

# Assessment of PROFIBUS Networks Using a Fault Injection Framework

José Augusto Carvalho  
Escola Superior de Tecnologia e de Gestão  
Campus de Santa Apolónia  
5301-857 Bragança - Portugal  
jac@ipb.pt

Adriano Silva Carvalho, Paulo José Portugal  
DEEC, FEUP  
Rua Dr. Roberto Frias  
4200-465 Porto - Portugal  
{asc, pportugal}@fe.up.pt

## Abstract

*Industrial control systems architectures have been evolving to the decentralization of control tasks. This evolution associated with the time-critical nature of these tasks, increases the dependability requirements for one of their most critical components: the communication system. Therefore, this is an important aspect on the control system design which must be properly evaluated.*

*In this paper the dependability of a PROFIBUS network is assessed. By using of a fault injection framework the network operation is disturbed with fault scenarios which are representative of industrial environments. From these experiments, the stability of the PROFIBUS logical ring is analyzed, the main outages causes are identified and their probabilities are obtained.*

## 1 Introduction

Fieldbuses are nowadays a widely used communication system oriented to automation and industrial applications [1]. Their use allowed the replacement of the traditional centralized control architectures based in analogue signals or proprietary communication protocols, and at same time the support of advanced functionalities.

Industrial environments are characterized by the existence of a high diversity of equipments that are source of large patterns of electromagnetic interference (EMI), which can induce faults in electronic circuits [2]. In the communication systems, these types of faults produce errors in the communication bus by corrupting transmitted data. To recover from these situations fieldbus networks implement several fault-tolerant mechanisms. However, this creates an overhead which introduces delays in the delivered messages and performance degradation in the control system operation [3,4]. In control systems with real-time requirements, message delays can disturb the system operation which can even lead to its failure [3,4].

In this context, an evaluation of fieldbus networks must be performed, enabling the identification of the most important attributes from dependability viewpoint. It is important to identify which fault-tolerant mechanisms are activated and their effects on the network behavior. By using this data, it becomes possible to de-

velop accurate dependability models (fault forecasting) which enable the evaluation of the distributed applications supported by these networks [5].

In this paper, a dependability evaluation of the PROFIBUS network is performed [6]. The behavior of the Medium Access Control (MAC) protocol in the presence of transient faults is analyzed. The main causes that lead to ring instability are identified and their probability for a given fault scenario is obtained.

The paper is structured as follows. In section 2 is given a description of the PROFIBUS network. A brief discussion of the related work on dependability and performance degradation in PROFIBUS networks is given in section 3. In section 4 a fault injection framework to experimentally evaluate the PROFIBUS behavior is proposed and the hardware platform to support this analysis is described. The methodology used for the implementation of the experiments is presented in section 5. The results obtained from the experiments are discussed in section 6. Finally the conclusions are presented in section 7.

## 2 The PROFIBUS

The Profibus [6] is a fieldbus designed for use at the low level of factory automation systems, where it performs high-speed data exchange between process controllers and field devices, such as sensors and actuators.

Two types of stations can be connected to the network: **(i)** Masters, usually performing automation and control tasks (e.g. PLCs). Their operation typically consists of polling a set of associated slave devices and executing control programs; **(ii)** Slaves, consisting on peripheral devices (e.g. I/O) exchanging data with the masters. Masters are referred as active stations, and slaves as passive ones. Communications can only be initiated by the active stations. Passive stations can only access to the medium in response to an active station request. The communication stack is organized according the OSI model, but using only with 3 layers: *Physical*, *Fieldbus Data Link* (FDL) and *Application*. The stations are interconnected according a bus topology.

The medium access control is achieved by a hybrid access medium method: a decentralized method accordingly to the principle of token passing is underlain by a

central method according to the master-slave principle. In order to manage the bus access, active stations have to build and manage a logical ring. Each active station has its own *List of Active Stations* (LAS), which represents all active members of the ring. According to the LAS, the token is passed on the ring from active to active station on ascending station address way, except if the token holder is the station with the higher address value. In this case, the token is passed to the station with lowest address value.

The station that possesses the token is referred as *this station* (TS). After receiving the token from the station immediately below in LAS, referred as *previous station* (PS), it is able to initiate the message cycles with the slaves. When it finishes, it has to pass the token to the immediate station on the LAS referred as *next station* (NS). If the token holder is the only active station on the ring it has to pass the token to itself. Each active station has two main types of tasks: an active phase where the station communicates with the associated slaves and a management phase, where the station performs maintenance ring operations. On the management phase the station has to keep its LAS updated in response either to a station insertion or removal from the ring.

### 3 Related Work

Fieldbus networks are usually used to support distributed real-time control applications [1]. Therefore in order to fulfill real-time system requirements, the communications protocols have to present small and predictable message latency with a minimal jitter [1,3,4]. Although the referred requirements are concerns of fieldbuses protocols, these are influenced by faults conditions such as the ones that lead to the corruption of transmitted data.

