Investigating Connector Faults in the Time-Triggered Architecture

P. Peti, R. Obermaisser, and H. Paulitsch
Vienna University of Technology, Austria
email: {php,ro,harald}@vmars.tuwien.ac.at

Abstract

In the context of distributed real-time systems as deployed in the avionic and the automotive domain a substantial number of system malfunctions result from connector faults. For instance, a middle class car has more than 40 Electronic Control Units (ECUs) interconnected by a heterogenous network infrastructure consisting of hundreds of wires and connections. Connector faults such as loose contacts impose a challenging task for the technician at the service station. This paper investigates to what extent the use of time-triggered communication protocols, in particular the TTP C2 communication controller, helps in identifying connector faults. We perform fault injection campaigns to judge whether the status information provided by the TTP C2 controller is sufficient for the detection of connector faults. The derived results constitute an important input for online analysis mechanisms.

1 Introduction

Wiring plays a central role in any distributed real-time system environment, since it provides the infrastructure for exchanging information between components. Like any other part of the system, the electrical interconnection system is exposed to environmental stress as well as assembly faults. Considering that a typical middle class car has about 40 ECUs and approximately 800 wires [19], the likelihood of connector problems is high. In fact, recent studies [22, 9, 21, 8] indicate that a substantial number of electrical failures can be attributed to connector problems. These faults are typically of transient nature and especially tricky to diagnose at the service station, since the testing procedure may itself be a corrective action [17].

Since time-triggered networks are gaining more and more momentum due to the deterministic and fault-tolerant transport of messages in combination with temporal composability, industry is currently shifting from event-triggered communication systems to time-triggered ones in the context of safety-relevant and safety-critical systems [1].

In this paper we investigate architectural means provided by the Time-Triggered Protocol (TTP) for the detection of connector faults in the Time-Triggered Architecture (TTA) using a bus topology. We analyze to what extent a priori knowledge about the transmission instants and the frame status information provided by a communication controller implementing TTP can be used to discriminate between internal faults, non-destructive external faults (e.g., Electromagnetic Interference (EMI)), and connector faults. For this purpose we employ a fault injection framework that allows perturbing a TTP cluster with EMI and injecting bus failures using a hardware device capable of injecting a large variety of physical faults. The results of the campaigns show that the status information of the TTP C2 controller can be used to distinguish between the above mentioned fault classes. In contrast to internal and non-destructive external faults that typically result in crash failures of nodes, for connector faults the arrival of syntactically incorrect (e.g., encoding error) frames has shown to be a suitable indicator.

The paper is structured as follows. In Section 2 we present recent studies on the problem of connector faults. The Time-Triggered Architecture (TTA) is discussed in Section 3 and our fault model for maintenance purposes is discussed in Section 4. In Section 5 we elaborate on mechanisms provided by TTP to detect possible connector faults. A description of the used fault injection framework follows in Section 6. In Section 7 the results of the fault injection campaigns are discussed and interpreted.

2 Connector Faults in Distributed Systems

Wiring and connector problems account for a significant proportion of electronic system failures. Several studies document the impact of connector and wiring failures in distributed embedded systems. Field data cited by Swingler et al. [22] indicates that more than 30% of electrical failures are attributed to connection problems. Considering, that a luxury car can have up to 400 connectors this number underpins the potential for failures due to connectors in the automotive domain. In the avionic domain, Galler and Slenski identify interconnection problems as the major cause of aircraft electrical equipment failures [9] with a percentage of 36%. These numbers are supported by a study of the US Air Force reporting that

43% of mishaps related to electrical systems were due to connectors and wirings [21, 8].

The reliability of the interconnection system is of great importance for the correct operation of an electronic system. The physical interface between the electronic control units (i.e., the components) and the interconnection system remains one of the weakest links in terms of reliability, implying potentially catastrophic consequences [16].

According to Tanner [23] several issues concerning the reliability of connectors can be identified, such as terminals backing out of housing, breakdown of contact at crimp, or failure of contact due to operational conditions [29]. A significant problem regarding connector and wiring failures is the fact that the failure analysis or the testing procedure may itself be a corrective action for the source of the failure [17]. The same is pointed out by [12], who states that analysis and repair is often difficult due to the possibility that any evidence of failure is inadvertently destroyed during extraction or inspection.

3 The Time-Triggered Architecture

In this section we present the system model that is based on the TTA. The TTA provides a computing infrastructure for the design and implementation of dependable distributed real-time systems [14].

3.1 System Structure

The basic building block of the TTA is a node computer, which is a self-contained composite hardware/software subsystem. A cluster is a set of nodes that are interconnected by two redundant communication channels. For the interconnection of nodes, the TTA distinguishes between two physical interconnection topologies, namely a TTA-bus and a TTA-star.

A TTA-bus consists of replicated passive buses. Every node has to bus guardians (one for each channel), which use the a priori knowledge about the points in time of communication activities to prevent communication outside a node's slot. In the TTA-star topology, the interconnection of nodes occurs via two replicated central guardians. The star topology has the advantage of a higher level of independence, since guardians are located at a physical distance from nodes. Furthermore, guardians reshape signals and support additional monitoring services [3].

3.2 Communication Protocol TTP

For providing the communication service, the TTA employs the fault-tolerant protocol TTP [13], which provides the following services:

Message Transport. TTP uses Time Division Multiple Access (TDMA) to control the media access to the two independent communication channels. Time is partitioned into slots, which are statically assigned to nodes. A sequence of slots that allows each node to send a message forms a TDMA round. A sequence of predefined TDMA rounds is called a cluster cycle. The cluster cycle also defines the periodicity of the message transmissions. Information about the points in time of all message transmissions during a cluster cycle are contained in the message schedule.

