ENFORCING REAL-TIME BEHAVIOUR ON LAN-BASED PROTOCOLS

INESC Technical Report AR/__-91

P.Veríssimo,J.Rufino,L.Rodrigues

September 1991

To appear in the Proceedings of the 10th IFAC Workshop on Distributed Computer Control Systems, IFAC, Semmering, Austria, September 1991

ESPRIT II P2252
DELTA4 - PHASE 3

Enforcing Real-Time behaviour on LAN-based protocols

Abstract:

Local area networks form the basis of a number of distributed real-time systems. While current standard technology does not solve all problems related with achieving real-time behaviour — in essence, bounded and known message delivery delays — designers should be able to build reliable LAN-based real-time systems now, using existing technology, VLSI, and namely, non-replicated architectures. Methods and techniques that make this possible are worthwhile being studied. This paper describes the concepts and design options used in Delta-4 to overcome the problems posed by reliable real-time operation over standard LANs, whose only redundancy exists at the physical medium level.

To appear in the Proceedings of the 10th IFAC Workshop on Distributed Computer Control Systems, IFAC, Semmering, Austria, September 1991

Partner: INESC

Authors: P. Veríssimo, J. Rufino, L. Rodrigues

Workpackage: 4

Date: November 1991 **Reference:**

Enforcing Real-Time behaviour on LAN-based protocols

Paulo Veríssimo, José Rufino, Luís Rodrigues
Technical University of Lisboa
INESC*
e-mail:...paulov@inesc.pt

Abstract

Local area networks form the basis of a number of distributed real-time systems. While current standard technology does not solve all problems related with achieving real-time behaviour — in essence, bounded and known message delivery delays — designers should be able to build reliable LAN-based real-time systems now, using existing technology, VLSI, and namely, non-replicated architectures. Methods and techniques that make this possible are worthwhile being studied. This paper describes the concepts and design options used in Delta-4 to overcome the problems posed by reliable real-time operation over standard LANs, whose only redundancy exists at the physical medium level.

1 Introduction

Local area networks form the basis of a number of distributed real-time systems. It is mandatory that they display real-time behaviour — in essence, bounded and known message delivery delays — and an interesting fact is that structure alone does not determine real-time behaviour of a LAN. As an example we find a discussion by Janetzky & Watson [1] on the need to account for individual offered load patterns, to avoid violation of the bounded access delay offered in principle by ISO 8802/4 token-bus networks. Peden shows settings where the scheme used by LANs such as the ISO 8802/5 token-ring, to enforce priority-based precedence on the network can fail, thus leading to priority inversion [2]. Kopetz dismisses the claim that Ethernet is not adequate for real-time, by superimposing a collision avoidance mechanism, based on global time, on the bare ISO 8802/3 protocol [3]. LeLann quite rightly points out that new LAN protocols are necessary for real-time and that the existing ones have shortcomings with regard to determinism [4].

These facts call for a systemic approach to substantiate any claims of real-time behaviour of LAN-based systems. Namely, to dress an elementary LAN with a model and a service, composing a subsystem on which the correctness of other system services lies — for the matter of this paper, time-domain correctness or *timeliness*. This approach was followed for example in the Mars system [3], which assumes a duplicated Ethernet LAN, and in the AAS system, which uses n-plicated token-rings [5]. Both architectures are based on a communication service with a well-defined subsystem interface offering certain properties. The fact that the network is replicated offers resilience against failure of a single medium.

Despite what was said in the opening paragraphs, designers should be able to build LAN-based real-time systems *now*, using existing technology, VLSI, etc. Methods and techniques that make this possible are worthwhile being studied. One question to be asked is whether one can reliably obtain real-time behaviour out of single, non-replicated LANs. The Delta-4[1] system architecture uses standardised LANs, whose only redundancy may exist at the physical — electrical signalling in the medium — level [6]. The reader should observe that this leads to a simpler system and to a cost-effective network infrastructure. The quality of service achieved satisfies a wide spectrum of applications [7], with exception

*Instituto de Engenharia de Sistemas e Computadores, R. Alves Redol, 9 - 6° - 1000 Lisboa - Portugal, Tel.+351-1-545150. This work has been supported in part by the CEC, through Esprit Project 1226 - DELTA-4.