In the case of PROFIBUS, errors in transmitted data lead to outage events. These outages are interruptions of the communications services which can affect either the entire ring, or single stations. They result both from token losses and station removals from the logical ring.

Although outage events have a strong impact in message latency times and in the Worst Case Response Time (WCRT), usual schedulability analyses don't include their effects [7,8]. Usual PROFIBUS behavioral analyses are performed based on the performance characteristics of the protocol [9]. Therefore, these analyses are only correct if no faults occur during the network operation.

In the case of PROFIBUS, few analyses were performed to verify how it behaves in fault scenarios. In [10,11,12] the ring stability of Profibus in error-prone links is analyzed. This work focuses in transient faults and proposes both a simulation model [11], based on a proprietary development environment, and an analytical model [12] as an approximation to [11], to evaluate the network behavior. Although the model presented in [10,11], identify several outages events, the causes of some of them are not totally identified, and in some cases

their explanations contradict the observed network operation.

Thus and in order to identify the main outages events and their causes, an analysis of the network behavior is performed supported by a real PROFIBUS network.

## 4 Fault Injection Framework

A fault injection framework was developed in order to characterize, quantitatively and qualitatively, the behavior of a PROFIBUS network in the presence of fault scenarios. This framework was already presented in [13]. In this paper some improvements are introduced, mainly in the fault injection mechanism. The improvements were performed with the objective to increase the controllability of the fault injection tasks.

### 4.1 Evaluation using Fault Injection Techniques

Dependability evaluation can be performed from different perspectives which are related with the type of techniques employed. There are two main classes of techniques: **(i)** The ones that doesn't require a physical model and are based on simulation model that emulates the system behavior; **(ii)** Those that require a physical model to perform the analyses.

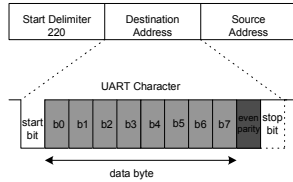
From dependability viewpoint it doesn't exist a technique that could be considered superior to the others. The advantages presented by one type of technique doesn't suppress the need of others techniques. On the contrary, they are complementary. Although simulation techniques are typically easy to use and to implement, they also present some disadvantages. One of the disadvantages is related with the fact that the results obtained depend of the model detail. Moreover the model needs to be validated, which in some scenarios imply a comparison of results with those obtained from the physical system. An example of the cooperation between the two types of techniques is presented in [14].

### 4.2 Framework Overview

Most of the faults in industrial environments are characterized by a transient temporal persistence [2]. Faults have its origins in external sources (e.g. motors, welding robots, etc) and in this context they affect both the communication bus and the transceivers. As result, data frames exchanged between stations has its contents corrupted by errors. PROFIBUS defines two main groups of frames:

- Action /Reply frames, which are used to exchange data between stations. Errors in these frames imply retransmissions, which lead to message delays. This can result in missed deadlines. The frame integrity is assured by means of a FCS (*Frame Check Sequence*) field (1 byte) which enables the stations to detect frame errors;
- Token frames, which are used to manage the logical ring. The token frame is composed by 3 UART

characters (Fig. 1). The token frame integrity is assured only by the character's parity bits. Therefore, it is possible that an error in SA or DA fields occur without being detected (e.g. 2 bit errors). An undetected error in those fields (wrong addresses) could lead to an erroneous token passing, which originates a disturbance in the logical ring. In other cases this can lead to a token loss.



**Figure 1 - Token frame and UART character formats.**

Since this paper focuses only the PROFIBUS MAC behavior, the discussion is restricted to token frames and FDL Status frames (action/reply) [6].

In order to evaluate the network behavior, it is necessary to produce a set of statistical independent experiments. In those experiments, faults are injected in the bus according a pre-defined distribution, which defines both their rate and length. By using probes, the network behavior is monitored and recorded. Dependability measures are obtained by a post-processing scheme applied on gathered data.

Since it is difficult to proceed with these experiments by using standard equipments (e.g. PLCs), because the protocol stack is normally hidden, the nodes used in the experiments were especially developed for it. The nodes are full compatible with the PROFIBUS protocol and totally accessible, which enables the access to all their internal functions.

### 4.3 Hardware Platform

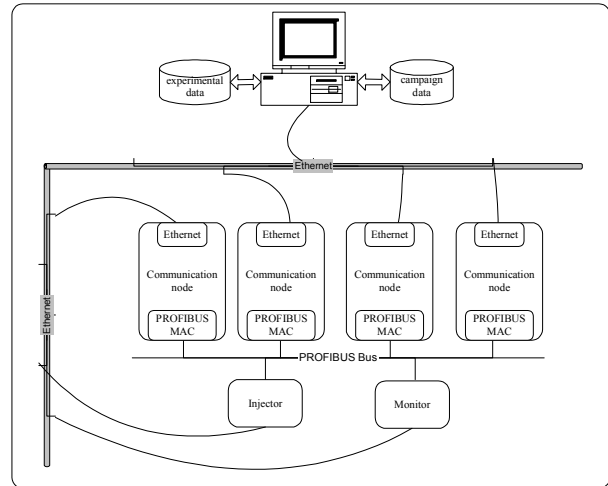
In this section the hardware platform used to support fault injection experiments is presented. To perform these experiments, a framework supported by a hardware platform is used (Fig. 2). This platform is composed by three main elements: **(i)** PROFIBUS nodes, which generate the network traffic; **(ii)** An injector, used to inject faults; **(iii)** A monitor node, used to gather and to process data. The PROFIBUS nodes are part of communication infrastructure. The injector and monitor modules compose the fault injection infrastructure.