Clock Synchronization. TTP provides fault-tolerant clock synchronization via the Fault Tolerant Averaging (FTA) clock synchronization algorithm [15]. The clock synchronization algorithm of the TTA has been formally verified in [20]. It differs from other algorithms by the fact that no special synchronization messages are used for the exchange of the local clock values of nodes. The difference between the expected and the actual arrival time of an incoming message is used to estimate the deviation between the local clock of the receiver and the sender.

Membership Service. The membership service provides nodes with consistent information about the operational state of every node in the cluster. In case multiple cliques with different membership views form, clique avoidance ensures that the minority cliques leave the membership [4].

4 A Fault Model for Maintenance

From a maintenance point of view, we are only interested in categorizing the type of fault of the experienced failure into classes that allow a determination whether a replacement of a Field Replaceable Unit (FRU) is the correct maintenance strategy. In order to take this type of faults into account, we extend the boundary classification of faults as introduced by Laprie [2] by adding the class of connector faults. Therefore, we devise the following three fault classes of a system component as illustrated in Figure 1. Faults that originate outside the component boundaries are denoted as *non-destructive external faults*. Non-destructive external faults are characterized by having no permanent effect on the functionality of the component. A restart of the component with subsequent state synchronization is a typical strategy to restore a correct state. An example for a non-destructive fault is EMI [11]. So-called *borderline faults* are the class of faults that cannot be judged to be external or internal with respect to the component boundary (a connector consist of two parts, one attached to the component, the other attached to the cable loom). Finally, *internal faults* cover those faults that originate from within the FRU boundary (e.g., crack in the Printed Circuit Board (PCB)). In contrast to external faults, these faults can at the service station only be eliminated by a replacement of the component. For more detailed information on the fault model see [18].

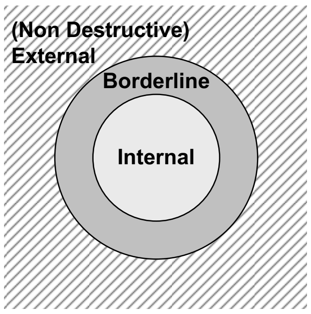
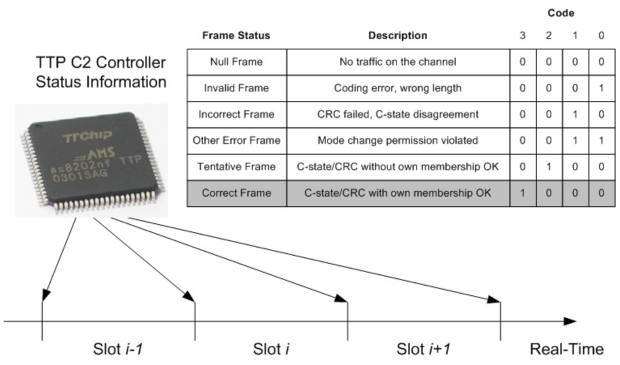


Figure 1. Component Fault Model



Frame Status	Description	Code 3	2	1	0
Null Frame	No traffic on the channel	0	0	0	0
Invalid Frame	Coding error, wrong length	0	0	0	1
Incorrect Frame	CRC failed, C-state disagreement	0	0	1	0
Other Error Frame	Mode change permission violated	0	0	1	1
Tentative Frame	C-state/CRC without own membership OK	0	1	0	0
Correct Frame	C-state/CRC with own membership OK	1	0	0	0

Figure 2. Frame Status Information Provided by the TTP C2 Controller

5 Error Detection Capabilities of the TTP C2 Communication Controller

The a priori known send instants of messages in a time-triggered communication system periodically provide diagnostic information. Each receive instant enables a receiver node to detect a message failure resulting from either a failure of the sender node, a failure of the communication channel, or a failure of the receiver node. A node not only detects syntactic errors of a message (e.g., invalid message header), but also message timing failures when a message fails to arrive at the predefined receive instant.

As depicted in Figure 2 the TTP C2 controller [24] determines the frame status for each received frame every TDMA slot. The so-called *Error Indication Field* allows an analysis of the status of each of the communication channels. The fault-tolerance layer as part of the operating system TTPos [28] compares the frames from both channels and declares the received frame as correct as long as one frame is correct (see also Section 6). This strategy is suitable for application transparent fault-tolerance, however, for diagnosis the status information of both channels must be taken into account to enable an investigation of possible physical faults of the replicated channels or bus drivers. In TTP [13] we can use the frame status information as described in the following for the detection of connector failures.

Invalid Frame: An Invalid Frame is a frame that is syntactical invalid, i.e., coding rules (e.g., coding or expected length) are violated. Additional information of other nodes is needed to decide on the cause of the Invalid Frame. For example, in case all other nodes of the cluster perceive the frame as correct, then the receiver or the cabling/connector is likely to be faulty. In contrast to a Null Frame, a signal on the bus is received (i.e., there are signal edges on the bus that cannot be decoded).

Null Frame: A Null Frame indicates either a faulty receiver, a faulty sender (or no sender at all) or a cabling defect. In contrast to an Invalid Frame, no signal on the bus has been observed.

Incorrect Frame: An Incorrect Frame is a syntactically valid frame (coding and size correct) for which all Cyclic Redundancy Code (CRC) checks have failed at the receiver.

Other Error Frame: An Other Error Frame is a frame which passed all CRC checks, but holds an invalid cluster mode change request.

Tentative Frame: A frame that is correct after the second CRC check. This means the first successor during the acknowledgment considers the controller faulty.

Correct Frame: A Correct Frame is a frame which passed all CRC checks at the receiver.

Besides the frame status field, the Communication Network Interface (CNI) of the C2 controller also provides controller registers (e.g., Failed Frame Counter, Agreed Frame Counter) indicating the relationship of the counter registers with respect to the TTP protocol errors as illustrated in Figure 3.

