D6.6 Evaluation of automotive demonstrator

Project number:	644080
Project acronym:	SAFURE
Project title:	SAFURE: SAFety and secURity by dEsign for interconnected mixed-critical cyber-physical systems
Start date of the project:	1st February, 2015
Duration:	40 months
Programme:	H2020-ICT-2014-1

Deliverable type:	Report
Deliverable reference number:	ICT-644080 / D6.6/ 1.0
Work package	WP 6
Due date:	31.05.2018
Actual submission date:	7th June, 2018

Responsible organisation:	MAG
Editor:	Stefania Botta
Dissemination level:	Public
Revision:	1.0

Abstract:	This deliverable describes the evaluation of the SAFURE automotive demonstrator. This demonstrator consists of a prototype of an automotive multicore control unit, and an automotive network prototype (deliverable D6.5 [49] gives more technical details about the prototypes). The evaluation is defined based on the requirements defined in the WP1 [35] .
Keywords:	Automotive, Tests, Network, Security, Safety, Real-time, Multicore

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 644080.

This work was supported by the Swiss State Secretariat for Education, Research and Innovation (SERI) under contract number 15.0025. The opinions expressed and arguments employed herein do not necessarily reflect the official views of the Swiss Government.

Editor

Stefania Botta (MAG)

Contributors/Reviewer

Fabrizio Lussiana, Paolo Giovannini (MAG)

Robin Hofmann, Borislav Nikolic (TUBS)

Edin Arnautovic, Eivind Christenson (TTT)

Bjoern Gebhardt (SYM)

André Osterhues (ESCR)

Marco Di Natale (SSSA)

Jaume Abella, Enrico Mezzetti (BSC)

Disclaimer

Executive Summary

This deliverable provides the evaluation details of the SAFURE xautomotive prototype. In particular, the automotive prototype consists of automotive multicore and automotive network prototypes.

For this reason, the first three chapters of the deliverable are dedicated to the evaluation of: automotive multicore prototype, network automotive prototype and the integration of automotive network and multicore prototype.

The multicore automotive prototype is mainly characterized by a control unit with an Aurix tri-core microcontroller [1] . The powertrain control unit integrated the SAFURE framework to guarantee “freedom from interferences”, secure communication over a CAN-bus and to exploit from one hand the new patterns and from the other the new multicore mechanisms provided by the SAFURE framework.

The network automotive prototype is focused on safety and security requirements required to enable mixed-critical communication in future in-vehicle Ethernet networks.

These two prototypes will be combined together introducing a hardware gateway which inserts CAN-messages into an Ethernet network.

For more details on the automotive prototype and on the SAFURE framework, please refer to the deliverable D6.5 [49] and D6.7 [50], respectively.

The other two fundamental chapters, of this deliverable, are focused respectively on: the requirement’s coverage defined in the WP1 and reported in the deliverables D1.1 [34] and D1.2 [35] . The other one provides potential and interesting evolutions of the SAFURE automotive prototype.

Contents

List of Figures

List of Tables

Chapter 1 Introduction

This deliverable provides the evaluation details of the automotive prototype. This prototype is composed by two sub prototypes: the automotive multicore prototype and the automotive network prototype. These two separate prototypes will be connected with a CAN-Ethernet gateway which translated CAN-messages into Ethernet frames and vice versa. The architecture of the automotive prototype is described in the deliverable D6.2 [48] and the detailed description of the prototype is provided in the deliverable D6.5 [49].

This deliverable reflects the structure of D6.2 [48] and D6.5 [49]. So, it is organized in the following way:

1. Evaluation of Multicore Control Unit (cf. Chapter 2)
    - Evaluation of safe protection mechanisms to guarantee the “freedom from interferences”
    - Evaluation of security library to guarantee a secure CAN communication
    - Application of security pattern and RTE generation
    - Evaluation of integration of multicore contention model for AURIX
    - Timing analysis

2. Evaluation of Automotive Network demonstrator (cf. Chapter 3)
    - Switches/gateways architecture
    - Interface definition to automotive multicore architecture
    - Automotive network simulation environment
    - Messages modeling extension

3. Evaluation of Combined Automotive Prototype (cf. Chapter 4)

Moreover, the Chapter 5 will show the coverage of the requirements established in the WP1 and reported in the deliverables D1.1 [34] and D1.2 [35] . Finally, Chapter 6 provides some potential and interesting evolutions for the SAFURE automotive prototype.

The automotive multicore prototype is focused on the integration of the SAFURE framework in the automotive industry. In particular, we applied the framework in a powertrain control unit based on Erika OS [15] and Aurix microcontroller [1] . In this document, we highlight the fundamental and innovative aspects that the ECU can provide after the integration of the SAFURE framework. In particular, it is able to guarantee:

- “freedom from interferences” at firmware level, according to the ISO 26262 [2] (chapter 2.1),
- secure communication on CAN in real time (Chapter 2.2),
- take advantages from multicore contention model and timing analysis (Chapters 2.4, 2.5),
- integrate RTE generation based on new secure pattern (Chapter 2.3).

For more details on the SAFURE framework please refer to the deliverable D6.7 [50].

In the automotive network architecture, an Ethernet network with traffic of various priorities and real time requirements is described. Apart from fault and failure tolerance, attack prevention mechanisms will be implemented.

Finally, some considerations and test details on the Ethernet gateway will be described in which the automotive multicore and automotive network architecture are combined in order to take in account the architectural aspects and requirements of these two separate prototypes.

Chapter 2 Evaluation of Multicore Control Unit

The automotive prototype consists of automotive multicore and automotive network prototype.

In particular, the multicore automotive prototype is mainly characterized by a control unit with an Aurix Tricore microcontroller [1] . The powertrain control unit integrates the SAFURE framework to guarantee the “freedom from interferences”, secure communication over a CAN-bus and to exploit from one hand the new patterns and from the other the new multicore mechanisms provided by the SAFURE framework.

The network automotive prototype is focused on safety and security requirements required to enable mixed-critical communication in future in-vehicle Ethernet networks.

These two prototypes will be combined together introducing a hardware gateway, which inserts CAN messages into an Ethernet network.

Refer to the D6.5 [49] for more details on the prototype.

2.1 Evaluation of safe protection mechanisms

The protection mechanisms in the automotive demonstrator guarantee the memory protection in two different scenarios: internal communication and external communication (cf. requirements S2-NF-004, S2-NF-008 - Table 9).

The first kind of memory protection is developed by MAG at software level as part of the SAFURE framework and described in detail inside the WP4 (see D4.1 [46] and D4.3 [47]). The aim is to protect a specific part of memory shared between the two cores from non-authorized access.

This second memory protection strategy is managed by the AURIX microcontroller (i.e., in hardware). In particular, all external requests to access at specific section of memory need to be approved by the HSM (Hardware Security Module).

2.1.1 Tests

The tests made on the memory protection solution implemented through the firmware level are described in the deliverable D4.3 [47].

The test strategy implemented to verify the memory protection against adversaries is made using the UDS (Unified Diagnostic Services) on CAN software testing simulator suite, we try to read specific areas of memory through a service provided by the standard ISO 14229-1:2013 [37].

2.1.2 Evaluation

These mechanisms developed by MAG in WP4 (see D4.1 [46] and D4.3 [47]) represent an optimized alternative to the mechanisms supported by an RTOS. Moreover, they allow obtaining an ASIL B level (refer to the ISO 26262 [2]) for the ECU Product also if the RTOS does not support these features.

2.2 Evaluation of secure real-time CAN communication

One of the goals for the automotive multicore demonstrator is to provide a powertrain control unit able to guarantee a secure real-time CAN communication.

The advantages of this solution are the exchange of encrypted information in real-time, avoiding possible malicious external attacks.

The solution, proposed in SAFURE and integrated in this control unit, establishes statically the classification of secure messages, integrates the cryptographic algorithms to send and receive encrypted and integrity-protected messages and implements a strategy to handle those messages that fail the security checks (refer to D6.5 [49] for more details).

In the following two sections, we will describe how we have implemented the tests to verify the secure communication of the control unit on CAN line and the proof that it is able to recognize and handle the corrupted messages received.

2.2.1 Tests

The test was implemented to show how our Secure CAN Communication solution guarantees the correctness of the information and the capability to catch each possible corruption or intrusion. It is conducted using a Powertrain ECU connected to a PC with a CAN Analyzer (Vector Tool) [38] on board.

The CAN Analyzer [38] has been programmed to send back each received message to the ECU with the correct address and the same information for each message type (A, B and C, cf. D.6.5 for details). In this way, the communication is always ok. As shown in Figure 1, we tried to simulate some cases of possible dangerous situation on Type A and B messages.

0.009337	CAN 1 A7	CAN Frame	Rx	8	8	01 06 00 00 FF 4D 41 47
0.000733	CAN 1 B2	CAN Frame	Tx	8	8	01 06 00 00 FF 4D 41 47
0.009260	CAN 1 A7	CAN Frame	Rx	8	8	02 F2 09 16 19 9B 1F B4
0.000678	CAN 1 B2	CAN Frame	Tx	8	8	02 F2 09 16 19 9B 1F B4
0.009327	CAN 1 A7	CAN Frame	Rx	8	8	03 6F 4A 79 80 B3 59 58
0.000709	CAN 1 B2	CAN Frame	Tx	8	8	03 6F 4A 79 80 B3 59 58
0.009295	CAN 1 A7	CAN Frame	Rx	8	8	04 36 94 55 55 55 55 55
0.000644	CAN 1 B2	CAN Frame	Tx	8	8	04 36 94 55 55 55 55 55
0.009381	CAN 1 A7	CAN Frame	Rx	8	8	01 06 00 00 00 4D 41 47
0.000735	CAN 1 B2	CAN Frame	Tx	8	8	01 06 00 00 00 4D 41 47
0.009258	CAN 1 A7	CAN Frame	Rx	8	8	02 6D 42 9A F3 EE C5 D6
0.000646	CAN 1 B2	CAN Frame	Tx	8	8	02 6D 42 9A F3 EE C5 D6
0.009363	CAN 1 A7	CAN Frame	Rx	8	8	03 DE 11 02 E2 2C 69 BA
0.000719	CAN 1 B2	CAN Frame	Tx	8	8	03 DE 11 02 E2 2C 69 BA
0.009282	CAN 1 A7	CAN Frame	Rx	8	8	04 E8 8F 55 55 55 55 55
0.000719	CAN 1 B2	CAN Frame	Tx	8	8	04 E8 8F 55 55 55 55 55
0.009303	CAN 1 A7	CAN Frame	Rx	8	8	01 06 00 00 01 4D 41 47
0.000731	CAN 1 B2	CAN Frame	Tx	8	8	01 06 00 00 01 4D 41 47
0.009262	CAN 1 A7	CAN Frame	Rx	8	8	02 BD AA D4 F1 2F BD 9D
0.000622	CAN 1 B2	CAN Frame	Tx	8	8	02 BD AA D4 F1 2F BD 9D
0.009387	CAN 1 A7	CAN Frame	Rx	8	8	03 9A 86 19 4D B1 1D FE
0.000721	CAN 1 B2	CAN Frame	Tx	8	8	03 9A 86 19 4D B1 1D FE
0.009284	CAN 1 A7	CAN Frame	Rx	8	8	04 A3 33 55 55 55 55 55
0.000606	CAN 1 B2	CAN Frame	Tx	8	8	04 A3 33 55 55 55 55 55
0.009414	CAN 1 A7	CAN Frame	Rx	8	8	01 06 00 00 02 4D 41 47
0.000721	CAN 1 B2	CAN Frame	Tx	8	8	01 06 00 00 02 4D 41 47
0.009275	CAN 1 A7	CAN Frame	Rx	8	8	02 B0 28 71 52 46 09 AE
0.000576	CAN 1 B2	CAN Frame	Tx	8	8	02 B0 28 71 52 46 09 AE
0.009429	CAN 1 A7	CAN Frame	Rx	8	8	03 7E E3 1C 67 02 1E 07
0.000723	CAN 1 B2	CAN Frame	Tx	8	8	03 7E E3 1C 67 02 1E 07
0.009280	CAN 1 A7	CAN Frame	Rx	8	8	04 25 E4 55 55 55 55 55
0.000584	CAN 1 B2	CAN Frame	Tx	8	8	04 25 E4 55 55 55 55 55

Figure 1: Communication without errors (CANalyzer view)

A “Message Corruption Test” has been performed (see Figure 2): on user request, one byte in a single frame is changed (not the first one, because that is the frame counter). It does not matter if in the message body or in the signature. In this case, the ECU could correctly build the whole message, but the signature check will fail, the ECU recognizes an error and discards the message.

0.009407	CA... A7	CAN Frame	Rx	8	8	01 06 00 00 A4 4D 41 47
0.000731	CA... B2	CAN Frame	Tx	8	8	01 06 00 00 A4 4D 41 47
0.009258	CA... A7	CAN Frame	Rx	8	8	02 5B 62 57 12 81 1F C5
0.000598	CA... B2	CAN Frame	Tx	8	8	02 5B 62 57 12 81 1F C5
0.009405	CA... A7	CAN Frame	Rx	8	8	03 B9 EE 2E 97 DE 23 5C
0.000721	CA... B2	CAN Frame	Tx	8	8	03 B9 EE 2E 97 DE 23 5C
0.009314	CA... A7	CAN Frame	Rx	8	8	04 07 93 55 55 55 55 55
0.000721	CA... B2	CAN Frame	Tx	8	8	04 07 93 55 55 55 55 55
0.009291	CA... A7	CAN Frame	Rx	8	8	01 06 00 00 A5 4D 41 47
0.000731	CA... B2	CAN Frame	Tx	8	8	01 06 00 00 A5 4D 41 47
0.009266	CA... A7	CAN Frame	Rx	8	8	02 66 7C 6E A5 C9 89 6E
0.000550	CA... B2	CAN Frame	Tx	8	8	02 66 7C 6E A5 C9 A5 6E
0.009457	CA... A7	CAN Frame	Rx	8	8	03 B9 EF 77 60 6C 17 9A
0.000721	CA... B2	CAN Frame	Tx	8	8	03 B9 EF 77 60 6C 17 9A
0.009286	CA... A7	CAN Frame	Rx	8	8	04 6F FD 55 55 55 55 55
0.000514	CA... B2	CAN Frame	Tx	8	8	04 6F FD 55 55 55 55 55

Figure 2: Failure communication (CANalyzer view)

A “Lost Frame Test” has been performed: on user request a single frame will not be replied; in this case, the ECU will recognize a frame sequence error, so it will recognize the error,

discard the whole message and put itself in “resynchronization” state, waiting for a counter value of “1”.

A “Man in the middle Test” has been performed, where, on request, different message frames substitute the correct ones. In this case, if the sequence counter is incorrect, the ECU will react as in “Lost Frame Test” case. If the sequence counter is correct, the ECU reacts as in “Message Corruption Test”, because the key is different. Figure 3 shows in the emulator window the case where the sequence number is correct, because it is equal to zero, after having received four complete messages.

Figure 3: Correct sequence counter case (Emulator view)

Figure 4 and Figure 5 show for messages of Category B and A, always in the emulator window, the case where the sequence counter is not correct, because it is different from zero.

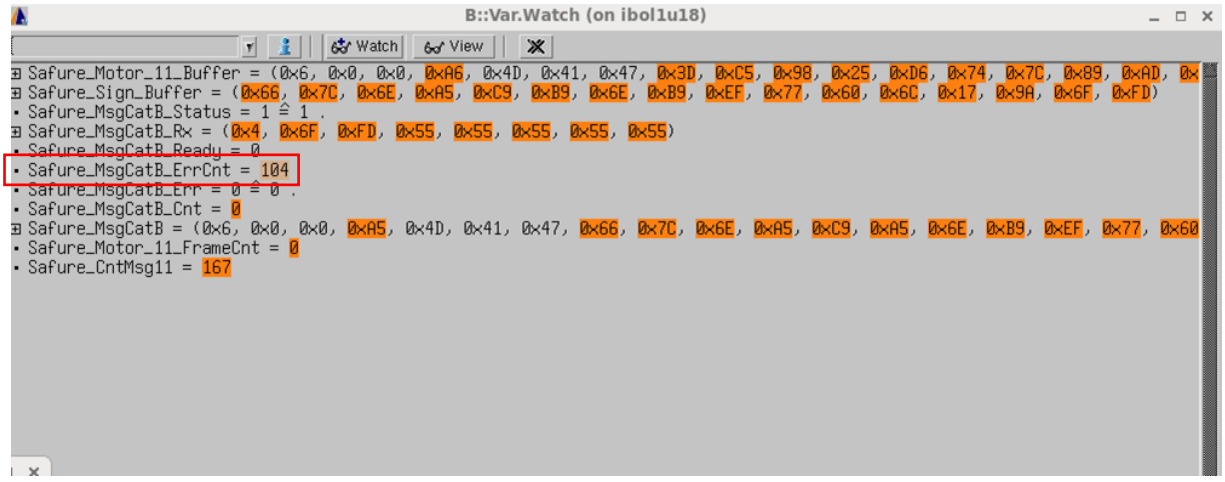


Figure 4: Incorrect sequence counter case for Cat B (emulator view)

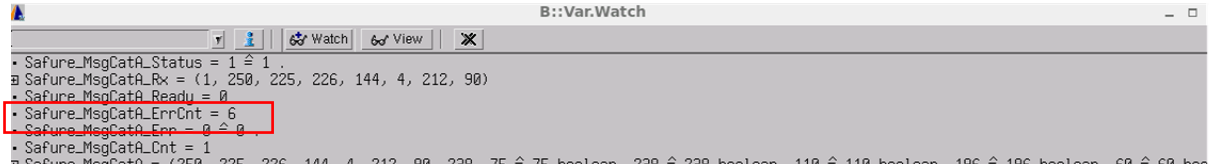


Figure 5: Incorrect sequence counter case for Cat A (emulator view)

In case of message Type C (single frame, not signed and not encrypted), no “Corruption Test” and no “Man in the middle Test” could find any error.