[1]Delta-4 is a CEC Esprit II consortium, formed by Ferranti-CSL (GB), Bull (F), Credit Agricole (F), IEI (I), IITB (D), INESC (P), LAAS (F), LGI (F), MARI (GB), NCSR (GB), Renault (F), SEMA (F), Un. of Newcastle (GB), designing an open, dependable, distributed architecture.

of those critical ones which require glitch-free continuity of service provision, or which have very narrow timeliness and synchronism specifications. For these exceptions, specialised space-redundant architectures are recommended.

This paper describes concepts, design options and lessons learned in Delta-4, to overcome the problems posed in achieving reliable real-time operation of standard LANs. In short, a number of conditions to achieve that aim are enumerated. Then, we establish a set of attributes for an abstract low-level frame delivery service and show that such a service fulfils the reliable real-time operation requirement. Finally, we show that "dressing" a given LAN with the necessary mechanisms and protocols to implement the service is easily achievable. One of these mechanisms is an innovative technique to tolerate partitions in real-time networks, based on a concept called *inaccessibility*.

2 Reliable Real-Time communications requirements

It is assumed that the faulty behaviour LANs display consists of timing failures (delays) due to overload; omission failures (lost frames), due to transmission errors; network partitions (eg. due to medium failure).

The requirement for reliable real-time operation of a communications subsystem is:

> **RT -** *A reliable real-time network displays bounded and known message delivery latency, in the presence of disturbing factors such as overload or faults.*

Secondarily, the subsystem should recognise *urgency*, i.e. the fact that some messages may get through head of others. This is one mechanism for propagation of priorities in a distributed setting, but practical systems will actually provide two urgency classes at this level: critical and non-critical traffic. All considered LANs provide this distinction through priorities.

In trying to obtain this behaviour out of a LAN, the failure modes just enumerated must be taken into account[2]. Their effects must in some way be tolerated or limited. The conditions presented below are sufficient to achieve requirement **RT**:

1. **enforce bounded delay from request to transmission of a frame[3], given the worst case load conditions assumed (avoid timing failures);**
2. **ensure a message[4] is delivered despite the occurrence of omission failures (tolerate omission failures);**
3. **control partitions.**

Condition 1 makes sure that any frame is sent within a known time bound, even if it does not arrive. Condition 2 ensures that a message is delivered, even if that implies, for example, the transmission of several frames to tolerate omissions. Condition 3 is mandatory for real-time systems. Since real-time systems are supposed to make progress within more or less rigid time constraints, they do not tolerate partitions in general. It is proposed to admit a class of *controlled* partitions, which can be tolerated with the necessary measures.

3 The Abstract Network Model

A model for a network displaying reliable real-time operation has been advanced in [8] and laid down in [9]. It was called the *abstract network*.

The idea is to consider a set of networks of a given type (LANs in the case) and be able to define a set of common properties abstracting from their physical particularities. The abstract network thus forms a low-level service, useful to build complex protocols on top of, and rendering them LAN-independent. In this case, it models the standardised ISO LANs[5] (8802/x, FDDI). The abstract network attributes concerning this discussion can be enumerated as follows:

An1 - Every frame queued for transmission is transmitted by the network within a bounded delay $T_{td} + T_{ina}$.

[2] As well as architectural factors such as the fact that the LAN is not replicated, making transmission errors unavoidable.
[3] LAN level information packet.
[4] User level information packet.
[5] Including the 8802/3 CSMA/CD, if the deterministic variant [10], although not standard, is considered.

(a) transmission-with-reply

```
tries := 0;    Resp := empty
do tries < nrTries  ∧ Resp ≠ full →
           Resp := empty;
           Tx(data,id_tries);
waitRepliesPutIn(TwaitReply, Resp);
tries := tries +1           od
```

(b) diffusion

```
tries := 0;
do tries < nrTries →
           Tx(data,id_tries);
tries := tries +1           od
```

Figure 1: k-omission tolerant protocol: (a) acknowledge based; (b) diffusion based.

