A Freshness Detection Mechanism for Railway Applications

Andrea Bondavalli[1], Enrico De Giudici[1], Stefano Porcarelli[2], Salvatore Sabina[3], Fabrizio Zanini[3]

[1]University of Florence, Dip. Sistemi e Informatica, via Lombroso 67/A, I-50134, Italy
a.bondavalli@dsi.unifi.it, enrico.degiudici@tin.it
[2]Italian National Research Council, ISTI Dept., via Moruzzi 1, I-56124, Italy
s.porcarelli@isti.cnr.it
[3] Ansaldo Segnalamento Ferroviario, Via dei Pescatori 35, I-16129, Genova, Italy
{Zanini.Fabrizio, Sabina.Salvatore}@asf.ansaldo.it

Abstract

Railway control systems are based on on-board and trackside subsystems for signaling purposes. Several factors demand for new design and implementation solutions for such railway control systems. These factors are related to the design of interoperable railway networks in Europe, the introduction of new technologies and equipment, and the competition in the market of railway products. The safety of such new design and implementation solutions should still be proved in accordance with the CENELEC recommendations. In this paper SFDA, Safe message Freshness Detection Algorithm among trackside subsystems, is deeply described. SFDA is included in a new message passing safety protocol stack and it allows the detection of "old" messages and the meeting of real time and safety requirements of trackside railway systems. It is demonstrated that the SFDA can detect all the old messages. Moreover a preliminary analysis of its availability characteristics to check whether it is suitable for railways systems is performed through simulation.

Keywords: railway control systems, safety, real-time

1. Introduction

Railway signaling is a safety critical application, which has been traditionally performed by the design and construction of often proprietary systems with ad hoc solution, both in hardware and software. In recent years, many European initiatives were carried out with the main objective of defining European Standards (e.g. [5], [6], [4], [7], [1] etc.) that provide functional and non-functional requirements such as interface requirements, safety, and interoperability requirements. In particular, the high demanding interface/interoperability/safety (quantitative) requirements are driving the technical development of newer systems. In such a difficult context, at the end of 2002 a message passing safety protocol stack has been proposed by Alstom and Ansaldo Segnalamento Ferroviario (ASF) to define a common communication infrastructure among some trakside subsystems. The main motivations are the following:

- these trakside subsystems must exchange vital-safe information, but no agreement exists on the physical interfaces of the different vendors and on involved software/hardware platforms;
- the safety targets must be identical for each interface but the way to reach these safety targets might be profoundly different;
- the communication layers must work adopting a common set of rules but they are not implemented in the same way; thus the interfaces are implementation independent whereas their implementation might change among vendors;
- the message passing protocol stack must be compliant with the European applicable standard for open networks.

To ease and to speed up the definition, the development, and, in particular, the validation of the proposed protocol stack, it was decided to re-use the Euroradio Safety Layer (defined for communications among trains and ERTMS Radio Block Centers) as one of the communication layer of the proposed protocol stack. To overcome some of the Euroradio Safety Layer limitations in the context of trackside safety communications, ASF have also defined and proposed the protocol layer in charge of coping with message freshness detection.

This paper describes the work performed in a research cooperation between ASF, the university of Florence and

the Italian CNR aiming at the definition of the freshness detection algorithm called SFDA: Safe Freshness Detection Algorithm, the verification of its safety properties and the analysis of its availability. SFDA is included in the Italian high speed railway signaling system based on an heterogeneous network infrastructure composed of interlocking subsystems and Radio Block Centers interconnected throughout an high speed fiber optic network. Eack network subsystem is responsible for checking whether the received information is fresh enough with respect to the application real time requirements or whether it is too old. SFDA, can discriminate the received messages judging whether they were produced too long ago to be used by the receiver subsystem or whether they contain still valid information and therefore it is safe for the application to use their content. SFDA is first described including the availability penalisation for guaranteeing safety; then, secondly, it is analysed to prove its safety properties (i.e., it never happen that an old message is used); third, a quantitative analysis regarding availability is performed to understand if availability is still valid for concrete applications (an algorithm that discardes all the received messages is a safe algorithm but it cannot be used). Such a quantitative availability analysis has been carried out by means of simulation to model the behavior of the algorithm under several conditions; preliminary availability figures are shown.

The rest of the paper is as follows. Section 2 provides the context overview where the freshness detection mechanism is applied, and defines the constraints for the algorithm. Section 3 describes SFDA and demonstrates its safety properties. Section 4 introduces the defined simulation model and the simulator developed, and presents some availability figures. Finally, Section 5 concludes this paper by describing the present state of the implementation, the validation, and the assessment of the proposed message passing safety protocol stack (which includes SFDA) on the ASF platforms.

2. The system and the algorithm requirements

ASF is involved in the design, implementation, and validation of the vital (safe) and non vital trackside subsystems of the Italian railway high speed network. For that it has started a project for the design, the implementation, and the validation of the High Speed Line Interlocking and Radio Block Center subsystems (see Figure 1).