These modules are interconnected by two communication networks: **(i)** An Ethernet network, used to configure the fault injection experiments and to retrieve data from it; **(ii)** A PROFIBUS network, used as part of the test bed for the experiments.

#### 4.3.1 Communication Infrastructure

The communication infrastructure provides a test bed for the fault injection experiments. It is composed by a PROFIBUS bus (physical medium) and by the commu-

nication nodes – PROFIBUS masters and slaves. This infrastructure can be used to perform two kinds of experiments: **(i)** Experiments where the main objective is to analyze only the logical ring behavior; **(ii)** Experiments where the main objective is to analyze the normal PROFIBUS behavior (full stack, e.g. messages exchange between masters and slaves). In the former one, analyses are performed only at FDL level and therefore their results are valid for both DP and FMS protocol profiles [6]. In the latter one, the network is configured to support the PROFIBUS-DP profile.



**Figure 2 - Hardware Platform.**

From the implementation viewpoint PROFIBUS masters are supported by the DSTni-LX-002 microcontroller [15]. The DSTni-LX-002 is a complete communication single chip solution that features high performance Turbo-186 compatible microprocessor. It includes a wide range of on-chip peripherals to support the most popular embedded networking technologies. The communication channels includes: Ethernet MAC, Dual CAN, PROFIBUS, SPI, and Dual Serial ports to handle the most demanding embedded applications.

The DSTni-LX-002 implements the ASPC2 ASIC from Siemens [16]. The ASPC2 is a MAC sub-layer that carries out the protocol of the PROFIBUS Fieldbus Data Link Layer (FDL). The interface service and management functions of the FDL are handled by software. The node configuration simplifies the access to the FDL services and user interface, allowing the collection of relevant data from the experiments. Information about the protocol behavior is supplied by a set of special registers of ASPC2, as follows:

- Detected faults by the MAC fault tolerance mechanisms, such as: *LAS Useless* (LAS inconsistency), *Pass-Token Error* (Error during token passing with the possibility of defective transmitter or receiver), *Response Error* (Error on a response frame), *Timeout* (Trigger of the timeout time in response to a token loss), *TS Address Error* (Multiple assignment of a station address);

- MAC states: *Hold-Token* (The station owns the token), *Listen-Token* (The station is in the listen token state), *Not Hold-Token* (The station doesn't own the token), *Offline* (FDL is in the offline state). These registers correspond to one or more states of PROFIBUS (FDL) state machine (Fig. 3).

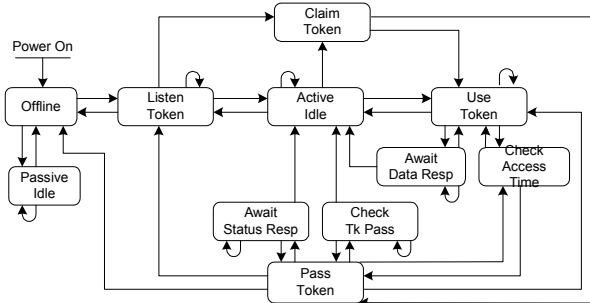


Figure 3 - FDL state diagram.

This information enables the characterization of the PROFIBUS behavior respecting to fault activation.

#### 4.3.2 Fault Injection Infrastructure

The fault injection infrastructure is composed by a set of components which perform the fault injection tasks and which observe their effects on the PROFIBUS nodes. In order to perform these activities the fault injection infrastructure is composed by two modules: the injector and the monitor.

##### 4.3.2.1 Injector Module

The injector is a dedicated module oriented to inject faults in the communication bus. A physical fault injection approach is used [14]. An electrical level it is imposed on a select location (electrical or electronic device), emulating therefore the fault effects. The type of the effect produced depends of the fault model used. The most usual physical fault injection models are: stuck at 0, stuck at 1, and bit flip. In the first two models it is imposed respectively a logical level 0 or 1 regardless of the original electrical level. The bit flip model corresponds to a logical level inversion. With this model it is possible to emulate transient faults. Therefore, this model is the natural choice to use in the injector module.

When a physical fault injection system is implemented there are some additional issues that must be considered. One is related with the real-time requirements of the fault injection process. In fact, it is not possible to slow down the system to inject the fault. Another important issue is the controllability of the fault injection process. This technique has inherent problems of controllability, namely to assure the respect of the fault injection time, location and desired value.