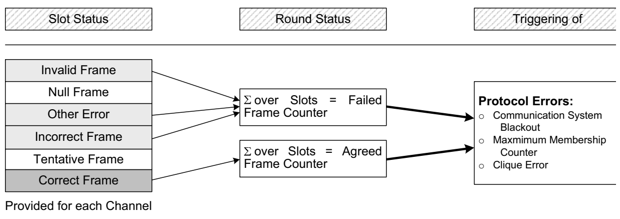

Slot Status
Round Status
Triggering of
Invalid Frame
Null Frame
Other Error
Incorrect Frame
Tentative Frame
Correct Frame
Provided for each Channel
Σ over Slots = Failed Frame Counter
Σ over Slots = Agreed Frame Counter
Protocol Errors:
○ Communication System Blackout
○ Maxmimum Membership Counter
○ Clique Error

Figure 3. TTP Protocol Errors

6 Overview of Fault Injection Framework

This section describes the constituting parts of the fault injection setup, namely a set of TTP nodes interconnected by a time-triggered communication bus and the two types of fault-injection devices (TTP Disturbance Node, NSG 1025 Fast Transient/Burst Generator). In addition, we discuss the software setup for controlling fault injection and performing error detection in the TTP nodes.

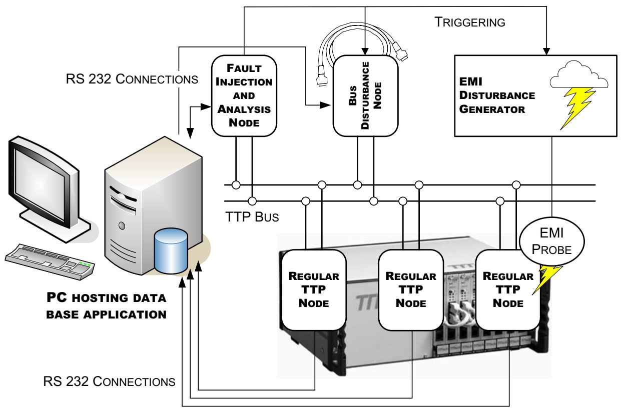


Figure 4. Setup of the Hardware Fault Injection Experiments

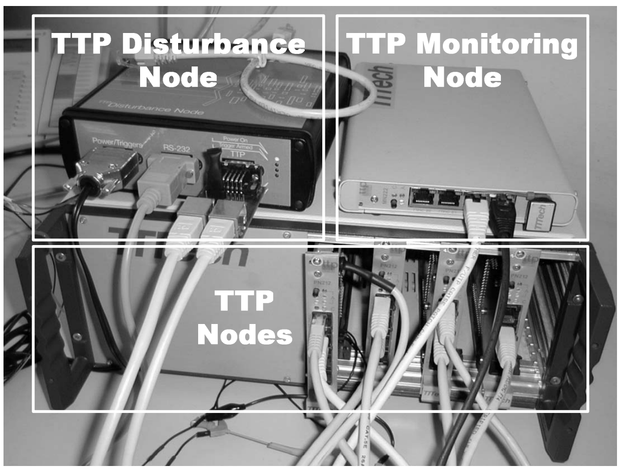


Figure 5. Fault Injection using the Disturbance Node

6.1 Setup

In the setup of the fault injection framework as depicted in Figure 4 we employ a TTP MPC555 cluster consisting of four nodes [25]. In the framework we utilize two hardware fault injection devices [10], namely an EMI disturbance generator and a bus disturbance node. While the latter is used to emulate connector faults, the EMI generator is used to simulate both internal and external component faults according to the fault model introduced in Section 4. The EMI generator exposes the target system to radiated energy, i.e., no power supply induced EMI.

6.1.1 TTP Cluster Description

In the framework we use a TTP-Development Cluster based on four PN212 TTP-Powernodes mounted in a rack and a TTP-Monitoring Node for downloading configuration data (i.e., the cluster schedule) and application code. Each PN212 TTP-Powernode [26] is equipped with a TTP C2 controller AS8202 (C2) and uses a Freescale MPC555 processor for the execution of application tasks. In addition to a 5 Mbps TTP interface, a broad variety of interfaces is supported: ISO 9141 (suitable for TTP/A, LIN, and ISO-K), CAN, digital I/O, and analog inputs. In the fault injection setup as depicted in Figure 4, the TTP interface is exploited and nodes are interconnected in a bus topology. TTPos is used as the operating system [28]. In the cluster we distinguish between two types nodes, the *regular nodes* and the *fault-injection and analysis node*. While the latter is used to control the fault injection experiments, the regular nodes are subject to perturbation. For a detailed discussion about the deployed software refer to Section 6.2.

6.1.2 The NSG 1025 Fast Transient/Burst Generator

The NSG 1025 fast transient/burst generator manufactured by Schaffner Instruments is used as an EMI-disturbance generator in our fault injection framework. The generator is designed for the use in lab and field-testing due to its capability of producing different types of disturbance signals. In particular, the ability to control the triggering of the EMI disturbances by software makes the system especially useful for lab testing, whenever a large number of experiments is required. Once the type of disturbances and parameters (i.e., the amplitude and frequency) has been selected, the fault injection campaign can be run automatically. All types of disturbances can be controlled using the trigger/gate port of the generator at TTL-voltage, thus the generator can be controlled using a standard output port of a node. Whenever no signal is provided, the generator transmits no disturbances. The EMI-generator can generate single pulses (5/50 ns), repeated single pulses and burst disturbances.