An2 - A network, in a known interval T_{rd}, may do at most k consecutive omission errors, either caused by a transmitter or a receiver or the medium[6].

An3 - A network, in a known interval T_{rd}, may be inaccessible at most i times.

T_{td} is the worst-case delay from request to end of transmission, in a normally operating network. T_{ina} is the maximum duration of a network inaccessibility period. Let us accept at this point that *inaccessibility* is a period when the network does not provide service, although remaining operational (take the example of the period from token-loss to recovery in a token LAN). It will be defined shortly.

Enforcing a bounded transmission time

Enforcing a bounded and known transmission time bound, T_{td} (An1), has not only to do with controlling the overall load offered by peer traffic, but also with distinguishing among classes of urgency. Higher urgency traffic may thus be guaranteed enough of the channel bandwidth to fulfil its latency requirements, in detriment of lower urgency ones. However, the latter may use the channel during idle periods, improving bandwidth efficiency of the communication system. This concerns:

- the own mechanisms of LANs — to ensure that load offered to the network by nodes is controlled in a way to achieve that aim;
- user-level load control — in terms of bounded and known average arrival rate and minimum inter-arrival time of transmission requests at each node.

LAN MAC-level priorities allow in principle urgent frames to overtake less urgent ones on the network although it has been shown in [2] that the scheme may be upset in real settings. We will not spend too much time with these issues, since they have been studied by a number of authors[1,11,12,13,14].

Handling omission failures

The **bounded omission degree**[7] assumption introduced in **An2** is very helpful as the foundation of basic error processing protocols with deterministic termination — crucial for real-time operation. Other properties of higher-level reliable broadcast/multicast protocols are easily implemented above the omission-free abstract network.

The assumption is realistic and it is based on the observation that omission errors are rare in LANs but may occur in bursts. Additionally, it is reasonable, for the limited interval of a protocol execution, to make the single fault assumption. In consequence, these omission bursts derive from the failure of a

[6] In receiver or medium failures, errors as perceived by the recipient are consecutive; in the case of consecutive transmitter failures, these may be interleaved with good transmissions from other points, from the recipient's viewpoint.

[7] We call *omission degree (Od)* to the number of consecutive omissions produced by a component.

single component [9]. A number of ways of handling omission failures are possible. Two alternative ways, based respectively on detection/recovery and masking of omission errors, are presented in figure 1. If k is the maximum omission degree as per An2, then $NrTries = k + 1$.

The detection/recovery algorithm is implemented through transmission-with-reply rounds[8]. Since error rate is expected to be low, this is optimal for the average case. After transmission, replies from a number of recipients are awaited for; replies are put in a bag $Resp$, which becomes full when it has all expected replies. In absence of errors there is only one try. The $waitRepliesPutIn$ function waits at most during $TwaitReply$, after which it returns with the replies it got. Note that the several tries to send message with reference id are identified by an index $_{tries}$ (l.53). Reference id allows detection of duplicates. The version in (a) of the figure was presented in [9]. It seeks a completely correct series, i.e. one where all recipients receive a given try and all replies are got, in order to enforce total order among competing LAN transmissions[9]. A minor variation consisting of deleting line 52, yields an unordered version.

The masking algorithm — (b) in the figure — is diffusion-based and in principle obtains the lowest worst-case delivery delay, given that it systematically repeats a transmission $k+1$ times, without waiting for any replies or error-detecting timeouts. However, note that if $T_{waitReply} \approx T_{td}$, both expressions for worst-case delivery delay will yield similar values. Given An2, after its execution at least one instance of message id arrives at every recipient. However it introduces a fixed overhead in processing, increases network load, and total order is not ensured.