2.2.2 Evaluation

We have shown that it is possible to protect both the integrity and confidentiality of CAN messages. Common attack types like “message corruption”, “lost frame”, and “man in the middle” are successfully detected. However, we have not considered sessions and potential resynchronisation issues. Also, our implementation does not use a full-blown protocol.

Radu and Garcia [45] recently proposed a lightweight authentication protocol (LeiA) for CAN. It uses MACs to protect the integrity of messages. It does not mandate a concrete MAC algorithm, but only states that it uses 64-bit authentication tags, in order to fit into the 8-byte payload field of a CAN data frame. The benefits of the approach are that it considers sessions, handles resynchronization, and proofs the security of the protocol under the unforgeability assumption of the underlying MAC algorithm, under chosen message attacks. However, it does not integrate confidentiality into the protocol.

We suggest using a mechanism similar to LeiA, but with additional confidentiality. This could be done by replacing the MAC algorithm with AES in GCM mode.

2.2.2.1 Secure code Load

Using the Lauterbach TRACE32 [43] environment (see Figure 6), we have measured the load of the secure code integrated into the demonstrator to guarantee the secure real-time CAN communication.

Figure 7 shows several Lautherbach windows that we have configured to highlight the encryption code activated when a message of Category A or B is ready to be transmitted by the ECU. In particular, two windows show the period of the encrypted code for Category A message (100 ms means that each 20ms one of the five frames is transmitted) and for Category B message (40ms means that each 10ms one of the four frames is transmitted). The other two ones show the distribution of the duration of the pattern represented in microsecond of the encrypted code for Category A and for Category B messages. It means that the Category A and Category B messages are encrypted with a load of 0.3% for each Category.

Figure 8 concerns the measurements collected in the transmission case of Category A and Category B messages.

Figure 6: Lauterbach TRACE32 environment

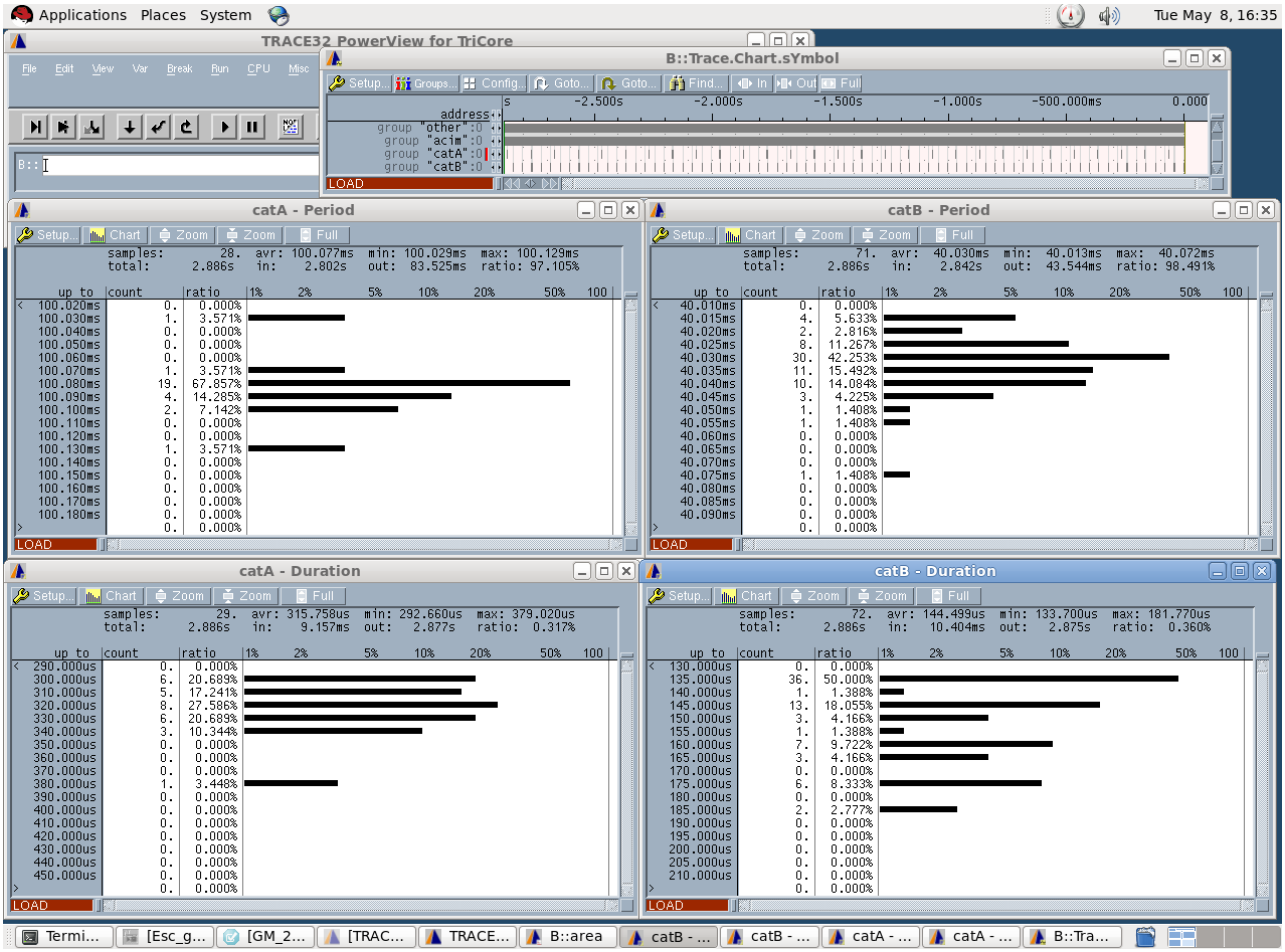


Figure 7: Secure code load for encryption - Lauterbach view

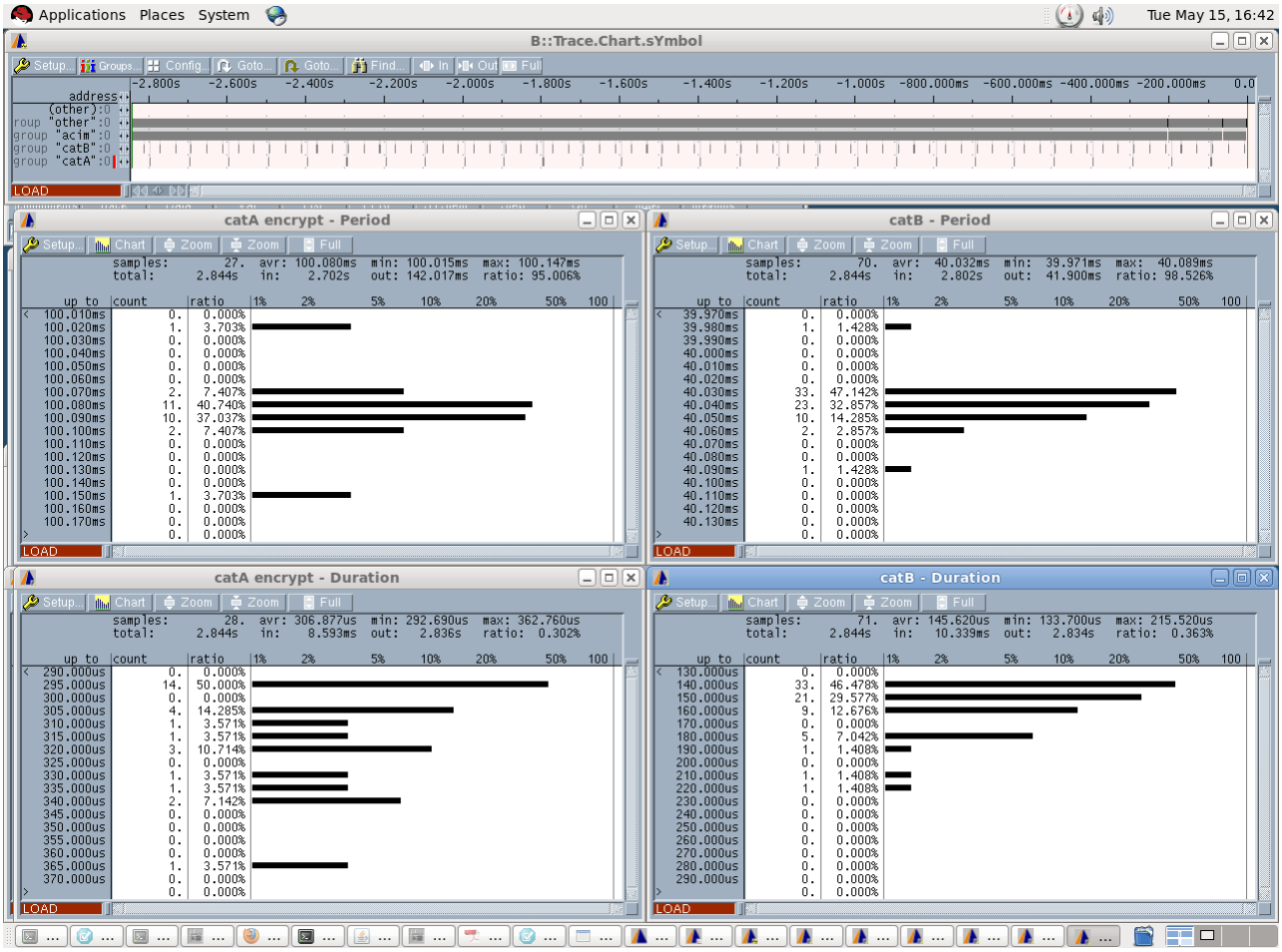


Figure 8: Secure code load for encryption - Lauterbach view

2.3 RTE generation

The subset model of the automotive application defined in D6.5 [49] has been represented in Rhapsody Modelling tool.

Figure 9 shows a screenshot with the ten application components. The AUTOSAR model includes several packages, with the component model, the implementation model (and the worst-case execution time estimates), the interfaces, the types and the constraints.

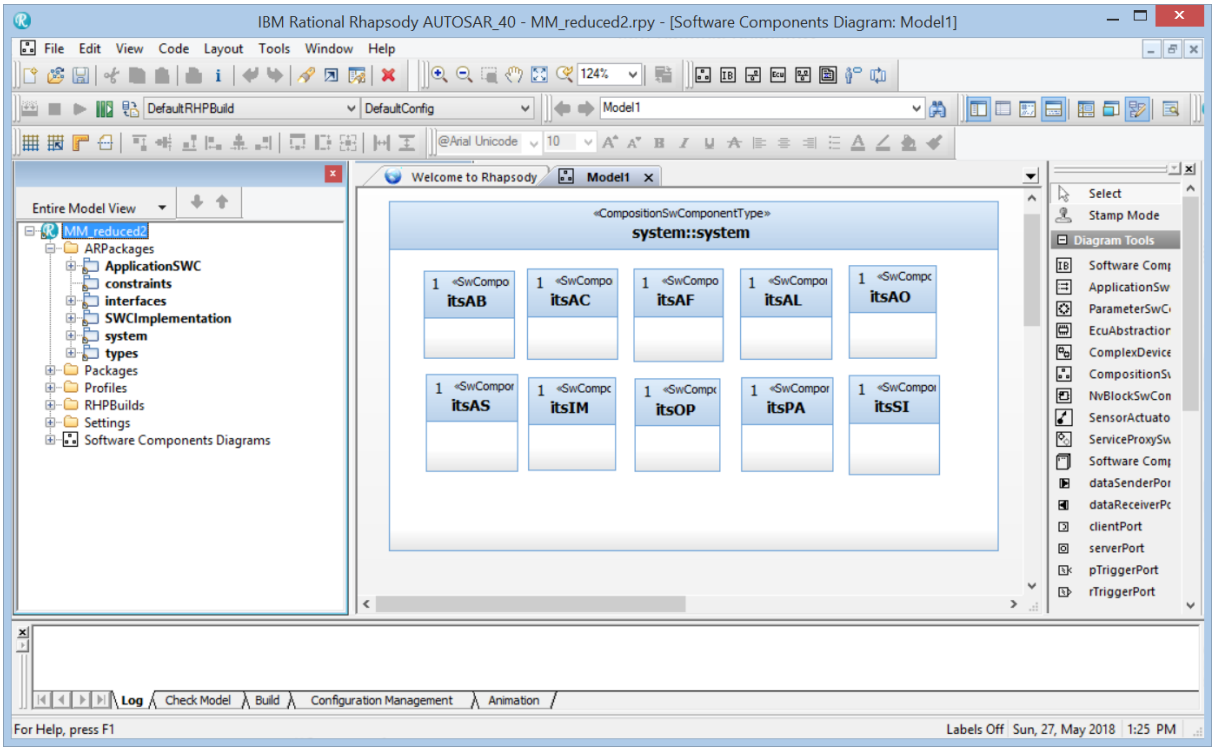


Figure 9: Rhapsody model for the case-study application: components and packages

In addition, the model contains the definition of the internal behaviour of all the components, with the runnables and the periodic events that are responsible for their activation. The left side of Figure 10 shows the details for the AF component.

The profiles identified for SAFURE are applied to the model and the stereotypes specifying the application of safety levels have been applied to selected runnables in the example. The runnable AF_Fast, is identified as executing at the safety level ASIL 3 in the model (left side of Figure 10).

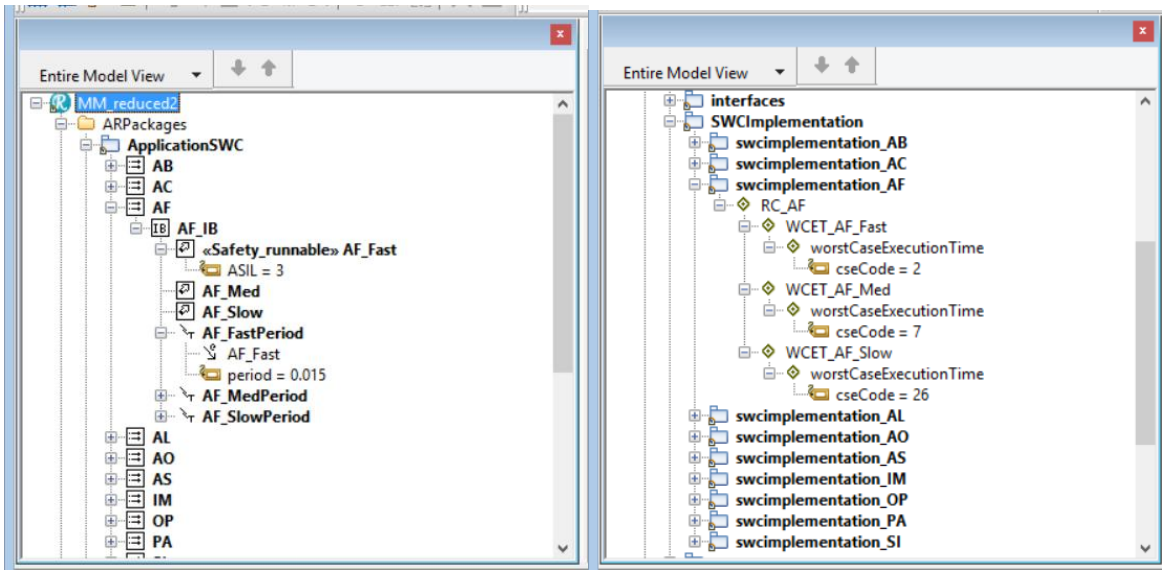


Figure 10: Rhapsody model for the case-study application: Behaviours, stereotypes and WCETs

Finally, the model includes an implementation model with the specification of the worst-case execution times of all the runnables (right side of Figure 10).

The model has been saved and exported as an ARXML. The ARXML has been then imported in Artop (an open source tool, based on Eclipse and created by the AUTOSAR consortium). We use Artop for the representation of the task model and the ECU configuration.

When imported in Artop, the AUTOSAR model contains the same information with a slight different format of the model tree view (as in Figure 11, showing the corresponding representation of the SW components, the runnables and the periodic events activating them).

