frames.

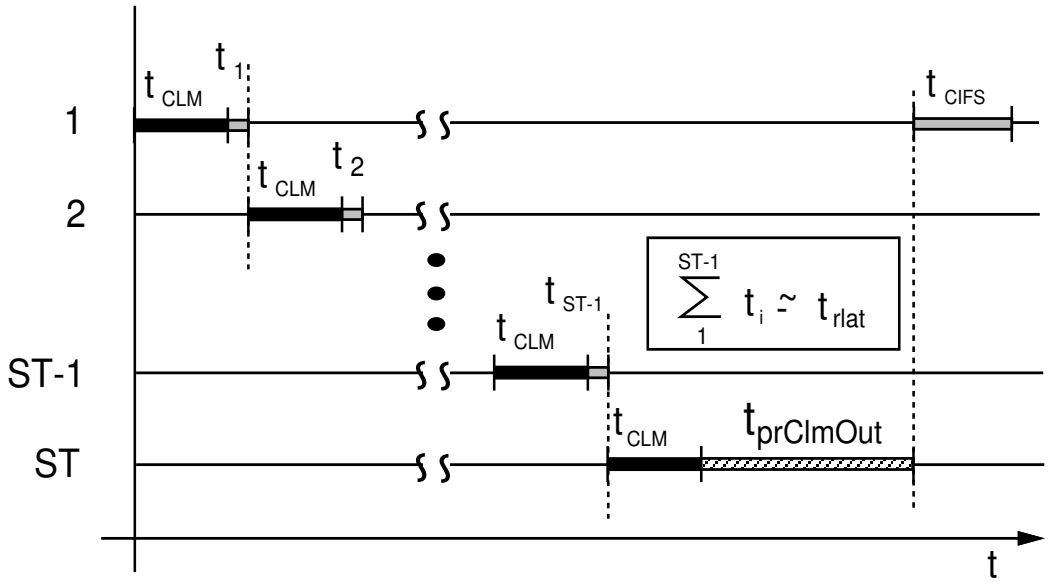

Figure 6: Contention resolution on token claim process (worst-case scenario)

Conversely, a contention process must be resolved when there are several stations in the *transmit claim* state. The resolution algorithm relies on station addresses discrimination: stations receiving a *claim_token* frame with a source address higher than their own, from upstream stations, abandon the process by making a transition into the *standby* state [20]. A worst-case resolution scenario is represented in Figure 6, where the duration of each *claim_token* frame is denoted by t_{CLM} and its propagation delay to the nearest downstream neighbor is identified by t_i. The process is leadered by the station having the highest address and we assume that all downstream stations marginally miss the *claim_token* transmission of its upstream neighbor, initiating then the execution of the corresponding *token claim process*. Furthermore we assume that *claim_token* transmissions cannot be preempted and that the exit from this state takes $t_{prClmOut}$. The duration of the *resolve contention process* is therefore upper-bounded by equation:

$$t^{wc}_{rcp} = N_{ST} \,.\, t_{CLM} + t_{rlat} + t^{wc}_{prClmOut} + t_{CIFS} \tag{44}$$

that added to the time spent in the remaining token claiming actions, provides a worst-case bound for the duration of the *token claim process*, in the absence of other network errors.

$$t^{wc}_{tcp} = t^{wc}_{prClm} + t_{CLM} + t_{rlat} + t^{wc}_{prAM} + t_{ctc} + t^{wc}_{rcp} \tag{45}$$

Signal Loss

In the previous scenario we have described the bare mechanism, forecasted in the standard specification, for the detection of faulty physical layer operation. In addition to that mechanism, some token-ring implementations [47] present special provisions for the detection of severe physical layer errors, like an input signal with insufficient energy or grossly out of phase.

First, a station detecting the reception of five consecutive half-bits of Manchester coded data without a phase change, begins to transmit "fill symbols" until it detects a valid signal. This prevents other stations of detecting the error. Secondly, a recovery procedure is entered if the error condition persists beyond a time period given by $t_{dect \leftarrow signloss} < t_{TNT}$. Usually, the station detecting the error directly enters into the *standby monitor: transmit claim* state, unless it is under a strongest recovery action or in the course of a *ring insertion process*.

This procedure can significantly improve network performability: not only reduces the error detection latency but also creates conditions to allow recovery actions to be started in isolation and, probably, finished without contention.

Ring Interruption

The loss of a station input signal is usually a consequence of physical ring interruption, either due to normal operational constrains or due to the failure of its physical layer components. Assuming a worst-case viewpoint, we will perform, in this scenario, a systematic analysis of data flow interruption that only takes into account their duration. The relation of this error analysis with their physical causes will be performed in subsequent scenarios.

Let us, therefore, assume that the flow of data in the physical ring is interrupted during a time given by t_{RgErr}. The set of procedures, to be executed on error recovery, rather depends on the ring interruption duration, as follows:

i) $t_{RgErr} < t_{Purge}$ — where t_{Purge} is given by equation (37). The impact of such a short-term error condition depends on the extension of corrupted data. Most likely, a worst-case scenario where the token is destroyed occurs, with its regeneration being carried out through the active monitor, as formerly explained.

ii) $t_{Purge} \leq t_{RgErr} < t_{Claim}$ — As we have described, the active monitor initiates, upon TVX time-out, the execution of a *purge ring process*. Whenever ring integrity is restored before the entering of some station into the *token claiming* state, this process allows a straight token regeneration. The deadline for purge ring termination, given by (46), actually depends on whether or not the station processing the error is provided with the aforementioned dedicated mechanism for signal loss detection.

$$t_{Claim} = \begin{cases} t_{dect \leftarrow signloss} & \text{signal loss detection} \\ \\ t_{TNT} - t_{rlat} & \text{no signal loss detection} \end{cases} \tag{46}$$

The resulting inaccessibility period is therefore given by:

$$t_{ina \leftarrow noring}(noClaim) = t_{RgErr} + t_{prp} \tag{47}$$

iii) $t_{Claim} \leq t_{RgErr} < t_{Beacon}$ — When the conditions described in the previous items are not met, some station in the network will initiate a *token claim process* for the election of a new active monitor. Provided that ring integrity is restored in a time that allows process termination, the resulting inaccessibility period will be given by:

$$t_{ina \leftarrow noring}(noBeacon) = t_{RgErr} + t_{tcp} + t_{prp} \tag{48}$$

with the upper-bound for error duration given by the sum of the ring error detection latency with the token claim process marginal termination time $t_{tcp}^{mt} = t_{TCM}$.

$$t_{Beacon} = t_{Claim} + t_{tcp}^{mt} \tag{49}$$

iv) $t_{Beacon} \leq t_{RgErr}$ — If the ring error condition persists beyond t_{Beacon} a recovery mechanism stronger than the *token claim process* is entered. The *beacon recovery process* is therefore only started under severe ring fault conditions. The process is initiated by repeatedly transmitting a *beacon* frame which carries information about the detected error (e.g. signal loss, TCM timeout) and the most reliable upstream neighbor address [20]. Stations in the *beacon* state, receiving *beacon* frames issued by other upstream stations abandon beaconing by making a transition into the *standby* state. At last, only the station located immediately downstream to the ring interruption remains beaconing, thus allowing the delimitation of the failure domain. Restoration of ring physical integrity usually requires network reconfiguration. The exact set of actions, to be performed for ring recovery, rather depends of each particular case and will be discussed in next scenarios. For now, we simply assume that the *beacon recovery process* timely terminates, upon reception of its own *beacon* frames, with a resolution period given by t_{brp}. The *beacon recovery process* is followed by the election of a new active monitor and by the corresponding token issuing.

$$t_{ina \leftarrow noring}(Beacon) = t_{Beacon} + t_{brp} + t_{tcp} + t_{prp} \tag{50}$$