The transmission-with-reply method allows detection of failure, i.e. absence of reply after $k + 1$ tries. This also works in the case of coverage failure of the bounded omission degree assumption (the abstract network detects the fact; it will be up to the higher-level protocol to handle it). In this sense, transmission-with-reply is safer than diffusion, in case coverage of An2 is not considered high enough. This is very important in real-life networks, where large noise bursts can arise. Even if the medium is fault-tolerant, reconfiguration may not occur fast enough to avoid all diffusion repetitions to fail.

The transmission-with-reply method, being bi-directional, requires a tighter coupling than diffusion, between sender and recipients. Overall scalability is similar, if a group orientation (multicast) is considered. In fact, average group dimension remains rather stable with system size increase. With regard to scalability of group dimension, diffusion is obviously better. An additional feature of transmission-with-reply is that the replies can also convey information from the recipients in a performance-efficient manner. This information may be useful for higher-level protocols built above the abstract network; this feature was used in [9]. A comparison between both methods is summarized in table 1.

Anyway, if bounded transmission delay T_{td} is ensured, either of these mechanisms implementing **An2** satisfy the second condition to attain real-time behaviour: message delivery despite omission failures.

The main feature of the bounded omission degree technique, whichever transmission method used, is that it has deterministic termination, i.e. it executes within a bounded and known time, in absence of partitions.

Controlling Partitions: Inaccessibility

The third condition is to control partitions. It is the trickiest problem to solve and it will be discussed at some length in the sequel of the paper.

Let a network be partitioned when there are subsets of the nodes which cannot communicate with each other[10]. Remember that the LAN is not replicated,

so there are a number of causes for partition in LANs: bus medium failure (cable or tap defect), ring disruption, transmitter or receiver defects; token loss; etc.

A first observation is that, although partitions are undesirable in real-time, the LAN will at least have glitches in operation, since it is not replicated. Some standard LANs have embedded means of recovering from some of the situations described above (eg. token regeneration for token-based LANs).

[8] This is different from the LLC type 3 service, namely because it is multipoint and because replies can convey semantically useful information.

[9] Besides, merely conferring transmission reliability, the ordering property of this low level protocol may be used to build very efficient atomic protocols [8].

[10] 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.

Features	Tx-with-reply	diffusion
execution time	lowest w/ no faults	lowest w/ w-case faults
w-case deliv. delay	$k.T_{waitReply}+$ $+T_{td}$	$(k+1).T_{td}$
no-fault deliv. delay	equal	equal
directionality	bi-	uni-
scalability (overall) (of groups)	equal	equal highest
proc. overhead		highest
network load		highest
total order	possible	not possible
failure detection	yes	no
upper-layer info in reply frame	possible	not applicable
resilience to lack of coverage	high (detects violation)	none

Table 1: Comparison between transmission-with-reply and diffusion methods.

The aim is to control all of them.

The solution is based on a very simple idea: if one knows for how long a network is partitioned, then synchronous, real-time operation of the system is possible, provided that those glitch periods are acceptably short. Let us call them periods of *inaccessibility*, to differentiate from classical partitions.

Inaccessibility has been introduced in [8]. Its formal definition in [9] is recalled here:

> *Certain kinds of components may temporarily* refrain *from providing service, without that having to be necessarily considered a failure. That state, that we call* **inaccessibility**, *is definable, if:*
>
> - **(i) it can be made known to the users of the component;**
> - **(ii) inaccessibility limits (duration, rate) are part of the component specification;**
> - **(iii) violation of those limits implies permanent failure of the component.**

We now explain our approach to control partitions, based on transforming them in inaccessibility periods. All that is necessary is for the abstract network implementation to complement LAN functionality in order to:

- recover from all conditions leading to partition, i.e. reestablish connectivity among affected nodes;
- ensure that the inaccessibility periods have a bound and that it is suitably low for the service requirements;
- accommodate inaccessibility in protocol execution and timeliness calculations.

This is not hard to implement, as shown in the next section. In consequence, inaccessibility will: represent a set of "allowed" temporary partitions; stipulate limits for the duration of the resulting glitches during a protocol execution; and require of the network components some self-assessment capability. This way, all partitions are *controlled*. Uncontrolled partitions are of course still possible, because systems do fail, but that event means the total and permanent failure of the real-time communications subsystem.