The Radio Block Center (RBC) subsystem will constitute the safe - vital part of the Central Satellite Posts (PCS), whereas the interlocking subsystem (IXL) will constitute the safe - vital part of the Fixed Peripheric Posts (PPF). The PCS and the PPF will therefore be part of the signaling and automation system of the Italian railway high speed network. For each trackside segment, there will be one PCS

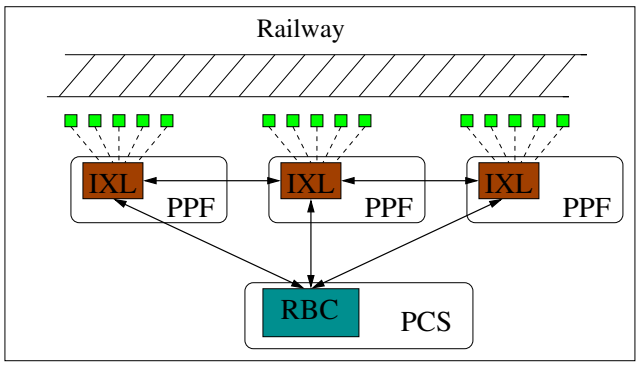

Figure 1. RBC and IXL.

and many PPF; the latter with different functional characteristics depending on their on the trackside segment location. Each PPF has to communicate with the adjacent PPF and all of them have to communicate with the PCS. For communications between trackside subsystems dedicated fiber-optic channels will be used.

Such subsystems have to comply with the recently defined railway CENELEC standards. In particular communications between trackside subsystems must obey all the requirements of an open communication system according to the CENELEC prEN 50159-2 [7] standard. This requires that open communication systems must be designed so to cope with the following threats ([7], pag. 14):

- Repetition of an old message performed by a hacker or due to an hardware-failure
- Deletion: a hacker/hardware-failure deletes a message, e.g. emergency, stop messages.
- Insertion: a hacker/authorized-third-part inserts a message;
- Resequence: a hacker/hardware-failure changes the message sequence;
- Corruption: a hacker changes the message content;
- Delay: the transmission system is overloaded due to normal traffic or an hacker generates false messages such that the provided service suffers delays;
- Masquerade: a hacker/hardware-failure masquerades the true source of messages.

[7] provides also a list of defenses applicable to each threat shown in Figure 2 ([7], pag. 22) although it leaves some degree of freedom to the designers by stating “new techniques may be developed in the future which offer new possibilities to the designer. Such new techniques may be used to provide protection against these threats, provided that the coverage of the techniques is well understood and has been analysed.”

Threats	Defenses							
	Sequence Number	Time stamp	Time out	Source and destination identifier	Fedd−back message	Identification Proc	Safety code	Cryptographic techniques
Repetition	X	X						
Deletion	X							
Insertion	X			X	X	X		
Resequence	X	X						
Corruption							X	X
Delay		X	X					
Masquerade					X	X		X

Figure 2. Threats and Defenses

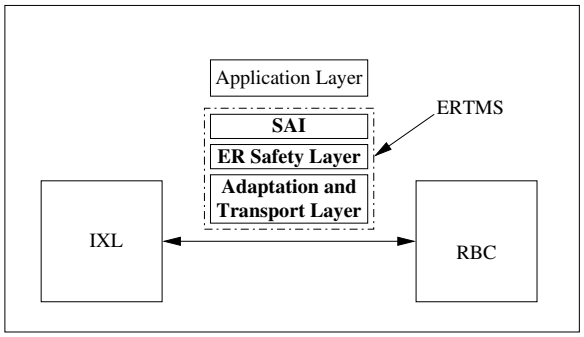

Figure 3. Euroradio FIS European Standard

The protocol stack [2] depicted in Figure 3, defined through a co-operation between ASF and Alstom, implements the required protections to address the threats defined in [7]. One can distinguish:

- Adaptation and Transport Layer: adaptation between the Euroradio Safety Layer and TCP Layer.
- SLE: Euroradio Safety Layer. It realizes defences to corruption and masquerade through cryptography (access protection).
- SAI: Safe Application Interface. It deals with the other threats (transmission protection). In particular, the use of a Sequence Number allows a node in the network to cope with Repetition, Deletion, Insertion and Resequence, while the use of a timestamp has the purpose to meet the Delay requirement. A more detailed view is in Figure 4.

For each received message, the SLE verifies that the source is the proper one and that the message is not corrupted; it forwards message to the SAI when there is no negative check. First, the SAI verifies the Sequence Number and, then, the freshness by using the Elaboration Cycle number (Ec) (which has the same role as a timestamp) and the freshness detection algorithm. When no check fails, the

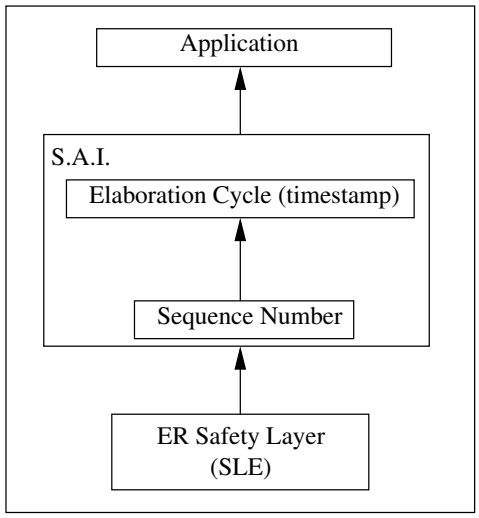

Figure 4. Details of the SLE-SAI protocol stack

message is passed to the application; otherwise, the SAI informs the Application Layer that implements its own fault management policy (e.g. connection closure and new open connection request). For the sake of simplicity, the Application Layer fault management policy assumed in this paper is connection closure and connection re-open.