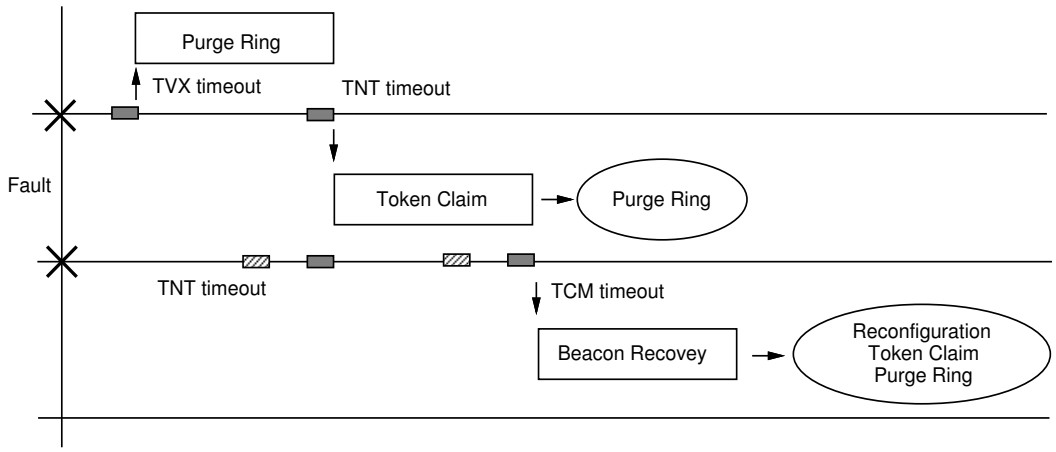

Figure 7: Token-ring recovery processes

Dumb Transmitter

Let us assume that a given station fails in the sense that it stops to transmit/repeat network data. Due to the permanent nature of the failure, network reconfiguration is required. This action is performed as a consequence of a *beacon recovery process* execution: a station receiving a *beacon* frame whose *upstream neighbor address* field matches the station's physical address, assumes that its transmitter is failed and removes it-self from the physical ring to execute a set of *confidence tests*. Since the station's transmitter is actually failed, the station will not re-insert.

The *beacon recovery process* is always wined by the nearest downstream neighbor of the failed station. Assuming that no other station, but the process winner issues *beacon* frames, no contention occurs and the propagation time of these data units, till the station with the failed transmitter, will be simply given by the network ring latency:

$$t_{bpd}^{bc} = t_{BCN} + t_{rlat} \tag{51}$$

In a less favorable scenario other stations in the network may also enter in this process and the *beacon propagation delay* – t_{bpd} – will be more long, as a consequence of station contention. For the computation of a worst-case bound, we consider that all downstream stations marginally miss the reception of its upstream neighbor *beacon* frame, initiating then its own *beacon* transmission. Furthermore, we assume that this transmission cannot be preempted and that the exit from this state takes $t_{prBcnOut}$. The time required for the propagation to the dumb station of a *beacon* frame issued by its nearest downstream neighbor is, under these circumstances, given by:

$$t_{bpd}^{wc} = N_{ST} \,.\, t_{BCN} + t_{rlat} + t_{prBcnOut}^{wc} \tag{52}$$

where t_{BIFS} is the inter-frame spacing between successive *beacon* frames.

The resulting inaccessibility period is obtained from equation (50) using a resolution period given by equation:

$$t_{brp\leftarrow dtx} = t_{prBcn} + t_{bpd} + t_{st\leftarrow leave} + t_{BIFS} + t_{BCN} + t_{btc} + t_{rlat} \tag{53}$$

where t_{prBcn} is the overhead required for entering into the beacon state and $t_{st\leftarrow leave}$ is the time required for switching the station's attaching relay from the *inserted* to the *bypassed* state. Notice that after dumb station removal, it is required that a beacon frame must be received back, in the beaconing station, to allow the *beacon recovery process* to terminate. The time accounted by t_{btc} – beaconing time correction – represents a corrective term that accommodates modeling of some token-ring implementations [47], where reception of more than one *beacon* frame is required for process termination. Denoting by N_{Bcn} the number of extra *beacon* receptions, such corrective term is given by:

$$t_{btc} = N_{Bcn} \,.\, (t_{BCN} + t_{BIFS}) \tag{54}$$

This description essentially follows the features of the token-ring implementation provided by [47], particularly in those issues concerning signal loss detection – as a result of the dumb transmitter – and station auto-removal for confidence testing. Similar considerations also apply to the next scenario.

Deaf Receiver

If the ring interruption is due to the failure of the beaconing station receiver, the auto-removal of its upstream neighbor does not restore physical ring integrity. As a matter of fact, after termination of their *confidence tests* that station re-inserts in the ring, since they will not detect any failed component. The station with the failed receiver will remain transmitting *beacon* frames until expiration of *Timer:Unresolved Beacon* (TUB), as implemented in [47]. At that time, the station with the failed receiver de-inserts from the ring and their transmission of *beacon* frames stops.

Usually, the time required to complete station auto-removal – $t_{st\leftarrow leave}$ – is much greater than the network ring latency – t_{rlat} – and therefore, any *beacon* frame that may be in repetition throughout the ring is, in principle, destroyed during reconfiguration. If this is not the case, circulation of a beacon frame issued by a station that does not belong to the ring anymore, will entirely block its operation, since it prevents the initiation of any other recovery action. To cope with this situation, the existing mechanism supporting detection of circulating packets is activated in a given station, upon reception of two consecutive *beacon* frames originated from its nearest upstream neighbor. If a *circulating beacon* frame is detected by this station a transition to the *transmit claim* state is made.

In a more realistic situation, no *beacon* frames subsist on the ring after beaconing station auto-removal, and a dedicated mechanism is provided for the detection of an abnormal absence of *beacon*

frames during the execution of a *beacon recovery process*. If no *beacon* frames are received for a period of $t_{dect\leftarrow noBcn}$, it is assumed that the conditions causing the ring interruption have been corrected and a transition is made to the *transmit claim* state.

This is the case considered in our analysis. The inaccessibility period, due to the presence of a deaf receiver, is given by equation (50), with a resolution period given by equation (55).

$$t_{brp\leftarrow drx} = t_{prBcn} + t_{TUB} + t_{st\leftarrow leave} + t_{dect\leftarrow noBcn} \tag{55}$$

Streaming Receiver

In this scenario, we consider that the receiver section of a given station, although recognising the presence of a correct input signal, fails in the provision of correctly decoded data at the PHY-MAC interface. Essentially, recovery from this error follows the steps used to bypass a deaf receiver, with the differences listed below:

- Error detection is performed by TNT timeout instead of by signal loss detection.
- Since TNT can expire in several stations, contention for the establishment of a *beacon recovery process* can occur. Beacon frames are transmitted either with a *received claim token* or *no received claim token* indications.
- Only the faulty station de-inserts from the ring, upon TUB timeout.

The resulting inaccessibility period will be expressed by equation (50), with a resolution period given by (55) and a t_{Beacon} value defined by:

$$t_{Beacon} = t_{TNT} - t_{rlat} + t_{TCM} \tag{56}$$

Jabbering Transmitter

Let us assume a token-ring implementation that, as in [47], is provided with a watchdog-timer monitoring station insertion. A reset of this timer must be performed, periodically, by the MAC protocol to ensure that the station remains inserted.

Additionally, let us assume that the MAC protocol functions stuck in the continuous transmission of frames, tokens or abort sequences and that, in consequence, it can no long ensure the timely reset of the *physical insertion watchdog-timer*. In these conditions, a jabbering station will de-insert from the ring, in a time given by t_{PhyOut}. The duration of the ring interruption is therefore given by:

$$t_{RgErr\leftarrow jabber} = t_{PhyOut} + t_{st\leftarrow leave} \tag{57}$$

The resulting inaccessibility period is given accordingly one of the expressions (47) through (50).

Station Insertion

Stations are physically inserted in the network ring through the switching of a dedicated relay from the *bypass* to the *inserted* state. In the course of this action, the network ring is interrupted during the period required for relay switching and bounce elimination. The duration of this disturbance, herein denoted by $t_{st\leftarrow join}$, is usually small but can, nevertheless, interrupt the normal token flow, due to its corruption. Assuming that $t_{st\leftarrow join} < t_{TVX}$, the resulting inaccessibility period will be equivalent to a token loss scenario.

Multiple Insertions

In the best-case all the stations will simultaneously switch their *attaching relays*, and the situation will not be distinguished from the insertion of a single station. In the worst-case, station insertions are serialised. An upper-bound for ring interruption duration can be obtained assuming a massive join to a network previously formed only by two stations.

$$t_{RgErr \leftarrow mjoin} = (N_{ST} - 2) \, . \, t_{st \leftarrow join} \tag{58}$$

The resulting inaccessibility period is either given by equation (36) or accordingly one of the expressions (47) through (50).