In order to satisfy these requirements the injector module was developed according to a hierarchical architecture (two levels). The bottom level is composed by an injector probe, which injects the physical faults. The top

level is composed by the fault injector control unit, which coordinates the fault injection experiments.

This architecture implies the use of dedicated hardware at the bottom level in order to fulfill the real-time requirements of the fault injection process, but has the advantage to reduce the real-time requirements at the top level.

##### 4.3.2.1.1 Injector Control Unit

The injector control unit is based on a microcontroller (DSTni-LX-002) whose software is responsible for the coordination of the hardware platform and for the management of the fault injection tasks. These activities can be grouped as following:

- Hardware platform synchronization. This is necessary to keep all system components synchronized, and it is performed in two phases. In the first one, the injector control unit interacts with others modules (PROFIBUS nodes and monitor module) to assure that they are correctly configured for the experiments. This is performed by using the Ethernet interface. In the second phase, a real-time synchronization with the PROFIBUS nodes is performed. This assures that the experiments start at the same time in all nodes. This is performed by means of PROFIBUS broadcast message;
- Dispatch of faults vectors. In order to manage the fault injection process, the injection control unit receives, at configuration time, a file with information about the injection parameters, including faults vectors and the associated injection times. A fault vector contains information about of the fault injection time instant and length. This vector represents a period of time equal to 16 *tbit*, (*tbit* is the time duration of one bit in the bus). By using this schema the time requirements at injector control unit are relaxed. Therefore, the dispatch of the injector control unit has only to send at the appropriate time the fault vectors to the injector probe.

##### 4.3.2.1.2 Injector Probe

The injector probe is used to inject faults in the communication bus in a controlled and accurate way. This requires that the injector probe has the capabilities to perform the injection tasks in real-time.

Since the smallest fault duration is equal to *tbit*, the injector probe must have a response time of a *tbit* fraction magnitude. As an example, for a 500Kbits/s network speed the response time is approximately 100ns.

It is also necessary to guarantee that each fault corresponds to an effective bit inversion. This requires permanent synchronization of the injector probe with the bus electrical signal. In order to fulfill these requirements all the tasks performed by the injector probe are implemented using high speed hardware.

A functional description of the hardware used is represented in the fig. 4. The injector probe is always reading the bus in order to obtain information about its signal. This information is processed by the *bit synchronization block*, which detects all transitions from logical level 1 to 0, excluding the ones that result from the fault injection process. Based on this information, the *oscillator & bus state acquisition trigger block* coordinates the remaining tasks of the injector probe. This is performed by the generating of the following signals: **(i)** The *serial clock*, to serialize the fault vectors (fault vectors are sent by the injector control unit through the host interface); **(ii)** The *enable signal*, to the *fault enable block*. This signal disables the injector probe during the period of time in which the injector acquires the bus signal; **(iii)** The *trigger signal*, for the *bus logical level detector*. Commanded by this signal the bus level is acquired and signal inversion is performed. Finally a fault is injected when both inputs of *fault enable block* are true. The latter implements a logical AND gate.

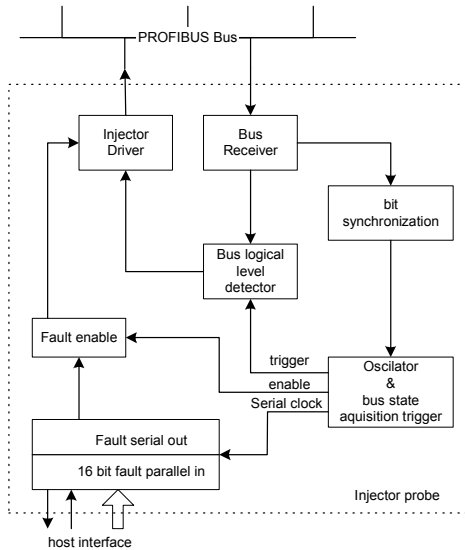


Figure 4 - Fault injection probe structure.

#### 4.3.2.2 Monitor Module

The monitor module is used to record all relevant events that occur in the bus (e.g. to verify fault activation and the corresponding effects). All PROFIBUS frames are recorded and time stamped.

At the end of an experiment recorded data is sent to a PC connected to the Ethernet network. Based on this data, a software application was developed to trace and log the system evolution. This application produces a list of events and identifies the most important ones (e.g. insertion and removal of stations, token lost, etc.).

Moreover, this application also permits to perform a behavioral analysis of commercial PROFIBUS nodes (e.g. PLCs) whose internal operation is normally not accessible. Therefore, the framework can also be used to

evaluate the internal and external behavior of commercial PROFIBUS networks placed in industrial environments. The main difference is that in this case faults are not injected but occur due to “natural” phenomenon.

## 5 Fault Injection Experiments

The experiments main goal is to analyze faults effects on PROFIBUS behavior, namely on the activation of fault-tolerant mechanisms and their impact on performance parameters such as outages events.