Thus, the system we are considering (see Figure 1) is a network system made of several trackside sub-systems (nodes) which a) communicate through networks which may induce random delays on messages and b) must be able to detect whether received messages are fresh or not. All subsystems can be thought of as cyclic executive subsystems, each one using a major cycle of a constant duration possibly different from that of other subsystems. Thus, each subsystem has an elaboration cycle duration (or period) and an elaboration cycle number as the only way to measure time and to synchronize each other.

Moreover, once a connection is up, the sender node has to send one message per cycle and the receiver node must process one message per cycle (the message with the highest sequence number). In general, a message might be processed more than once (if the next message sent gets lost for example) and some messages may not be processed at all (if they are slow and when they arrive to destination more recent messages are already available). However, if the most recent message is too old, the receiver node must not forward it to the application layer but has to raise a signal to the application layer, that, as described earlier, forces disconnection.

Our system can thus be classified as a time-driven real-time architecture [14] such as TTA [11]. Moreover variable and unpredictable communication delays coupled with tim-

ing requirements resembles very much the Timed Asynchronous model [9]. Clearly, by time stamping messages, all these models and related systems allow detection of a too long delay in delivery of messages. Solutions based on timestamping have been proposed in such contexts for rejecting old data [10, 9, 8]. In addition, a freshness mechanism must cope with variations of the communication delays, with the long term clock drifts and with approximation errors of the algorithm computation. The freshness detection requirements state that each network node must not process old messages. That is, each node must recognize whether a message is too old.

3. SFDA: a Safe Freshness Detection Algorithm

The Freshness Detection Algorithm we have devised, SFDA, is based on the concept of Elaboration Cycle (Ec). Each message (one per cycle) is sent by attaching the sender Ec of the cycle in which the message is sent which thus constitutes a timestamp for that message. The receiver subsystem could easily judge the message freshness if the system would be fully synchronized. However this would be a sufficient condition far more demanding than necessary. In fact for fulfilling the requirement it is enough that each subsystem, for each connection in which it acts as receiver, has knowledge of the sender Ec. If it knows the sender Ec a simple comparison is sufficient. In reality the receiver can only compare the Ec in the message with its own estimate of the sender Ec (VEC: Virtual Elaboration Cycle, i.e. expected Ec number). A proper estimate of the VEC is the key issue we have addressed. This estimate must be conservative to guarantee that all old messages are discarded. A conservative estimation means that VEC must be greater or equal to the Ec of the sender. The price to pay if the estimate is not precise is that some "fresh" messages are considered "old" and that some signal will be raised unnecessarily with a consequent closure of the connection. SFDA is introduced first without considering clocks drift and approximation errors due to real values processing: these effects will be considered and addressed later. The receiver side is described in full details in the following. In fact it is the receiver which has to fulfill the freshness requirement with a small cooperation from the sender.

The Virtual Elaboration Cycle (VEC) is computed based on the relationship among the sender and receiver elaboration cycle periods. We assume that before the data transmission takes place, beside its own elaboration cycle time (referred to as RCL), the receiver subsystem knows the sender elaboration cycle time (SCL in the following), and the minimum reaction time of the sender. These may be configuration parameters or may be communicated by the sender at connection set up. If the receiver subsystem has a cycle duration R times that of the sender, R messages will normally be received in a cycle by the receiver.

Thus SFDA has two main phases: the initialization and connection set up, and the working phase in which received messages are checked for freshness.

SFDA Initialization and Set Up

During the synchronization phase, once a SLE connection has been set up, the receiver computes an estimation of the Virtual Elaboration Cycle (VEC) of the sender. This happens through message exchanges to determine the round trip time and **the receiver adjudicated delay** of the first data message.

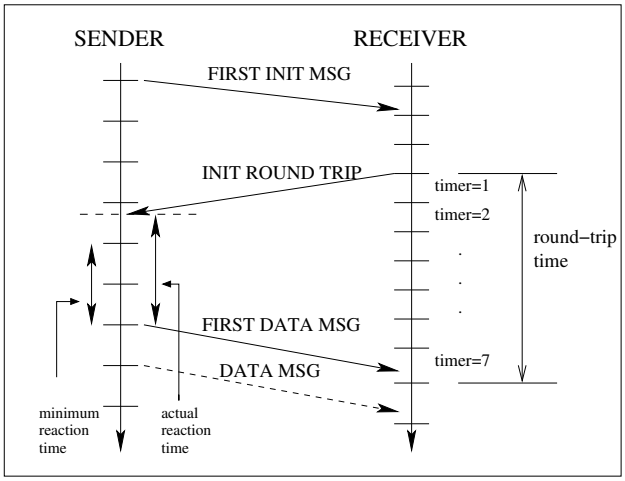

Figure 5. Algorithm initialization

Figure 5 shows the messages exchanged in this phase and the notation used hereafter. The first message from the sender to the receiver, called FIRST INIT MSG, has the purpose to trigger the receiver to compute the round trip time. After the reception of the FIRST INIT MSG the receiver sets a timer to measure the round trip time (measured in receiver cycles) and sends back a message called INIT ROUND TRIP. As soon as the sender receives this message starts to send data messages called DATA MSG, one per cycle. In particular, the first message that reaches the receiver is used to compute the round trip time, and is called FIRST DATA MSG. Synchronization is performed by two handshakings with a shared message (the INIT ROUND TRIP). Since messages might get lost, both the sender and the receiver have to set a timer so to avoid waiting forever. The use of the sequence number (at the SAI layer) allows to discard out of order messages delayed by the network so they do not interfere in the connection set up.