Station De-Insertion

Ring interruption also occurs when a station switches its *attaching relay* from the *inserted* to the *bypass* state. The duration of ring interruption is denoted by $t_{st \leftarrow leave}$ and accounts the time required for relay switching and bounce elimination.

Multiple De-Insertions

In the best-case, all the stations simultaneously de-insert from the ring and the situation cannot be distinguished from the withdraw of a single station. In the worst-case, these events are serialised and upon a massive withdraw of $(N_{ST} - 2)$ stations, the duration of ring interruption is given by:

$$t_{RgErr \leftarrow mleave} = (N_{ST} - 2) \, . \, t_{st \leftarrow leave} \tag{59}$$

Broken Lobe

Upon an abrupt breakage of a station's lobe connection, a total loss of the *DC phantom drive current* is detected and the station's *attaching relay* is disconnected, in a period given by $t_{st \leftarrow leave}$. More subtle faults, originated by the interruption of only one line in the transmit/receive pair and that can cause an abnormal *binary error rate* in this link, are detected by monitoring the balance of the DC current. If an unbalanced drive current persists beyond a period given by $t_{WireFault}$, the station removes it-self from the ring. The corresponding ring interruption period is thus given by:

$$t_{RgErr \leftarrow brklobe} = \begin{cases} t_{st \leftarrow leave} & \text{best-case scenario} \\ \\ t_{WireFault} + t_{st \leftarrow leave} & \text{worst-case scenario} \end{cases} \tag{60}$$

Broken Backbone

In the presence of a broken backbone the auto-removal of those stations located in the nearest neighborhood of the failure domain[27], performed in an attempt to restore ring integrity, will not succeed. These stations will re-insert on the ring, after completion of their confidence tests and we return back to the situation existing prior to station's auto-removal.

Using today's technology, restoration of ring integrity only occurs after manual reconfiguration. Such situation is clearly unacceptable for real-time applications. The study of automatic reconfiguration mechanisms able to recover from these faults fall out from the scope of this work, but are of mandatory importance for an ISO 8802/5 LAN with enhanced availability.

[27] Upstream and downstream.

Token-Ring Analytic Results

In order to complete our study of networking accessibility we now evaluate the inaccessibility time bounds, for a given token-ring network. The network length is $C_l = 500m$ and the total number of stations $N_{ST} = 32$. We have assigned a typical value of $200\mu s$ to all the processing times, with an overhead of 25% for their worst-case figures. These values are estimates, supported by our personal experience on the evaluation of communication protocols performance. However, an experimental measurement of these parameters is a mandatory condition when a highest degree of accuracy is required. To all the remaining implementation dependent parameters we have assigned the set of values which characterise the behaviour of [47] ($lat_{AM} = 27$, $lat_{st} = 2.5$, $N_{Clm} = 2$, $N_{Bcn} = 7$, $t_{CIFS} = t_{BIFS} = 20ms$, $t_{dect \leftarrow signloss} = 200ms$, $t_{PhyOut} = 8.7ms$ and $t_{WireFault} = 5s$). Further estimates, which are essentially dependent on the performance of physical layer components, are: $t_{st \leftarrow join} = 5ms$ and $t_{st \leftarrow leave} = 10ms$. The results of the evaluation, for all the relevant studied scenarios, are summarized in Table 8.

Data Rate - $4Mbps$		
Scenario	t_{ina} (ms)	
	min.	*max.*
Stripping Errors	4.6	4.7
Persistent Stripping	4.1	14.7
Token Loss	14.6	14.7
Active Monitor Loss	1044.9	15068.4
Dumb Transmitter	1416.0	1441.7
Deaf Receiver	27455.2	27478.3
Streaming Receiver	28255.2	28278.3
Jabbering Transmitter	23.3	23.4
Station Insertion	14.6	14.7
Multiple Insertions ($N_{ST} = 32$)	14.6	154.7
Station De-Insertion	14.6	14.7
Multiple De-Insertions ($N_{ST} = 32$)	345.2	368.3
Broken Lobe	14.6	5078.4

Table 8: Token-Ring Inaccessibility Times ($l_{add} = 48$)

The worst-case figures are extremely high, like for example the 15 seconds that may take the establishment of a new active monitor and the 28 seconds required for the shutdown of a streaming receiver. However, this study not only provides a basis to know what to expect from token-ring performance in the presence of failures, but also provides guidance on where to act in order to improve the situation and *justifiably* achieve better performability, and thus better respect any bounded delay requirements.

With these grounds, we can recognise that longest inaccessibility periods are essentially due to an over dimensioning of the standard MAC timers. The definition of inaccessibility minimizing strategies falls out from the scope of this work. Nevertheless, we can envisage that a realistic tuning of MAC protocol timers with each specific network configuration can optimise error-handling performance and bring the inaccessibility figures drastically down. This will be a mandatory condition for the eventual utilization of ISO 8802/5 Token-Ring LANs in real-time applications.

5 Case Study: FDDI

The *Fiber Distributed Data Interface* (FDDI) is a high speed LAN [39] developed under the auspices of ANSI[28] within its X3T9.5 committee, and approved as international standard ISO 9314.

The FDDI was originally conceived to be used either as a backend network within the computer center environment or as a backbone network for LAN interconnection. The original FDDI was supported on multi-mode fiber, needed to accommodate a high data rate transfer. Along these last years the FDDI has evolve through the definition of extensions in order to support isochronous traffic (FDDI-II), and to expand their geographic scope for the metropolitan area, using either single-mode fiber [12] or dedicated lines [31]. On the other hand a serious effort has been placed in bring FDDI to the desktop environment through the use of either low-cost fiber or shielded/unshielded cooper cables [11, 14].

The FDDI uses a timed-token protocol access method. Two classes of service are provided: synchronous and asynchronous. Synchronous frames can be transmitted whenever a station receives the token. Asynchronous frames can only be transmitted using the bandwidth not consumed by synchronous class. Multiple levels of priorities can be assigned within the asynchronous class. The timeliness characteristics of the FDDI LAN presumably allows its utilization in control and automation, as frontend networks.

The performance of the FDDI access method has been widely studied [42, 43, 7, 27, 28]. 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 ring operation is affected, usually severely, in the sense that it can no longer ensure the calculated frame transmission delay bound.

A comprehensive analysis of the FDDI error handling mechanisms is provided in [26]. However, it is essentially qualitative. This work presents an exploratory quantitative analysis of the FDDI error handling mechanisms. Future evolution of the FDDI specification, particularly in issues concerning fault detection and fault isolation policies, as well as network reconfiguration, may lead to modifications of the model to enhance its adherence to the standard.

MAC Frame	Symbol	Duration (μs)
Token	t_{Tk}	0.88
Frame header/trailing	t_{HrTr}	2.24
Claim Token	t_{CLM}	2.56
Beacon	t_{BCN}	3.04

Table 9: Duration of MAC FDDI Protocol Data Units ($l_{add} = 48$)

Network Characterization

For our purposes, the following set of two parameters characterize the basic operation of any ISO 9314 FDDI network:

- ◇ **Data Rate** - The rate of data signalling, on the ring (100 Mbps). It gives a meaning to the *bit* ($t_{bit} = 0.01\mu s$) and *octect* times ($t_{oct} = 0.08\mu s$).
- ◇ **Network Ring Latency** - This parameter accounts the time required for a data unit to propagate once around the ring. It is an aggregate variable accounting for station delays and network size, as expressed by equation:

[28] American National Standards Institute.