6.1.3 The TTP Disturbance Node

In the test setup the TTP-Disturbance Node [27] as depicted in Figure 5 is used to study the effects of connector faults. A key advantage of the disturbance node is the reproducibility of the test cases. The disturbance node is easily configured using XML description files and can be triggered directly by the fault injection and analysis node via the output port pins. The disturbance node can be used to inject a variety of classes of faults, thus allowing the analysis of the behavior of the physical, logical, and application layer. For instance, the following faults can be injected: loss of transmissions, short-circuits of bus-lines to VCC and ground, mismatched termination of bus wires, noise burst, loss of specific data (reproducible experiments possible), or a wrong CRC.

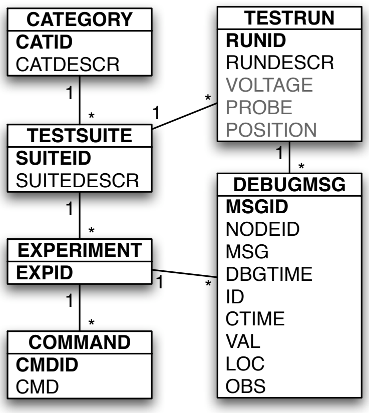


Figure 6. Extended Entity Relationship Model

6.1.4 PC hosting Database

The database application running on the PC constitutes the main control instance of the fault injection environment and also provides the main user interface to define and conduct the fault injection experiments. The application provides graphical user interfaces to configure the serial interfaces, to define the fault injection campaigns and to monitor the test runs. A standard SQL'92 database (InterBase v.6.01) serves to store the experiment definitions and the outcome of the fault injection campaigns, i.e., a set of debug messages.

The database is implemented according to the Extended Entity Relationship (EER) model depicted in Figure 6. The categories allow to differentiate between the two fault injection devices. A test suite comprises all experiments run at once in a test run. During a test run debug messages are generated that are linked to the experiments that have provoked them.

A test run includes the optional attributes `VOLTAGE`, `PROBE`, and `POSITION` that are used in the case of a test run with the EMI generator and describe the voltage and probe chosen. Furthermore the attribute `POSITION` allows to describe the positioning of the probe relative to the unit under test.

The attributes `RUNDESCR`, `CATDESCR`, and `SUITEDESCR` are human readable names for the numerical identifiers.

A debug message has several attributes. The attributes `ID`, `CTIME`, `VAL`, `LOC`, and `OBS` store the values of the diagnostic message. These values are type, time, value, location, and observer and map to the attributes of a `DEBUGMSG` in this order. All these attributes are integers.

The attribute `MSG` allows storing the whole diagnostic message received. The attribute `NODEID` holds the number of the node that forwarded the diagnostic message to the computer; in our setup always Node 0. Furthermore in the attribute `DBGTIME` a timestamp is generated when the debug messages is written into the database.

Finally, the attribute `CMD` holds a command string, which embodies a command for one of the two fault injection devices or a command for the embedded triggering application (e.g., wait for a certain amount of time).

6.2 Embedded Software

In the following we discuss the software running on the four TTP nodes.

6.2.1 Embedded Triggering Application

The embedded triggering application running at the fault injection and analysis node (see Figure 4) is controlled by user requests received via the serial connection from a PC. After processing the input, the real-time task forwards the request to the trigger interface of the selected fault injection device by the application of TTL-Signals. Both fault injection devices (disturbance node and EMI disturbance generator) trigger a disturbance at the rising edge of the signal applied to their trigger interface and stop it at the falling edge. In the following we discuss embedded triggering application for controlling the EMI disturbance generator.

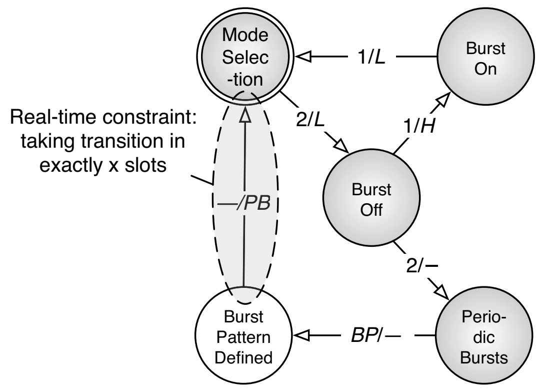


Figure 7. State Diagram of the Embedded Triggering Application

The embedded triggering application is modeled as a Finite State Machine (FSM) consisting of 5 states. The FSM is illustrated in Figure 7. Each transition is augmented with two symbols. The symbol shown before the slash represents the input required to shift from one state to another in the direction of the arrow. The symbol after the slash denotes the action executed upon the transition.

Figure 8 describes the symbols used in the state diagram of the FSM. Either a single burst can be activated and deactivated manually, or a specific burst pattern can be executed. The burst pattern (input symbol *BP*) is defined by eleven digits. The first three denote the period in slot granularity at which the disturbance is repeated, the forth

Symbol	Description
H	The signal is set to HIGH.
L	The signal is set to LOW.
PB	Periodic burst: the required signal to produce the requested periodic burst pattern is generated. Finally, the signal is LOW.
—	No input/action.
1	ASCII digit 1.
2	ASCII digit 2.
BP	Burst pattern: the 11 digit burst pattern.

Figure 8. Descriptions of the Symbols

and the fifth digit denote the offset/slot at which the first disturbance starts, the sixth till the eighth digit denote the number of bursts, and the last three the length of the bursts in slot granularity. The burst pattern 016 02 010 005, for instance, denotes 10 bursts with a period of 16 slots and each burst is 5 slots long and the first burst starts in slot 2.

In order to synchronize the fault injection with the time-triggered communication schedule, the transition from the state Burst Pattern to the state Mode Selection must be completed in exactly x slots, where x depends on the burst pattern *BP*.

$$x = \text{offset} + (\text{number} - 1) * \text{period} + \text{length}$$

The selected fault injection device is triggered at the beginning of the relevant slots defined by the burst pattern *BP*.

The embedded triggering application for controlling the disturbance node is modeled in the same way.