Then, the receiver estimates the delay of the FIRST

DATA MSG and using this estimation and the sender cycle number included in it, is able to check whether a message is older than a given threshold (MAXD) or not. The estimation is made as follows. The round trip time is computed as the number of receiver cycles from the sending of INIT ROUND TRIP to the reception of FIRST DATA MSG multiplied by the receiver cycle time. Then the sender minimum reaction time (a number K of sender cycles multiplied by the sender cycle time) is subtracted to obtain the time employed by the network to deliver the two messages INIT ROUND TRIP and FIRST DATA MSG. If a minimum limit for the delivery of a message (MND) exists, a safe way to estimate an upper bound for FIRST DATA MSG is to assume this minimum delay for the delivery of INIT ROUND TRIP (otherwise 0 delay could be assumed). Thus the expression for D, the delay attributed to FIRST DATA MSG, is the following:

$$D = (timer * RCL) - K * SCL - MND \tag{1}$$

D is an overestimation of the real delay experienced by FIRST DATA MSG. Obviously if the estimate is higher than the threshold MAXD, FIRST DATA MSG is discarded and the receiver sends back a DISCONNECT message. The receiver can set a timeout Tout so that such a situation never happens: the timer expires before such a slow message arrives. The expression for setting the timer is the following:

$$Tout = \left\lfloor \frac{MAXD + K * SCL + MND}{RCL} \right\rfloor + 1 \tag{2}$$

The Virtual Elaboration Cycle (VEC), the receiver estimate of the sender cycle number, can be initialized when FIRST DATA MSG is received as follows:

$$VEC = Ec + \frac{D}{SCL} \tag{3}$$

where Ec is the elaboration cycle of the sender included in FIRST DATA MSG, that is the sender clock at the time the message was sent, and D is the estimation of the delay of FIRST DATA MSG.

SFDA Working Phase

The receiver updates VEC in each cycle, using R the ratio between the sender cycle time and the receiver cycle time:

$$R = \frac{RCL}{SCL} \tag{4}$$

At each cycle VEC is increased by R.

$$VEC = VEC(\textit{at the previous cycle}) + R. \tag{5}$$

For each message received the difference $\Delta = VEC - Ec$ gives an over estimate of its age. In particular for a message suffering a delay greater than MAXD, $VEC - Ec$ becomes greater than

$$\Delta_{Max} = \frac{MAXD}{SCL} \tag{6}$$

At each cycle the receiver actually has to process only the freshmost message (the one with the highest sequence number). If it is too old the receiver does not forward it to the application layer and issues a DISCONNECT instead.

3.1. SFDA Steps

Summarizing, SFDA (at the receiver side) is composed by the following steps:

Init 1: Upon the receipt of the FIRST INIT MSG, the receiver evaluates Δ_{Max}, R and Tout using equations (6), (4) and (2), respectively.

Set up 1: send the INIT ROUND TRIP MESSAGE.

Set up 2: Upon the receipt of the FIRST DATA MSG, the receiver evaluates D, Ec, and VEC. D is evaluated using equation (1). Ec is initialized with the Ec of the FIRST DATA MSG and VEC is evaluated using (3).

At each cycle steps 3-6 are repeated until disconnection:

Step 3: At each own cycle the receiver updates VEC as in (5).

Step 4: If one or more messages have been received, the variable Ec of the receiver is updated with the value of the message received with the highest Ec (the freshmost message).

Step 5: Δ is evaluated as $VEC - Ec$.

Step 6: freshness is evaluated as follows:

$$\begin{cases} \Delta \leq \Delta_{Max} \rightarrow \textit{“fresh”} \rightarrow \text{pass to appl.} \\ \Delta > \Delta_{Max} \rightarrow \textit{“old”} \rightarrow SIGNAL \end{cases} \tag{7}$$

After a SIGNAL the receiver re-executes the algorithm starting from Set up 1.

3.2. SFDA Safety

We now prove that SFDA is safe, i.e. it is able to detect and discard all the old messages. More formally it never happens that a message which is old is accepted as a good one and processed. The subsystems communicate over a network which may induce random delays and the receiver, through a round trip, makes a read of the sender clock (the cycle number) with a bounded error [3].

