Design and Evaluation of a Safe Driver Machine Interface

ANDREA BONDAVALLI[5], ANDREA CECCARELLI[5], JESPER GRØNBÆK[2], DANILO IOVINO[1], LUCIE KÁRNÁ[3], ŠTĚPAN KLAPKA[3], TATIANA K. MADSEN[2], MELINDA MAGYAR[4], ISTVÁN MAJZIK[4], ANNA SALZO[1]

[1]*Ansaldo Segnalamento Ferroviario, Torino, Italy*
[2]*Aalborg University, Aalborg, Denmark*
[3]*AŽD Praha s.r.o., Prague, Czech Republic*
[4]*Budapest University of Technology and Economics, Budapest, Hungary*
[5]*Institute ISTI-CNR / University of Florence, Italy*

(Received on June 10, 2008.)

Abstract: Driver Machine Interface (DMI) is a slave unit of the train onboard computer in the ERTMS automatic train control system. The SAFEDMI project aimed at the development of a DMI which fulfills the requirements of Safety Integrity Level 2 according to the CENELEC development standards. The main challenges were (i) the reduction of the hardware complexity and costs by implementing the safety mechanisms in software and (ii) provide a safe and secure wireless communication interface to support diagnostics and maintenance. This paper presents the requirements, the design of the architecture and the wireless communication protocol, and the evaluation of the applied solutions.

Keywords: Safety-critical systems, safe wireless communication, quantitative evaluation

1. Introduction

Railway Automatic Train Control (ATC) systems are based both on trackside and onboard systems and their evolution has followed the introduction of new technologies. The increasing level of train traffic and the spread of high-speed rail lines are demanding an increasing safety level in the ATC systems. In order to ensure compatibility and interoperability between the ATC systems produced in the various European countries, the European Rail Traffic Management System (ERTMS/ETCS) programme has been set up to provide unique functional and non-functional standard requirements [1].

The ERTMS train-borne equipment includes, among others, the European Vital Computer (EVC) and the Driver Machine Interface (DMI). The EVC is the core of the onboard ATC system; it supervises the movement of the train by using the information received from the trackside systems. The DMI is the interface between the EVC and the driver; it acquires driver's commands and transforms EVC commands in graphical and audible information according to [3]. The standard ERTMS DMI does not have to fulfil safety requirements although it operates in a critical context. These days many railway operators are beginning to require higher safety levels and reduced costs. Moreover, the use of emerging wireless communication technologies is demanded to speed up the DMI configuration and software/firmware upgrade, thus avoiding mechanical operations.

The SAFEDMI project[1] aimed at developing a DMI that fulfils the requirements of Safety Integrity Level 2 (SIL 2) according to the CENELEC development standards

[1] IST FP6 STREP-031413 Safe Driver Machine Interface (DMI) for ERTMS Automatic Train Control. http://www.safedmi.org

[4][5][6]. The main challenges of this project were (i) the reduction of the hardware complexity and costs by implementing the safety mechanisms in software and (ii) provide a safe and secure wireless communication interface to support diagnostics and maintenance. To accomplish this task, the SAFEDMI project defined and organized the DMI requirements into four main groups: (i) non functional requirements related to the hardware characteristics; (ii) functional requirements related to ERTMS functionalities and DMI operational modes; (iii) safety requirements mainly focused on LCD display, keyboard, on-line device testing; (iv) wireless requirements related to malicious attacks, integrity checking, authentication, and performance.

The proposed solution is compliant with CENELEC standards. More specifically, it ensures that no single random hardware component failure mode is hazardous by using (i) composite, (ii) reactive or (iii) inherent fail-safety techniques as suggested in EN 50129 [4]. Furthermore, as suggested in EN 50128 [5], defensive programming, fault detection mechanisms, error detecting codes have been used in the design phase to fulfil SIL 2 requirements. Finally the DMI satisfies the requirements specified in EN 50155 [7] and in EN 50121-3-2 [8], respectively, for electronic equipments used on rolling stock for control, regulation, protection supply, and for electro-magnetic compatibility.

The paper is organized as follows. Section 2 reports a description of the design task in terms of fail-safe fault tolerant architecture and protocols for wireless maintenance. The feasibility of the technical solutions adopted have been proved with the realization of a prototype. In Section 3 the evaluation task is described to demonstrate that the envisaged solution allows reaching SIL 2. Finally the conclusions are reported in Section 4.

2. Design of the DMI

The internal architecture of the DMI was designed to address several demands. To reduce the costs, safety was ensured mainly by software mechanisms, and the wireless communication architecture was designed to include off-the-shelf solutions. The following subsections present an overview of the design decisions.

2.1 DMI operational modes

The DMI has five operational modes: *Start-up*, *Configuration*, *Normal*, *Suspect* and *Safe* mode (state). Fig. 1 presents these operational modes and how they are interrelated.

In *Start-up mode* the initialization procedures and the thorough testing of all devices are performed. Diagnostic functionalities are also available. In *Configuration mode* safe software upload and configuration are performed by means of wireless communication. After a successful configuration session, a restart of the DMI is needed. In *Normal mode* the DMI produces graphical and audio information to support train driving, as well as it acquires and processes driver's commands. Moreover, periodic testing activities are performed and diagnostic functionalities are available.