MAC Timers	Symbol	Default Values
Token Holding Timer	t_{THT}	
Transmission Valid Timer	t_{TVX}	2.5 ms
Token Rotation Timer	t_{TRT}	T_min=4.0 ms
		T_max=165.0 ms
SMT Timers	Symbol	Default Value
Entity Coordination	t_{TEC}	
Physical Connection	t_{TPC}	
Idle Detection	t_{TID}	
Noise Event	t_{TNE}	
Ring Management	t_{TRM}	
Timing Values	Symbol	Default Value
PCM signalling timeout	T_Out	100 ms
No operational ring	T_Non_Op	1.0 s
Stuck beacon time	T_Stuck	8.0 s
Direct beacon time	T_Direct	370 ms
Scrub time	t_{Scrub}	7.1 ms

Table 10: FDDI Protocol Timing References

$$t_{rlat} = t_{PD} + N_{ST} \,.\, t_{st\_lat} \tag{61}$$

- t_{PD} – End-to-end ring cable propagation delay. It is a ring length-dependent variable with a typical value of 5 $\mu s/km$.
- t_{st_lat} – Station latency, i.e. the delay that each individual station introduces in the repetition of data units. We consider $t_{st\_lat} = 0.6\mu s$.
- N_{ST} – Maximum number of stations in the network. Equation (61) defines therefore a worst-case bound for the ring latency instead of their current value. However, notice that its variation with this parameter is usually small.

◇ **Station Delay** (t_{SD}) - This parameter represents the overhead due to transmitter idle timer after token capture. We consider $t_{SD} = 3.5\mu s$ and in order to avoid the introduction of an additional parameter we also use the station delay to represent the overhead associated with the processing of any other MAC protocol frame.

MAC Protocol Data Units

Besides token passing only two more types of protocol data frames are exchanged between peer MAC[29] entities: *claim token* frames for the negotiation of the *operational target token rotation time* and *beacon* frames for the signaling of ring breaks. The transmission of *beacon* frames can be made autonomously by the MAC protocol or under *Station Management* request. These *beacon* frames only differ in their contents. The duration of these MAC protocol frames is presented in Table 9, for an address length (l_{add}) of 48 bits. The values were computed based on frame length and data rate.

[29]Medium Access Control.

FDDI Timers

A set of timers as well as some timeout values are defined in the satndard, to be used by the MAC and SMT protocols. These are listed in Table 10 together with the the corresponding default values.

FDDI Accessibility Constrains

A comprehensive set of scenarios leading to network inaccessibility will be analysed in this section. As a general rule, we start with very simple single error situations that progressively evolve to less restrictive – and thus more realistic – operating/fault assumptions. For most of the cases, the main analysis is completely general, being particularised for best and worst cases, afterwards.

Frame Stripping

Each FDDI station is responsible for the elimination, from the network ring, of every single frame that it has transmitted. This procedure – known as frame stripping – aims to avoid the drawbacks arising from the repetition of a frame beyond the originating station: managing of frame duplicates at the MAC sub-layer, resource wastefulness, etc.

Frame stripping is initiated when a frame with a source address matching the station individual address is detected. Upon this event, the frame is deliberately truncated by replacing all the remaining data fields with *idle* symbols. This procedure actually leaves a frame remnant on the ring since a set of fields, at the beginning of the frame, were already repeated. However, this does not represent a real disadvantage: no station receives such truncated frames and all them will be eliminated whenever encountering a transmitting station.

When the emitting station fails, before stripping from the network all its transmissions, these no-owner frames will tend to persist on circulating around the ring. Since a FDDI station immediately releases the token after its last transmission, the subsequent failure of that station does not directly lead to network inaccessibility. In a best-case scenario, ring operation is not disrupted: the token will be captured by a downstream station and transmissions originated in the failed station will be eliminated when they arrive to the then-current token user. In a worst-case scenario, the token may be lost as a consequence of the failure[30] and inaccessibility will then occur.

Frame Corruption

Unlike other ring architectures [20], *all* the frame fields of particular relevance for MAC protocol robustness are covered by a *frame check sequence* (FCS). This means that a frame error can be detected with the coverage provided by the *cyclic redundancy check* (CRC) polynomial already employed in the standard ISO 8802 protocols [21]. These frames will not be received by the corresponding MAC entity. Therefore, corruption of MAC protocol data units usually degenerates in other error scenarios.

No Valid Transmissions

The FDDI LAN presents a decentralised control strategy for monitoring ring functionality. A specific timer — *Valid Transmission Timer* (TVX) — is re-started in each station, every time a

[30] This is particularly true in small networks when the token is not captured by any station, within the round immediately following station failure.

non-restricted token[31] or valid frame is received. In the absence of such data units, the TVX timer is not re-started and consequently it will expire in some station. This signals a ring error situation, whose recovery will be tried through the execution of a *token recovery procedure* that includes: a *token claim process* for the election of the new token generator, with a duration given by t_{tcp}, as well as a *token restoration process*, of duration t_{trp}. Assuming that this process succeeds in providing service restoration, the corresponding inaccessibility period will present a duration given by the sum of the time required to detect the error – $t_{dect \leftarrow noVtx}$ – plus the time required for its recovery, as described by equation:

$$t_{ina \leftarrow noVtx} = t_{dect \leftarrow noVtx} + t_{tcp} + t_{trp} \tag{62}$$

The time elapsed between the occurrence of the fault and expiration of TVX represents the ring error detection latency, for this scenario. In the worst-case the fault just occurs after TVX re-start, while in the best-case the data unit triggering TVX re-start is able to complete almost an entire ring round before the occurrence of the fault. Under such conditions the error detection latency is described by equation (63). The value of TVX timer should be made large enough, in order to allow its expiration only in the presence of long-term ring errors, such as token loss. Short-term random noise bursts shall never cause TVX timeout.

$$t_{dect \leftarrow noVtx} = \begin{cases} t_{TVX} - t_{rlat} & \text{best-case scenario} \\ \\ t_{TVX} & \text{worst-case scenario} \end{cases} \tag{63}$$

Upon error detection, a recovery process is entered. Execution of the aforementioned *token claim process* is completely distributed and aims two goals: negotiate among all the network stations the next *operational token rotation time* (T_Opr) and bid for token regeneration [8]. The process is started when the station where TVX has expired, initiates the continuous transmission of *claim_token* frames, carrying its proposal for the *target token rotation time* (TTRT), as well as its individual address. Downstream stations that receive this frame can either accept or reject this proposal.

The proposed *target token rotation time* is accepted by a downstream station whenever its own *target token rotation time* is higher than the proposed one, i.e. this station requests a slower token round time. In such a case, it stores the received *target token rotation time* as the current bidding value for the token rotation time[32], starts to repeat the received *claim_token* frames and, eventually, stops a *token claim process* formerly initiated.

Otherwise, the proposal will be rejected. A downstream station which requests a *target token rotation time* lower than the carried by the *claim_token* frame will either continue to leader a *token claim process* formerly initiated or starts, at this time, their execution, with the transmission of its own *claim_token* frames. An eventual tie between contenting stations will be resolved by station addresses, with the station with the highest address taking precedence.

The *token claim process* successfully terminates when a given station receives its own *claim_token* frames. The exact duration of the *token claim process* depends on whether or not there is a process leadership replacement and on how quickly this leadership propagates to the winning station. In the best-case, token claiming is initiated by the winning station, i.e. by the station with the highest address among those requesting the fastest token rotation. The duration of the *token*

[31] The FDDI LAN allows a restricted token utilization, previously negotiated among stations comited with asynchronous data transfers. Normal operation is performed using non-restricted tokens. Further details can be found in [8].

[32] The MAC receiver stores this value in the T_Bid_Rc register and passes the final negotiated value (T_Neg) to the MAC transmitter, where it becomes the *operational target token rotation time* (T_Opr) upon successful ring initialization. Further details can be found in [8].

claim process is, in this case, essentially defined by the time required to receive back from the ring their own *claim_token* frames and is given by equation:

$$t_{tcp}^{bc} = t_{SD} + t_{CLM} + t_{rlat} \tag{64}$$

A different situation occurs when token claiming is initiated by the lowest precedence station, i.e. by the station with the lowest address among those requesting the slowest token rotation. The previously described algorithm for *target token rotation time* negotiation requires the replacement of the process initiator in the token claiming leadership. Despite the simplicity of this operation, it consumes time and therefore increases the duration of the *token claim process*. So, a worst-case scenario is obtained when this process of leadership replacement successively continuous throughout all the network stations, with each station taking the place of its nearest upstream neighbor, until token claiming control reaches the highest precedence station.