6.2.2 Task Structure

Within each TTP node, we employ a set of time-triggered tasks as shown in Figure 9 for controlling fault-injection, evaluating the frame status information of the C2 communication controller, and transferring diagnostic information via a serial connection to the PC hosting the data base application:

- **Detection Task.** The detection task is executed during each slot in the fault-injection and analysis node, as well as in regular nodes. The purpose of this task is the evaluation of the frame status information provided by the C2 communication controller. The detection task is triggered at the beginning of each communication slot in order to process the status of the frame received during the previous slot. The execution of the detection task must be finished before the next frame arrives at the beginning of the subsequent slot.

- **Communication Task.** The communication task is also executed in the fault-injection and analysis node, as well as in the regular nodes. This task outputs via a serial connection an error that has been detected by the detection tasks (i.e., debug message), thus enabling the filling of the database in the PC.

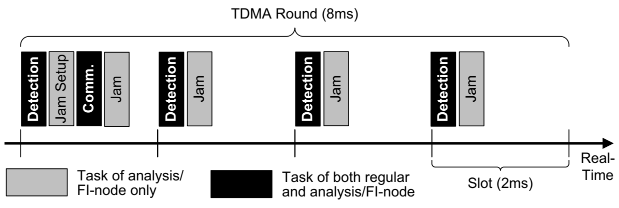


Figure 9. Task Structure

- **Jam Setup.** The task Jam Setup converts the parameters (i.e., period, offset, number of bursts, and length) into a generic form, so that the task Jam no longer has to deal with specific parameters like period or length, but is given the more generic values `sigUptime` and `sigDowntime` that represent the number of slots that the signal must be held HIGH or LOW before being inverted. Contrary to the task Jam which is invoked in every slot the task Jam Setup is only invoked in Slot 0, which allows to easily align the disturbance with the start of the TDMA round. The task Jam Setup receives its input via the RS232 connection from the PC.

- **Jam.** The task jam is activated every slot and triggers the selected fault injection device according to the parameters `sigUptime` and `sigDowntime` provided by the task Jam Setup.

All tasks are scheduled by the time-triggered operating system TTPos [28] at a priori specified points in time. The durations of TDMA rounds and communication slot determine the deadlines for these tasks.

7 Fault Injection Campaigns

The presented fault injection campaigns aim at investigating the behavior of the TTP cluster under faults generated by the disturbance node and the EMI disturbance generator. During the experiments both connector faults and component external/internal faults are injected in order to judge to which extent the frame status field of the TTP C2 controller can be used to discriminate between the introduced fault classes.

7.1 Hypotheses

The ability to identify error detection mechanisms that allow discrimination between component and bus/connector failures is a prerequisite for applying the maintenance-oriented fault model introduced in Section 4. Therefore, the following four hypotheses are investigated by the conducted fault injection campaigns:

Hypothesis 1 *The Invalid Frame status field provided by the TTP controller of the deployed TTP PN212 is a suitable indicator for a connector fault affecting a node computer.*

Hypothesis 2 *The Null Frame status field provided by the TTP controller of the deployed TTP PN212 is a suitable indicator for a connector fault affecting a node computer.*

Hypothesis 3 *The Incorrect Frame status field provided by the TTP controller of the deployed TTP PN212 is a suitable indicator for a connector fault affecting a node computer.*

Hypothesis 4 *The Tentative Frame status field provided by the TTP controller of the deployed TTP PN212 is a suitable indicator for a connector fault affecting a node computer.*

7.2 Description of Fault Injection Campaigns and Results

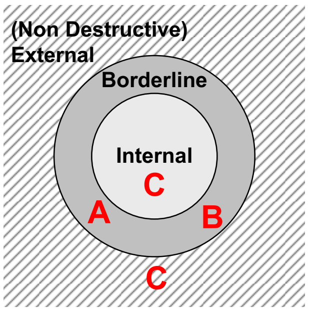


Figure 10. Relationship between the Fault Injection Campaigns and the Maintenance-oriented Fault Model

Three fault injection campaigns have been conducted. As depicted in Figure 10 campaigns A and B have aimed at the borderline (connectors) of the component (via the bus), while campaign C has subjected the component directly to EMI.

The direct exposure of a component to EMI is used to simulate internal/external faults. In contrast to external component faults that occur randomly, internal faults are considered to exhibit a relatively high occurrence rate after their first appearance [5, 6]. Thus, the discrimination is mainly based on the rate and location of occurrence [7]. In our experiments, internal and external faults are subsumed in the fault injection campaign C, since the focus of our investigation are connector faults.

In the campaigns A and B the EMI generator and the Disturbance Node are used to inject faults on the bus-level in order to simulate connector faults. The faults injected during these two campaigns have resulted predominantly in Invalid Frames. Besides, in campaign A the EMI propagation from the bus to the internals of the component has lead to 35 component crashes during 13,000 experiments, which were perceived by other nodes via Null Frames. Furthermore, in campaign B in two of eight types of fault injection experiments the idle state was reproduced, which again other nodes perceived via Null Frames.

In campaign C the component has been directly exposed to EMI. In 64% of all experiments Null Frames have been detected, while only in 0.3% of all experiments Incorrect or Invalid Frames have been observed (cf. Figure 15).