In each of the previous modes, whenever an error is detected, a transition to the *Suspect mode* is forced. Suspect mode allows going back to Start-up mode, to perform attempts to restart. Using a specific mechanism [19], it is possible to set a maximum number of allowed consecutive restart attempts. Moreover, a maximum time to restart the DMI is set. This mechanism allows increasing the availability of train missions, since DMI can automatically restart from transient faults and become fully working again, thus avoiding that the EVC activates the emergency brakes and interrupts the train mission.

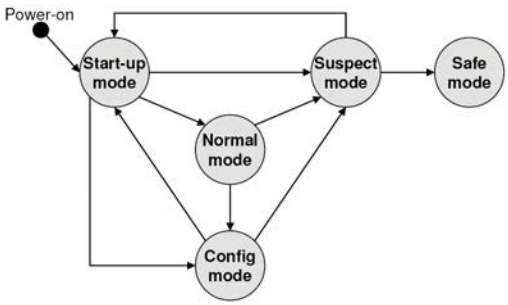

Fig. 1: SAFEDMI operational modes

Safe mode is entered when a permanent malfunctioning is detected and attempts to restart the DMI have failed. This mode prevents further operations of the DMI.

2.2 DMI Architecture and Safety Mechanisms

Regarding the hardware, the DMI has a non-redundant architecture including OTS components. Absence of hardware redundancy imposes a more sophisticated software architecture, since safety requirements need to be accomplished only by software mechanisms.

Regarding the software architecture, for each operational mode we defined five *modules* that are activated/deactivated when the DMI transits from one mode to another. *Global monitoring* performs the role of an execution monitor, a software watchdog, a log manager, and a diagnostic manager; it recognizes the conditions that cause an operational mode change of the DMI and performs the operations needed to reconfigure the system when a mode change happens. *Checks and tests* performs checking and testing activities. *Operations* module contains the software objects involved in the implementation of the functional requirements of the operational modes, as visualization and audio emission procedures. *Communications* module handles the diagnosis, maintenance and EVC related communications, and *I/O manager* contains the drivers of the DMI hardware devices.

Regarding the safety mechanisms applied to guarantee SIL 2, in SAFEDMI the *reactive fail-safety* principle was adopted [4]. As a consequence, efficient fault detection plays a crucial role. According to the safety standards, the potential hazardous effects of *systematic faults* (especially software design faults) was handled by means of fault prevention and removal during development, i.e., the quality and safety management conditions corresponding to SIL 2. Fault detection addressed *random faults* that cover both permanent and transient faults.

In Start-up mode *permanent hardware faults* are addressed by efficient test procedures with high fault coverage, for example march test algorithms [21] for RAM and flash memory. In Normal and Configuration modes on-line (run-time) fault detection is applied. For detecting latent *random permanent faults* in the hardware components periodic testing is performed. In these operational modes the execution time of the tests is an important issue, this way fast (but less efficient) versions of the test procedures applied in the Start-up mode were developed. For example, instead of a *march C- test* covering an extensive fault model, a simpler version is applied [21]. Considering passive objects (files, configuration data) an integrity checking procedure is instead adopted.

Active *permanent faults* and *random transient faults* in the hardware components are detected on the basis of their *effects on software execution*. Considering (i) the recommendations of the safety standards [4] [5], (ii) the SAFEDMI specific requirements and

(iii) the analysis of the peculiarities of the different modules of the DMI, we identified the following three categories of safety mechanisms: *data acceptance/credibility checks*, *control flow monitoring* and *multiple computation and comparison*. The first category is applied mainly in configuration operations (by error detection codes and integrity checking executed to check transmitted files). The second category is applied in the test and checks module, in the global monitoring module and in the operations module (in this last module, control flow monitoring is applied in the audio management function and in the state change procedure). Finally, the third category is applied in the visualization and keyboard processing functions. A more detailed description of the DMI architecture can be found in [20].

2.3 The Wireless Communication Architecture and Protocols

Traditionally, maintenance operations are performed by maintenance personnel physically attaching a maintenance device to the on-board equipments. To improve procedures, a motivation exist to replace cables with a wireless connection enabling remote execution of these operations. In this way a wireless link is introduced between a *Maintenance Center* (MC) and one or more DMIs located in trains in a maintenance area or in open tracks.

To enable a cost-efficient solution off-the-shelf (OTS) wireless equipment has been incorporated. However, typical OTS wireless equipments do not inherently target the dependability, safety and security requirements existing in industrial applications. Thus, the project aimed at applying new functions. The additional functions required to provide a safe, secure and dependable wireless communication solution lead to increased resource consumption on the peer systems. The primary operations being performed in the DMI are safety critical and efforts must be made to minimize impact on such operations when introducing wireless communication capabilities. To manage these issues the basic concept of the proposed communication architecture is the introduction of a *Bridge Device* (BD) installed next to the DMI in the train. The BD separates an *open* wireless communication system [6] from the MC to the BD, and a *closed* wired communication system from the BD to the DMI. Thus, the BD manages the most resource requiring functions in relation to establishing and maintaining wireless communication. This approach allows reducing the processing load on the DMI and enables the use of state-of-the-art methods for cryptographic protection of wireless transmissions. In addition the BD can act as a firewall and traffic policing entity toward the DMI to minimize impact from faulty equipment and malicious attacks. Altogether, the entire communication system is composed of three devices: the DMI itself, the BD and the MC.