In order to perform the experiments the hardware platform was configured with 9 communication nodes. All this nodes are configured as masters, and their activity is only related with the maintenance of the logical ring, i.e. they only generate *Token* and *FDL Request Status* frames [6]. Their addresses are fixed and were generated according to normal distribution:  $S_{ad}=\{9, 20, 25, 32, 35, 38, 51, 69, 83\}$ . All parameters related with the FDL layer are maintained constant between experiments (Tab.1).

Table 1. FDL parameters.

Parameter	Value
Bit rate	500kbit/s => tbit=2μs
$T_{TR}$ - Target rotation time	20ms -10000 tbit
$T_{ID1}$ - Idle Time 1	37 tbit
$T_{ID2}$ - Idle Time 2	100 tbit
$T_{SL}$ - Slot Time	200 tbit -
$T_{RDY}$ - Ready Time	11 tbit
HAS	126
$G=6$	$T_{GUD} = G \times T_{TR}$

Different scenarios were analyzed by changing of the following parameters:

- **Bit error rate (ber).** To verify error sensitivity, faults are injected into the communication bus according to a geometric distribution with different error rates. This distribution was chosen because faults are injected at discrete time instants (*tbit* multiples) and because it has no memory (memoryless property) [17];
- **Error length.** In order to verify the effects of fault length in the behavior of fault-tolerant mechanisms, and in particularly the effects of undetected token errors, a set of different error lengths were used: 1, 2 and 4 bits.

Data gathered from experiments is analyzed and the main events are identified and classified as following:

- **System outage.** This event is caused by a token loss. This hampers, temporarily, the possibility of all network nodes to execute their message cycles, leading to a system outage. The generation of a new token is triggered by a timeout mechanism. To avoid a simultaneous triggering a mechanism assures that a

new token is generated in the station with the lowest address;

- **Station outage.** This event only affects the stations that were removed from the logical ring by the protocol mechanisms. Outside the ring, the station cannot perform their communication functions which lead to their outage. The station only recovers from this situation when it is inserted in the ring by another master.

## 5.1 System Outage

System outage has its origins on errors during token pass procedures. The token pass procedure takes place whenever the master has finished its message cycles. The token is passed to its successor (NS) by transmitting a token frame. If after transmitting the token frame and within a *slot time* ( $T_{SL}$ ) the token transmitter receives a valid frame it assumes that the NS owns the token. During this period of time the master also monitors their transmission by using a loopback mechanism. This mechanism allows the station to verify the state of the transceivers.

When the token pass procedure is disturbed several outage events may occur. These are discussed in the following sections.

### 5.1.1 Fatal Token Error

If during a token transmission the station doesn't receive its own token frame, than there is a fatal error in the transceiver. The station stops its activity in the logical ring, enters in the *Offline* state and the token is lost. In this scenario the LAS contents are also lost.

Although this behavior results apparently from a permanent fault, transient faults can also produce the same results. This occurs when the token *start delimiter* (SD) is affected by faults. The ASPC2 implementation assumes a fatal error when two consecutive SD errors occur.

### 5.1.2 Normal Token Error

If two consecutive token frames contains errors (not fatal ones), the station leaves the ring and enters in the *Listen token* state. In this scenario the LAS contents are maintained.

### 5.1.3 Slot Time Error

If after a token transmission and within the *slot time* an error pattern occurs during the remaining bus *idle time*, the station assumes that another master owns the token. This is due to the fact that these errors generate UART characters which are interpreted as a frame header. If the token wasn't successfully passed then this leads to a token loss.

### 5.1.4 Ring Initialization

The ring initialization is a special event which repre-

sents the worst recovering scenario from a system outage. This event occurs according the following conditions. Whenever a station leaves the *Offline* state and enters in the *Listen Token* state, it needs to build their LAS. If during the building process a token loss occur and the address of station is the lowest address, this station performs a ring initialization. This procedure implies the removal of all stations from the logical ring and the establishment of a new LAS.

## 5.2 Station Outage

Station outages events were observed as consequence of 3 scenarios in which token frames are affected by errors: *TS Address Error*, *LAS Inconsistency* and *Station Jump Over*.

The *TS Address Error* and *LAS Inconsistency* events are caused by undetected errors. *Station Jump Over* is caused by a combination of factors that lead the station to be skipped in the token pass procedure.

### 5.2.1 TS Address Error

When two consecutive tokens with Source Address (SA) equal to the station address are monitored, the station leaves the logical ring and enters in *Listen Token* state. In the experiments this event was observed for the stations  $S = \{32, 35, 38, 51, 83\}$ , which have addresses that are interchangeable by a two bit error length pattern.

### 5.2.2 LAS Inconsistency

In order to keep the LAS updated, active stations have to constantly to monitor token frames in the bus. Nevertheless, some errors can lead to a *LAS Inconsistency*. In this case a station verifies that its LAS contents are inconsistent with the real logical ring organization. When this error occurs the station leaves the logical ring until re-establishing their LAS.