7.2.1 Fault Injection Campaign A

Type of Burst	Probe	Position	Voltage	# of Experiments
Burst 0.5L	L (15 ms)	Channel 1	500	13000

Node #	Frame in Slot of Node #	Channel	Invalid Frames	Null Frames	Incorrect Frames	Tentative Frames	Other Error Fr.
0	1	0	0	5	0	0	0
0	1	1	93	5	0	0	0
0	2	0	0	6	0	0	0
0	2	1	48	6	0	0	0
0	3	0	0	24	0	0	0
0	3	1	1908	24	0	0	0
1	0	0	0	0	0	0	0
1	0	1	213	0	0	0	0
1	2	0	0	2	0	0	0
1	2	1	58	2	0	0	0
1	3	0	0	24	0	0	0
1	3	1	1936	24	0	0	0
2	0	0	0	0	0	0	0
2	0	1	689	0	0	0	0
2	1	0	0	1	0	0	0
2	1	1	417	1	0	0	0
2	3	0	0	24	0	0	0
2	3	1	1997	24	0	0	0
3	0	0	0	0	0	0	0
3	0	1	1975	0	0	0	0
3	1	0	0	5	0	0	0
3	1	1	1961	5	0	0	0
3	2	0	0	6	0	0	0
3	2	1	1889	6	0	0	0

Figure 11. Results of EMI Experiments (Target: Channel 1, Mode: Burst (15 ms))

Figure 11 shows the results of the EMI experiments targeting channel 1 of the bus of the TTP cluster. For these 13,000 experiments the voltage has been set to 500 Volts and a large (L) probe (length: 16 cm, width: 10 cm, height: 0.2 cm) has been used to disturb channel 1 (bus line TTH) with 15 ms bursts. The probe has been placed on channel 1 of the bus between node 2 and node 3. In the first column the node of the cluster that is observing either an Invalid Frame or Null Frame on the respective channel 0 or 1 is listed. For each row the “Frame in Slot of Node #” entry contains the slot number of the node in which a frame type other than a correct one has been detected. The data shows that perturbing channel 1 of the bus using the EMI probe causes a significant number of Invalid Frames on channel 1. While Invalid Frames have only been detected on the channel subjected to disturbances, Null Frames have been simultaneously detected on both channels. This has always been reported by a majority of the nodes in the cluster. The received Null Frames have been due to crash failures of the respective nodes. After restart and reintegration the nodes have operated correctly again, i.e., regain of membership. The results in Figure 11 show that node 3 has crashed 24 times, node 2 six times, and node 1 five times. Notice that node 1 and node 2 observe lower restart numbers of each other due to 4 simultaneous crash failures.

7.2.2 Fault Injection Campaign B

Type	Busline TTL	Busline TTH	# of Exp.	Duration	Detected Frames
DN1	GND	GND	1000	40 ms	Invalid
DN2	VCC	VCC	1000	40 ms	Invalid
DN3	GND	VCC	1000	40 ms	Null, Invalid
DN4	VCC	GND	1000	40 ms	Invalid
DN5	GND	—	1000	40 ms	Invalid
DN6	VCC	—	1000	40 ms	Invalid
DN7	—	GND	1000	40 ms	Invalid
DN8	—	VCC	1000	40 ms	Null, Invalid

Figure 12. The Fault Injection Experiments using the Disturbance Node

During the fault injection campaign B using the disturbance node as shown in Figure 5 a total of 8000 experiments has been investigated to analyze the effects of short circuits on the bus of the cluster. Figure 12 lists the eight different types of experiments that have been executed. Each of the eight configurations of short circuits to GND and VCC lasted for 40 ms starting at the beginning of the slot of node 0 on channel 0. For each type a total of 1000 experiments has been injected and analyzed. During the runs DN1, DN2, DN4, DN5, DN6, and DN7 the nodes monitor Invalid Frames for the duration of 4 TDMA rounds (a TDMA round has been configured to 10 ms).

DN1						
Node #	Frame in Slot of Node #	Invalid Frames	Null Frames	Incorrect Frames	Tentative Frames	Other Error Fr.
0	1	4000	0	0	0	0
0	2	4000	0	0	0	0
0	3	4000	0	0	0	0
1	0	5000	0	0	0	0
1	2	4000	0	0	0	0
1	3	4000	0	0	0	0
2	0	5000	0	0	0	0
2	1	4000	0	0	0	0
2	3	4000	0	0	0	0
3	0	5000	0	0	0	0
3	1	4000	0	0	0	0
3	2	4000	0	0	0	0

Figure 13. Results for Campaign DN1

Figure 13 presents the results of campaign DN1. The results are the same compared to the results of the run DN2, DN4, DN5, DN6, and DN7. During fault injection only Invalid Frames have been detected.

The results of the fault injection runs DN3 and DN8 deviated from the results of the other fault injection campaigns: the injected short-circuits have shown the same characteristics on the bus like the idle state of the bus driver and thus the received frames were classified as Null Frames. The start and the end of the disturbance coincided with a frame transmission, so that the frames at the start and the end of the disturbance have been truncated, which explains the Invalid Frames.

DN3						
Node #	Frame in Slot of Node #	Invalid Frames	Null Frames	Incorrect Frames	Tentative Frames	Other Error Fr.
0	1	0	4000	0	0	0
0	2	0	4000	0	0	0
0	3	0	4000	0	0	0
1	0	2000	3000	0	0	0
1	2	0	4000	0	0	0
1	3	0	4000	0	0	0
2	0	2000	3000	0	0	0
2	1	0	4000	0	0	0
2	3	0	4000	0	0	0
3	0	2000	3000	0	0	0
3	1	0	4000	0	0	0
3	2	0	4000	0	0	0

Figure 14. Results for Run DN3

In Figure 14 the results for the fault injection run DN3 are presented (the experiments for run DN8 have shown the same results). In all 1000 experiments all receiving nodes of the cluster have detected in the slot of node 0 (in which the fault injection starts) an Invalid Frame. Then, for 3 TDMA rounds Null Frames have been consistently monitored and again an Invalid Frame has been observed at the end of the experiment.