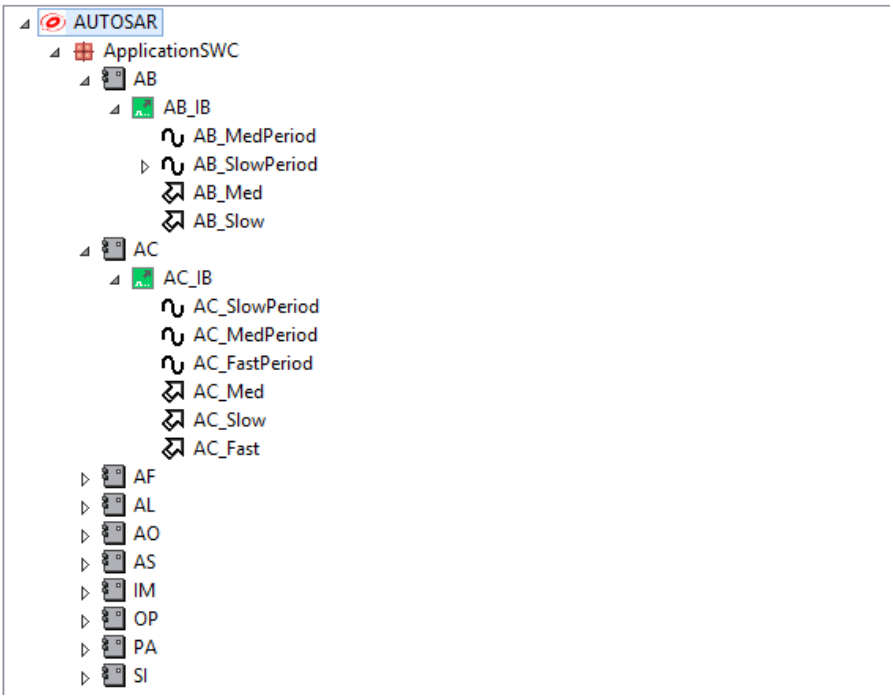


Figure 11: Artop model for the case-study application

The task model has been added manually in Artop, since Rhapsody is not meant to be a modeller for RTE and OS configuration and generation.

In ARTOP we added the description of the tasks and the other information related to the OS configuration.

Figure 12 shows an example of the imported information on the left, and an outline of the additional information on the EcuConfiguration on the right. The ECU Configuration part includes information on all the tasks of the system, the mapping of the runnables into the tasks, and the OS features that need to be defined to execute the tasks in the model (Counters, Alarms).

The protection data was generated starting from the modelling extensions in Rhapsody and backannotated to the corresponding ECU Configuration part, defining the criticality level for each task.

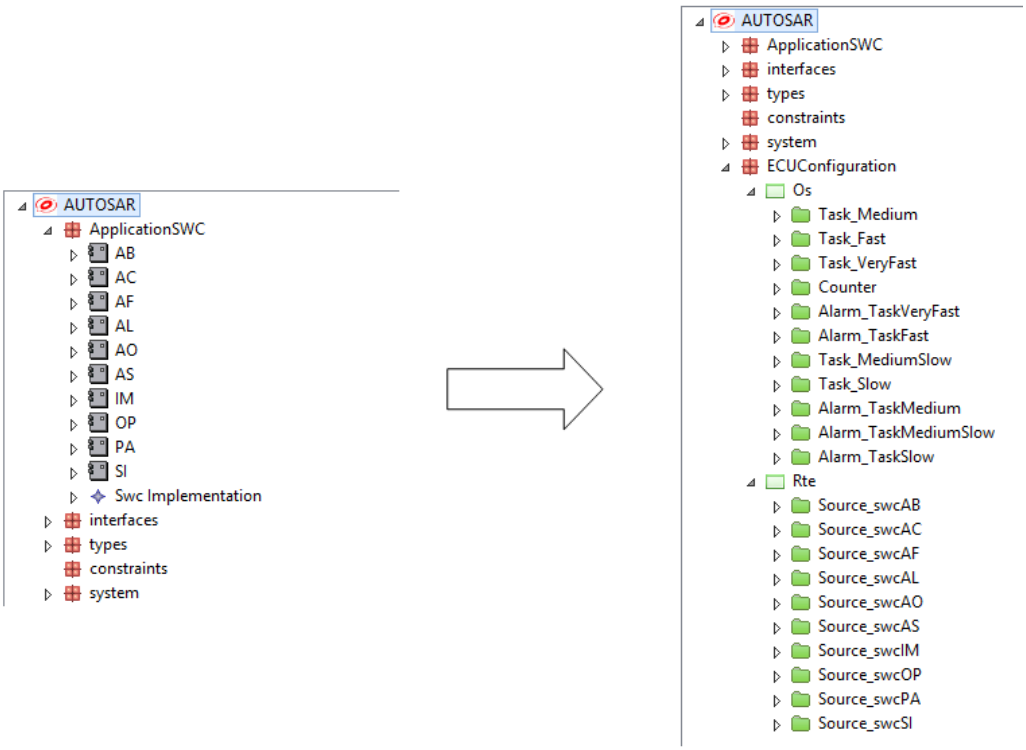


Figure 12: The extension in Artop to the ECU Configuration part

As shown in the Figure 13, the final Artop model is backannotated with the SAFURE AUTOSAR modelling extensions to specify the criticality level of the tasks. In particular, `Task_Fast` has been assigned a criticality level 3, thus implying a timing protection for both `Task_Fast` and `Task_VeryFast`.

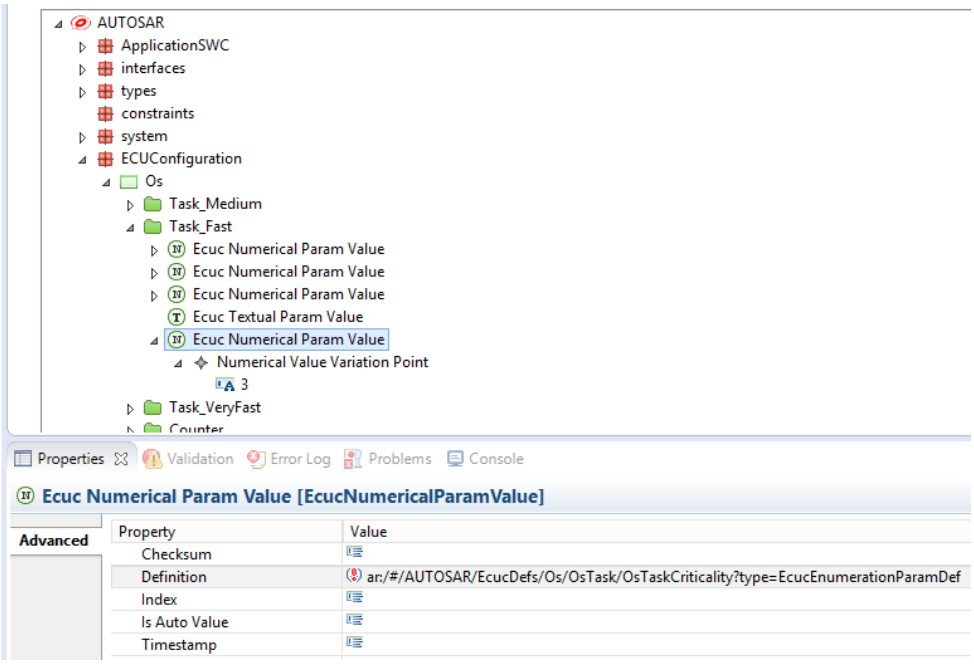


Figure 13: Artop model enrichment with the SAFURE AUTOSAR extensions

Next, the RTE generator has been invoked.

As a result of the specification provided in the model, the generator produced the RTE `.c` and `.h` C files, and the OIL configuration of the operating system.

For example, the code snippet of a task is reported below, where the corresponding runnables are sequentially called.

```
TASK(Task_Medium)
{
    {
        SI_Med();
    }
    {
        AB_Med();
    }
    {
        AC_Med();
    }
    {
        OP_Med();
    }
    {
        AS_AL_Med();
    }
    {
        AF_Med();
    }

    /* end this task */
    TerminateTask();
}
```

An excerpt from the generated OIL configuration is also reported (the timing values are expressed in seconds):

```
 TASK Task_Fast {
        ACTIVATION = 1;
        PRIORITY = 4;
        SCHEDULE = FULL;
       TIMING_PROTECTION = TRUE {
               EXECUTIONBUDGET = 0.00234;
       };
    };

    TASK Task_Medium {
        ACTIVATION = 1;
        PRIORITY = 3;
        SCHEDULE = FULL;
    };
```

```
TASK Task_MediumSlow {
    ACTIVATION = 1;
    PRIORITY = 2;
    SCHEDULE = FULL;
};

TASK Task_Slow {
    ACTIVATION = 1;
    PRIORITY = 1;
    SCHEDULE = FULL;
};

TASK Task_VeryFast {
    ACTIVATION = 1;
    PRIORITY = 5;
    SCHEDULE = FULL;
   TIMING_PROTECTION = TRUE {
          EXECUTIONBUDGET = 0.00002;
   };
};
```

Mock code for the runnables has been written by hand. “For” loops with `nop` instructions have been adopted to simulate their execution. The number of iterations performed by each loop has been tuned to obtain a first-order approximation of the execution times provided in the model. To simulate a timing fault, the code of runnable `AL_VeryFast` (executed by `Task_VeryFast`) includes a conditional branch to double the number of for-loop iterations if a global flag is set.

Figure 14 shows a trace of the regular execution of the application (taken from the Lauterbach TRACE32 [43] environment).

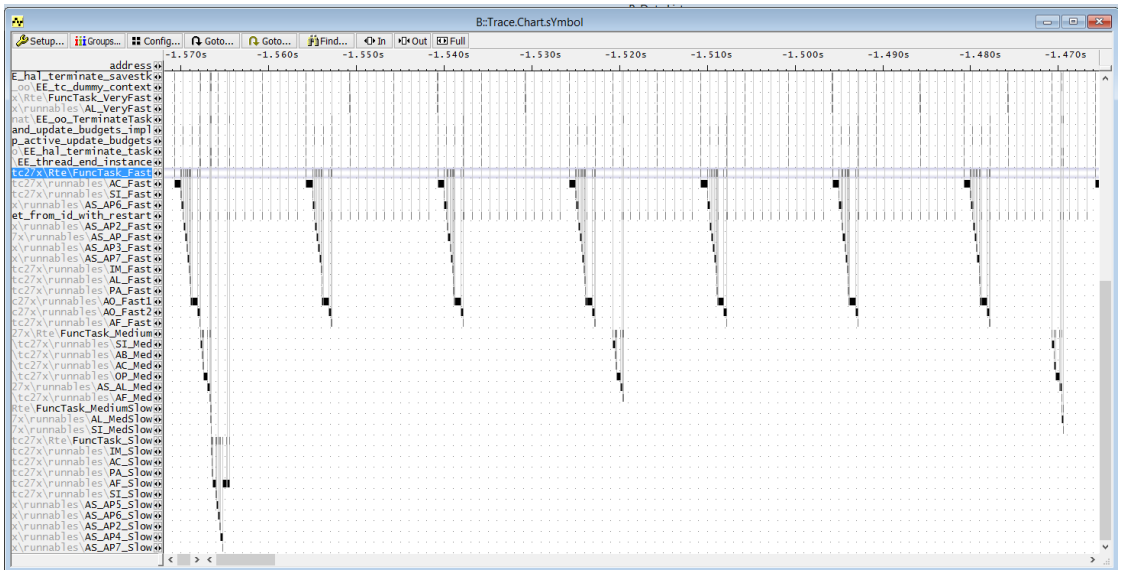


Figure 14: Trace of a regular execution of the case-study application

As it can be observed in Figure 15 when the fault is injected (causing `Task_VeryFast` to exceed its execution budget) the ProtectionHook is correctly invoked. In our testing setup, the ProtectionHook has been configured to react at a timing fault by shutting down the system.

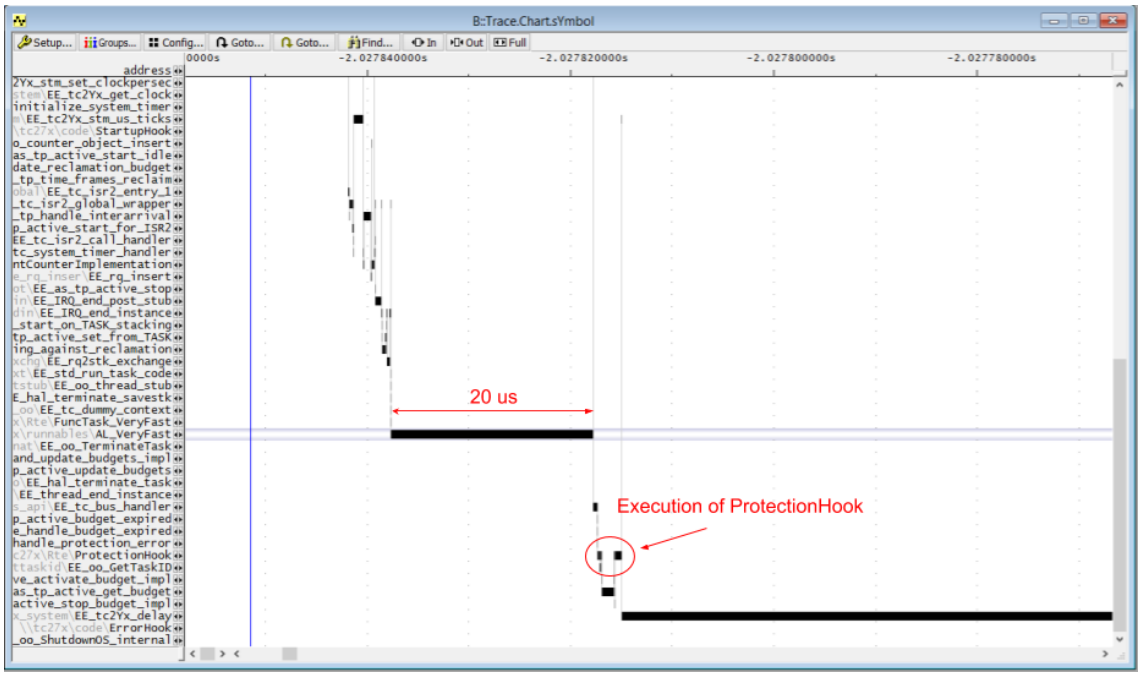


Figure 15: Trace of an execution where a timing fault is injected

Also, the LEDs on the board were used to signal the occurrence of a timing fault.

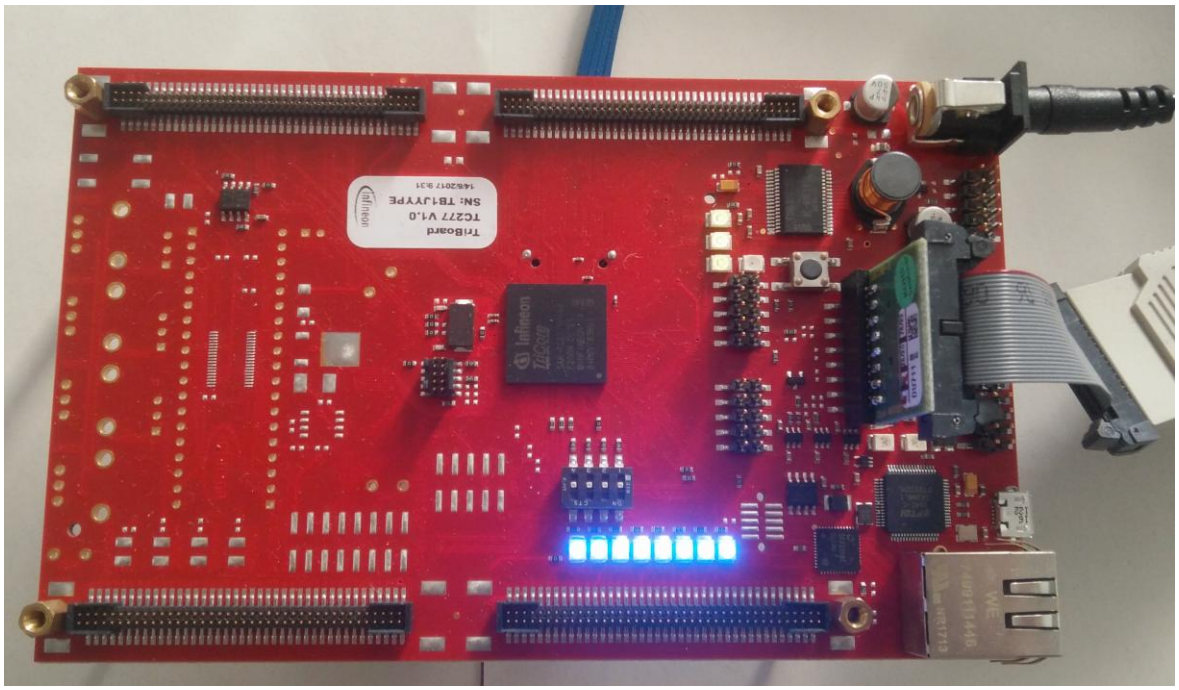


Figure 16: All LEDs of the Triboard start blinking when a timing fault occurs

2.4 Evaluation of integration of multicore contention model for AURIX

The proposed contention model has been assessed against small and medium size functions extracted from a complete automotive application. The evaluation was not performed on an end-to-end automotive task as it has been observed that the approach itself is particularly efficient (and naturally applied) at the level of software units, hence during the unit-testing verification effort, rather than on run-time entities.

Two functions have been selected for conducting the model assessment. Their specific deployment on the platform mimics a specific application deployment scenario. In the AURIX TC27x platform, both application code and data can be mapped to different memory areas, where all areas correspond to different interfaces in the cross-bar interconnect. Since the cross-bar supports parallel transactions on different interfaces, contention may happen only between requests targeting the same interface. Consequently, the location where code and data are mapped is determining the potential contention in the system. The analysed functions characteristics are reported in Table 1 below.