In the experiments this error was observed when a token frame was corrupted and the resulted frame violates the insertion and removal PROFIBUS station rules. As consequence, the logical ring partially collapses and in most situations the only stations which are not affected are the ones involved in the token pass procedure.

### 5.2.3 Station Jump Over

This event has its source in the mechanisms that allow removing stations from the ring.

In the case of an absence of reaction from the NS, the station that owns the token retries to pass the token for two times. When this value is reached and if the token wasn't passed, the station tries to pass it to the next station in LAS. This process is repeated until the token pass procedure succeeds. The stations which are skipped in the token pass process are then removed from the LAS. This behavior can happen due two reasons: **(i)** undetected token errors (SA or DA errors) by other stations; **(ii)** Undetected errors by the token owner loopback mechanism but which are detected by others stations.

Undetected errors by the loopback mechanism can have its origins in physical factors that are present in real applications, such as:

- Electrical signal strength. The electrical signal that creates the errors (fault injection) is influenced by the bus electrical parameters, such as: capacitance, resistance and inductance. Thus a signal attenuation can interfere with the error detection;
- The configuration of the RS-485 transceiver used by the PROFIBUS communication nodes. When the loopback mechanism is activated it is possible that the injector signal level is too low to produce an undetected error by the token sender, but which is detected by the other stations.

## 6 Experimental Results

In order to quantify the main causes of ring instability a group of metrics were defined. Those metrics are oriented to quantify events probability, rather than their temporal implications.

Let  $n$  be the number of independent experiments,  $E_{ij}$  the number of observed events of type  $i$  (*Fatal Token Error, Normal Token Error, Ring Initialization, TS Address Error, LAS Inconsistency, Station Jump Over*), in the experiment  $j$  and  $TK_j$  the total number of observed token frames in the experiment  $j$ . By assuming that at the beginning of every experiment the ring is in steady-state, the event probability estimator  $\hat{p}_i$  is given by:

$$\hat{p}_i = \frac{1}{n} \sum_{j=1}^n \frac{E_{ij}}{TK_j} \quad (1)$$

The experiment length was fixed in 3.5 seconds in order to guarantee that the majority of the samples (events) are acquired when the network is in a stochastic steady-state. Behavioral data (samples) were obtained according to the *independent replication method* [17]. Estimators were defined using a 95% confidence interval with 5% of relative error (interval width) for the most frequent event. Less frequent events were obtained with a relative interval of 20%.

To present the results in a compact way, data gathered by the above metrics were grouped according to their effects in the two event classes defined previously: *System Outage* (§5.1) and *Station Outage* (§5.2). Since the *Ring Initialization* metric represents a special case in which is necessary to establish a new logical ring, its value is represented in separate.

The behavior of *System Outage*, *Station Outage* and *Ring Initialization* metrics are presented respectively in figs. 5, 6 and 7. These graphics shows the event probability versus two parameters: bit error rate ( $ber$ ) and bit error length ( $bel$ ). In the next sections these results are analyzed from two viewpoints: one parameter is maintained fixed and while the other varies.

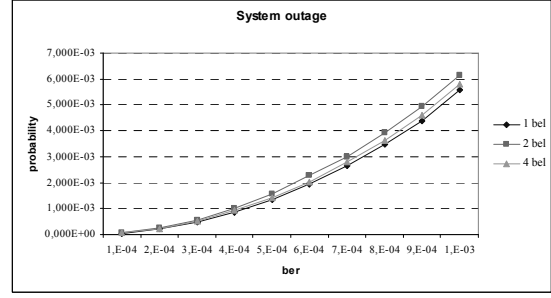


Figure 5. System Outage.

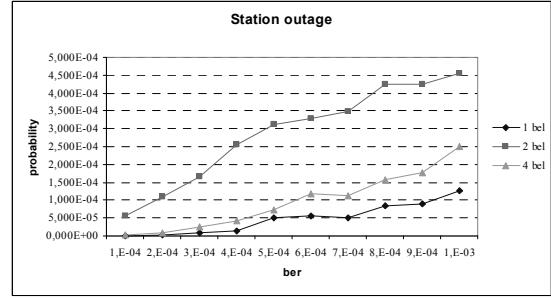


Figure 6. Station Outage.

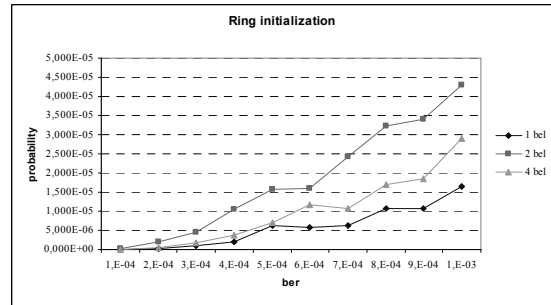


Figure 7. Ring Initialization.