In fact, as depicted in Figure 6, the receiver can measure $T_4 - T_1$ through its clock, while $T_3 - T_2$ is provided by the sender. Thus D(m1) the delay of m1 (corresponding to D,

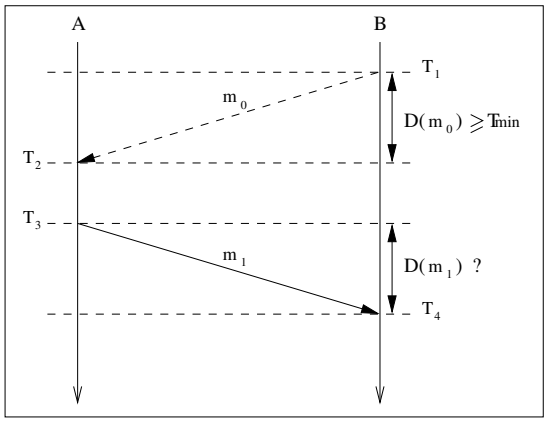

Figure 6. Round Trip Time estimation [14]

the delay of FIRST DATA MSG for SFDA) lies between the two extremes:

$$Tmin \leq D(m1) \leq (T_4 - T_1) - (T_3 - T_2) - Tmin \quad (8)$$

Many estimations are possible and in SFDA we have chosen for D the highest extreme:

$$D = (T_4 - T_1) - (T_3 - T_2) - Tmin. \quad (9)$$

This choice allows to guarantee the safety of the proposed algorithm as proven by the following theorems.

Theorem 1: If a receiver subsystem overestimates the delay experienced by FIRST DATA MSG the receiver is capable to build an image of the sender clock (VEC) so that this image is always equal or ahead of the sender clock (Ec).

Proof. Let t be the instant of time in which the sender sends its time (Ec) to the receiver. When the receiver gets this message the sender clock will have reached the value $t + \delta t$ (the actual delay suffered by the message) whereas the receiver will attribute the value t+D to its image of the sender clock. Since D is greater or equal to δt, the receiver image of the sender clock is not behind it. ■

Theorem 2. Let us consider a sender and a receiver in our system. Assume that the sender timestamps every message it sends with its cycle number and that it sends at most one message per cycle to the receiver. The receiver must accept messages which have experienced a delay less than or equal to MAXD and discard those suffering a delay greater than MAXD. If the receiver has an image of the sender clock (VEC) so that this image is always equal or ahead of the sender clock (Ec), then messages experiencing a delay greater than MAXD are never accepted.

Proof. Suppose a message sent at time t experiences a delay MAXD + ϵ (that is the message is old). At time t the receiver has an image of the sender clock equal to $t_0 \geq t$. The receiver will receive this message when its image of the sender clock holds t_0 + MAXD + ϵ. The message has the value t as its timestamp. Let suppose the message is accepted, then the difference between its arrival time and its timestamp must be less or equal to MAXD. That is t_0 + MAXD + $\epsilon - t \leq$ MAXD. Adding t and subtracting MAXD at both terms of the expression leads to $t_0 + \epsilon \leq t$ contradicting the assumption $t_0 \geq t$. ■

3.3. Long Term Effects and How They are Solved

Two factors can affect the safety of SFDA: the clock long term drift and the approximation errors of the real variables. In fact their effect might be such that VEC goes slower than Ec invalidating the basis on which the protocol safety builds. They can be addressed so that the safety property of the algorithm is preserved, again at the price of further reducing the availability of the system. That means that surely, sooner or later, "fresh" messages will be judged "old" dropping unnecessarily an ongoing connection. In fact, the rationale behind the method for avoiding the long term effects is still to overestimate VEC (see (3)) so that **Theorem 1** is still valid. If ρ is the clock tolerance in part per million (ppm), ρ_0 is its drift (ppm/ °C), and $tempT$ is the range of working temperature for the system, the length of the sender and receiver cycle should be underestimated and overestimated, respectively, as in the following:

$$\begin{cases} \text{New SCL} = (1 - \rho - \rho_0 * tempT) * \text{SCL} \\ \text{New RCL} = (1 + \rho + \rho_0 * tempT) * \text{RCL} \end{cases} \quad (10)$$

Also, to cope with the approximation errors introduced by the floating point operations in order that **Theorem 1** still remains valid, the SFDA real-valued variables should be approximated as in the following (for example using the facilities of the GNU C library [13]):

$$\begin{cases} \Delta_{Max} \rightarrow \text{DOWNWARD approximation} \\ \Delta \rightarrow \text{UPWARD approximation} \\ VEC \rightarrow \text{UPWARD approximation} \\ R \rightarrow \text{UPWARD approximation} \end{cases} \quad (11)$$

where the DOWNWARD approximation consists in approximating a real number with the biggest number lesser than it representable by the computer while the UPWARD approximation consists in approximating a real number with the smallest representable number greater than it.

4. Availability

We have described SFDA and proved it is safe. Due to the randomness of the network it is impossible to reach a perfect synchronization so to judge correctly all the incoming messages. The choice made in SFDA has been to overestimate the delay of messages to be sure of not accepting

old messages. The price to be paid is to discard sometimes fresh messages, thus impacting availability. In order to evaluate the availability of SFDA we have built a discrete event simulator [12] written in C which models the behavior of the system and of SFDA and performed a preliminary analysis of the availability related figures. The results achieved have been evaluated with a confidence level of 95% and relative confidence interval of 10%. The objective of the quantitative analysis has been to evaluate:

- the mean number of trials (handshakings) and mean time to set up a connection
- the mean duration of a connection;

In doing such evaluation we have accounted for varying networks characteristics (such as messages delay) and clock long term drift and approximation errors of the real variables. We have not yet accounted for message loss over the network.