Table 1: Characterization of the automotive functions

Function	Size	Code	Data	Characterization
Fun A	Small	PSPR	DSPR	Application is small enough to fit in the local scratchpads (PSPR and DSPR). No activity is expected on the cross-bar and model should predicted no contention
Fun B	Medium	PFlash0	DSPR (Stack) PFlash1 (Constants)	Application is accessing the cross-bar for fetching code and data. Both code and data are mapped to the PFlash, but on separate areas that are accessed from different interfaces.

The contention model allows to compute a fully time-composable upperbound to the (worst-case) contention effect based on the number of cross-bar accesses and target thereof of the analysed tasks. Such an upperbound is valid under any possible deployment scenario, regardless of the functions that are concurrently executed on the platform. This exceptionally wide validity scope comes at the cost of some degree of often unnecessary pessimism. Having the same information on the actual co-runners allows deriving much more realistic (and tighter) bounds.

The model inputs comprise the readings of nine (9) performance monitoring counters (PMCs) on the target platform. The set of hardware events of interest are summarized in Table 2.

Table 2: Hardware events and monitoring counters of interest

Counter	Event	Relevance
CCNT	Clock count	Executed cycles describe the baseline timing behaviour that must be inflated to account for inter-core contention
ICNT	Instruction count	Used for sanity check between different runs of the same program
PCACHE_HIT	Program cache hit	Indicator of good cache performance

Counter	Event	Relevance
PCACHE_MISS	Program cache miss	Event triggering a cross-bar request to the interface where program code is deployed
DCACHE_HIT	Data cache hit	Indicator of good cache performance
DCACHE_MISS_CLEAN	Data cache miss clean	Event triggering a cross-bar request to the interface where program data is deployed
DCACHE_MISS_DIRTY	Data cache miss dirty	Event triggering a cross-bar request to the interface where program data is deployed
PMEM_STALL	Cycles when the program interface has been stalled for whatever reason	Strongly related to stalls suffered due to cross-bar requests to the interface where program code is deployed
DMEM_STALL	Cycles when the data interface has been stalled for whatever reason	Strongly related to stalls suffered due to cross-bar requests to the interface where program data is deployed

The combination of cache-related events and stall cycles are used to conservatively derive the number and type of all requests over the cross-bar. Each type of request is associated a worst-case latency.

Owing to the Performance Monitoring Unit (PMU) implementation on the AURIX, readings over the target PMCs could not be collected on a single experiment. Since the PMU provides three configurable registers (besides the CCNT and ICNT), we were required to perform and capture PMC values from three different executions. The same PMC collection process has been applied to both functions under analysis. Raw numbers are reported in Table 3, where the three different executions for each function are identified as “profile *n*”.

Table 3: PMC readings from the experiments

Function	Profile 1		Profile 2		Profile 3	
Fun A	PCACHE_HIT	0	DCACHE_HIT	0	TOTAL_BRANCH	640
	PCACHE_MISS	0	DCACHE_MISS_CLEAN	0	PMEM_STALL	0
	MULTI_ISSUE	5629	DCACHE_MISS_DIRTY	0	DMEM_STALL	7
	CCNT	16361		20969		20969
	ICNT	17538		23734		23734
Fun B	PCACHE_HIT	9614	DCACHE_HIT	263	TOTAL_BRANCH	1525
	PCACHE_MISS	173	DCACHE_MISS_CLEAN	9	PMEM_STALL	1380
	MULTI_ISSUE	5119	DCACHE_MISS_DIRTY	0	DMEM_STALL	156
	CCNT	20969		20969		20969
	ICNT	23734		23734		23734

The MULTI_ISSUE and TOTAL_BRANCH counters were collected as a by-product of the measurement protocol but are not fed to the contention model.

As expected, the small function (Fun A) completely fits into the core scratchpads and does not generate cross-bar traffic (except for 7 cycles counted in the DMEM_STALL that are triggered by the measurement protocol). The medium function (Fun B) instead is fetching code and data from the Flash device and, despite the good cache usage, makes use of the cross-bar which in turn exposes to inter-core contention.

The objective of Fun A was to verify the behaviour of the counters and to show that the model can detect and adapt to the deployment scenario. In fact, after removing the probe effect, the analytical contention bound computed by the model is zero. We expected more interesting results from applying the model to Fun B. The analytical model, which in its latest release has been implemented as an ILP problem, was used to compute a fully time-composable bound to the multicore contention. Under the considered configuration scenario, Fun B seems to be relatively robust against inter-core contention as the model determines that its execution time can only increase up to 7.2% due to multicore contention (cf. Table 4 below) when run together with other tasks in the other high-performance core, regardless of the characteristics of those tasks. Hence, typically contention experienced will be below 7.2% despite of what shared resources (and when) are accessed by other tasks in the other core.

Table 4: Contention model results

	Isolation MOET	*Analytical Bound*	*Inflated MOET*	*Inflation ratio*
Fun A	20969	0	20969	0%
Fun B	20969	1504	22473	**7,17%**

It would have been interesting to use the model not only on the task but also on a set of contenders so that to compute an even less pessimistic partially time-composable bound. However, results obtained already prove that the potential execution time increase due to running other tasks in the other cores simultaneously is very low.

Overall, the integration and evaluation of this methodology allows reaching the following positive conclusions:

- No roadblock is foreseen to integrate the methodology on industrial use cases.
- Multicore contention bounds can be applied at unit testing and provide information independent of the final integration of the whole system, thus enabling the application of the methodology in early design stages, so that potential violations of timing bounds can be addressed soon in the design process.
- Results, if software is deployed efficiently, provide evidence (supported by the methodological approach and the quantitative assessment) showing that the inflation factor to use on top of the MOET is low, thus providing guarantees with low potential impact on the utilization of the hardware resources.
- While some integration steps need some consultancy for their application the first time, the application of this methodology on further software units can be carried out by end users on their own, thus providing them with independence to analyse their software.

2.5 Timing Analysis: SymTA/S

The electronic components in the automotive industry need to be capable of meeting the timing requirements of the functions that are implemented on them. This is important in order to select right CPUs and busses with a suitable topology and optimized traffic configurations. On one hand, this influences the reliability or correctness (from a timing perspective) of a system. On the other hand, the timing properties of a system determine the unused performance reserves to a large part. They can be optimized or explicitly kept for later extensions.

So, timing and specifically scheduling analysis are very important. They make it possible to control and verify timing during implementation, integration, and verification.

The SymTA/S tool [41] is the core product of Symtavision and it is able to define the components of an ECU graphically and interactively. On this basis, the program generates a mathematical model based on the timing behaviour. After being solved quickly, it provides information about the system timing behavior, and identifies worst case configuration parameters automatically.

For these reasons, we have applied the SymTA/S tool on the automotive multicore prototype. In this chapter, we will report some results and useful considerations on the analysis provided by SymTA/S, applied to the powertrain control unit.

2.5.1 Analysis Scenario

As input for the timing analysis, we have imported a part of the AUTOSAR software architecture model that represents our powertrain control unit. In particular, we have imported and set up the scheduling part of the architecture. In this sense, in our engine control, there are two kinds of tasks: periodic (activated by a timer at fixed rate) and angular (activated at specific rotation angles). Respectively, the involved tasks have the following periods and are listed in a decreasing fixed priorities way:

- Task Pms → angular task
- Task Time Fast → 4 ms
- Task Medium Time → 12 ms
- Task Time Slow → 100 ms
- Task Time Very Slow → 1000 ms

According to this scheduling, we have applied the WCRT (Worst Case Response Time) analysis made by SymTA/S, introducing the execution times collected simulating the control unit system with a RPM (revolution per minute) value equal to 5000 so the “Task Pms”, for a 3 cylinders application, has a period of 8 ms. In order to measure the execution time, we have used the Lauterbach emulator [43].

Figure 17 and Figure 18 show two different views for the calculated worst case load. Just some clarification on Figure 17, the grey “Idle” piece means that for the 61,7% the system is not loaded, instead “DummyTask_IDLE” and “DummyTask_CRUISE” can be ignored for the timing analysis purpose. They collect the one shot tasks like Power On and Power Off tasks. The first one gives an overview of the whole load of the system; the second one is more focus on each task load.

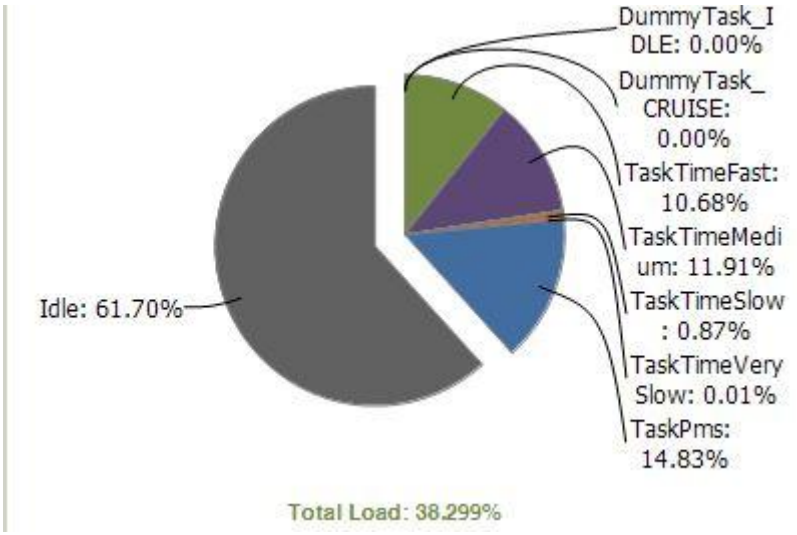


Figure 17: Worst Case Load

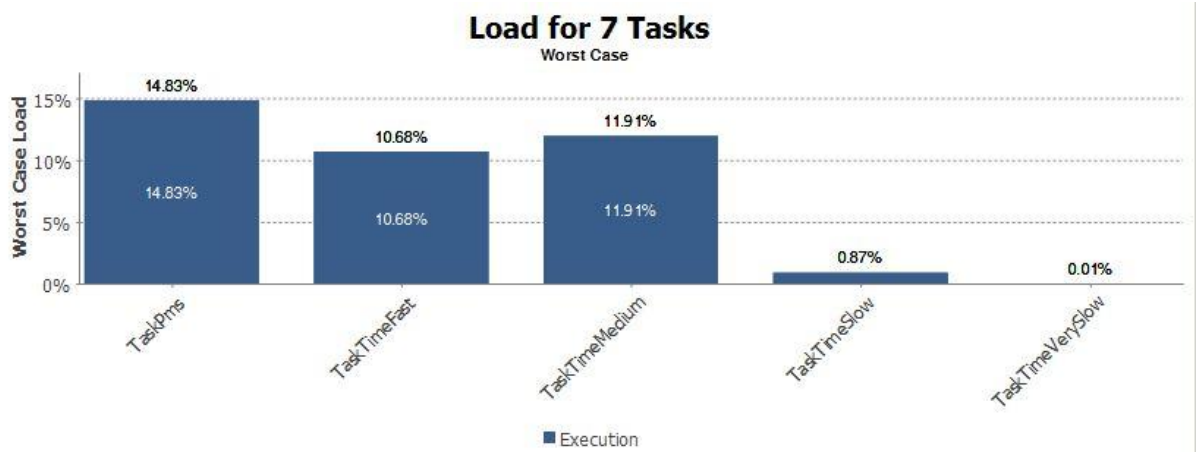


Figure 18: Worst Case Load for each Task

Moreover, SymTA/S provides also a table view (see Figure 19) of the jitters, in the response time, to be considered during the scheduling phase. The calculated jitters are shown also in another useful view, named “Worst Case Gantt”. The view takes the priorities, the activation times and the jitters into account. The “Worst Case Gantt” shows the worst-case response time of specific tasks you like to check (in our case is the “TaskTimeMedium”, see Figure 20). In particular, it highlights which interferer tasks exist (e.g. tasks with higher priorities) and are responsible for WCRT of the focussed task.

	Load		Output Events			Resource Consumption Time	Response Time		Status
	Total	Execution	Activation	S...	O.	Virtual TCore	Value	Jitter	Status
1 DummyTask_IDLE	0	0	<n/a>	<...	<..	[0 ms;0 ms]	[UNDEFI...FINED]	<n/a>	Success
2 DummyTask_CRUISE	0	0	<n/a>	<...	<..	[0 ms;0 ms]	[UNDEFI...FINED]	<n/a>	Success
3 TaskTimeFast	0.1068	0.1068	P(4 ms)+J(1.6136 ms)	<...	<..	[0 ms;0.4272 ms]	[0 ms;1.6136 ms]	1.6136 ms	Success
4 TaskTimeMedium	0.1191	0.1191	P(12 ms)+J(3.0428 ms)	<...	<..	[0 ms;1.4292 ms]	[0 ms;3.0428 ms]	3.0428 ms	Success
5 TaskTimeSlow	0.0087	0.0087	P(100 ms)+J(3.9128 ms)	<...	<..	[0 ms;0.87 ms]	[0 ms;3.9128 ms]	3.9128 ms	Success
6 TaskTimeVerySlow	0.00008998	0.00008998	P(1000 ms)+J(4.42998 ms)	<...	<..	[0 ms;0.08998 ms]	[0 ms;4.42998 ms]	4.42998 ms	Success
7 TaskPms	0.1483	0.1483	P(8 ms)+J(1.1864 ms)	<...	<..	[0 ms;1.1864 ms]	[0 ms;1.1864 ms]	1.1864 ms	Success
8	<n/a>	<n/a>	<n/a>	<...	<..	<n/a>	<n/a>	<n/a>	<n/a>

Figure 19: SymTA/S Outputs

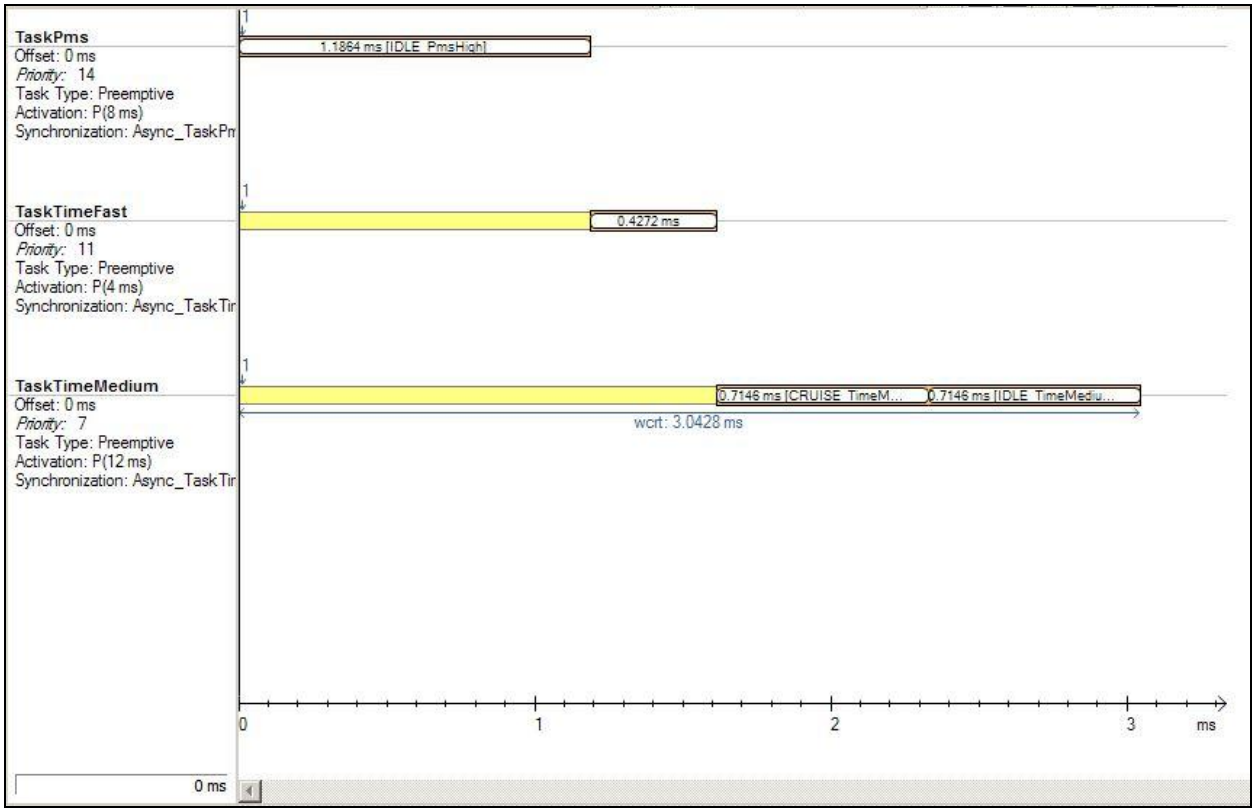


Figure 20: Worst Case Gantt for Task Time Medium

2.5.2 Evaluation

The results obtained using SymTA/S tool to calculate the WCRT analysis highlight two fundamental advantages in term of robustness and reduction of costs in the development process of a product. In particular:

- at design level, to simulate the software architecture model to estimate the WCRT analysis that can be useful for better deployment of CPU load on different tasks or cores,
- at testing level, verifying the CPU load to satisfy system timing requirements to guarantee the correctness of task schedule.

Chapter 3 Evaluation of Automotive Network demonstrator

3.1 Evaluation of the Ethernet Simulator

In the Ethernet simulator implemented in OMNeT++, the focus of the evaluation was on two aspects:

1. Different Ethernet transmission schemes and
2. The effects of the FRER (IEEE802.1CB-2017) protocol

In this section, we present the results of an exemplary system.