7.2.3 Fault Injection Campaign C

Type of Burst	Probe	Position	Voltage	# of Experiments
Burst 4L	L (15 ms)	Node	4000	10000

Node #	Channel	Invalid Frames	Null Frames	Incorrect Frames	Tentative Frames	Other Error Fr.
0	0	9	6444	12	0	0
0	1	29	6444	12	0	0
1	0	12	6442	12	0	0
1	1	9	6442	12	0	0
2	0	19	6357	13	0	0
2	1	26	6382	13	0	0

Figure 15. Results of 10,000 EMI Experiments (Target: Node, Mode: Burst (15 ms))

Figure 15 lists the results of 10,000 fault injection experiments (Burst4L) targeting the complete node using the large probe for injecting 15 ms EMI bursts in the slot of the node under investigation (i.e., node 3). The vast majority of the resulting failures are crash failures of the node. However, during a few test runs also channel failures (resulting in Invalid and Incorrect Frames) have been detected. As shown, node 0 has monitored 6444 Null Frames on both channels, 9 Invalid frames (i.e., wrong encoding) on channel 0, and 29 Invalid Frames on channel 1, and 12 Incorrect Frames (i.e., incorrect CRC) on both channels. Node 1 has monitored 6442 Null Frames on both channels, 12 Invalid Frames on channel 0 and 9 Invalid Frames on channel 1, and 12 Incorrect Frames on both channels. Finally, node 2 has monitored 6357 Null Frames on channel 0 and 6238 Null Frames on channel 1, 19 Invalid Frames on channel 0 and 26 Invalid Frames on channel 1, and 13 Incorrect Frames on both channels. Since node 2 has been in close physical proximity to node 3 in the bus configuration of the fault injection experiments, it can be concluded that the effects of the EMI disturbances have propagated from node 3 to node 2.

7.3 Interpretation and Discussion

In the following we interpret and discuss the previously described results of the fault injection experiments.

Hypothesis 1 has not been falsified. During fault injection campaign A a total of 13,184 Invalid frames compared to a total of 194 Null Frames have been detected (cf. Figure 11). The Null Frames have always been observed simultaneously on both channels by all other alive nodes. Investigation of the result has shown that these 194 observed Null Frames result from 35 node crashes during the 13,000 experiments due to the propagation of EMI to the internals of the nodes. Therefore, all EMI that affected channel 1 exclusively only resulted in Invalid Frames.

In 6,000 experiments of fault injection campaign B only Invalid Frames have been observed (cf. Figure 12). A closer investigation of the physical layer of the cluster (MFM encoding) has revealed that in the remaining 2,000 experiments the short-circuits resembled the idle state of the bus. The bus driver considers the bus idle and reports a Null Frame when the bus line TTH has a higher voltage level than bus line TTL during the whole frame transmission window. Therefore all injected short-circuits, except from those that reproduced the idle state, resulted in Invalid Frames.

On the contrary, during campaign C which has exposed the component directly to EMI in order to simulate internal/external faults only in 0.3% of all 10,000 experiments Incorrect or Invalid Frames have been observed (cf. Figure 15).

Considering the connector and internal/external faults we have covered in our fault injection campaigns the frame status Invalid Frame is a suitable indicator for a connector fault and therefore the general Hypothesis 1 is not falsified.

Hypothesis 2 has not been falsified. In campaign A 194 Null Frames have been observed. As discussed previously an investigation of the results has shown that these Null Frames can be adjudged to 35 node crashes during the 13,000 experiments because of EMI propagation to the internals of the component.

During campaign B Null Frames have been observed when the injected short-circuits reproduced the bus idle state of the bus during the whole frame transmission window.

By contrast, during campaign C predominantly Null Frames have occurred. In 64% of all 10,000 experiments Null Frames have been detected (cf. Figure 15).

Considering the connector faults we have covered in our fault injection campaigns the frame status Null Frame is not considered to be a suitable indicator. However, since we do not have access to field data that covers the percentage of short-circuits that reproduce the idle state and thus are recognized as Null Frames, we cannot conclude that Hypothesis 2 is falsified.

Hypothesis 3 has been falsified. No Incorrect Frames have been monitored during fault injection campaigns A and B. Consequently, Hypothesis 3 has been falsified.

Hypothesis 4 has been falsified. No Tentative Frames have been observed during all fault injection campaigns (A, B, and C). Consequently, this type of frame cannot be used to distinguish between the fault classes of the introduced fault model.

7.4 Topology Considerations

In our fault injection setup using a bus topology the exact location of a short circuit cannot be traced, since the complete bus is disturbed for the duration of the fault injection.

Note, that these results are a consequence of using a TTP cluster with a bus topology. In case of full duplex point-to-point connections, as used in a star topology with a central bus guardian, a significantly better accuracy with respect to the error location is expected [3]. However, additional fault injection campaigns are necessary to validate this claim.

8 Conclusion

In the context of distributed embedded systems connector faults are responsible for a significant number of system failures. In particular, the environmental stress in the automotive domain results in more than 30% of failures attributed to connectors. Connector faults are especially tricky to identify, since the maintenance action may itself be the corrective action. With the shift from event-triggered to time-triggered communication systems, future automotive systems offer the chance to significantly improve the detection coverage of connector faults. A high detection coverage is a key element for increasing the accuracy of online analysis strategies and handling connector faults in maintenance.

In this paper we have investigated to what extent the mechanisms of the TTP C2 communication controller can be used to reveal connector faults. Fault injection experiments have shown that the frame status information is a suitable indicator, since the frame status has been different for faults induced by EMI simulating component external/internal faults and connector faults induced by the disturbance node targeting the system bus.

Acknowledgments

This work has been supported in part by the Austrian Advanced Automotive Technology Project under project No. 809437 and the European IST project ARTIST2 under project No. IST-004527 and the European IST project DECOS under project No. IST-511764.

References

[1] Analysis of the european automotive in-vehicle network architecture markets. Technical report, Frost & Sullivan, Oct. 2004.