We evaluated the availability of SFDA both during the connection set up and the working phases. Concerning the connection set up the following measures have been evaluated: i. the mean time to successfully complete the connection set up and, ii. the probability mass function of the number of attempts needed to reach a successful synchronization. Concerning the working phase the mean connection duration has been evaluated. The several parameters involved in the simulation are set up as shown in Table 1.

Parameter	Value
RCL	350 msec.
SCL	500 msec
Sender minimum reaction time (K)	1 cycle
MAXD	1.2 $\div$ 2.1 sec.
MND	0.01 sec.
Mean Message Delay	0.1 $\div$ 0.3 sec.
$\phi = \rho + \rho_0 * \Delta T$	0 $\div$ 20 ppm

Table 1. Simulation parameters

We considered the receiver and the sender having different elaboration cycle length (500 and 350 msec., respectively). In this case R is a real variable and needs to be approximated as suggested in eq. 11. If cycles have the same length, R is an integer making the case much simpler. We assume also that the sender has a minimum reaction time of one cycle ($K = 1$).

MAXD is a configuration parameter and it is allowed to span in a significant range of the application domain. The networks characteristics and the elaboration processing time of the protocol layers below the freshness mechanism are modeled by an exponential distribution. Moreover, it is assumed that a message has a minimum delivery time (MND). The ϕ parameter gathers the effects of both clock tolerance and drift. The approximations (11) against the errors introduced by the floating point operations are directly built in the simulator.

4.1. Connection Set Up Phase

Figure 7 shows the mean time to successfully complete the connection set up at varying values of the mean message delay for different MAXD.

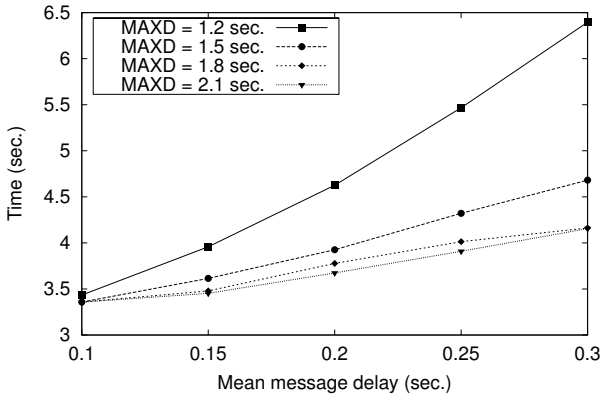

Figure 7. Mean connection set up time (sec.)

Not surprisingly, the connection set up time increases as the mean message delay increases and/or the MAXD decreases. It appears to be relatively low, in our settings it varies from 3.4 sec. to a maximum of less of 6.5 sec.. Such a low set up time allows to reach good availability also in those cases in which the duration of the connection will be relatively short.

Figures 8, 9, 10, and 11 show the probability mass function of the number of attempts needed to reach a successful synchronization at varying values of the mean message delay and for four different values of MAXD, respectively.

Figure 8 (MAXD = 1.2) shows that the probability of synchronizing the sender and the receiver decreases as the mean message delay increases. In particular, the probability of synchronizing within 3 attempts is always over 94%. Figures 9, 10, and 11 have a similar behavior of 8 (but they have different scale) and show how the probability mass function of the number of attempts needed to reach a successful synchronization is closer and closer to 100% as MAXD increases.

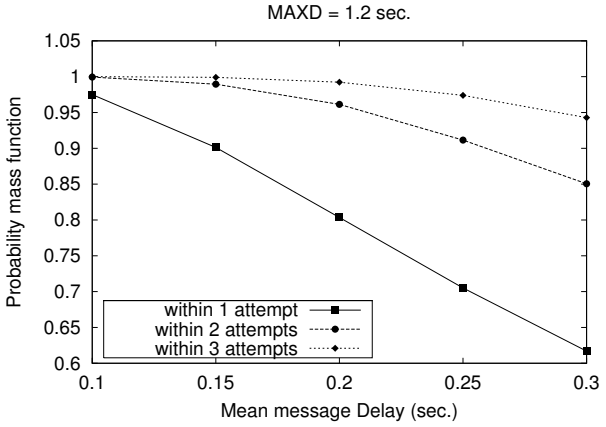

Figure 8. Probability mass function of the number of attempts needed to reach a successful synchronization for MAXD = 1.2

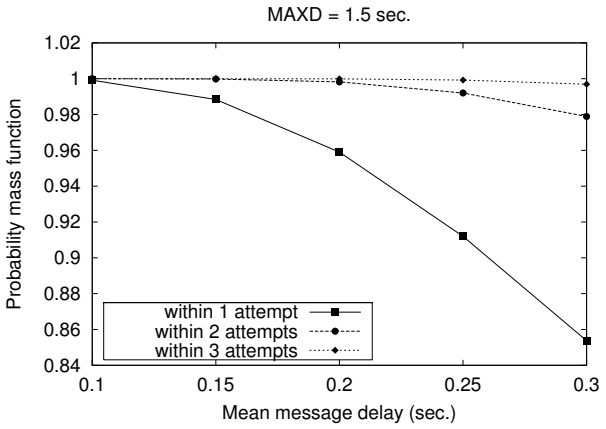

Figure 9. Probability mass function of the number of attempts needed to reach a successful synchronization for MAXD = 1.5