An3 is attained through this methodology, and a network exhibiting An3 satisfies the third condition to achieve real-time behaviour enumerated in section 2.

4 Implementing inaccessibility control

Given the concept and shown it is the foundation for reliable real-time operation of single LANs subject to partitions, design criteria and protocols to set-up an abstract network fulfilling **An3** are presented next.

First, one must recover from all conditions leading to partition. For example, recovering from physical — mechanical or electrical — partition by means of medium and physical layer redundancy is assured in

Rate Mbps	t_{dl} μs	t_{SD} μs	t_{Sl} μs	Scenario		t_{ina} (ms)
5	36	11	27	*Token Loss*		5.98
				No Successor		0.32
				Join	*Contention*	3.70
					No Conten.	0.10
10	18	22	49	*Token Loss*		9.98
				No Successor		0.37
				Join	*Contention*	5.49
					No Conten.	0.14

Table 2: Inaccessibility Times For Token Bus. t_{dl}: delimiter (header/trailer) duration; t_{SD}: station delay; t_{Sl}: slot-time; t_{ina}: inaccessibility periods.

standard FDDI, through a dual-reconfiguring ring, capable of surviving one interruption of the ring [15]. In a token-bus network some similar measures should be custom-implemented, for instance dual-media, since it has no standardised redundancy.

In general, one needs to show that all the recovery glitches are time-bounded and determine the upper bound. Those include operating situations, typical to each LAN, such as re-establishment of the logical ring after loss of token and physical reconfiguration subsequent to ring breakage, or switch-over to an operational spare after bus failure, etc. To illustrate this engineering procedure, a study of ISO 8802/4 token-bus inaccessibility is presented in table 2. The scenarios are the causes for inaccessibility foreseen in the standard specification. Absolute worst-case figures are given in a detailed study (cf. [16]) but the LAN can be configured and operated so as to limit them to the figures above. Token loss accounts for the maximum duration of an isolated occurrence. It corresponds to token-bus T_{ina} and is, in a 10Mb/s bus, $T_{ina} \simeq 10ms$. Medium failure is not accounted for, because we devised a method for real-time switch-over between busses of a dual-media token-bus which is glitch-free [17].

Finally, inaccessibility must be included in the timeliness model. Let us consider a system only with local clocks (timers), used to implement timeouts. Timeouts detect timing, omission and crash failures in the abstract network protocols. They can also be used by upper-layer protocols to perform surveillance of remote parties, upon waiting for the reply to a request (eg. RPC). Two things are necessary: calculation of real execution times (for real-time processing purposes) and dimensioning of timeouts taking inaccessibility into account.

Calculations for real worst-case protocol execution times will have terms which are a function of T_{ina}, corresponding to worst-case inaccessibility periods that may occur during an execution. According to **An3**, in an interval chosen at random, T_{rd}, there may be i inaccessibility periods. When engineering the protocol, this must be taken into account. When calculating timeliness however, more workable assumptions can be made. Let us simplify by assuming that $i = 1$ and that during a whole protocol execution there can be at most one inaccessibility period. This is realistic for most settings, and implies that T_{ina} be simply added to the worst-case protocol execution time expression computed without taking inaccessibility into account. We will use this assumption in the sequel of the text.

Timeout values depend on execution times. Protocol timers must in general include T_{ina}, or else they timeout too early should inaccessibility occur. In this case, the protocol may fail[11].

At this point, note that incorporating T_{ina} in the timers is a sufficient condition for running synchronous (real-time) protocols over the abstract network, using the transmission-with-reply protocol. The diffusion protocol requires additional measures, since it does not use timeouts.

Before going any further, note that the timers are now undesirably long, since T_{ina} may be much greater than T_{td}. A solution hiding inaccessibility away would be welcome, with two obvious advantages: giving the programmer the simple and elegant abstraction of a virtual distributed environment which is always connected; yielding shorter timeouts.