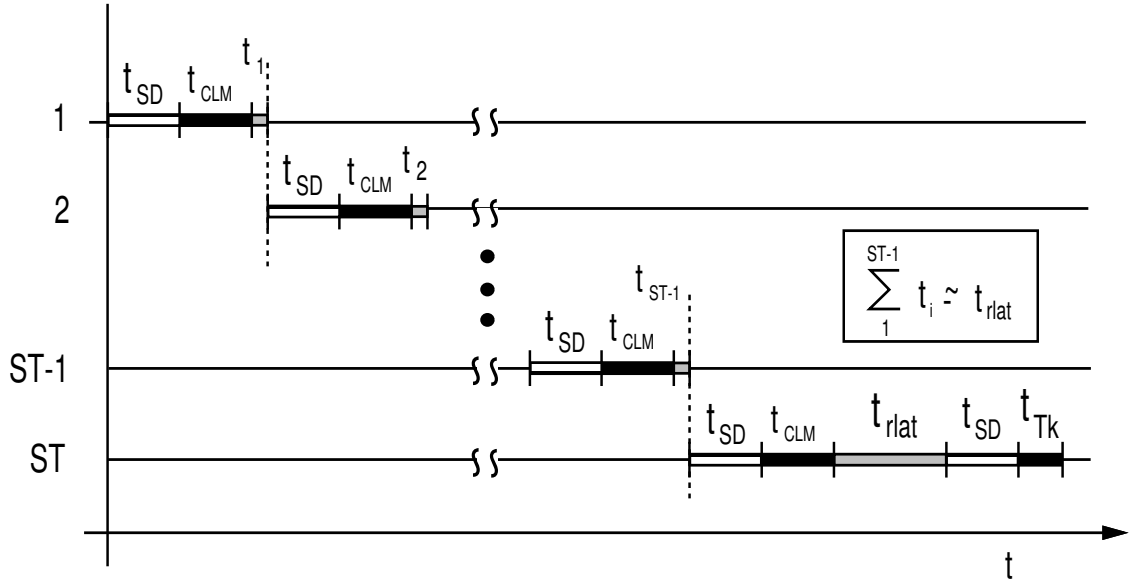

Figure 8: Contention resolution on token claim process (worst-case scenario)

A representation of this process is provided in Figure 8. Such situation only occurs if attachment of individual stations to the network ring is ordered by increased precedence and assumes that the TVX timer does not expire in any other station, but in token claiming initiator. This allows the establishment of an upper bound for *token claim process* duration, as defined by equation:

$$t_{tcp}^{wc} = N_{ST} \,.\, (t_{SD} + t_{CLM}) + 2 \,.\, t_{rlat} \tag{65}$$

Upon termination of the token claim process, the winning station issues a token, that circulating around the ring, performs a very important role in the completion of its recovery. In fact, this initial token is not captured by any station[33], since its purpose is not provide service to transmission queues but rather trigger, in each station, the update of specific MAC operational variables and align the *Token Rotation Timer* (TRT) throughout all network stations. The exact set of actions include the assertion of the *Ring_Operational* flag, the setting to one of the *Late_Ct* counter, as well as, the set of the *operational target token rotation time* (T_Opr) to the newly negotiated T_Neg value[34].

[33]The absence of token capture is due to the non-assertion of *Ring_Operational* flag.

[34]A detailed description of these protocol elements can be found in [8] or, for some of them, later in the paper.

The TRT timer is then set to T_Opr. The time required to perform these actions, in all network stations, an operation that we designate by *token restoration process*, is given by equation:

$$t_{trp} = t_{SD} + t_{Tk} + t_{rlat} \tag{66}$$

Service of transmission queues is started thereafter, but since $Late\_Ct \neq 0$, this service is restricted to the synchronous access class. During the second token rotation, each station accounts the current synchronous bandwidth utilization, and resets the $Late\_Ct$ counter. This enables service of asynchronous latency classes[35] on the third and subsequent token rotations. Therefore, the inaccessibility period is not, for those access classes, merely represented by the time time required to recover the token. Its worst-case value, must be consolidated with the addition of the network access delay upper-bound.

As a matter of fact, such correction is required even in a more restrictive scenario, where only the existence of synchronous traffic is considered. This arises from some side effects that token recovery may have on the normal token circulation. To illustrate this, consider the FDDI network, of Fig. 9. Let us assume that station $\mathcal{S}_1$ has the highest precedence and that the token is garbled, due to random noise, at the output of station $\mathcal{S}_{n-1}$. Ring recovery is initiated upon TVX timeout in some station and the subsequent *token claim process* is wined by the highest precedence $\mathcal{S}_1$ station. Station $\mathcal{S}_n$, which was about to receive the token when the fault occurs, is passed over. So, in addition to the token recovery time, station $\mathcal{S}_n$ must wait an extra token rotation to get access to the network and, therefore, it sees an inaccessibility time added by that delay.

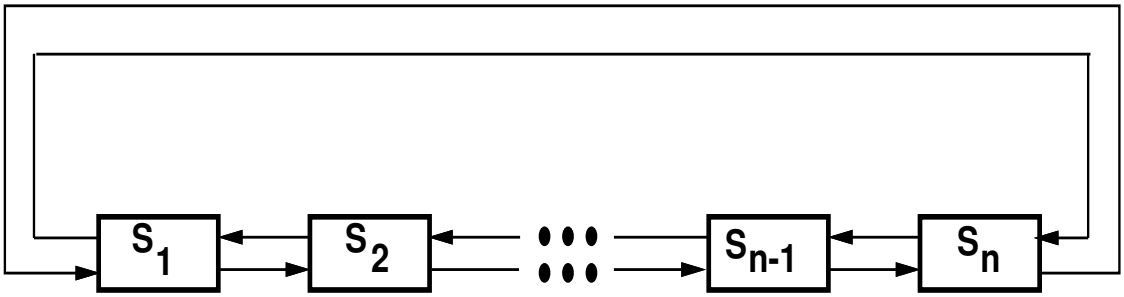

Figure 9: Example of a FDDI network

No Valid Tokens

The mechanism described in the previous scenario has been designed to cope with an abnormal absence of valid frames or with the erroneous persistence of a *restricted* token in the ring [26]. Although that mechanism may represent, for most of the cases, an efficient speed-up alternative to the method herein described for the detecting of incorrect ring activity, it does provide full coverage.

Let us analyse a particular example: a station fails, after having transmitted a frame but before token release. Assuming that the transmitted frame is not garbled, due to an eventual ring interruption, it will persist on the ring. Incorrect ring operation is not detected by TVX timeout, since the persistent frame will cyclely reset that timer.

A decentralised mechanism is provided by the FDDI protocol to cope with such problems. It is based on the timeliness characteristics of token circulation. The affected resources, that remain active in each network station, are the following:

[35] A detailed description of the FDDI latency classes can be found in [8].

- **Token Rotation Timer** (TRT) – under normal ring operation it is re-started with the current value of the *operational target token rotation time* (T_Opr), each time a valid token (restricted or non-restricted) is received with $Late\_Ct = 0$ or when it expires.
- **Late Counter** (Late_Ct) – under normal ring operation this counter holds the number of TRT timeouts since the last received token (restricted or non-restricted). This counter is reset on the arrival of a valid token (restricted or non-restricted) and is incremented on each expiration of TRT. It is set to one upon station initialize or ring recovery.

These MAC elements, together with the *Token Holding Timer*[36], support, within each station, the execution of the timed-token access protocol, scheduling ring transmissions accordingly with the following rules:

- If the token is on time, i.e. if $Late\_Ct = 0$, then THT is set to the current value of TRT, before its re-start with the value stored in T_Opr. Both synchronous and asynchronous frames may be transmitted.
- If the token is late, i.e. if $Late\_Ct \neq 0$, then $Late\_Ct$ is reset but not TRT, in order to retain the accumulated lateness. Only synchronous frames may be transmitted, in this case.
- The time spent in the transmission of synchronous frames may not exceed the station's synchronous bandwidth allocation[37] The time taken by transmission of asynchronous frames is upper-bounded by THT. Service of both classes is prevented after TRT expiration.

This methodology of accessing network medium presents timeliness characteristics guaranting that, under normal operation, a token always return to any given station in a time upper-bounded by $2 \times T\_Opr$. This claim has been formally proofed in [29] and furnishes the foundations for a method aiming the detection of incorrect ring activity:

> Ring recovery is initiated upon TRT has expired twice without any token has been received, i.e. recovery is started when TRT expires and $Late\_Ct > 0$.