In order to fulfill the requirements on wireless communication between the DMI and the MC, a special purpose protocol layer, called *Safety Layer* is developed. The role of the safety layer is to manage interactions between the OTS protocols and services as well as to add additional functions enabling a required *integrity level* of transferred data. This layer is split into two sub-layers: *Low Safety Layer* (LowSL) located at MC and BD and *High Safety Layer* (HighSL) located at MC and DMI. This split is motivated by a classification of communication safety requirements into two levels: (i) communication at the application level; (ii) message transmission over open wireless communication system. The LowSL is responsible for providing a secure connection between BD and MC. The purpose of HighSL is to guarantee a safe, secure and dependable end-to-end connection. The main tasks of HighSL include:

- Performance monitoring: monitoring of wired and wireless link conditions to enable autonomous and user initiated fault recovery actions,

- Session control, including the ability to terminate sessions, e.g., in case of unrecoverable faults identified in performance monitoring,
- Key management: management of hardware based keys used to authenticate secure BD-MC connection,
- Safety aware application interface providing information about unrecoverable faults and monitoring of excessive communication events for additional fault management.

Fig. 2 illustrates the communication protocol stack and the traffic flows.

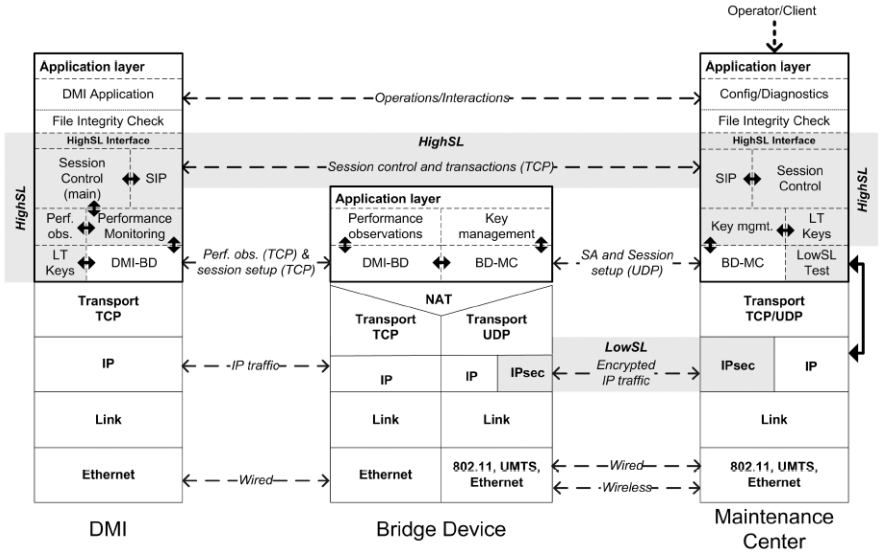

Fig. 2: Protocol components, connections and functional components

Using an open medium the data transferred on the wireless link needs to be protected. For safety reasons it must be ensured that a malicious party cannot access the DMI or alter data being transferred to and from the DMI. This is achieved by strong authentication and integrity checking methods. In addition, encryption must ensure that a third party cannot retrieve the contents of the data being transferred. IPsec [25] is an OTS security framework which has been applied in the LowSL. Its specification provides numerous configuration options making IPsec a versatile solution and allowing fine-tuning the algorithmic choices to particular needs. Table 1 contains the configuration chosen to provide a secure channel between the MC and the BD.

Requirements	**IPsec configuration**
IP	IPv4
Confidentiality	ESP using AES-CBC to provide strong standardized encryption
Protection against data insertion, re-sequencing, corruption, masquerade, deletion	Data integrity and origin checks are provided in ESP with HMAC using the cryptographic hash function SHA-2
Repetition/ replay protection	ESP with packet sequence numbering
Host-to-host security	Transport mode

Table 1: IPsec configuration used for the LowSL safe communication

Considering L2 protocols, different wireless technologies (e.g. UMTS, WiMAX,

WLAN 802.11) can be used depending on the varying requirements of different use case scenarios. This has been accounted for in the design of the BD: it can contain multiple wireless interfaces while the Safety Layer may adapt its service level to the properties of a given wireless link. The redundancy provided by multiple interfaces also includes a wired link as a back-up solution. This may be necessary in cases when the wireless connection cannot be established or cannot provide the required throughput.

The communication stack has been designed to be generic and not only support existing diagnostic and maintenance applications; i.e., the communication stack is not tightly coupled to applications. However, the knowledge of application communication requirements enabled the introduction of different supported traffic classes representing different requirements for throughput and integrity levels.

2.4 The DMI Prototype

The design activities provided inputs to the implementation task aimed at demonstrating the feasibility of the envisaged technical solution by means of a prototype. To allow a parallel implementation, the prototype was organized in three main parts: (i) safety related architecture; (ii) wireless maintenance protocols; (iii) overall final integration.

The first prototype focused on train mission and safety mechanisms. The prototype was realized using OTS hardware components and the VRTX operating system [16]. The DMI was connected to the ASF ERTMS on-board equipment with a protocol compliant to the UNISIG standard [2]. Moreover an ASF *track simulator* was used for generating the events of a train mission. By means of a diagnostic tool (that showed the value of software variables) the operation of the main safety mechanisms were observed, namely the operational mode changes under the control of the execution monitor, the activation of duplicated computation and comparison applied to safe acknowledgment, and the activity of control flow checking in relation to audio output management.