[2] A. Avizienis, J. Laprie, and B. Randell. Fundamental concepts of dependability. Research Report 01-145, LAAS-CNRS, Toulouse, France, Apr. 2001.

[3] G. Bauer, H. Kopetz, and W. Steiner. The central guardian approach to enforce fault isolation in a time-triggered system. In *Proceedings of the 6th International Symposium on Autonomous Decentralized Systems (ISADS 2003)*, pages 37–44, Pisa, Italy, Apr. 2003.

[4] G. Bauer and M. Paulitsch. An investigation of membership and clique avoidance in TTP/C. In *Proceedings of 19th IEEE Symposium on Reliable Distributed Systems*, pages 118–124, Nürnberg, Germany, Oct. 2000.

[5] A. Bondavalli, S. Chiaradonna, F. D. Giandomenico, and F. Grandoni. Discriminating fault rate and persistency to improve fault treatment. In *Proceedings of The Twenty-Seventh Annual International Symposium on Fault-Tolerant Computing (FTCS'97)*, pages 354–362, Washington - Brussels - Tokyo, June 1997. IEEE.

[6] A. Bondavalli, S. Chiaradonna, F. D. Giandomenico, and F. Grandoni. Threshold-based mechanisms to discriminate transient from intermittent faults. *IEEE Transactions on Computers*, 49(3):230–245, Mar. 2000.

[7] C. Constantinescu. Impact of deep submicron technology on dependability of VLSI circuits. In *Proceedings of the International Conference on Dependable Systems and Networks*, pages 205–209. IEEE, 2002.

[8] U. A. Force. Aircraft mishap investigation handbook for electronic hardware. Technical Report WL-TR-95-4004, Materials Directorate, Wright Laboratory, Air Force Material Command, Jan. 1995.

[9] D. Galler and G. Slenski. Causes of aircraft electrical failures. *Aerospace and Electronic Systems Magazine*, 6(8):3–8, Aug. 1991.

[10] M. Hsueh, T. Tsai, and R. Iyer. Fault injection techniques and tools. *IEEE Computer*, 30(4):75–82, Apr. 1997.

[11] H. Kim, A. White, and K. Shin. Effects of electromagnetic interference on controller-computer upsets and system stability. *IEEE Transactions on Control Systems Technology*, 8(2):351–357, Mar. 2000.

[12] K. Kimseng, M. Hoit, N. Tiwari, and M. Pecht. Physics-of-failure assessment of a cruise control module. *Microelectronics Reliability*, 39:1423–1444, 1999.

[13] H. Kopetz. *Specification of the TTP/C Protocol*. TTTech, Vienna, July 1999. Available at http://www.ttpforum.org.

[14] H. Kopetz and G. Bauer. The time-triggered architecture. *IEEE Special Issue on Modeling and Design of Embedded Software*, Jan. 2003.

[15] J. Lundelius and N. Lynch. A new fault-tolerant algorithm for clock synchronization. In *Proceedings of the Third Annual ACM Symposium on Principles of Distributed Computing*, pages 75–88. ACM Press, 1984.

[16] J. McBride. Electrical contact and connectors in automotive systems. In *Proceedings of the IEE Colloquium on Connectors on Vehicles*, pages 3/1–3/7, Nov. 1993.

[17] M. Pecht and V. Ramappan. Are components still the major problem: a review of electronic system and device field failure returns. *IEEE Transactions on Components, Hybrids, and Manufacturing Technology*, 15(6):1160–1164, Dec. 1992.

[18] P. Peti, R. Obermaisser, and H. Kopetz. A maintenance-oriented fault model for the DECOS integrated diagnostic architecture. In *Proceedings of the Workshop on Parallel and Distributed Real-Time Systems 2005 (WPDRTS)*. IEEE, Apr. 2005.

[19] P. Peti, R. Obermaisser, F. Tagliabo, A. Marino, and S. Cerchio. An integrated architecture for future car generations. In *Proceedings of the 8th IEEE International Symposium on Object-oriented Real-time distributed Computing*, May 2005.

[20] H. Pfeifer, D. Schwier, and F.W. von Henke. Formal verification for time-triggered clock synchronization. In *Proceedings of 7th IFIP Working Conference on Dependable Computing for Critical Applications*, pages 207–226, San Jose, CA,USA, Nov. 1999. Fakultat fur Inf., Ulm Univ.

[21] S. Sullivan and G. Slenski. Managing electrical connection systems and wire integrity on legacy aerospace vehicles. In *Proceedings of the FAA Principal Inspectors and Engineers Workshop*, Seattle, USA, 2001.

[22] J. Swingler, J. McBride, and C. Maul. Degradation of road tested automotive connectors. *IEEE Transactions on Components and Packaging Technologies*, 23(1):157–164, Mar. 2000.

[23] P. Tanner. Automotive connectors. In *Proceedings of the IEE Colloquium on Connectors on Vehicles*, pages 6/1–6/10, Nov. 1993.

[24] TTTech Computertechnik AG, Vienna, Austria. *TTP/C Controller C2 Controller-Host Interface Description Document, Protocol Version 2.1*, Nov. 2002.

[25] TTTech Computertechnik AG, Vienna, Austria. *TTP Development Cluster - The Complete TTP System*, 2004.

[26] TTTech Computertechnik AG, Vienna, Austria. *TTP Powernode – The TTP Development Board*, 2004.

[27] TTTech Computertechnik AG, Vienna, Austria. *TTP Disturbance Node: Testing Fault Tolerance Properties of Safety-Critical Distributed Real-Time Systems*, 2005.

[28] TTTech Computertechnik AG, Vienna, Austria. *TTPOS: The OSEKtime-Based Operating System for Safety-Critical Real-Time Applications*, 2005.

[29] P. van Dijk and F. van Meijl. Contact problems due to fretting and their solutions. *AMP Journal of Technology*, 5:14–18, June 1996.