Establishment of an expression for inaccessibility duration is straightforward. The ring error detection latency is essentially defined by the value of the TRT timer, as expressed by equation (67). The value associated with TRT corresponds to the previously negotiated *target token rotation time*, which is lower and upper bounded by technological and functional constrains associated either with interoperability issues or with network size [8].

$$t_{dect \leftarrow noTk} = \begin{cases} 2 \,.\, t_{TRT} - t_{rlat} & \text{best-case scenario} \\ \\ 2 \,.\, t_{TRT} & \text{worst-case scenario} \end{cases} \tag{67}$$

Ring recovery is based on the aforementioned *token recovery procedure* and therefore, the corresponding inaccessibility period is given by equation:

$$t_{ina \leftarrow noTk} = t_{dect \leftarrow noTk} + t_{tcp} + t_{trp} \tag{68}$$

Ring Interruption

The previously described mechanisms are able to recover ring operation, whenever its faulty behaviour is not due to a physical cause, like cable break or defective transmitter/receiver. However they are helpless when dealing with such problems.

[36] The *Token Holding Timer* (THT) controls how long a station may transmit asynchronous frames. A station may start an asynchronous transmission as long as THT has not expired and it is lower than the associated priority threshold value. Further details can be found in [8].

[37] Not checked by MAC. Supported by a dedicated protocol, inside SMT [13].

The impact of physical ring interruption on network operation depends on its nature (physical or logical interruption) and duration. Ring interruption errors are essentially recovered by *Station Management* (SMT) protocols, with the eventual cooperation of dedicated MAC sub-layer processes and station self-test capabilities.

Special mechanisms are provided, within SMT, for detecting defective link connections and breaks in the logical ring. The methodology dealing with the first problem is based on the continuous monitoring of inbound line states, whereas the second one requires the intervention of MAC sub-layer. This latter process runs several orders of magnitude slower than line state monitoring [18].

In the next scenarios we perform an analysis of these error detection methods. Further details on the protocols where they are built-in, can be found in the standard documents [13, 8]. Since the corresponding recovery actions are not fully covered by the standard specification we made the reasonable assumption of considering algorithms that can be generically implemented, without violating FDDI constrains and inter-operability issues. Evolution of the standard specification may, nevertheless, originate the revision of this model in order to ensure a more accurate description of network inaccessibility behaviour.

Line State Monitoring

A dedicated protocol inside SMT is responsible for establishing and managing the connection between neighboring ports. This protocol – known as *Physical Connection Management* (PCM) – initializes such a connection and manages the required signalling.

Due to their inherent functions, the PCM is particularly suitable to support the detection of abnormal operating conditions, originated by the failure of PHY[38] and/or PMD[39] layer components. Under correct operation, a stream of *idle* symbols should be received within a period defined by T_Out, from each inbound active link. Should this not occur or otherwise, should *quiet* symbols, *halt* symbols or an extended noise condition be received, and the PCM will attempt re-initialize port connection, since these can be either symptoms of a defective/broken connection or an indication that the other end has just started a connection procedure, probably due to the joining of a new station.

However, in the presence of a failure such an attempt to restore port connection does not succeed and the PCM will either stuck in its entry-point or will cyclely perform successive attempts to re-establish port connection. A high-level SMT entity[40] can monitor these situations, based on indications furnished through the CMT-SMT interface[41] and take a decision on subsequent actions, eventually after having performed some sort of self-testing.

Together with the PCM, another *Connection Management* entity, the *Link Error Monitor* (LEM), performs a role of outstanding importance in the observation of link quality. In fact, the LEM examines the *link error rate* (LER) of an active link, aiming the detection of an inadequate *binary error rate* (BER) due to a marginal link quality, link degradation or connector unplugging. These tests complement any off-line *link confidence test* [13] and their results can be used by the aforementioned SMT entity, namely for fault diagnostics and fault isolation purposes.

[38] The *Physical* layer of FDDI [10].

[39] The *Physical Medium Dependent* layer of FDDI [9].

[40] Currently not specified.

[41] The CMT - *Connection Management* - is a group of *Station Management* protocols that aims to assure port insertion/removal and the connection of port PHY entities to MAC entities. Besides PCM, it includes the *Entity Coordination Management* (ECM) and the *Configuration Management* (CFM).

Physical Ring Recovery

Upon detection of a failure, the network should be reconfigured in order to remove the failed component, from the data path. The reconfiguration of each local station is executed by a CMT entity, the *Configuration Management* (CFM), which is responsible for the interconnection of PHY and MAC components.

Any legal station configuration can be selected. This selection is triggered by relevant changes in the PCM status and is performed accordingly with well defined criteria, taking as a base the description of component's state, held the *Management Information Base* (MIB) [13]. The present standard specification does not clearly define the way how these MIB objects should be managed, i.e. does not define the criteria that rule component's state update, from the indications furnished upon PCM status change. This means that a fault detection and fault isolation policy should be formally defined, hopefully, in the near future. Meanwhile, and in order to complete our study of networking inaccessibility, we postulate a set of reasonable assumptions, to which any fault detection and isolation policy should yield.

- Networking components failures can be reliably detected through SMT protocols, eventually assisted by self-test procedures.
- Fault detection procedures are time bounded.
- Error recovery, to be performed either by station removal or through network physical reconfiguration, is also time bounded.

In order to define an expression for the inaccessibility period due to the previously described reconfiguration procedure, let us define the following parameters:

- $t_{PCM-Dect}$ – The PCM error detection latency.
- $t_{st \leftarrow join}$ – The interruption on ring data flow, due to the joining of a given station.
- $t_{st \leftarrow leave}$ – The duration of ring interruption, due to the withdraw of a station.
- $t_{self-test}$ – The period required for a station reliably test their network components.
- t_{scrub} – The time required for the execution of the *scrub function*. The *scrub* function is an error prevention measure that aims to purge from the ring all the data units, issued before reconfiguration takes place.

The physical interruption of the network ring is therefore detected by *line state monitoring* and recovered through network reconfiguration. The duration of the ring interruption is given by equation (69), where we have specifically considered the time taken by station withdraw, for self-testing, and by the correct station join or, alternatively, the establishment of a connection between their upstream and downstream neighbors, if both stations have failed.

$$\begin{aligned} t_{PhyBrk} \quad = \quad & t_{PCM-Dect} + t_{st \leftarrow leave} + t_{self-test} + \\ & t_{st \leftarrow join} + t_{scrub} \end{aligned} \tag{69}$$

Additionally, notice that interruption of physical ring directly leads to the collapse of logical ring, whose systematic analysis is performed in the next scenario. The inaccessibility period due to the need of network physical reconfiguration can then be easily obtained from the expressions of next scenario, by considering that physical and logical ring interruption present the same duration.

Logical Ring Recovery

A set of faults may lead to data flow interruption, although the physical ring remains intact. Let us to group such errors under the generic designation of *logical ring interruption*, whether or

not they may have a physical cause. The two methodologies earlier described for restoration of token circulation can be view as particular cases of a logical ring collapse, where the interruption is essentially due to transient operational conditions rather than to severe failures.