### 6.1 Constant Bit Error Rate versus Variable Bit Error Length

From figs. 5, 6 and 7, it becomes clear that the *System Outage* event has the highest probability. It is also show that this event has a low sensitivity to error length variations (Fig. 5).

A very small difference exists between the *System Outage* probabilities for 1 *bel* and 4 *bel* cases (Fig. 5). With 2 *bel* only a slight increase is verified. This difference is justified by the fact that in some scenarios there is the possibility of additional token losses. This occurs when some stations have its LAS inconsistent. In this case if the token owner doesn't know its successor in the logical ring and it receives a corrupted frame then the token is lost.

It can be concluded that *System Outage* event is independent from the error length and it is only marginally affected by undetected errors which cause a *LAS Inconsistency* event.

The results also show that *Station Outage* is highly sensitive to undetected errors (Fig. 6). This is observed by comparing the 2 *bel* results with the 1 *bel* and 4 *bel* ones. This behavior is justified by the fact that some conditions that lead to a *Station Outage* event are caused by undetected errors. The events which are affected by the referred conditions are: *TS Address Error* and *LAS Inconsistency*. The *TS Address Error* is a special case since is only observed for 2 *bel* patterns.

In the experiments the *Ring Initialization* event only happens after a previous *System Outage* occurrence. Thus in terms of error length sensitivity, the *Ring Initialization* event should present a behavior which is similar to the *System Outage* one. However curves in fig. 7 presents some oscillations, which are caused by the relative error interval of 20%. There are significant differences between 1, 4 and 2 *bel* scenarios. This difference has the same justification of the *System Outage* case and it is increased by the fact that when a *LAS Inconsistency* event occurs the station with the lowest address is usually in the group of affected stations.

## 6.2 Variable Bit Error Rate versus Constant Bit Error Length

In fig. 5 it is observed that the *System Outage* event is highly sensitive to the *ber*, and its probability increases according to an exponential law.

This behavior is strongly correlated with the *slot time* ( $T_{SL}$ ) value. For a small *ber* the interval between errors tends to be higher than  $T_{SL}$ . Therefore, the number of error within  $T_{SL}$  becomes smaller. For a high *ber*, the interval between errors tends to be less or equal than the token frame length plus bus *idle time*. In those conditions the probability to occur a *Slot Time Error*, *Fatal Token Error* or a *Normal Token Error* is higher.

Due to their dependence of the *System Outage* event, the *Ring Initialization* event presents a similar behavior, but with a small magnitude.

If the interval between errors has the same magnitude of  $T_{SL}$ , the probability of two or more consecutive faulty tokens increases. This condition leads to the occurrence of *Station Outage* events, and therefore this type of event also depends from the  $T_{SL}$  value.

## 6.3 Comparative Analysis

From the analysis performed on previous sections the *System Outage* event was identified as the outage with the highest probability. Since this event is composed by several events (*Fatal Token Error*, *Normal Token Error* and *Slot Time Error*), it is necessary to characterize the contribution of these individual events for the global system outage. Their relative values are represented in figs. 8 and 9.

From fig. 8 it is clear that the *Slot Time Error* is the dominant event. Their ratio represents between 97.19% and 97.98% of all token losses. The *Fatal Token Error* presents a residual value in 2.02% to 2.81% range.

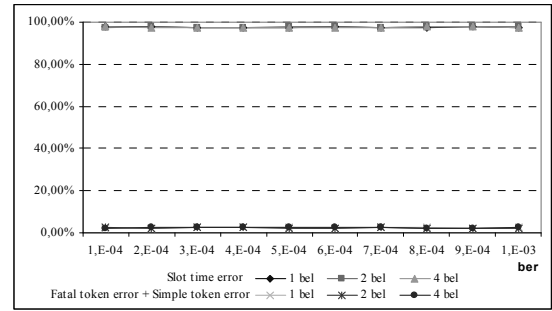


Figure 8. System Outage.

A comparison between the two outage classes (*System Outage* and *Station Outage*) is presented in fig. 9. From a probability viewpoint, the results show a bipolarization for 1 *bel* and 4 *bel* cases. In this case the results are in 94.53% to 99.58% range. The *Station Outage* component represents between 0.42% and 5.47%, of all observed outages.

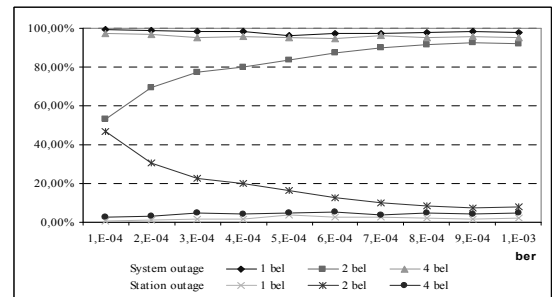


Figure 9. System Outage vs Station Outage.