[11] Consider the example of the protocol of figure 1(a): if the $TwaitReply$ timer does not include T_{ina}, it may timeout all the $k + 1$ times during an inaccessibility period, and recipient failure is wrongly detected. The one in figure 1(b) makes the higher-level user believe the message was indeed delivered after some estimated delay; in fact, it may be waiting in some queue long after.

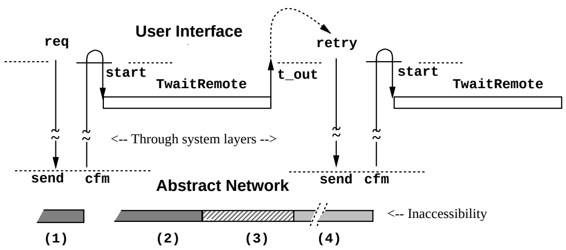

Figure 2: Timing of the inaccessibility masking mechanism

4.1 Timer-freezing mechanism

The mechanism proposed to hide inaccessibility is very simple. It concerns all timeouts controlling timeliness of remote interactions, be them transmission-with reply timeouts or upper-layer timeouts. The value of the timeouts does not include inaccessibility. The mechanism is based on stopping (freezing) all timers when inaccessibility is detected and while it lasts, and restarting them when the network becomes accessible again.

All protocols can then be constructed as if the network were always accessible. Both algorithms of figure 1 will work unchanged. As an example, a protocol over a token-bus can assume that the LAN is always in steady state with the ring formed, and that there are no station joins or leaves, and no token losses. Then, the worst-case frame transmission time in all situations, from the programmer or protocol designer's point of view[12], is merely the maximum queue plus access plus transmission delay, T_{td}, under those operating conditions. This yields an optimal sizing of timers, with regard to detection of other errors, such as omission and node failures.

On the other hand, worst-case timeliness calculations, i.e. determining the real duration of activities, are tuned by adding T_{ina} — which must be determined for each LAN — to the computed protocol execution time expressions.

4.2 Inaccessibility masking mechanism

Freezing the timers is an engineering issue, which implies: inaccessibility to be detected by the communications adapter part; information between the adapter and the timers to flow quickly enough; timers to be stopped/resumed on arrival of in-/accessibility notifications.

While this mechanism can be easily implemented on specially built hardware, the conditions named above are not always met in existing hardware/software platforms. Namely: the timer package may not have the functionality for lightweight start-restart; some LAN controllers do not signal all inaccessibility situations, or do not do it consistently at all nodes.

Thence, an alternative had to be seeked, since porting of the abstract network to existing environments was desired. We devised a mechanism which masks inaccessibility at the network interface level. We explain it next.

Assume that the following system requirements are fulfilled: (i) in the interface with the abstract network, all requests are positively confirmed *when the frame is transmitted.* (ii) in a layered architecture, the higher-level request is also confirmed: the low-level confirmation that the request was served propagates up the layers; (iii) timers to control a remote request are only started after the confirmation that it was issued. In consequence, when the network is inaccessible the confirmation does not come. Let us follow the several situations depicted in figure 2:

Timer *TwaitRemote* equals the desired waiting time at any level of the system (to receive an RPC result, or a transmission reply).

Situation (1): Since a remote interaction is started by a **send** request, a timer controlling it at whatever level will be pending of the confirmation from the network. Should it be inaccessible, timer start will be postponed until the network becomes accessible again.

Situation (2): After the timer is started, it will count up to *TwaitRemote*. In this situation, the network becomes inaccessible after the timer is started and accessible again before it expires. If there

[12] I.e. for the sake of dimensioning timeouts.

is not enough time for the remote frame to arrive, the timer will expire.

Situation (3): Inaccessibility lasts beyond $TwaitRemote$, but ends before anything is attempted in order to recover. A timeout is issued here as well.

Situations (2) and (3) are analogous. They are not disturbing: the timer is there to detect these facts. Inaccessibility has just been transformed into an omission. Given An2 and An3, whatever methods used to take care of k omissions in the system, will keep working for $k + i$ (i - number of inaccessibility occurrences, see An3). This means that inaccessibility is in fact masked out of the system algorithms above the abstract network. The system programmer can, for example, be given a consolidated $K = k+i$ omission bound figure.