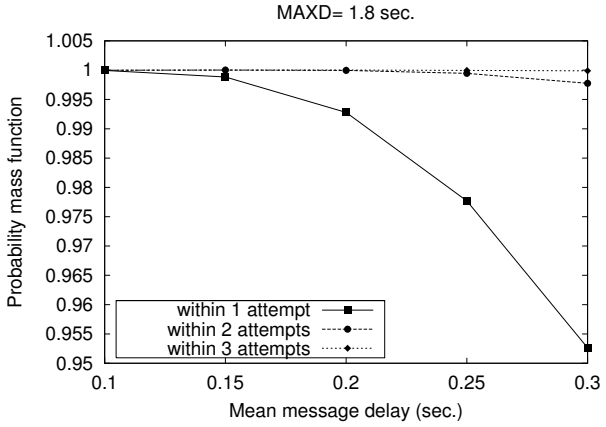

Figure 10. Probability mass function of the number of attempts needed to reach a successful synchronization for MAXD = 1.8

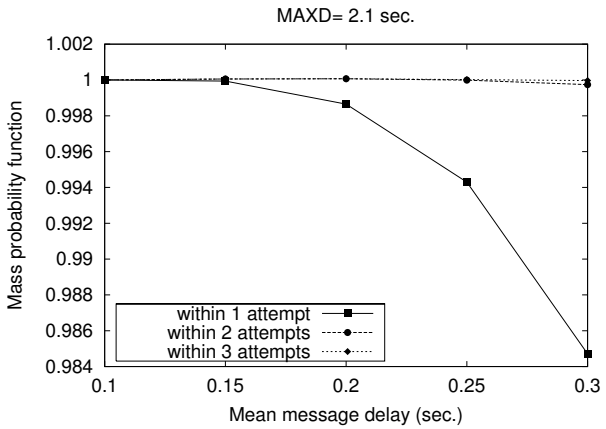

Figure 11. Probability mass function of the number of attempts needed to reach a successful synchronization for MAXD = 2.1

4.2. Working Phase

Figures 12, 13, and 14 show the mean duration of the working phase of a synchronization at varying values of the mean message delay, MAXD, and ϕ. The value in Figure 12 for mean message delay = 0.1 sec. and MAXD = 2.1 sec. is missing because it took too long time to converge. Actually, in this case, the connection lasts quite long (more than 100000 seconds) but the simulator needed too much time to collect enough observations in order to achieve a statistical estimation.

In all cases the mean duration of a connection decreases as MAXD reduces and mean message delay and ϕ increase. The settings we have chosen here include combinations of values for which, also without any drift of the clock (such as MAXD=1.5 with mean message delay = 0.3 in Figure 12), the availability is very poor. Moreover, as the Figures show, the effect of ϕ is quite relevant and points out either the need for using a very stable clock or the need of modifying the algorithm (e.g. recomputing VEC) to limit the difference between VEC and Ec. Actually, with $\phi = 20$ ppm after 10 hours (36000 sec.) the clock of the sender and of

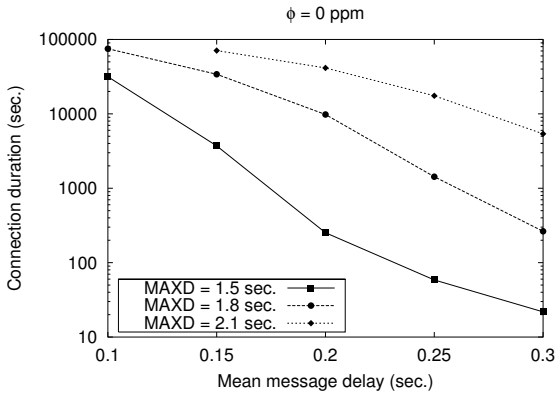

Figure 12. Mean connection duration for $\phi = 0$ ppm

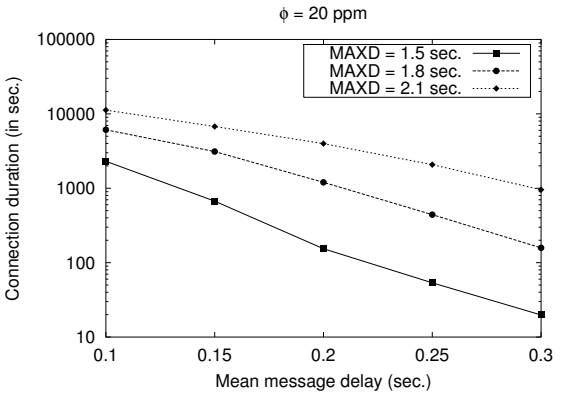

Figure 14. Mean connection duration for $\phi = 20$ ppm

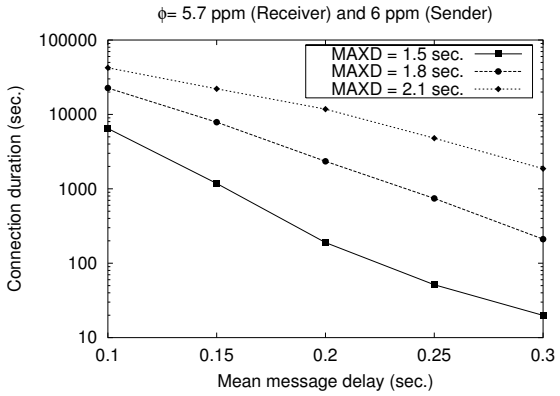

Figure 13. Mean connection duration for $\phi = 6$ ppm

the receiver may be back or ahead of 0.72 sec. with respect to their nominal values (see (10)). This means that in the worst case $VEC - Ec$ is at least 1.44 sec. and the connection will very likely be close.