A *DMI Simulator* was also developed to anticipate and simplify final integration activities. It is an application running on a PC and includes the same software modules like the physical prototype. This application was useful to test the suspect state and the related restart policy. In fact, the implementation in the physical prototype was more complex due to the need of reinitializing all hardware and software components without cutting the connection with the EVC.

The second prototype focused on wireless maintenance functionalities and included PCs corresponding to the BD and MC roles. Linux Gentoo [17] was selected as operating system thanks to its good performance in embedded platforms. Furthermore, Linux has open source code for wireless communications and provides better accessibility to hardware related information needed for performance monitoring and error detection.

An ad-hoc mode was selected for the wireless link between BD and MC. WPA-PSK was available as authentication method in ad-hoc mode with a strengthened security employing PBKDF2 key derivation functions. For securing wireless communication between BD and MC, LowSL was based on IPsec protocol thus achieving confidentiality, authenticity and integrity of the transmitted messages. Most of the IPsec software architecture is already included in the Linux kernel. *OpenIKEv2* [18] library was chosen for managing setup of IPsec connections and the specific key exchange procedure.

Finally, from the third *integrated prototype* important information was derived about the required characteristics of maintenance tools to be used with a wireless communication. At least longer delays have to be considered in such a situation, especially in the preliminary phase for the session establishment.

3. Evaluation of the DMI

The evaluation task concentrated on four targets. The DMI was evaluated focusing on the safety and availability of the architecture (Section 3.1), understanding the behavior and efficiency of the error detection techniques (Section 3.2), and analyzing the real-time properties. The communication between the DMI and its environment was evaluated focusing on the performance and dependability properties of the wireless communication (Section 3.3), and the detection properties of the applied safety codes (Section 3.4).

3.1 Model Based Evaluation of the Architecture and the Safety Mechanisms

The quantitative safety, reliability and availability requirements necessitated the evaluation of the DMI architecture and the related safety mechanisms. A model-based approach was adopted by constructing the *state-based stochastic dependability model* of the architecture. This model represents how the activation of a fault in a hardware component may lead to a detected error (that is either successfully handled by the recovery mechanism or leads to the Safe mode), or potentially to an undetected error at system level (that is considered as a hazard). Parameters of the model include the components' fault occurrence rates, the error propagation probabilities, the coverage and latency of the periodic tests and other error detection mechanisms. Stochastic Activity Networks (SAN, [10]) was selected as the formalism for the construction of the dependability model since (besides representing states and state transitions) it allows specifying complex conditions and actions, general firing time distributions, reward functions and easy composition of sub-models.

Since the DMI has a relatively complex architecture including several hardware resources and software tasks, we followed a modular modeling approach: dependability sub-models were assigned to (i) the hardware resources, (ii) the software components, (iii) the error propagation among them, (iv) the error detection mechanisms, and finally (v) the system level error handling (i.e., the suspect mechanism). The hazardous failures were considered as undetected failures at the task level. The dependability model construction approach can be summarized in the following steps:

1. The dependability sub-models enumerated above were constructed manually, integrating expert knowledge with regard of failure modes, ways of error propagation, error detection and error handling mechanisms, following the principles described in [11]. The corresponding SAN subnet templates were collected in a subnet library. In Fig. 3 we present the subnet assigned to a hardware component. Transition *fault-Occurrence* models the occurrence of faults, its parameter is the fault rate. These faults may be relevant to different tasks ($Task_i\_Errors$) that use this resource, the propagation probabilities are characterized by the ratio of the usage of this resource. The occurrences of permanent faults are registered separately to model how periodic tests (with given rate and coverage parameters) can detect these faults. A detected error triggers a change in the DMI operational mode (*HWTrigger*). Periodic tests and startup tests can be activated only in the corresponding operational mode.
2. To support the composition of the subnets, the UML model of the DMI architecture (that included the hardware components, the software components and the service usage relations among them) was extended: stereotypes and tagged values were used for specifying the components' types and the related dependability parameters.
3. The dependability model was constructed by our automated tool [12] which identified the different components, relations and parameters on the basis of the UML model then instantiated and connected the corresponding SAN subnets from the library.

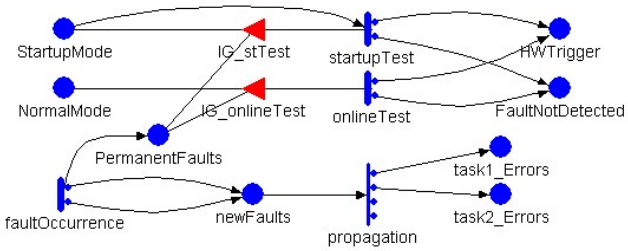

Fig. 3: Subnet belonging to a hardware component

The probabilistic calculations, i.e., solution of the dependability model by computing the *mean time to failure* (reaching Safe mode) and *mean time to hazard* were performed by the Möbius tool [9] applying simulation. Sensitivity analyses were performed by specifying the ranges of the different design parameters and examining their effects on the system level measures. As an example, Fig. 4 presents how the hazardous failure rate depends on the error detection coverage of the applied control flow checking technique.