3.1.1 Traffic description

The traffic consists of four Ethernet streams, originating on individual ECUs, which are connected to a single switch and being forwarded to the same target. As all streams compete at the connected switch for the port, we focus on the behaviour of the shared switch port for this evaluation.

Each stream is activated in a burst of 140 frames, a minimum intra-burst distance of 200 µs. Their respective periods are chosen so that all frames of each burst get transmitted before the next burst is sent.

3.1.2 Ethernet transmission schemes

The evaluation of a single transmission scheme is performed in the context of the following aspects: stream activation and transmission profiles, end-to-end stream latencies and switch buffer occupancy levels.

3.1.2.1 Weighted Round Robin

Configuration: Stream P4 was assigned a weight of 3, streams P3 and P2 a weight of 2 and stream P1 a weight of 1.

Transmission pattern: As can be seen from the transmission pattern (Figure 21), the transmission of frames from different traffic classes is interleaved. The traffic classes with a higher weight can send a higher number consecutive frames and hence finish the transmission of all frames ahead of streams with a lower weight. Streams with the same weight (P3, P2) finish their transmission roughly at the same time.

End-to-end latencies: Frames of streams with a higher weight are leaving the switch with a higher frequency. Therefore, all of these frames finish their transmission well ahead of other streams. On the other hand, some of the low weight streams are already transmitted alternating with the other traffic classes, hence some of these frames also have very low end-to-end latencies. The results are illustrated in Figure 22.

Buffer occupancy levels: As soon as frames arrive at the analysed switch, they are subject to output arbitration towards their destiation. As some streams have a higher weight than others, more frames of this class are being transmitted. Hence, their buffers effectively do not fill as quickly as for streams with smaller weights. The results are illustrated in Figure 23.

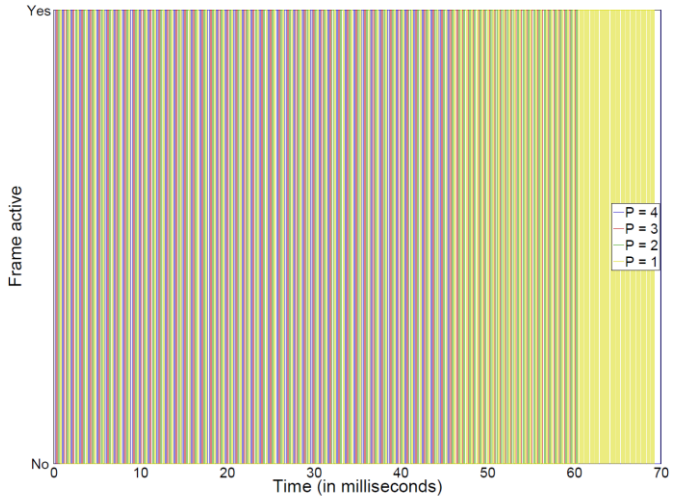


Figure 21: Transmission pattern for WRR

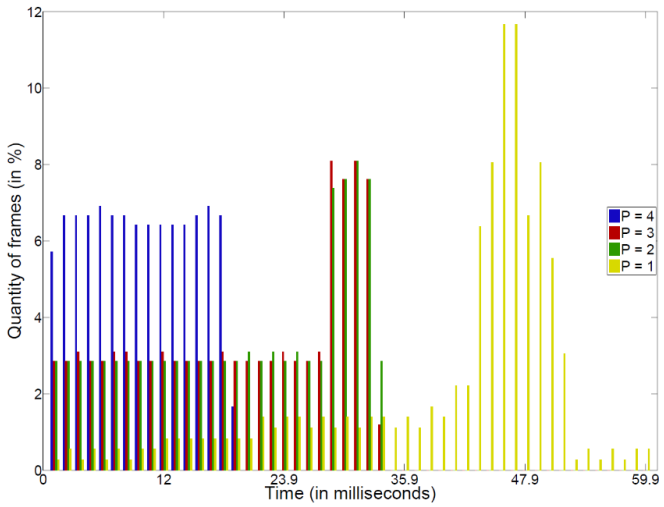


Figure 22: End-to-end latencies for WRR

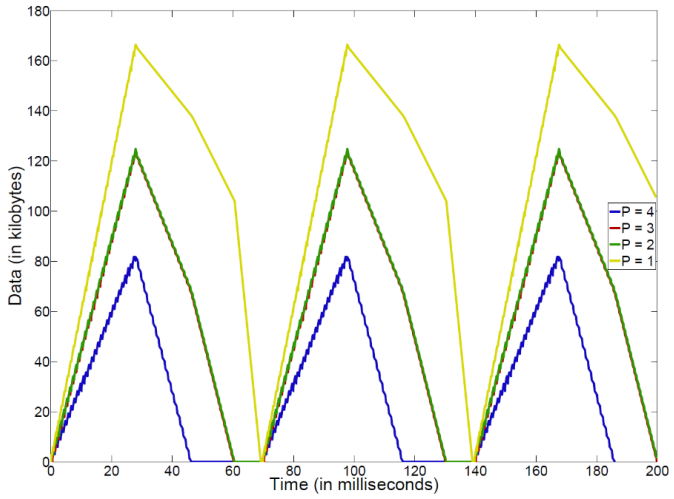


Figure 23: Switch buffer occupancy levels for WRR

3.1.2.2 Static Priority Non Preemptive (SPNP)

Similar to the previous example, there are 4 streams, each with a distinctive priority.

Transmission pattern: If a frame of stream P4 is available, it is transmitted. However, since frames arrive with an intra-burst distance, some P3 frames can be transmitted until the next P4 frame arrives. After all frames of P4 have been transmitted, all backlogged frames of P3 are transmitted, follwed by P2 and P1, i.e. starting with the highest priority and finishing with the lowest. The results are illustrated in Figure 24

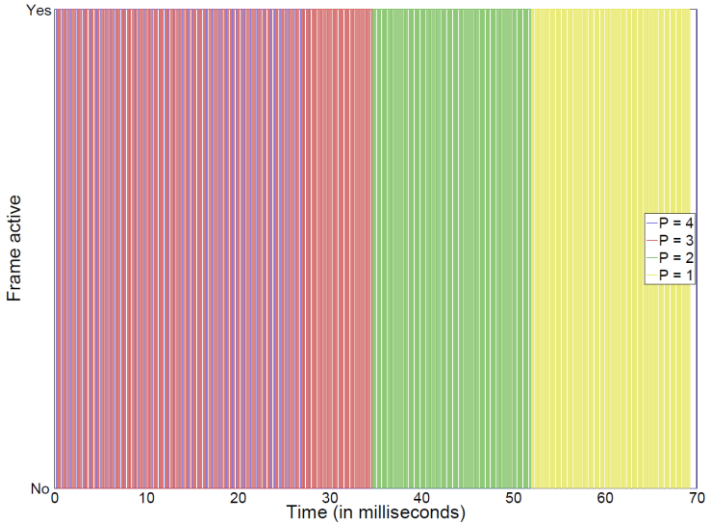


Figure 24: Transmission pattern for SPNP

End-to-end latencies: Frames of P4 only have at most one lower priority blocker to wait for before they are sent, therefore the distribution of end-to-end latencies is very small. The end-to-end latencies of frames of P3 are partially overlapping with the ones of P4 as they are partially transmitted interleaved. The frames of P2 and P1, on the other hand, have to wait for the accumulated time caused by the transmissions of all other frames with higher priorities. The results are illustrated in Figure 25.

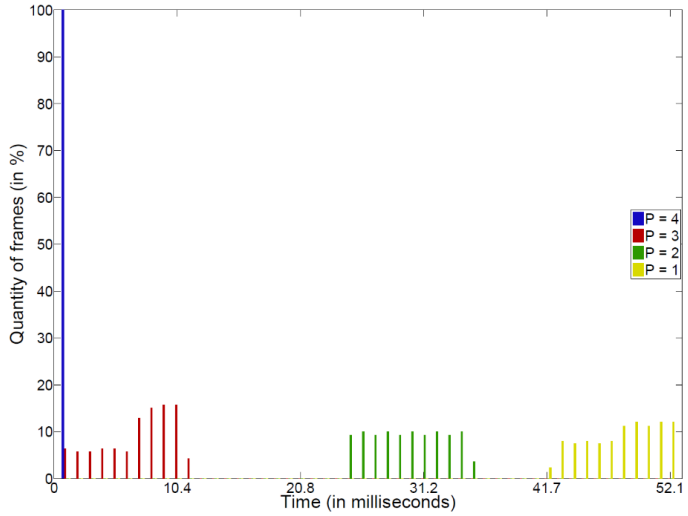


Figure 25: End-to-end latencies for SPNP

Buffer occupancy levels: Frames of P4 are transmitted as soon as they arrive and the switch port is free. As there is an intra-burst distance, some frames of P3 can be transmitted interleaved with P4. Streams of P2 and P3, however, remain in the buffer until all their respective higher priority streams have been transmitted. The results are illustrated in Figure 26.

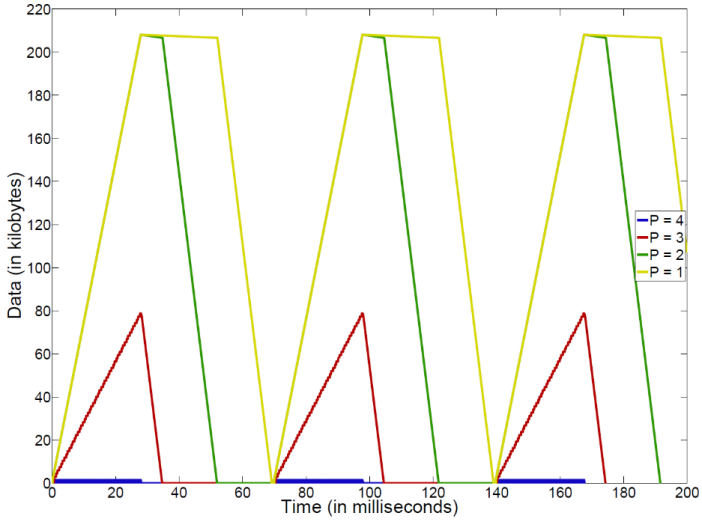


Figure 26: Buffer occupancy levels for SPNP

3.1.2.3 Credit based shaper (CBS/AVB)

The same set of streams from the previous experiments is used. Configuration of shapers is as follows: Bandwidth (P4) = 25%, Bandwidth (P3) = 25 %, P2 and P1 unregulated.

Transmission pattern: For the first half of the transmission interval the streams P4, P3 and P2 are transmitted alternating. In the second half, the streams P4, P3 and P1 are transmitted alternating. This is due to the fact, that P4 and P3 each have a quarter of the bandwidth available. As P2 has a higher priority than P1 it will take all the remaining available bandwidth (50%). Only after P2 has finished its transmission, any bandwidth is available for P1. The results are illustrated in Figure 27.

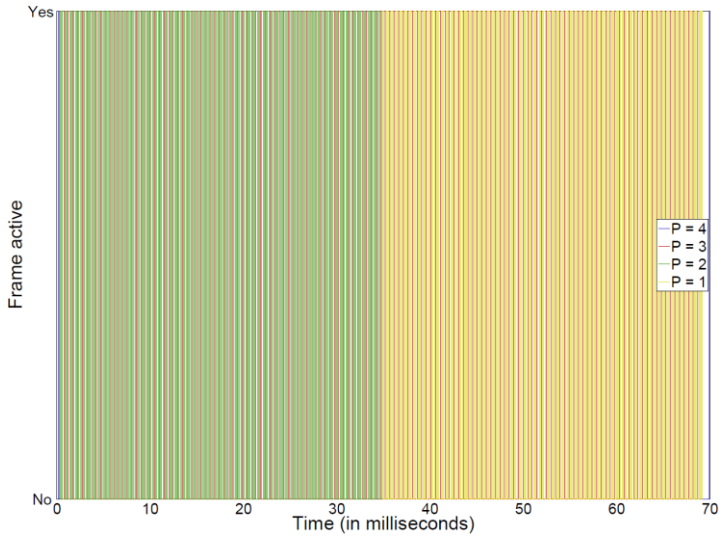


Figure 27: Transmission pattern for AVB

End-to-end latencies: As frames of stream P2 leave the switch with a higher frequency, their end-to-end latencies are lower than those of frames of any other stream. Streams P4 and P3 have a constant 25% bandwidth assigned and therefore transmit evenly until the end of the transmission period. The results are illustrated in Figure 28.

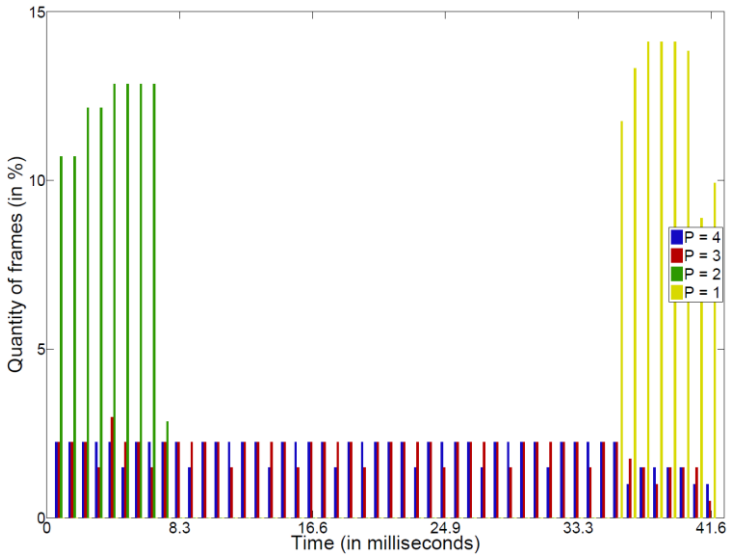


Figure 28: End-to-end latencies for AVB

Buffer occupancy levels: The streams with the highest priorities (P4 and P3) are limited to 25% of the bandwidth respectively, while the remaining 50% are used by the next highest priority, i.e. P2. Hence, P2 sends more frames per time interval and finishes its transmission before streams which have a higher priority but are shaped. The results are illustrated in Figure 29.

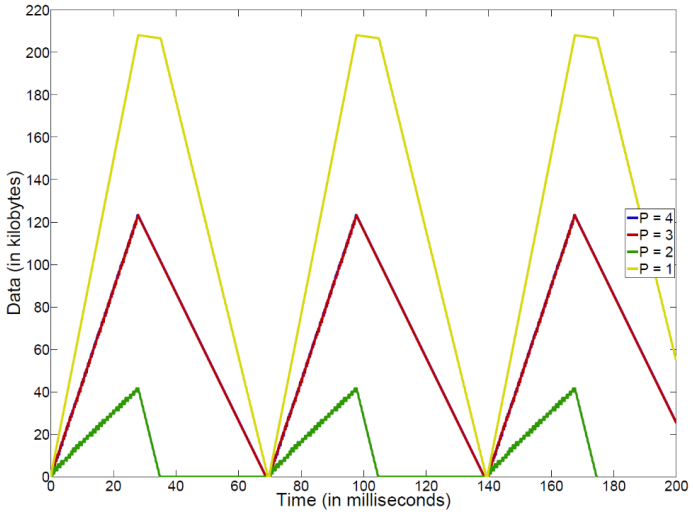


Figure 29: Buffer occupancy levels for AVB

As it can be observed from the evaluation results, different transmission schemes have their adventages and limitations with respect to end-to-end latencies, buffer occupancy levels, work-conservation property etc. Therefore, the preference and the choice of one method over the other is highly dependent on the use-case. The main conclusions are briefly summarised in Table 5.

Table 5: Advantages and limitation of transmission schemes

Arbitration scheme	Positive sides	Negative sides
SPNP	Strict prioritisation	High E2E latencies of low-priority streams
	Low E2E latencies of high-priority streams	Higher buffer requirements than WRR for low-priority streams
	Amenable to real-time analysis	High critical traffic must be trusted
	Suitable to periodic, control traffic with tight timing requirements	Susceptible to misbehaving traffic on high priorities
	Work-conserving	
WRR	Comparable E2E latencies for all streams	Limited possibilities to prioritise traffic
	Lower buffer requirements than SPNP	Less amenable to real-time analysis than SPNP
	Suitable to traffic with similar criticality and lose timing	

Arbitration scheme	Positive sides	Negative sides
	requirements	
	Resilient to misbehaving of individual streams	
	Easy implementation	
	Work-conserving	
AVB	Highly configurable approach, can mimic both SPNP and WRR	Analytically complex, less amenable to real-time analysis than SPNP
	CAN achieve both prioritisation and equality if necessary (via proper configuration)	Performance heavily depends on the proper configuration, and misconfiguration can lead to poor performance
		Non-work-conserving

3.1.3 FRER Protocol

In this section, the evaluation of the FRER mechanism is presented. The following FRER configurations are being evaluated.

1. **Baseline system**: No FRER protection, hence no replication, nor elimination of frames.
2. **Temporal FRER**: Redundant frame copies are sent via the same path in succession. The replication and elimination is performed in each traversed switch. This configuration does not require redundant hardware.
3. **Spatial FRER**: Each traversed network element (switch, gateway) is duplicated, forming a twin network including identical connecting links. After replication, each frame copy is sent via a different network copy to the receiver, which automatically removes redundant copies. This approach requires additional hardware.
4. **Spatial+Temporal FRER**: This configuration is a combination of the two previously mentioned approaches. The network is duplicated the same way as in Spatial FRER and each network element transmits two redundant frame copies as in Temporal FRER.