For the 2 *bel* case, a significant number of *Station Outage* events are observed. This occurs for a low *ber* value, but it tends to converge for the 1 *bel* and 4 *bel* cases when the *ber* value increases. This behavior results from the characteristics of the 2 *bel* errors (undetected errors) and the interval between errors, which tend to be higher than  $T_{SL}$  at high *ber* values. In these conditions, the *TS Address Error* and the *LAS Inconsistency* have a major importance.

## 7 Conclusions

The paper presents a dependability evaluation of PROFIBUS networks. This evaluation was carried out in order to verify the stability of logical ring and identify qualitatively and quantitatively the main outages causes. To perform this evaluation a hardware platform able to inject faults and to emulate real PROFIBUS networks was developed.

The network operation was disturbed by a set of fault injection experiments and the following events were identified: (i) Fatal Token Error, Normal Token Error and Slot Time Error are associated with the token loss, which leads to system outages; (ii) TS Address Error, LAS Inconsistency and Station Jump Over are errors that lead to station outages.

The analysis of these results allows establishing the following conclusions:

- Most of PROFIBUS outages are caused by token losses;
- There is a high relative difference between the sources of system outages: **(i)** Slot Time Error with a valuable occurrence probability; **(ii)** Fatal and Normal Token errors have a very low probability of occurrence in comparison the previous one;
- High sensitivity of PROFIBUS to errors within the slot time.

Another issue associated to the identified outage events is related with its implication on the PROFIBUS temporal response in those conditions. Although not addressed in this paper the outage events have significant impact in attributes such as: station availability, token cycle time jitter and message latency time. Thus the influence of outage events in such attributes should be taken into account in order to build dependable applications.

## References

- [1] J. P. Thomesse, "A review of the Fieldbuses", *Annual Reviews in Control*, Vol. 22, pp. 35-45, 1998.
- [2] H. Kim, A. White, K. Shin, "Effects of Electromagnetic Interference on Controller-Computer Upsets and System Stability", *IEEE Transactions on Control Systems Technology*, Vol. 8, pp. 351-357, 2000.
- [3] H. Kim, K. Shin, "On the Maximum Feedback Delay in a Linear/Nonlinear Control System with Input Disturbances Caused by Controller-Computer Failures", *IEEE Transactions on Control Systems Technology*, Vol. 2, No. 2, pp. 110-122, 1994.
- [4] K. Shin, H. Kim, "Derivation and Application of Hard Deadlines for Real-Time Control Systems", *IEEE Transactions on Systems, Man and Cybernetics*, Vol. 22, No. 6, pp. 1403-1413, 1992.
- [5] P. Portugal, A. Carvalho, "A Simulation Model Based on a Stochastic Petri Net Approach for Dependability Evaluation of PROFIBUS-DP Networks", *Proceedings of 10th Int. Conf. on Emerging Technologies and Factory Automation – ETFA'05*, 2005.
- [6] EN 50170, *General Purpose Field Communication System*, Volume 2/3 (PROFIBUS), CENELEC, 1996.
- [7] E. Tovar, F. Vasques, "Real-time Fieldbus Communications using Profibus Networks", *IEEE Transactions on Industrial Electronics*, Vol. 46, No.6, pp. 1241-1251, 1999.
- [8] E. Tovar, F. Vasques, "Setting Target Rotation Time in Profibus Based Real-Time Distributed Applications", *Proceedings of the 15th IFAC Workshop on Distributed Computer Control Systems*, 1998.
- [9] H. Seung, K. Ki, "Implementation and Performance Evaluation of Profibus in the Automation Systems" in *Proceedings of the IEEE International Workshop on Factory Communication Systems*, 1997.
- [10] A. Willing, A. Wolisz, "Ring Stability of the PROFIBUS Token-Passing Protocol Over Error-Prone Links", *IEEE Transactions on Industrial Electronics*, Vol. 48, No. 5, pp. 1025-1033, 2001.
- [11] A. Willing, "Analysis and Tuning of the PROFIBUS Token Passing Protocol for Use over Error Prone Links", TKN Technical Report TKN-99-001, 1999.
- [12] A. Willing, "Markov Modeling of the PROFIBUS Ring Membership over Error Prone Links", TKN Technical Report TKN-99-004, 1999.
- [13] J. Carvalho, P. Portugal, A. Carvalho, "A Framework for Dependability Evaluation of PROFIBUS Networks", *Proceedings of International Symposium in Industrial Electronics – ISIE'03*, 2003.
- [14] J. Arlat, Y. Crouzet, J. Karlsson, P. Folkesson, E. Fuchs, G. Leber, "Comparison of Physical and Software Implemented Fault Injection Techniques", *IEEE Transactions on Computers*, Vol. 52, No. 9, pp. 1115-1133, 2003.
- [15] *DSTni-LX Data Book, Revision E*, Grid Connect, 2003.
- [16] *Simatic Net ASPC2 / Hardware User Description Advanced PROFIBUS Controller According to EN 50170, Version V2.0*, Siemens, 1999.
- [17] J. Banks, *Discrete-Event System Simulation*, Prentice Hall International Editions, 1996.