Note the very little difference between the effect of inaccessibility in situations 2 or 3, and that of a slightly longer LAN access time. In fact, 'inaccessibility' would not be needed for this, it would suffice to make timers a little longer. However, this approach is justified by the bi-modal characteristic of LAN operation, with the more serious inaccessibility situations lasting much longer than T_{td} and thus $TwaitRemote$. That is the case of:

Situation (4): Inaccessibility still lasts when recovery is attempted (the second send request).

The protocol failures due to short timers described earlier in a footnote occurred because the protocols might proceed during an inaccessibility period. In order for this mechanism of *short timers* and *transforming inaccessibility into omissions* to work, it is necessary that each inaccessibility period is seen as *one* "omission". The confirmed send request performs this role, preventing the protocol from proceeding until the period ends. Note also why the duration of inaccessibility does not affect system timers: it is either simultaneous with the $TwaitRemote$ period, or with the request-to-confirmation period.

We have shown that inaccessibility is masked out of all system activity: its duration is hidden; its occurrence is transformed into an omission. Looking back at the algorithms of figure 1, both will work:

- the diffusion algorithm is throttled by the abstract network interface in case of inaccessibility, if correctly used, i.e. the user protocol should not proceed beyond a point where a reliable delivery is assumed, without receiving the collection of $k + 1$ transmission confirmations corresponding to that delivery;

- the transmission-with-reply algorithm works practically unchanged; the only modification required is to make $nrTries = k + i + 1$ instead of just $k + 1$. An individual transmission in case of inaccessibility works like the example in figure 2, with $TwaitReply$ substituted for $TwaitRemote$.

On the quantitative side, execution times are not increased by T_{ina} anymore when faults occur (the case with including T_{ina} in the timer values). Besides, when inaccessibility occurs, in lieu of increasing by T_{ina}, they increase *at most* by t_{ina} (the effective duration, which may be much lower than T_{ina}).

The disadvantage with regard to the timer freezing scheme, which "stops" the time in this part of the system, is that timeouts do occur due to inaccessibility, and recoveries have to be initiated.

5 Conclusions

Methodologies for enforcing reliability and real-time behaviour of standard LANs have been discussed. The definition of a LAN-based communication sub-system with a well-defined set of properties provides, in addition to several non-functional attributes, very simple low-level reliable frame delivery services. These have proved to ease construction of complex protocols such as reliable multicast. The handling of failures which may hinder timeliness, including a class of *controlled* partitions, was shown to be possible in non-replicated networks, in order to yield reliable real-time operation.

Such mechanisms substantiate the correctness of the approach, followed by the Delta-4 architecture, of designing robust real-time distributed systems using standard communication components.

References

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

[2] 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.

[3] H. Kopetz, G. Grunsteidl, and J. Reisinger. Fault-tolerant membership service in a synchronous distributed real-time system. In *IFIP WG10.4 Int. Working Conference on Dependable Computing for Critical Applications*, Sta Barbara - USA, August 1989.

[4] 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.

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

[6] D. Powell, D. Seaton, G. Bonn, P. Veríssimo, and F. Waeselynk. The Delta-4 approach to dependability in open distributed computing systems. In *Digest of Papers, The 18th International Symposium on Fault-Tolerant Computing*, IEEE, Tokyo - Japan, June 1988.

[7] D. Powell, editor. *Delta-4 - A Generic Architecture for Dependable Distributed Computing. ESPRIT Research Reports*, Springer Verlag, November 1991.

[8] P. Veríssimo, L. Rodrigues, and M. Baptista. AMp: a highly parallel atomic multicast protocol. In *SIGCOM'89 Symposium*, ACM, Austin-USA, September 1989.

[9] 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.

[10] G. Le Lann. *The 802.3D Protocol: A variation of the IEEE802.3 Standard for Real-time LANs*. Technical Report, INRIA, France, 1987.

[11] 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.

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

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

[14] 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.

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

[16] 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).

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