The evaluation is performed for two distinctive scenarios: (i) there are no transmission errors, and (ii) there are transmission errors which comply with the selected BER (bit-error-rate) value, applied to the transmission of each frame.

End-to-end latencies: In this experiment, we evaluated the impact of different FRER configurations on end-to-end latencies of streams. It is visible from Figure 30 that the temporal component of FRER inflates the traffic resulting in increased end-to-end latencies. The only spatial variant suffers no protocol overhead but requires additional hardware.

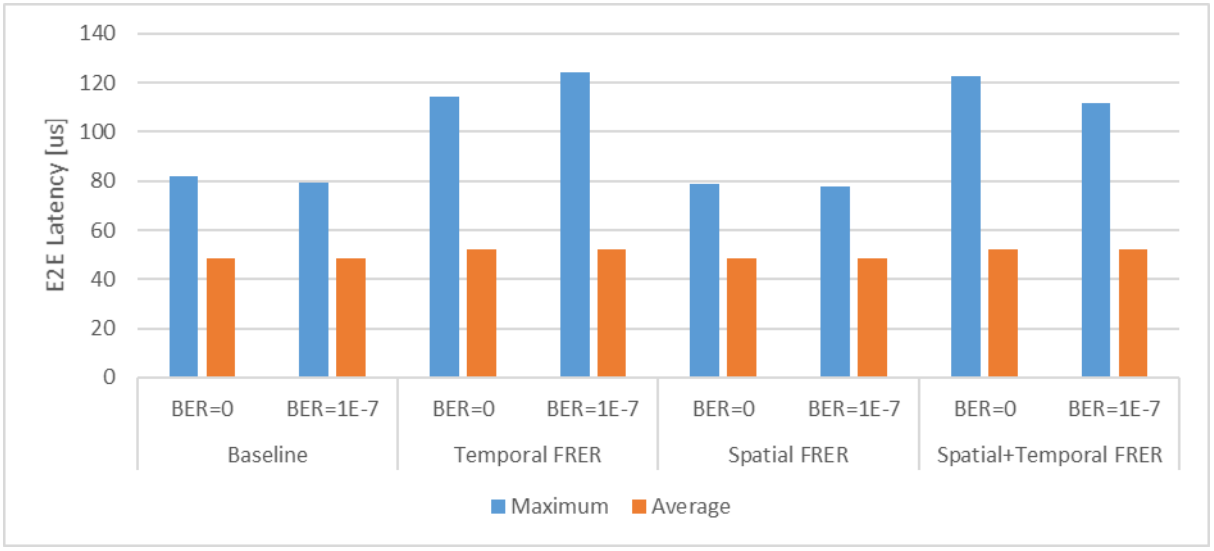


Figure 30: End-to-end latencies under different FRER configurations

An alternative presentation of end-to-end latencies is also available in Figure 31.

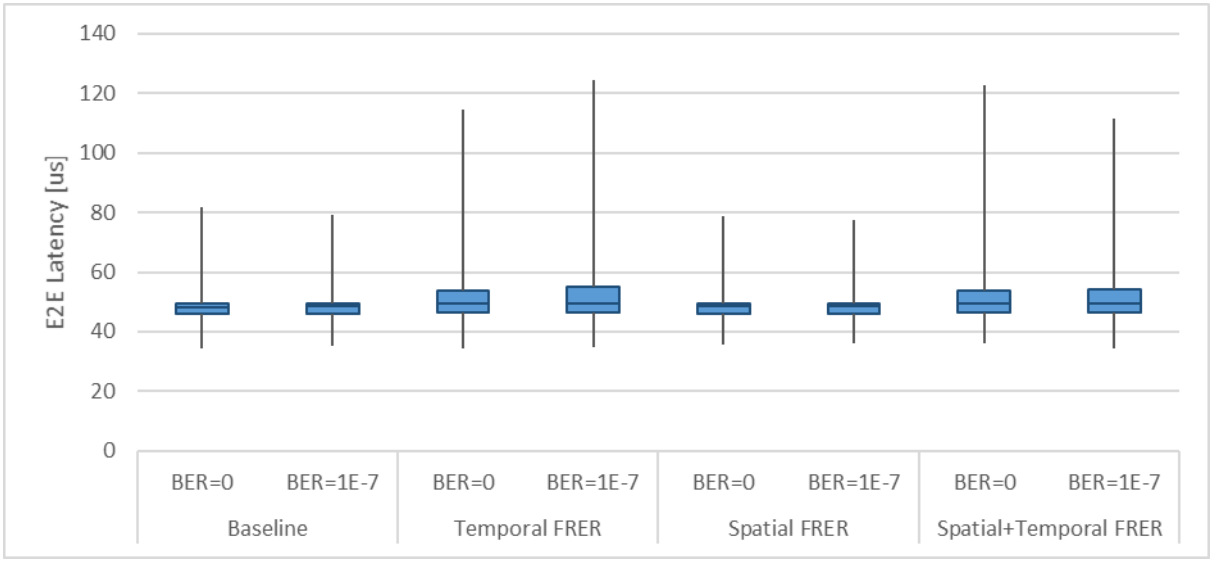


Figure 31: End-to-end latencies under different FRER configurations

Buffer occupancy levels: Of interest are the average and the maximum buffer utilisations for the following configurations: (i) No protection (Baseline), (ii) Spatial FRER (with the end-to-end replication/elimination), (iii) Temporal FRER (with the hop-by-hop replication/elimination) and (iv) Spatial + Temporal FRER. Moreover, for each configuration we have analysed two scenarios, without errors (BER = 0) and with errors (BER = 1E-7).

The results are illustrated in Figure 32. It is visible that in all analysed scenarios there is a significant difference between the average and the maximum utilisation. This is because in majority of cases the frames are served as soon as they arrive, so on average the buffers do not hold more than a single frame. This is the consequence of a sufficient capacity of downstream links, which successfully cope with the traffic which is produced in the analysed scenarios. The maximum buffer utilisations are reached in cases where contentions occur,

due to concurrent arrival of multiple packets from different input ports. In such cases, frames from one input direction have to be stalled, and any subsequently arriving frames from the same direction will be queued behind them. Nonetheless, due to the sufficient capacity of downstream links, the contentions for the output port are always quickly resolved, which is evident from the fact that only few frames get queued (at most 2 camera frames in the baseline and the spatial FRER cases, and at most 4 camera frames in the temporal and temporal + spatial cases). Moreover, it is visible that the schemes which include the temporal replication require bigger buffering requirements (twice as much space to store the queued frames). This is expected because these schemes produce double the load. Finally, it is evident that the presence of errors does not have an effect on the buffer occupancy levels, simply because the FRER mechanism is error-agnostic. Note, that this is not the case for the ARQ approach, which is error-sensitive.

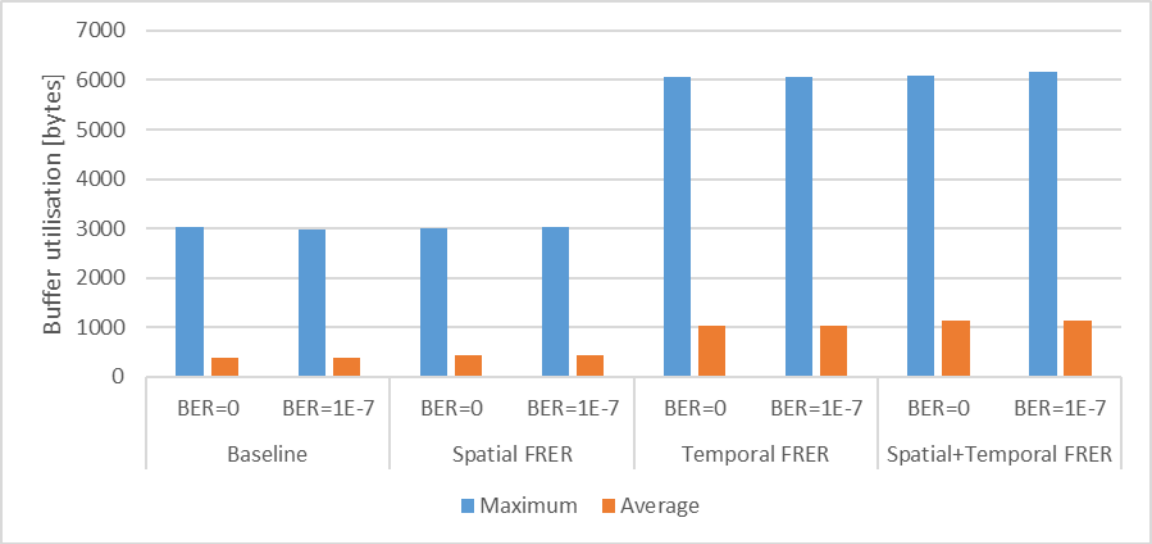


Figure 32: Buffer occupancy levels under different FRER configurations

Link utilisation: Of interest are the average and the maximum link utilisation for the following configurations: (i) No protection (Baseline), (ii) Spatial FRER (with the end-to-end replication/elimination), (iii) Temporal FRER (with the hop-by-hop replication/elimination) and (iv) Spatial + Temporal FRER. Moreover, for each configuration we have analysed two scenarios, without errors (BER = 0) and with errors (BER = 1E-7). The results are given in Figure 33. It is visible that in all analysed scenarios the average and the maximum link utilisation are very similar. This is because of the nature of the analysed traffic. Specifically, all traffic sources produce and send the data in a periodic manner, which results in a constant supply of frames into the analysed switch. At the same time, the capacity of the analysed link is sufficient to handle the produced traffic, so no network congestion, nor bottlenecks are created. The traffic in the scenarios with the temporal replication is twice as in scenarios without it (440 Mb/s and 220 Mb/s, respectively), which is expected. Nonetheless, as mentioned, both these desired link capacities are successfully met with the analysed link, with the capacity of 1Gb/s. However, in scenarios where the link capacity is limited and cannot cope with the doubled amount of traffic, temporal replication may not be a desired approach. Finally, it is visible that scenarios with and without errors have the same results, which again confirms the previous observations that FRER approaches are error-agnostic.

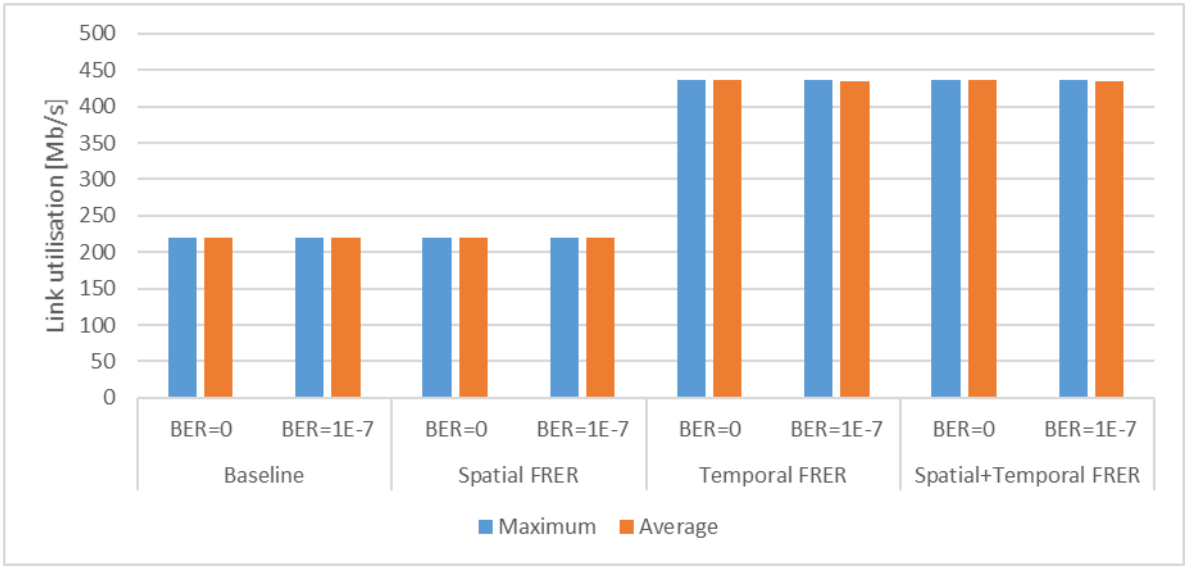


Figure 33: Link utilisation under different FRER configurations

The findings regarding the applicability of different FRER mechanisms are summarised in Table 6.

Table 6: Advantages and limitation of different FRER configurations

Evaluated scheme	Positive sides	Negative sides
Temporal FRER	Reduced packet loss rate	Doubles number of transmitted frames
	No additional hardware required	Doubles required buffer space & bandwidth
	Easy to configure	Increased E2E Latencies
		No protection against permanent faults
Spatial FRER	Reduced packet loss rate	Significant hardware overhead
	No increase in buffer and bandwidth requirements	
	No increase in E2E latencies	Requires explicit switch configuration
	Provides protection against permanent faults	
Spatial+Temporal FRER	All positive aspects from Spatial & Temporal FRER	Significant hardware overhead
	Offers best protection against transient and permanent faults	Explicit switch configuration required
		Increase E2E latencies
		Doubles required buffer space & bandwidth

3.2 Evaluation of Demonstrator System in SymTA/S

The main metrics to evaluate any Ethernet systems including the automotive demonstrator (and the virtual demonstrator mentioned in 6.5) in SymTA/S were the main SAFURE project results of Symtavision. This includes a worst case data rate analysis for Ethernet messages, a worst case Ethernet port load analysis, a worst case response time analysis for Ethernet messages and a worst case buffer fill level analysis for ports and switches. All analyses come with additional charts to simplify the viewing of result data for the customer. In the following sections all four analyses were executed on the second demonstrator to evaluate it. They show the resulting numeral values and charts for the implemented Ethernet schedulers. Even if the SPNP (Static Priority Non-Preemptive) and AVB (Audio/Video Bridging) scheduler were implemented, the evaluation of the demonstrator concentrates to SPNP. Evaluation of the AVB would work in a similar way.

3.2.1 Data Rate

In Figure 34 the calculated data rates for the traffic elements (Ethernet messages) of the demonstrator system are shown. Even if only the first ten Ethernet messages are shown, they are representative for the whole demonstrator system. The data rate is basically the quotient of message transmission frequency and message size. Even with all 914 Ethernet messages the network manages all this data rates very well and is not overloaded.

List of all Ethernet Messages

	Element	Data Rate		
	Name	Total	Execution	Schedul...erhead
1 EthernetMessage#1	EthernetMessage#1	67.2 kbit/s	57.6 kbit/s	9.6 kbit/s
2 EthernetMessage#2	EthernetMessage#2	67.2 kbit/s	57.6 kbit/s	9.6 kbit/s
3 EthernetMessage#3	EthernetMessage#3	12.218...bit/s	10.472...bit/s	1.7454...bit/s
4 EthernetMessage#4	EthernetMessage#4	838.8 kbit/s	831.6 kbit/s	7.2 kbit/s
5 EthernetMessage#5	EthernetMessage#5	838.8 kbit/s	831.6 kbit/s	7.2 kbit/s
6 EthernetMessage#6	EthernetMessage#6	838.8 kbit/s	831.6 kbit/s	7.2 kbit/s
7 EthernetMessage#7	EthernetMessage#7	58 kbit/s	55.6 kbit/s	2.4 kbit/s
8 EthernetMessage#8	EthernetMessage#8	57.8 kbit/s	55.4 kbit/s	2.4 kbit/s
9 EthernetMessage#9	EthernetMessage#9	67.4 kbit/s	65 kbit/s	2.4 kbit/s
10 EthernetMessage#10	EthernetMessage#10	58 kbit/s	55.6 kbit/s	2.4 kbit/s

Figure 34: Worst Case Data Rate of Ethernet Messages of the Demonstrator

3.2.2 Ethernet Port Load

The Ethernet port load analysis basically shows the load at the data-transmitting Ethernet ports. In Figure 35 the Ethernet ports with the highest load in the network are shown. As you can see, the port with the highest load exceeds 80%, which is quite high for one port. Due to the fact that 80% is a suggested “virtual” maximum value for load, the customer should reconfigure the network to reduce the load for this port. Which “adjusting screw” the customer can take use is mentioned in D6.5 [49].

Even the second and third ports (regarding load) are quite at the maximum and should be observed and not grow anymore.

*List of all Ethernet Ports

	Element	Egress Parameters	Load		
	Name	Port Transmit Rate	Total	Execution	Scheduli...verhead
1 EthernetPort#24	EthernetPort#24	100 Mbit/s	0.8391155151515152	0.8240623272727273	0.01505...787879
2 EthernetPort#34	EthernetPort#34	100 Mbit/s	0.7915936484848485	0.7765319272727273	0.01506...121212
3 EthernetPort#28	EthernetPort#28	100 Mbit/s	0.7905237777777778	0.7768654666666667	0.01365...111111
4 EthernetPort#20	EthernetPort#20	100 Mbit/s	0.7046201333333333	0.6909684	0.01365...333333
5 EthernetPort#16	EthernetPort#16	100 Mbit/s	0.6009016888888889	0.5909189333333333	0.00998...555556
6 EthernetPort#31	EthernetPort#31	100 Mbit/s	0.5894515555555556	0.5788277333333333	0.01062...222222
7 EthernetPort#30	EthernetPort#30	100 Mbit/s	0.4622276	0.4528564	0.0093712
8 EthernetPort#14	EthernetPort#14	100 Mbit/s	0.3966097333333333	0.3903724	0.00623...333333
9 EthernetPort#15	EthernetPort#15	100 Mbit/s	0.3820272	0.3737952	0.008232
10 EthernetPort#29	EthernetPort#29	100 Mbit/s	0.3709269818181818	0.3639495272727273	0.00697...454545

Figure 35: Worst Case Load of transmitting Ethernet Ports of the Demonstrator with highest Load

In Figure 36 you can see the load of all switches in the network as a bar chart, which we included in the tool during the SAFURE project. In this case it gives an overview over all switches and the distribution of the load in the network and helps to detect bottle necks.