In those scenarios, error recovery is started immediately after its detection, but that will not be the case when the error condition persists beyond their detection. Let us denote by $t_{RingBrk}$ the duration of ring interruption. The possible error scenarios are as follows:

i) $t_{Dect} \leq t_{RingBrk} < t_{MacBeacon}$ — with $t_{MacBeacon} \simeq t_{Dect} + T\_max$, where t_{Dect} represents the ring error detection latency bounds, as given by equations (63) and (67), and T_max is the maximum value for the *target token rotation time* [8]. This last parameter is several times the maximum ring initialization time, in order to allow a stable ring recovery, and places a deadline for termination of token claiming recovery actions. Should the network ring data path be restored in a time that allows the successful execution of token claiming and its operation is able to be recovered strictly through the set of actions associated with the aforementioned *token recovery procedure*. The corresponding inaccessibility period is directly proportional to the logical ring interruption duration, as given by equation:

$$t_{ina \leftarrow noRing}(noBeacon) = t_{RingBrk} + t_{tcp} + t_{trp} \tag{70}$$

ii) $t_{MacBeacon} \leq t_{RingBrk} < t_{DirBeacon}$ — where $t_{DirBeacon}$ is defined ahead. Termination of the *token claim process* previously used to recover the logical ring is monitored through the TRT timer. Upon that process initiation, the TRT timer is always loaded with the T_max value. Should the error condition persist beyond this timeout value and TRT timer will expire. This usually means that a more severe fault has occurred on the ring and that a stronger method for its recovery is required. Such recovery procedure, known as *MAC beacon process*, is started with the continuous transmission of *MAC beacon* frames and ends when a station receives back their own *MAC beacon* transmissions. The *MAC beacon process* aims to provide a simple logical ring recovery method for medium-term breaks [32], like for example the withdrawal of active stations from the network ring. Provided that the ring break duration lies within the aforementioned limits, the *MAC beacon process* successfully terminates and the corresponding inaccessibility time is given by equation:

$$t_{ina \leftarrow noRing}(MacBeacon) = t_{RingBrk} + t_{brp} \tag{71}$$

where t_{brp} – *beacon recovery process* – represents the time required to complete the restoration of network service. It includes the propagation of a *beacon* frame around the ring, the bidding of a new *operational target token rotation time* and the corresponding token issuing.

$$t_{brp} = t_{BCN} + t_{rlat} + t_{tcp} + t_{trp} \tag{72}$$

iii) $t_{DirBeacon} \leq t_{RingBrk}$ — A dedicated protocol within SMT monitors ring activity. This protocol, known as *Ring Management* (RMT), tracks MAC activity in order to detect when the ring is not operational (claiming or MAC beaconing). Among other functions [13], RMT aims to detect "stuck beacon" conditions, i.e. situations of logical ring interruption that are unable to be recovered by the *MAC beacon process*, presumably because a reconfiguration that cannot be performed by the methods early described is required. Under such scenario we may find the failure of components located above the PHY level. Execution of *MAC beacon process* is allowed during a time given by:

$$t_{DirBeacon} = T\_Non\_Op + T\_Stuck \tag{73}$$

where T_Non_Op is the time allowed for ring recovery, before examining any other error condition and T_Stuck is the time granted, after that, for *MAC Beacon* transmissions [13]. Execution of a *direct beacon process* is entered after this period. This process is used to notify the ring that a "stuck beaconing" condition have been detected through the transmission of *direct beacon* frames, usually addressed to that station upstream neighbor. These frames are transmitted only for a

short period of time, before the *PC-Trace* function is invoked. The *PC-Trace* function uses the PHY level signalling to identify the fault domain, which is defined between the MAC of the beaconing station and its nearest upstream MAC. Execution of the *PC-Trace* function requires the cooperation of the PCMs and ECMs within the fault domain. The recovery actions are actually only triggered upon execution of the *PC-Trace* function. After the *trace request* has been propagated to nearest upstream station with an inserted MAC, this function will be acknowledged downstream to initiating station. Testing of *data paths* [13] will be performed, at this point, by stations located at the edges of the fault domain. Failed components and/or stations should then be removed in order to restore network operation.

Let us to denote by $t_{PC-Trace}$ the time required for the execution of the *PC-Trace* function and by t_{path_test} the time needed to reliable test the data path local to a given station, as suggested in [13]. The duration of the inaccessibility period for this scenario is given by:

$$t_{ina \leftarrow noRing}(DirBeacon) = \begin{array}{c} t_{DirBeacon} + T\_Direct + \\ t_{PC-Trace} + t_{st \leftarrow leave} + t_{path\_test} + \\ t_{st \leftarrow join} + t_{scrub} + t_{brp} \end{array} \quad (74)$$

where T_Direct is the period of time during which *direct beacon* frames are transmitted, before the *PC-Trace* function is invoked. Equation (74) accounts the time required for station withdrawal before its testing, the time needed for the join of the station that remains correct or, alternatively, for the establishment of a connection between their upstream and downstream neighbors, if both stations have failed. Naturally it also accounts the time required to purge the ring of all the data units issued before reconfiguration.

Dumb Transmitter

Having generically studied FDDI inaccessibility, let us now consider some particular failure scenarios. We start with a scenario where a station is not able to transmit a signal to the medium. This may include failures in the transmit section of the PHY layer, as well as, the failure of optical transmitter, at the PMD sub-layer.

This station downstream neighbor detects this situation upon reception of *quiet* symbols originated by the absence of medium signalling. This starts the aforementioned PCM actions for fault detection on both sides of the connection, which allows the station with the dumb transmitter to detect its fault. Let us assume that detection of the abnormal signalling condition is performed by the PCM in a time given by $t_{PCM-Dect}(noTx)$. Known a numeric value to this parameter, and the corresponding inaccessibility period can be easily evaluated.

Deaf Receiver

Let us assume that the optical receiver or the associated section of the PHY layer fail by starting to report only *quiet* symbols. This situation is essentially equivalent to the previous scenario, so we made $t_{PCM-Dect}(noRx) = t_{PCM-Dect}(noTx)$.

Broken Cable

The symptoms of this failure are equivalent to those reported in the previous scenarios. However, we assume that fault detection is more complex and thus $t_{PCM-Dect}(noLink) > t_{PCM-Dect}(noTx)$. Recovery from a broken cable is achieved by wrapping the data path into the secondary ring.

Jabbering Transmitter

Let us assume that the transmitter of a given station fails by stucking in the send of a continuous stream of data. Clearly, the *token claim process* does not succeed, since that station is not able to

repeat *claim token* frames. The problem is detect by PCM timeout, after a period of T_out without any *idle* symbol. The diagnose and recovery procedures are similar to those of previous scenarios. The PCM latency must now obey to relation $t_{PCM-Dect}(JabTx) > T\_Out$.

Streaming Receiver

In this scenario we consider that the receiver section fails in the decoding of network control signal, thus feeding a continuous stream of data to both PCM and MAC inputs. The scenario is quite similar to the jabbering transmitter and therefore, we consider $t_{PCM-Dect}(StrRx) = t_{PCM-Dect}(JabTx)$.

Streaming MAC Receiver

In this scenario, we consider that the receiver section of a given station, although feeding a correct signal to PCM inputs, fails provision of correctly decoded data to the MAC interface. In this case, the PCM will not detect any abnormal condition and therefore does not initiates any fault isolation procedure.

From MAC operation viewpoint, the absence of correct data reception leads the station to a "stuck beacon" condition. The RMT will then start *direct beacon* and *PC-Trace* executions. The corresponding inaccessibility duration is given by equation (74).

Station Join

Stations are physically inserted in the network ring through the switching of a dedicated optical *bypass* switch. The PCM of the just arrived station, as well as, of their neighboring ports will then initiate the connection procedure. We consider that execution of a station join takes a time given by $t_{st \leftarrow join}$. The resulting inaccessibility period is obtained by considering a ring interruption with that duration.

Multiple Joins

In the best-case all the stations will simultaneously join to the network ring and the situation will not be distinguished from the insertion of a single station. In the worst-case, station joins are serialised. An upper-bound for ring interruption duration can be obtained assuming a massive join to a network previously formed by only two stations.

$$t_{RingBrk \leftarrow mjoin} = (N_{ST} - 2) \, . \, t_{st \leftarrow join} \tag{75}$$

Station Leave

A station leaves the ring by switching their optical relay to the *bypass* state. Afterwards, the new adjacent ports located upstream and downstream of the missing station must complete a connection procedure. The corresponding inaccessibility period is obtained by considering a ring interruption duration given by $t_{st \leftarrow leave}$.

Multiple Leaves

In the best-case, all the stations simultaneously abandon the network ring and the situation cannot be distinguished from the withdraw of a single station. In the worst-case, these events are

serialised and upon a massive withdraw of $(N_{ST} - 2)$ stations, the duration of ring interruption is given by:

$$t_{RingBrk \leftarrow mleave} = (N_{ST} - 2) \,.\, t_{st \leftarrow leave} \tag{76}$$

FDDI Analytic Results

In order to complete our study of networking accessibility we now evaluate the inaccessibility time bounds, for a FDDI installation, 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 $N_{ST} = 32$. Estimates for other network parameters are defined: $t_{st \leftarrow join} = 30ms$, $t_{st \leftarrow leave} = 20ms$. The implementation dependent parameters associated with self-test procedures are also estimated: $t_{self\_test} = t_{path\_test} = 5ms$. Finally we also estimate a set of parameters associated with PCM behaviour under different operation conditions: the parameters associated with the detection of faults triggered by reception of *quiet* symbols $t_{PCM\_Dect}(noTx) = t_{PCM\_Dect}(noRx) = 15ms$, $t_{PCM\_Dect}(noLink) = 25ms$; and the parameters associated with the detection of faults by timeout in the reception of *idle* symbols $t_{PCM\_Dect}(JabTx) = t_{PCM\_Dect}(StrRx) = 115ms$. The results of the evaluation, for each one of the studied scenarios, are summarized in Table 11.