The main results of the evaluation can be summarized as follows: The DMI prototype fulfils the reliability and safety requirements. The error detection coverage of the on-line error detection mechanisms has a crucial effect on the hazardous failure rate: if the duplicated execution and comparison is replaced by a less efficient technique, or the error detection coverage of control flow checking decreases below 50% (see Fig. 4) then the SIL 2 safety requirement (i.e., that the hazard rate remains less than 10^{-6} per hour) cannot be satisfied. The rate and coverage of periodic testing activities have less significant effects, the reasonable values allow to fulfill the SIL 2 requirement.

A separate model was constructed to investigate the effects of DMI failures on train missions and QoS parameters observable by the passengers. On the basis of analyzing the influential effects, additional sub-models were constructed that represent the interactions with the on-board computer, the potential switchover to on-board spare DMI (that can increase availability), and the train missions. Two types of communication protocols were analyzed that could be applied in the interactions with the on-board computer: in a *cyclic exchange of messages* the EVC periodically checks the operational status of the DMI, while in the *acyclic case* the failure of the DMI is detected only if an aperiodic request is sent to the DMI and no proper answer is received. The probability to successfully complete a mission without unscheduled train stops, and the steady-state availability of the overall system were computed. The evaluation pointed out the improvements in adopting an acyclic protocol (in case of a DMI failure this protocol increases the chance to boot a spare DMI and switch to it between two EVC requests). Sensitivity analysis quantified the effects of applying different aperiodic timeouts and faster boot procedure.

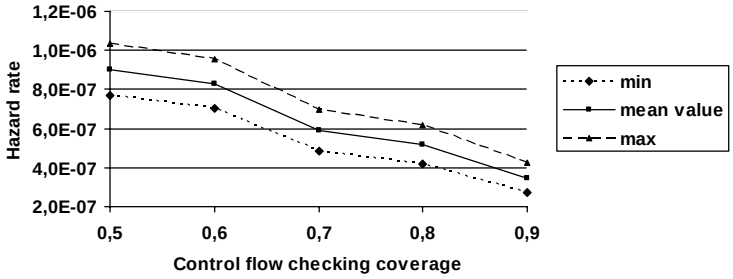

Fig. 4: Hazardous failure rate (per hour) vs control flow checking coverage

3.2 Experimental Evaluation by Fault Injection

Experimental evaluation was performed by fault injection [22] campaigns to understand the efficiency of the safety mechanisms adopted in the DMI prototype. Due to risk of damaging the DMI hardware, only software fault injection was applied (i.e., effects or hardware faults were simulated by injecting changes in software). The *fault injection framework* was composed basically of (i) a low-priority thread (composed of a few instructions to guarantee negligible perturbation) that performs the run-time injection of faults or fault sequences, (ii) a fault library that enlists the faults to inject and (iii) a monitor (based on the existing logging facilities of the DMI) that collects events and allows off-line analysis of the results. The workload was generated by means of an *EVC Packet Generator* that simulates the EVC by sending a pre-determined set of packets to the DMI.

The faults can be subdivided in two categories: (i) faults implemented adding extra code in existing DMI functions (these faults are activated in run time by setting a specific parameter to true), they allow testing specific safety mechanisms of the DMI, and (ii) faults injected by additional software functions (e.g., injecting erroneous bytes in random areas of the RAM).

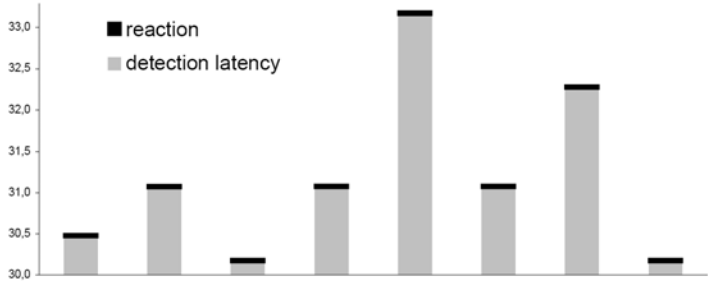

Fig. 5: Results of eight fault injection tests (seconds)

Both qualitative assessment of the correct behavior of the safety mechanism under test (*yes/no verdict* of proper detection and *coverage*), and quantitative evaluation of the error detection techniques were performed. The main measurands identified for quantitative evaluation are two time intervals: (i) *detection latency* (time elapsed from the injection of an error and its detection) and (ii) *reaction* (after detection, time needed to transit to Suspect mode). A Type-B uncertainty according to [24] for the time intervals was computed to be ±4 milliseconds. As example, in Fig. 5 the "reaction" and "detection latency" for eight executions of a fault injection causing erroneous control flow in the audio management function are shown. The wrong signature is injected at second 30. While reaction is a stable value, detection latency varies, because of fluctuations in scheduling activities and in the timing of the EVC Packet Generator. Full details of the fault injection tool and experiments are reported in [23].

3.3 Evaluation of the Wireless Communication Properties

Evaluation of the designed wireless communication solution focuses on the study of system behavior in terms of commonly used parameters, such as throughput, delay, packet error and packet drops at different layers of the protocol stack. The main emphasis is on the throughput estimation, as this estimation can be directly linked with the requirements on communication protocols. It is assumed that a maximum of 20 Mb is to be uploaded

distributed on 10 files of approx. 2 Mb each. Considering the performance requirement of maximum download time of 5 min, the corresponding mean throughput is at least 67 Kb/s.