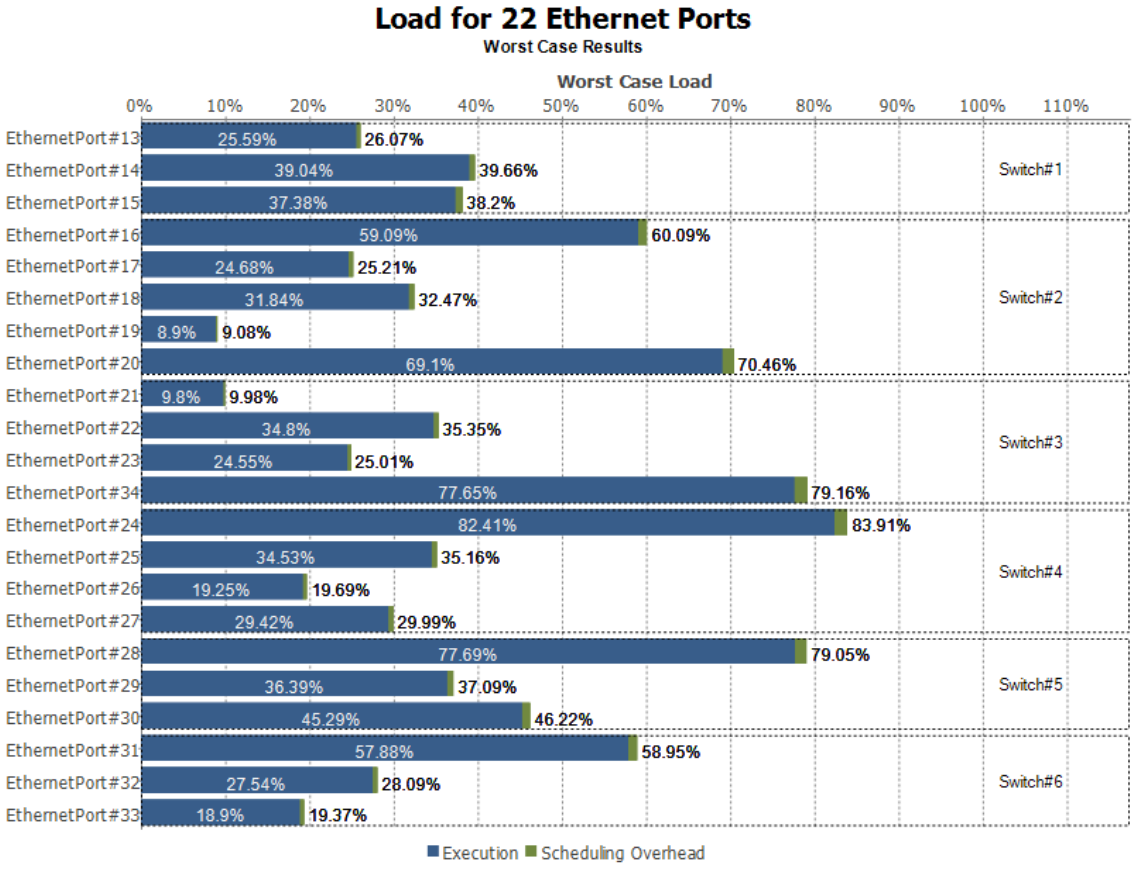
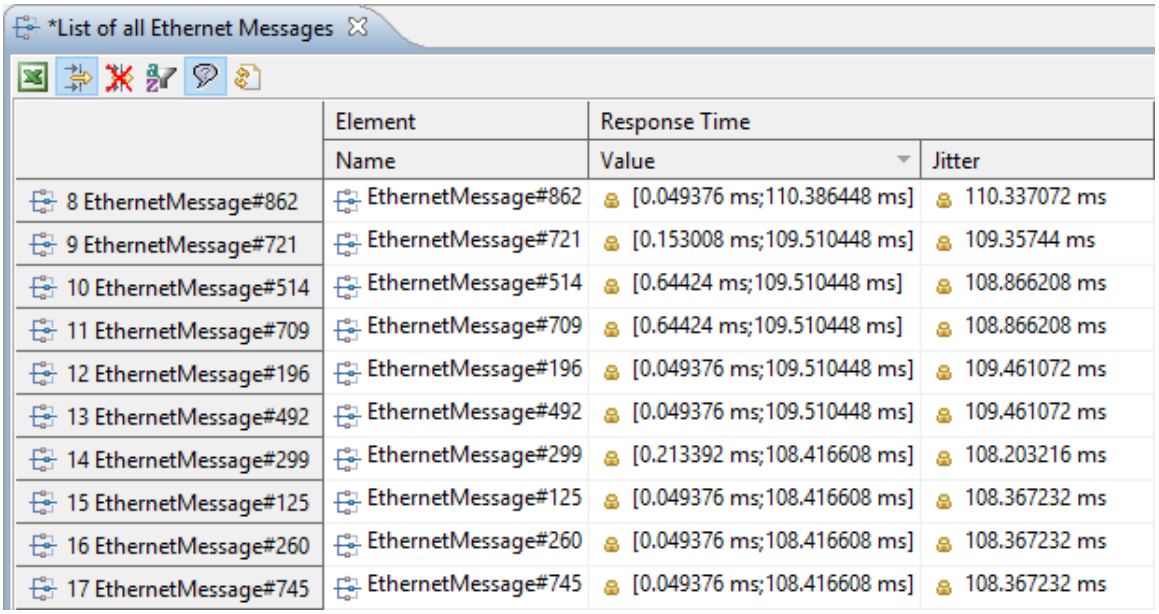


Figure 36: Worst Case Load of all Switches of the Demonstrator

3.2.3 Worst Response Time (Latency)

In Figure 37 you can see the ten Ethernet messages with the highest response time (latency) of the demonstrator system. The response time is given as an interval, the best and the worst-case. Usually the response time is a value somewhere between these intervals. The worst-case response time is quite high for the shown ten Ethernet messages. So, this message should not transport any time critical data. But if this would be the case the traffic configuration should be reconfigured again to fit shorter deadlines than 100 ms. See D6.5 [49] for more details what is possible to reconfigure here.


*List of all Ethernet Messages

	Element	Response Time	
	Name	Value	Jitter
8 EthernetMessage#862	EthernetMessage#862	[0.049376 ms;110.386448 ms]	110.337072 ms
9 EthernetMessage#721	EthernetMessage#721	[0.153008 ms;109.510448 ms]	109.35744 ms
10 EthernetMessage#514	EthernetMessage#514	[0.64424 ms;109.510448 ms]	108.866208 ms
11 EthernetMessage#709	EthernetMessage#709	[0.64424 ms;109.510448 ms]	108.866208 ms
12 EthernetMessage#196	EthernetMessage#196	[0.049376 ms;109.510448 ms]	109.461072 ms
13 EthernetMessage#492	EthernetMessage#492	[0.049376 ms;109.510448 ms]	109.461072 ms
14 EthernetMessage#299	EthernetMessage#299	[0.213392 ms;108.416608 ms]	108.203216 ms
15 EthernetMessage#125	EthernetMessage#125	[0.049376 ms;108.416608 ms]	108.367232 ms
16 EthernetMessage#260	EthernetMessage#260	[0.049376 ms;108.416608 ms]	108.367232 ms
17 EthernetMessage#745	EthernetMessage#745	[0.049376 ms;108.416608 ms]	108.367232 ms

Figure 37: Worst Case Latency of Ethernet Messages of the Demonstrator with highest Latency

3.2.4 Buffer Fill Level

In Figure 38 the maximum buffer fill levels of all switches of the demonstrator are shown. Even if the highest load is produced on “Swtich#4”, the highest buffer fill level is observed on “Switch#2”. This could have different reasons. Mostly this depends on more density in time MAC frame arrivals at the switches buffer. So, called bursts lead often to more buffer usage at switches, but do not necessarily have an influence on the load of the Ethernet ports. Burst situations can be prevented by the Ethernet AVB scheduler, which introduced shaping at the transmission Ethernet ports.

List of all Switches

	Element		Switch	Buffer Fill Level
	Name	Parents	Latency	Maximal [Bytes]
1 Switch#1	Switch#1	Etherne...work#1	[4 µs;4 µs]	118980
2 Switch#2	Switch#2	Etherne...work#1	[4 µs;4 µs]	679978
3 Switch#3	Switch#3	Etherne...work#1	[4 µs;4 µs]	581973
4 Switch#4	Switch#4	Etherne...work#1	[4 µs;4 µs]	310429
5 Switch#5	Switch#5	Etherne...work#1	[4 µs;4 µs]	195301
6 Switch#6	Switch#6	Etherne...work#1	[4 µs;4 µs]	93454

Figure 38: Worst Case Buffer Fill Levels of Switches of the Demonstrator

Chapter 4 Evaluation of Combined Automotive Prototype

The goal of the combined prototype is to guarantee a secure real time communication between more nodes located inside a car that are connected together using different protocols, in this case we have used Can and Ethernet. So, the proposal solution guarantees a real time secure inter-communication inside a vehicle system in the automotive industry.

In this chapter we will describe the whole equipment involved in this demonstrator and how we have implemented the man in the middle attacks to show that the malicious attacks are recognized and discarded.

4.1 Test environment

As described in the D6.5 [49], in the automotive demonstrator we have combined the multicore control unit with the network use case. This has been possible thanks to the introduction of the CAN-Ethernet Gateway provided by TTTech. In fact, this gateway has the purpose to convert CAN messages in Ethernet messages and vice versa (refer to D6.5 [49] for more details on this conversion).

So, the combined use case scenario consists of the following main parts (see Figure 39):

- MAG multicore control unit connected to Hw Hermes Gateway provided by TTTech via CAN line which is able to send and receive CAN messages. In order to test and verify the messages in the ECU, we have introduced a Lautherbach emulator and to stimulate the ECU, the usage of a static simulator of the engine was been necessary.
- Hermes Gateway communicates via CAN to the multicore control unit and via Ethernet to the end system (a TTTech PC). In particular, a custom cable (Hermes-to-CAN/UART (RS232)) is made for the switch by TTTech.
- TTTech's Project PC, where two software services are implemented that send and receive Secure CAN messages (Categories A and B – cf. chapter 4.2), and also the clean CAN messages. The services use the same algorithms integrated and used in the multicore ECU and the same shared key.

Moreover, to implement the tests on this demonstrator we have included the usage of a laptop whit installed Wireshark tool [42] as Ethernet protocol analyser. This laptop has been connected with a standard cat5e cable to the gateway in order to sniff the Ethernet messages (see Figure 41 and Figure 42). Instead, we have monitor the CAN messages using the CAN Analyzer connected to the CAN line. The whole equipment involved in this test's scenario is visible in the picture of Figure 40.

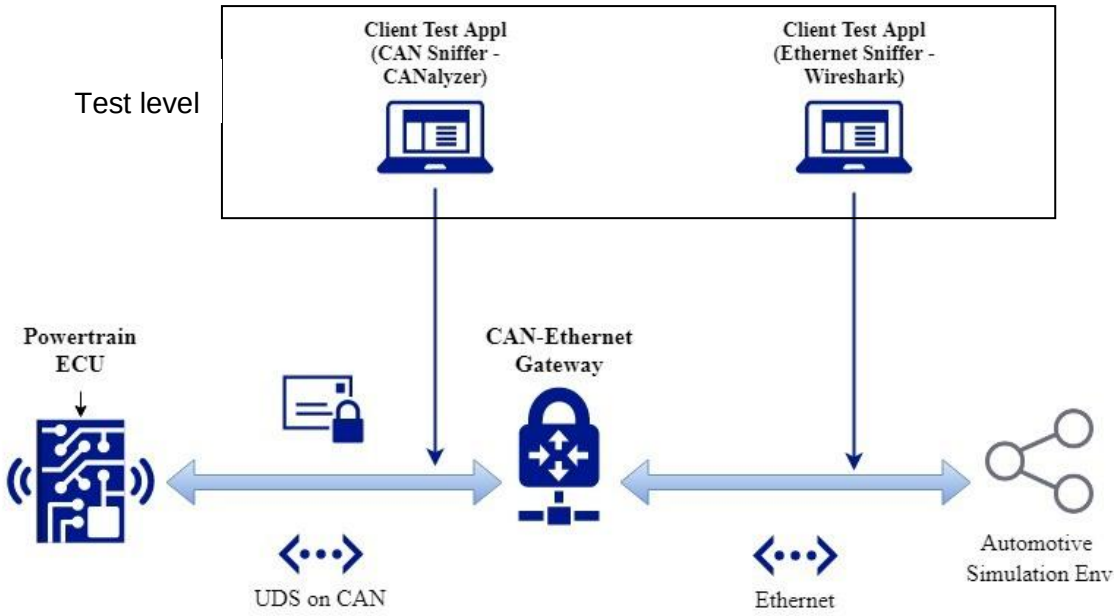


Figure 39: Combined Automotive Scenario

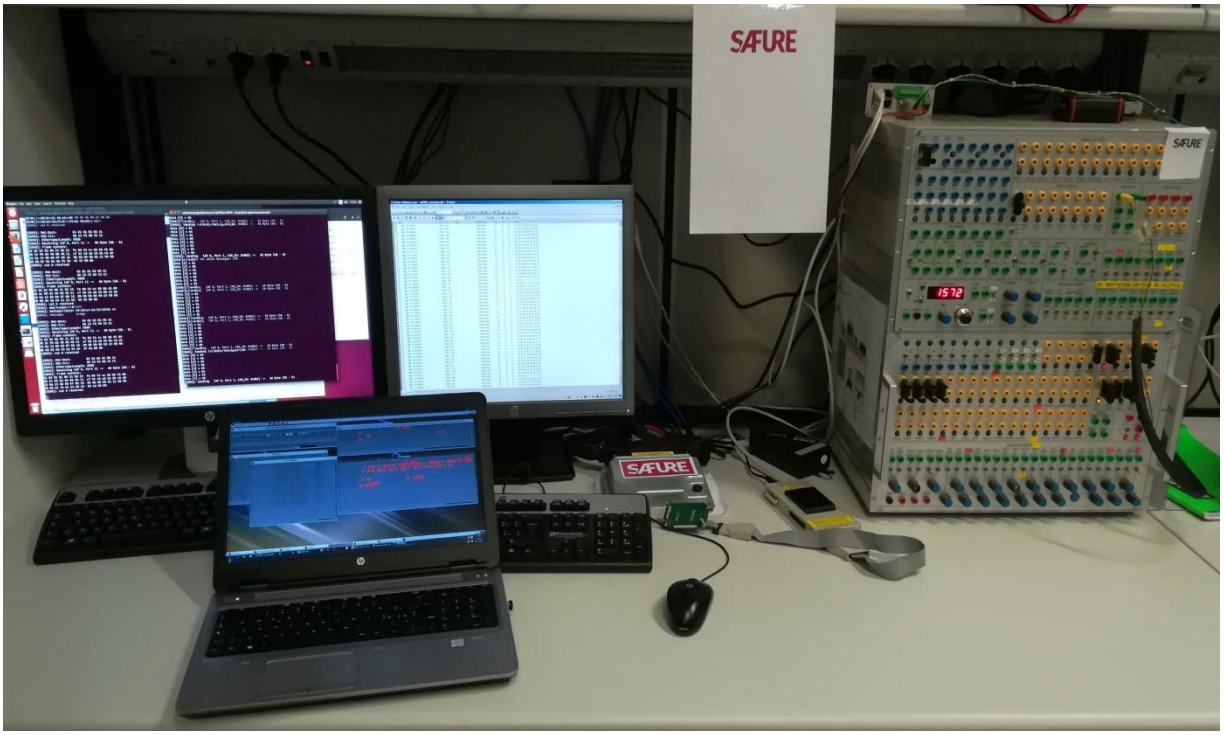


Figure 40: Full equipment of combined demonstrator

4.2 “Man-in-the-middle” Tests

The tests made in this scenario have had two main purposes:

1. checks that the gateway was be able to convert the messages in real-time without corrupt them;
2. the corrupted Ethernet messages era recognized and discarded by the powertrain control unit.

To verify the first purpose, we have introduced two protocol sniffers: one for Ethernet (i.e. Wireshark) and the other for CAN (i.e. CANalyzer).

Figure 41 and Figure 42 show the traces of the Ethernet protocol and the relevant information for our tests are the ‘ID’ into the message (‘A7’ for Category B and ‘104’ for Category A), because it represents which node is sending, the index of the frame (in Figure 41 is highlighted in yellow the ‘02’ for a message and in the other you can read, in the same position ‘01’) and the content of the messages.