Data Rate - $100Mbps$		
Scenario	$t_{ina}\ (ms)$	
	min.	*max.*
No Valid Transmissions	2.53	2.76
No Valid Tokens	15.03	15.26
Dumb Transmitter	77.15	77.36
Deaf Receiver	77.15	77.36
Broken Cable	87.15	87.36
Jabbering Transmitter	177.18	177.39
Streaming Receiver	177.18	177.39
Streaming MAC Receiver	9457.18	9457.33
Station Join	30.03	30.24
Multiple Joins ($N_{ST} = 32$)	30.03	900.29
Station Leave	20.03	20.24
Multiple Leaves ($N_{ST} = 32$)	20.03	600.29

Table 11: FDDI Inaccessibility Times ($l_{add} = 48$)

The worst case figures are rather high, like for example the $9.5s$ required for the removal of a streaming MAC. However, note three points: the parameter estimates are quite conservative, the network parameters are based on the default values and in consequence inadequate for a network with at most 32 stations and finally the probability of occurrence of some worst-case scenarios is very small. As a minimizing strategy we envisage the tuning of timing references with network size. This can bring some of the figures drastically down.

The study is interesting on the grounds that it provides a basis to know what to expect from FDDI 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.

6 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[50].

This paper does an exhaustive study of the inaccessibility characteristics of a set of standard token-based local area networks, namely ISO 8802/4 Token-Bus, ISO 8802/5 Token-Ring and ISO 9314 FDDI, addressing the problem of temporary LAN partitioning in a comprehensive way. The derived error-handling performance models allows the evaluation of corrective terms for computing transmission delays in various situations, for each of these LANs.

References

[1] D. W. Andrews and G. Schultz. A token-ring architecture for local area networks. In *IEEE COMPCON*, October 1982.

[2] Werner Bux. Local-area subnetworks: a performance comparison. *IEEE Transactions on Communications*, 29, October 1981.

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

[4] M. Alex Colvin and Alfred C. Weaver. Performance of single access classes on the IEEE 802.4 Token Bus. *IEEE Transactions on Communications*, 34(12), December 1986.

[5] Flaviu Cristian. *Synchronous Atomic Broadcast for Redundant Broadcast Channels.* Technical Report , IBM Almaden Research Center, 1989.

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

[7] Doug Dykeman and Werner Bux. Analysis and Tuning of the FDDI Media Access Control Protocol. *IEEE Journal on Selected Areas In Communications*, 6(6), July 1988.

[8] FDDI. *FDDI Token-Ring Media Access Control (MAC).* ANSI X3.139, 1987.

[9] FDDI. *Physical Layer Medium Dependent (PMD).* ANSI X3.166, 1990.

[10] FDDI. *Physical Layer Protocol (PHY).* ANSI X3.148, 1988.

[11] FDDI. *Shielded Twisted Pair Physical Medium Dependent (STP-PMD).* ANSI X3T9.5/ 91-166 IBM Initial Draft Proposal, 1991.

[12] FDDI. *Single-Mode Physical Layer Medium Dependent (SMF-PMD).* ANSI X3T9.5/ 88-155 Draft Proposal Rev 3, 1989.

[13] FDDI. *Station Management (SMT).* Draft Proposal ANSI X3T9.5/91 SMT-LBC-161, January 1992.

[14] FDDI. *Unshielded Twisted Pair Physical Medium Dependent (UTP-PMD).* ANSI X3T9.5/ 91-170 Draf Proposal Rev 0, 1991.

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

[16] 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.

[17] 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.

[18] J. Hamstra. *FDDI Ring Reconfiguration.* SMT 40 - X3T9.5 working document, January 1987.

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

[20] *ISO DP 8802/5-85, Token Ring Access Method.* 1985.

[21] Raj Jain. Error characteristics of Fiber Distributed Data Interface. *IEEE Transactions on Communications*, 38(8), August 1990.

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

[23] Dittmar Janetzky and Kym S. Watson. Token bus performance in MAP and Proway. In *IFAC Workshop on Distributed Computer Protocol System*, 1986.

[24] Anura P. Jayasumana. Comment on "performance of single access classes on the IEEE 802.4 Token Bus". *IEEE Transactions on Communications*, 36(2), February 1988.

[25] Anura P. Jayasumana. Throughput analysis of the IEEE 802.4 priority scheme. *IEEE Transactions on Communications*, 37(6), June 1989.

[26] Marjory Johnson. Reliability mechanisms of the FDDI high bandwidth Token-ring protocol. *Computer Network and ISDN Systems*, 11(2), 1986.

[27] Marjory J. Johnson. Analysis of FDDI synchronous traffic delays. In *Procs. of Systems Design and Networks Conference: Patting Local Area networks t Work*, January 1988.

[28] Marjory J. Johnson. Performance analysis of FDDI. In *Procs. of EFOC/LAN'88*, Amsterdam, April 1988.

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

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

[31] Lawrence J. Lang and James Watson. Connecting remote FDDI installations with single-mode fiber, dedicated lines, or SMDs. *Computer Communication Review (ACM Sigcomm)*, 20(3):83–94, July 1990.

[32] My T. Le. *FDDI Fault Detection and Recovery Mechanisms.* SMT 287 - X3T9.5 working document, February 1989.

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

[34] *Manufacturing Automation Protocol (MAP) specification V2.1.* March 1985.

[35] H. Okada, T. Yamamoto, Y. Nomura, and Y. Nakanishi. Comparative evaluation of Token-Ring and CSMA/CD medium access protocols in LAN configurations. In *Computer Networking Symposium*, IEEE, December 1984.

[36] Jeffery H. Peden and Alfred C. Weaver. Performance of Priorities on an 802.5 Token Ring. In *SIGCOM'87 Symposium*, ACM, 1987.

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

[38] 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.

[39] Floyd Ross. An Overview of FDDI: The Fiber Distributed Data Interface. *IEEE J. on Selected Areas in Comm.*, 7(7), 1989.

[40] J. Rufino and P. Veríssimo. A study on the inaccessibility characteristics of ISO 8802/4 Token-Bus LANs. In *IEEE INFOCOM'92 Conference on Computer Communications*, IEEE, Florence, Italy, May 1992. (to appear).

[41] J. Rufino and P. Veríssimo. Minimizing Token-Bus inaccessibility through network planning and parameterizing. In *EFOC/LAN92*, IGI, Paris, France, June 1992. (to appear).

[42] Alexander Schill and Martin Zieher. Performance Analysis of the FDDI 100Mbit/s Optical Token Ring. In *IFIP TC6/WG 6.4 Workshop on HSLAN*, Aachen, February 1987.

[43] K. C. Sevcik and M. J. Johnson. Cycle time properties of the FDDI token ring protocol. *IEEE Transactions on Software Engineering*, 13(3), March 1987.

[44] N. Strole. A local communications network based on interconnected token-access rings: a tutorial. *IBM J. Res. X Development*, 27(5), September 1983.

[45] Jay K. Strosnider and Thomas E. Marchok. Responsive, deterministic IEEE 802.5 token-ring scheduling. *Real-Time Systems*, 1(2), September 1989.

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

[47] *Token-Ring Adapter Chipset TMS380 User's Guide*. 1985.

[48] Jũrgen Tusch, Heinrich Meyr, and Erwin A. Zurfluh. Error handling performance of a Token Ring LAN. In *13th Conference on Local Computer Networks*, IEEE, Minneapolis, USA, October 1988.

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

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

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

[52] On-Ching Yue and Charles A. Brooks. Performance of the timed token scheme in MAP. *IEEE Transactions on Communications*, 38(7), July 1990.