The wireless communication protocols were evaluated using analytical, simulation and experimental approaches. Analytical and simulation approaches have a controllable environment; however, their accuracy depends on how well the underlying models reflect the reality. A simulation model was constructed that corresponds to the SAFEDMI wireless communication solution and its operating environment. It has been implemented in NS-2 [13] with relevant channel model extensions to enable detailed wireless link properties at layers 1 and 2 including bit error rate (BER), interference, Rician fading, and accurate MAC functions. Furthermore, to support the analytical and simulation results real experiments were conduced to determine achievable performance of a 802.11 based wireless link in the SAFEDMI setup. The testbed was similar to the one described in Section 2.3, its basic functions and components are as follows:

- PCs represented the DMI, BD and the MC. BD and MC were connected using 802.11 a/b/g enabled network interface cards setup in ad-hoc mode. To enable the configuration case an FTP server (client) were installed in the MC (DMI, respectively).
- The wireless connection was pre-configured with IPsec security associations. Network address translation was defined for IPsec secured ports in the BD for forwarding traffic between the DMI and MC.

The analytical modeling of the communication system behavior is a challenging task as it is a system exhibiting complex behavior. We adopted a modular approach: the overall model is split into several sub-models that are tied together. Defining multiple sub-models enables their individual development primarily with the purpose of complexity reduction. The output of one sub-model is used as an input for the subsequent sub-models. The used feed-forward approach is illustrated in Fig. 6. The definition of sub-models can partially be based on layers of the protocol stack. This allows a starting point in already studied models such as channel models. The complete description of the sub-models used and the complete set of results can be found in [23].

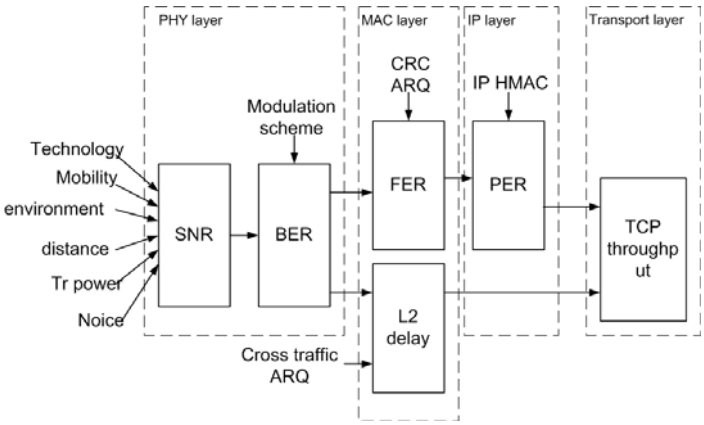

Fig. 6: The feed-forward modeling chain

Measurement, simulation and analytical results corresponding to the case of good channel conditions (wireless transmitter and receiver are located close; signal strength about -50dBm) are presented in Table 2. Looking at goodput and Round Trip Time (RTT), there is a good resemblance between the results. As we observed in our studies, the most significant influence to RTT is caused by the waiting time in the queue. The

good resemblance in the results indicates that queue size in simulation and analytical model is reasonably defined with respect of the real setup. Another reason is that TCP maximum window size in the simulation and the analytical model, as presumably in the real setup, is limited to 64Kb of unacknowledged segments. This clearly also limits how many packets/frames are put in the queue effectively.

	Transmission rate	Goodput, KB/s	Round Trip Time, s	Frame Error Rate	Packet Error Rate
M	11	630	0.072	0.011	0.003
	5.5	397	0.100	0.018	0.004
	1	89	0.501	0.021	0.011
S	11	636	0.105	0.059	0
	5.5	418	0.160	0.057	0
	1	102	0.637	0.055	0
A	11	607	0.107	0.018	0
	5.5	410	0.159	0.017	0
	1	104	0.624	0.016	0

Table 2: Evaluation results (M measurement; S simulation; A analytical)

From our studies considering different environmental and interference parameters we could conclude that the performance requirements are satisfied under a large range of conditions. From measurements and modeling we have observed that packet losses are rare. Frame losses on link layer increase delay due to the retransmission mechanism, but do not lead to significant packet losses. As we are dealing with a non delay-sensitive application, increase in delay has almost no impact on TCP performance.

3.4 Analysis of the Detection Properties of the Safety Codes

This subtask focused on the analysis of the detection properties of the applied detection codes. The detection codes are used in the DMI in communication functions and for the purpose of performing integrity checks of data files or internal data structures (e.g., in ROM). This is essential for ensuring the error detection coverage needed to satisfy the quantitative safety target. A malfunction of a detection code is assessed by computing the probability of an undetected error. In order to quantify this parameter for the linear detection codes, the model of a Binary Symmetrical Channel (BSC) was used.

The investigated codes can be split into several groups, depending on the purpose of their use. The most important group comprises the detection codes of the safety protocol layers (safety codes). This group contains the safety codes used in the on-board communication (CRC_SL2 and CRC_SL4) and the detection codes used for file integrity checks.

The majority of the investigated codes are shortened cyclic codes. For this type of codes, the inherent computational complexity of the BSC calculation can be reduced by several techniques. Our analysis was based on the calculation of weight distribution with the help of a dual code [15] and on studies of the extreme behaviour of the *undetected error probability* $P_{ud}(p_e)$ [14] where p_e is the *bit error probability*. Our results point out that the complexity of the BSC analysis is viable for up to 48 redundant bits and for a maximum of 5MB data on a current PC.