These preliminary evaluation suggests hat SFDA suffers from a quite relevant availability penalization. In addition we expect that considering message loss (as we intend to do as our next step) will even worsen the situation. In fact it can be observed that only with a low ϕ and reasonable combinations of mean message delay and MAXD, the mean duration of a connection is hours or days, which, combined with the few seconds necessary to set up a connection, makes the attained availability satisfactory for railways systems. Mechanisms to improve availability still maintaining safety, have to be devised for being able to use SFDA in less favorable settings and with higher ϕ.

5. Conclusions

Communications in railway control systems have stringent message delivery constraints. The communicating subsystems considered in this study are not synchronized with a common clock: they have only their own cycle duration as notion of time. The paper proposed SFDA a safe freshness detection algorithm to detect "old" messages during the communications among these systems. The basic idea is that the receiver has always to overestimate the value of the clock of the sender. The safety of SFDA has been demonstrated. Moreover, it has also been shown how to cope with the effects of clock tolerance and drift as well as error approximation in floating point operations. The counterpart of the introduced necessary approximations is a decrease of the availability of the communication. That means that "fresh" messages can be deemed "old" dropping the connection. Our analysis has shown that a high clock drift may have severe consequences on availability. Thus, we are currently analyzing online synchronization algorithms (while the communication is ongoing) in order to improve communications availability.

The safety protocol stack has been specified, designed, and implemented in accordance with the CENELEC 50128 and 50159-2 standards by the ASF development Department. Moreover, the ASF independent V&V Department has concluded the V&V process and delivered the Safety Case. It has been proved the protocol stack suitability for SIL 4 railways systems. At the moment, the Independent Assessor Sciro TUV is carrying out the assessment of the Generic Product which includes the protocol stack and SFDA.

6. Acknowledgments

This work has been partially supported by MIUR (the Italian Ministry for research) through the ARGENTO project. The authors wish to thank the other components of the ASF team.

References

[1] ADTRANZ, A. SEL, ALSTOM, A. SIGNAL, I. RAIL, and SIEMENS. *EURORADIO FIS*, March 2002. Issue 2.2.0.

[2] A. ALSTOM. *SATURNO Interfaccia standard unificata - Specifica Funzionale (Parte 2)*, September 2002. Rev-B.

[3] A. Casimiro, P. Martins, P. Veríssimo, and L. Rodrigues. Measuring distributed durations with stable errors. In *Proceedings of the 22nd IEEE Real-Time Systems Symposium (RTSS 2001)*, pages 319–325, London, U.K., Dec. 2001.

[4] CENELEC. *Railway Applications - Safety Related Electronic Systems for Signalling*, May 1994. EN 50129.

[5] CENELEC. *Railway Applications - The Specification and Demonstration of Reliability, Availability, Maintainability and Safety (RAMS)*, May 1998. EN 50156.

[6] CENELEC. *Railway applications - Communications, signalling and processing systems - Software for railway control and protection system*, March 2001. EN 50158.

[7] CENELEC. *Railway Applications - Part2: Requirements for Safety-Related Communication in Open Transmission Systems*, March 2001. EN 50159-2.

[8] D. Essame, J. Arlat, and D. Powell. Padre: A protocol for asymetric duplex redundancy. In C. B. Weinstock and J. Rushby, editors, *Proc. of the 7th IFIP Working Conf. on Dependable Computing for Critical Applications (DCCA-7)*, pages 229–248, Los Alamitos, CA, USA, 1999. IEEE CS Press.

[9] C. Fetzer and F. Cristian. Fail awareness: An approach to construct fail-safe applications. In *Proc. 27th Int. Conf. on Fault-Tolerant Computing Systems (FTCS-27)*, pages 282–291. IEEE CS Press, 1997.

[10] H. Kopetz. Fault containment and error detection in the time-triggered architecture. In *The Sixth International Symposium on Autonomous Decentralized Systems (ISADS'03)*, pages 139–147, Pisa, Italy, 2003. IEEE CS Press.

[11] H. Kopetz and G. Bauer. The time-triggered architecture. *Proceedings of the IEEE*, 91(1):112–126, August 2003.

[12] A. M. Law and W. D. Kelton. *Simulation Modeling and Analysis*. McGraw-Hill, Third edition, 2000.

[13] S. L. R. M. S. R. M. A. Oram and U. Drepper. *The GNU C Library Reference Manual*, 0.10 edition. For version 2.2.x.

[14] P. Verissimo and L. Rodrigues. *Distributed Systems For System Architects*. Kluwer Academic Publishers, 2001.