The detailed analysis showed that the SAFEDMI related safety targets are fulfilled. However, the studies demonstrated that none of the investigated CRC codes ensures that the probability of an undetected error is less than the usually used bound 2^{-c} (where c is the number of redundant bits) for all possible lengths of codewords. At the worst case

(CRC_SL2 with the codeword length 72 bits) the result is *50 times higher* than the mentioned bound [14].

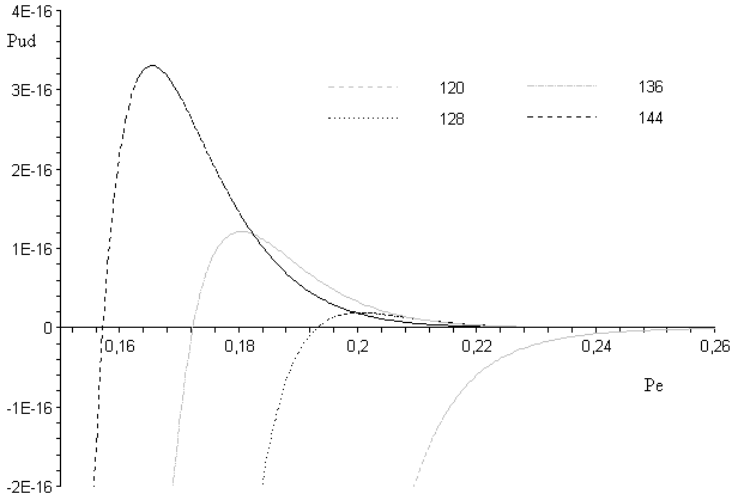

Fig. 7: $P_{ud}(p_e)$-2^{-32} for CRC_Cast (codeword lengths vary from 120 to 144 bits)

One of the proposed CRC codes for file integrity checks was the 32 bit long CRC_Cast analysed by Castagnoli et al. [15]. This CRC is closer to the 2^{-c} bound than other CRCs, but the probability of an undetected error of this code is not upper-bounded by this value (at the worst case the result is 1.0000014 times greater than the bound).

To get the results presented in Fig. 7, it was necessary to apply high precision arithmetic for the calculation of the extreme behaviour: the Newton's method for finding the zero points of the derivation of the function $P_{ud}(p_e)$ was used in the Maple tool.

4. Conclusion

The design, prototyping and evaluation activities of the SAFEDMI project demonstrated that (i) a DMI fulfilling the SIL 2 requirements can be realized using off-the-shelf hardware components and software implemented safety mechanisms, and (ii) providing a safe and secure wireless maintenance is feasible. The main results of the project include the thoroughly evaluated architecture design, the quantitative evaluation methodology integrating analytic, simulation based and experimental techniques, and also a comprehensive testing framework including a set of re-usable testing tools.

References

[1] UIC Subset-026 - rev.2.2.2, 2002 - ERTMS-ETCS Class 1 - System requirements specification.

[2] UIC Subset-033 - rev.2.0.0, 2000 - ERTMS-ETCS Class 1 - FIS for the Man-Machine Interface.

[3] ERTMS – *Driver Machine Interface Part 1-6*, CLC/TS 50459, 2005.

[4] EN 50129 *Railways applications – Communications, signalling and processing systems – Safety related electronic systems for signalling*, 2000.

[5] EN 50128 *Railways applications – Communications, signalling and processing systems – Software for railways control and protection system*, 2001.

[6] EN 50159-2 *Railway applications - Communication, signalling and processing systems - Part 2 – Safety related communication in open transmission systems*, 2001.
[7] EN 50155 *Railways Applications: Electronic equipment used on rolling stock*, 1997
[8] EN 50121-3-2 *Railway applications – Electromagnetic compatibility Part 3-2: Rolling stock – Apparatus*, 2001.
[9] Daly, D., D. Deavours, J. M. Doyle, P. G. Webster and W. H. Sanders, *Möbius: an extensible tool for performance and dependability modeling*. In Proc. TOOLS 2000, LNCS-1786, Springer Verlag, pp 332-336, 2001.
[10] Sanders, W. H., and J. F. Meyer, *Stochastic Activity Networks: Formal definitions and concepts*. Lectures on Formal Methods and Performance Analysis, vol. 2090 of LNCS, Springer Verlag, pp 315-343, 2001.
[11] Serafini, M., P. Lollini, and A. Bondavalli, *Modeling On-line Tests in Safety-critical Systems*. In Guedes, Soares and Zio (eds.), *Safety and Reliability for Managing Risk*, Vol. 1, Taylor & Francis Group, pp 231-238, 2006.
[12] Majzik, I., P. Domokos and M. Magyar, *Tool-supported Dependability Evaluation of Redundant Architectures in Computer Based Control Systems*. In Proc. FORMS/FORMAT 2007, GZVB, Braunschweig, pp 342-352, 2007.
[13] NS-2 - The Network Simulator. http://www.isi.edu/nsnam/ns/, 2005.
[14] Kárná, L., Š. Klapka, and M. Harlenderová, *Quantitative Assessment of Safety Code*. In Proc. FORMS/FORMAT 2008, l'Harmattan, Budapest, pp 249-255, 2008.
[15] Castagnoli, G., S. Brauer and M. Hermann, *Optimization of Cyclic Redundancy-Check Codes with 24 and 32 Parity Bits*. IEEE ToC Vol. 41, pp 883-892, 1993.
[16] Microtec, *VRTXsa Real-Time Kernel. Programmer's Guide and Reference*.
[17] *Gentoo Linux distribution*. Description available at http://www.gentoo.org
[18] *Open IKEv2*. Description available at http://openikev2.sourceforge.net
[19] Serafini, M., A. Bondavalli, and N. Suri, *Online diagnosis and recovery: On the choice and impact of tuning parameters*. IEEE TDSC, 4(4):295–312, 2007.
[20] Ceccarelli, A., I. Majzik, D. Iovino, F. Caneschi, G. Pintér and A. Bondavalli, *A Resilient SIL 2 Driver Machine Interface for Train Control Systems*, In Proc. DepCos-RELCOMEX 2008, Szklarska Poreba, Poland, pp 365-374, IEEE Press, 2008.
[21] Thatte, S.M. and J. A. Abraham, *Test generation for microprocessors*. IEEE Trans. Comput., 29(6):429–441, 1980.
[22] Hsueh, M.-C., T. K. Tsai, R. K. Iyer, *Fault Injection Techniques and Tools*. IEEE Computer, 30(4):75–82, 1997.
[23] SAFEDMI Consortium, *Quantitative Evaluation Methodology*, Deliverable 4.1, http://www.safedmi.org, 2008.
[24] BIMP, IEC, IFCC, ISO, IUPAC, IUPAP, and OIML. *ISO International Vocabulary of Basic and General Terms in Metrology (VIM)*. Second edition, 1993.
[25] Kent, S., and R. Atkinson, *Security Architecture for the Internet Protocol*, RFC 2401, November 1998.

Andrea Bondavalli is Professor in the Department of Systems and Informatics at the University of Florence. Previously, he was a researcher at the Italian National Research Council, working at the CNUCE Institute in Pisa, where he was responsible for the Dependable Computing Group. His research activity and interest are focused on the design and validation of critical systems and infrastructures. He is author of more than 130 refereed publications. He has been PI in many projects funded by the European Community, currently HIDENETS, SAFEDMI and AMBER. He acted as an expert for the European Community, and served as program chair of the most important conferences in the area.

Andrea Ceccarelli took his Minor degree in Computer Science at the University of Firenze in 2006 and is walking toward his Master degree. He has been involved in the European projects HIDENETS and SAFEDMI, and his research interests are in design and quantitative evaluation of dependable systems.

Jesper Grønbæk is a PhD student at Aalborg University Denmark, Networking and Security research section. He received his M.Sc.E.E. degree in Distributed Systems at Aalborg University in 2007. His research interests are in methods for fault detection and diagnosis in converging wired and wireless networks to enable dependable service provisioning. His current research activities are conducted in collaboration with TietoEnator IPS. Jesper has been involved in the EU-funded projects SAFEDMI and HIDENETS.

Danilo Iovino received his degree in Electronic Engineering from the University of Palermo. He worked for Alenia Spazio in the Payload Integrated Team on projects related to the International Space Station. Actually he has been working for Ansaldo Segnalamento Ferroviario in the R&D department on ERTMS and SCMT train-borne equipments. His experience refers to real-time embedded systems and ranges from requirement specification to software design, from implementation to testing.

Lucie Kárná was born in Prague. She received the diploma in Mathematical Analysis from the Charles University in Prague, Faculty of Mathematics and Physics. She is presently with the Research and Development division of AŽD Praha s.r.o.

Tatiana K. Madsen is an Associate professor at Aalborg University, Networking and Security research section. After her PhD studies in Moscow State University, Russia, she joined Dept. of Electronic Systems, Aalborg University, Denmark in 2001. Her research interests lie within the areas of wireless networking with the focus on performance optimization and mathematical modeling of wireless protocols behavior. She has been involved in a number of industrial, national and international projects.

Istvan Majzik is Associate Professor at the Budapest University of Technology and Economics. His research interests include the design, verification and validation of dependable computing systems. He was involved in several international projects funded by the European Community. He is co-author of more than 50 research papers and was served as PC member and reviewer of important conferences in the area.

Melinda Magyar received her diploma in CE from the Budapest University of Technology and Economics in 2006. Her current activities are related to the evaluation of the DMI architecture and implementation of automated tools that support the construction of dependability models. She is co-author of several related conference papers.

Štěpán Klapka was born in Prague, Czech Republic. He received the diploma in Numerical Analysis in 1987 and the Ph.D. degree in Scientific Computing in 2002, both from the Charles University in Prague, Faculty of Mathematics and Physics. He is presently with the Research and Development division of AŽD Praha s.r.o.

Anna Salzo received her degree in Computer Science from the University of Bari in 1996 and a Ph.D. in Computer Science at the same university in 2001. From 2001 to 2003 she worked for the Italian Consortium CINI as a researcher in the field of handwriting recognition and document analysis. Since 2003 she has been working in Ansaldo Segnalamento Ferroviario within the R&D department. She is involved in the specification, design and implementation of ERTMS and Italian automatic train protection systems.