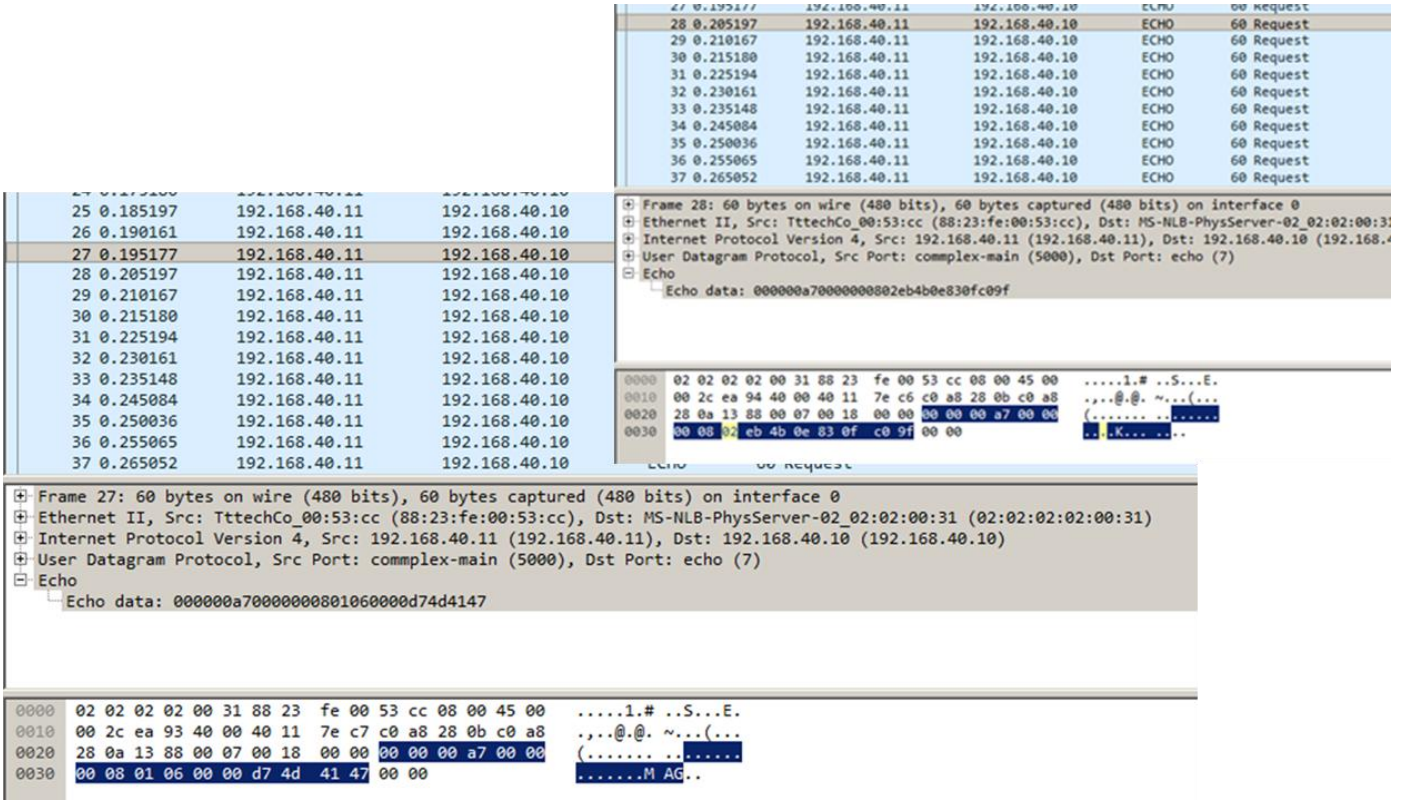


Figure 41: Wireshark view (message Cat B)

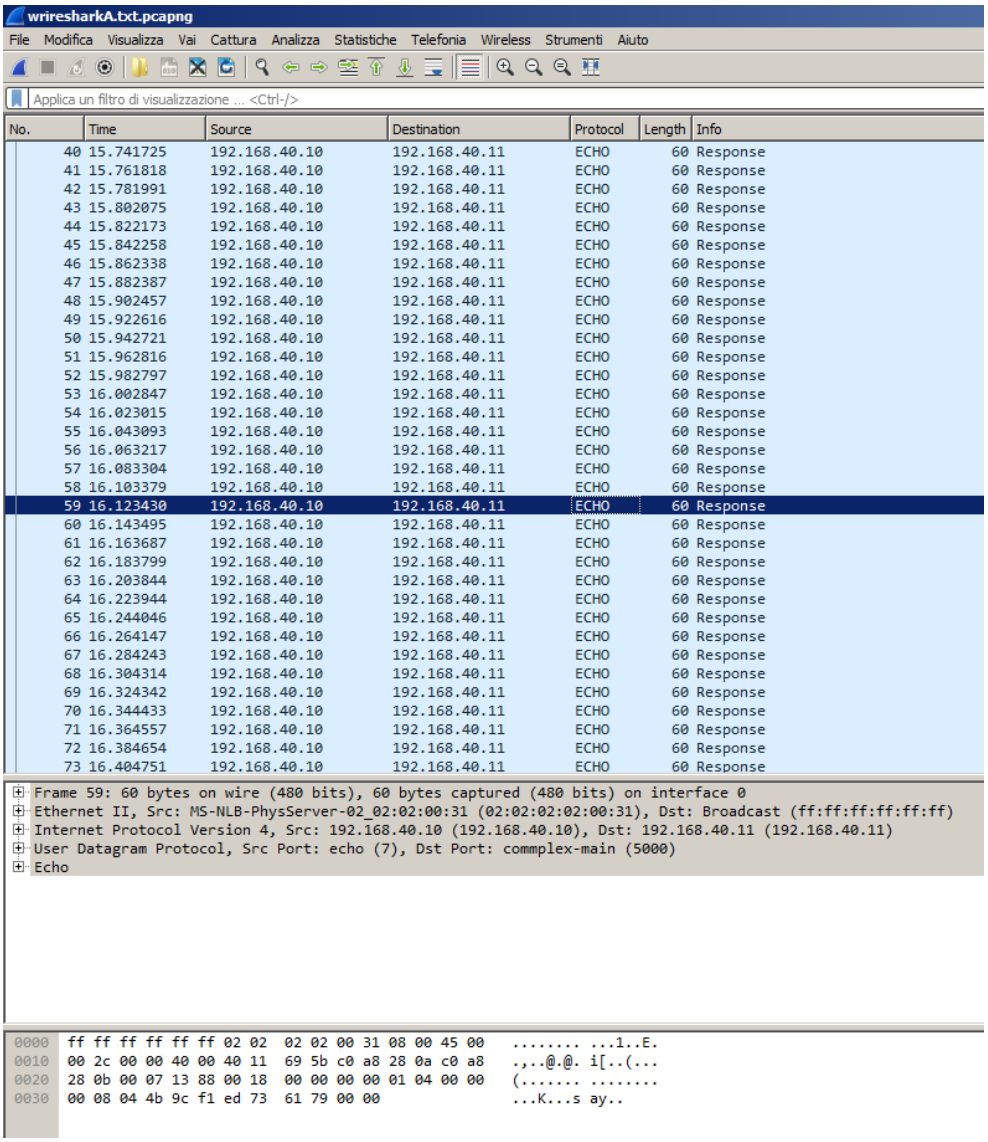


Figure 42: Wireshark view (message Cat A)

To test the second purpose, we have followed the same test strategy implemented to test the control unit on CAN line (see Chapter 2.2.1). The difference here is that from the TTTech's project PC, we have sent and verified the Ethernet messages that are converted by the HW gateway (without to do any secure checks) into CAN messages and we have verified the incremented counter in the control unit side, using the Lauterbach emulator [43].

In D6.5 [49] are showed the received and the sent message's mechanisms from the TTTech's project PC side.

Chapter 5 Requirements coverage

According to the deliverables D1.2 [35] and "D1.2_Improvement", we have reported in this chapter the requirements related to the automotive use case. The structure of D1.2 [35] is maintained here.

Just two clarifications, the requirements S1-NF-003, S1-NF-004, S1-NF-005 are moved from the telecommunications use case to the automotive use case, according to the document "SAFURE-D4.2-delay-justification-M24-V2". Moreover, in this document we do not report the tables of requirements, which have been already integrated into SAFURE project at the time of the delivery "SAFURE-D1.2-PU-M06_Improvement".

5.1 Common Requirements

5.1.1 Functional Requirements

Table 7: Common Functional Requirements for All Scenarios

ID	Description of Requirements	Comments	Coverage
CR-F-001	Mixed-critical safety requirements and time critical requirements need to be coupled in at least one of the use-case supporting PikeOS, including the possibility to run concurrently different tasks with different safety levels, or the ability to support a degraded mode for lowest critical tasks.	Requirement for the research performed in WP4. Else WP4 will use a dedicated prototype. Integrated in the WP4 prototype.	Refer to D4.3.
CR-F-002	The use-cases should quantify their usage and requirements in term of accesses to the different shared hardware resources of the target platforms for the adaptive solution to guarantee the associated requirements based on observed behaviour.	Requirements for QoS algorithm developed in WP3.	Refer to D3.2, Chapter 4.5.

5.1.2 Non-functional Requirements

The coverage column of Table 8 in this deliverable is based on automotive Demonstrator. The same requirements are listed also in the deliverable D6.4 [51] concerning the Telecom demonstrator.

Table 8: Common Non-Functional Requirements for All Scenarios

Type	ID	Description of Requirements	Comments	Coverage
Real-Time Operating System	CR-NF-002	All the use cases should use tools and SW that are an expression of an acknowledged standard or have a reliable open source implementation		For the Automotive demonstrator ERIKA OS is used.
Time analyses	CR-NF-005	System description (topology, etc.) must be available in an accessible format	Applies to all use cases for which timing analysis shall be performed.	See the Chapters 2.5 and 3.2.
	CR-NF-006	System configuration (communication, tasks, etc.) and timing properties (execution times, frame sizes, etc.) must be available in an accessible format	Applies to all use cases for which timing analysis shall be performed.	See the Chapters 2.5 and 3.2.
	CR-NF-007	System constraints (deadlines, max. load, etc.) should be available in an accessible format	Applies to all use cases for which timing analysis shall be performed.	See the Chapters 2.5 and 3.2.
	CR-NF-008	Timing behaviour must be known/specied for all arbitration points (CPU scheduler, network arbitration, shared resource access, etc.)	Applies to all use cases for which timing analysis shall be performed.	See Chapter 2.4.
	CR-NF-009	For unknown time consumers (attackers), constraints should be specified (e.g. what resources are affected).	Applies to all use cases for which timing analysis shall be performed.	See Chapter 2.4.
	CR-NF-010	Standard arbitration protocols should be used for OS and networks (e.g. AUTOSAR, OSEK, Ethernet).	There will likely be no support from SYM for non-standard / custom protocols for timing analysis.	The evaluation of the automotive network includes all relevant

Type	ID	Description of Requirements	Comments	Coverage
				standard arbitration schemes defined in the IEEE802.1 Q and IEEE802.1 Qbv standard.
	CR-NF-011	Timing properties should be derived via tracing, static analysis or budgeting.	Applies to all use cases for which timing analysis shall be performed.	For multicore control unit see Chapters 2.2.2.1 and 2.5. For Network demonstrated see Chapter 3.2.
	CR-NF-012	WCET analysis techniques and dedicated isolation techniques should provide Time Composability in target multicore systems by providing features allowing us to compute or bound the co-running interference overhead.		In D6.5 (Chapter 2.5) is explained in details the mechanism that, when the runnable will exceed the WCET provided, this trigger the AUTOSAR timing protection mechanisms (as implemented in the Erika open source operating system [15]).
Security	CR-NF-015	The hypervisor shall support		Refer to

Type	ID	Description of Requirements	Comments	Coverage
		secure boot of the whole system and each partition separately.		D4.3 [47], Chapter 4.2.
	CR-NF-016	The hypervisor shall provide secure update of a partition.		Refer to D4.3 [47], Chapter 4.3 and D5.2 [52], Chapter 2.3.2.
	CR-NF-018	The SAFURE platform must provide services for cryptographic mechanisms and handle cryptographic objects (i.e. keys, certificates). The services must include the following features: a) Managing cryptographic keys. (Generating, deleting and storing keys) b) Calculation of cryptographic functions: - Signature generation and verification - Message Authentication Codes (MACs) - Encryption and decryption c) Management of cryptographic certificates. (Storing and updating certificates)	This requirement needs to be ful-filled if a system wants to provide security like confidentiality, integrity, and authenticity.	Refer to D4.3 [47], Chapter 4.1 and D5.2 [52], Chapter 2.3.3.
	CR-NF-019	The cryptographic services must provide a configuration mechanism to define the access methods and rights to the cryptographic objects. a) The configuration shall only be done by authorized entities. b) The access rights shall be enforced by the security architecture. c) Access rights must be	This requirement needs be ful-filled if a system wants to provide access control.	Access control is very platform-specific and therefore needs to be configured for each respective platform

Type	ID	Description of Requirements	Comments	Coverage
		definable for - Roles and Users - Services - Domains d) Access rights shall define: - Overall access - Access to individual functions using the cryptographic objects.(i.e. generating or deleting keys) e) Usage rights of cryptographic objects should be defined: - Keys for encrypting, decrypting, signing, verifying. - If keys can be deleted, exported, derived or not.		individually.
Safety	CR-NF-021	A software component should not be allowed to alter, contaminate or delay another software component's code, I/O, scheduling, or data storage areas in uncontrollable ways, especially from the less critical components to the most critical ones. Time isolation and Spatial isolation have to be ensured. New isolation mechanisms can be introduced to ensure software independence in multicore systems, enablingthe safe execution of softwarecomponents with different criticalitylevels.	Generic from safety definition.	These concepts are covered by the memory and timing protection mechanism implemented in the WP4.
	CR-NF-022	Failure on hardware unique to a software component should not cause adverse effects on any other software component.	Generic from safety definition	Covered by freedom of interferences in the IS26262 context, refer to D4.1 [46], chapter 6

Type	ID	Description of Requirements	Comments	Coverage
				and D6.2 [48], chapter 2.4.
Mixed-Critical	CR-NF-024	Mixed-criticality must be supported in hardware.	Mixed-criticality should be sufficiently isolated.	Refer to the D6.5 [49], Chapter “Safe Protection Mechanisms”.
	CR-NF-026	Incremental changes should be supported in the design and verification. The tools should exploit the isolation to keep the effects of incremental changes as small as possible for the higher levels of criticality. This feature is required for incremental certification.	Generic from mixed-critical definition	Refer to D5.3 [53], Chapter 4.
Hardware platform	CR-NF-027	The hypervisor shall support the platform selected in the telecom use case.		See deliverable D6.4 [51].
	CR-NF-028	The selected hardware platform has to provide monitoring features such as Performance Monitoring Counter (PMC) or hardware counters, allowing to monitor the timing behavior, the runtime workload on the different hardware resources, and power consumption or energy related features.	For monitoring features required by WP3 and WP4	For Automotive demonstrated refer to Chapter 2.4.

5.2 Functional and Non-functional Requirements for Automotive Multi-Core Use Case

5.2.1 Functional Requirements

Table 9: Functional Requirements for Automotive Multicore UC

ID	Description of Requirements	Comments	Coverage
S2-F-001	The functional architecture of the automotive use cases should be defined (at least in part) by means of a formal (possibly standard and commercial) modelling language.		A part of the functional architecture, is been modelled using AUTOSAR 4.x as formal language. In particular we have modelled the management of the idle in the engine control unit. As commercial tool we have used in first instance in Rhapsody tool, for a first generic description and after we have imported the arxml in the Davinci Toolchain to generate the Rte, using in particular Da DaVinci Configurator Pro, where we have specified the Task Mapping, that can be used also as input for the SymtAS/S tool to calculate the WCET analysis during the design phase. Moreover, the new pattern introduced by SAFURE framework and described in the Deliverables of WP2 are modelled in Rhapsody tool, where we have the possibility to extend the stereotypes and SSSA worked to generate ad hoc Rte for these new parts. Please refer to the chapter 2.2.2 and to the D6.5 [4949] for more details.
S2-F-003	The Electronic Control Unit (ECU) must be able		The final demonstrator is able to manage a three

ID	Description of Requirements	Comments	Coverage
	to manage a four cylinders engine and simulate the control of automatic transmission gearbox.		cylinders engine. Because this is a more requested by the automotive industry as product: the cost are less and the efficiency is comparable for economy car sector.

5.2.2 Non-functional Requirements

Table 10: Non-functional Requirements for Automotive Multicore UC

Type	ID	Description of Requirements	Comments	Coverage
Architectural Design	S2-NF-001	Modelling all the components should be required to simulate the entire system and allow a predictable time analysis and task/runnable allocation.	The simulation is mandatory for ISO26262. The time analysis is a new requirement.	We have imported the management of the “idle handle” as part of the function architecture. Moreover, we have modelled the scheduling of the control unit into the SymTA/S tool and we added the estimated timing measurements to calculate the WCRT. Refer to Chapter 2.5 for more details.
Safety	S2-NF-002	The automotive use case should provide at least one example of communication or interaction with safety concerns/issues that can be expressed in a quantitative and formal way.		Secure and safe communication based on deterministic Ethernet is implemented in WP5 and is a basis for future automotive applications (although not directly integrated in WP6 demonstrator). To ensure safety properties of a communication system, this communication system must guarantee the deterministic communication (predictive latency). This is shown in the measurements in D5.2 [52].

Type	ID	Description of Requirements	Comments	Coverage
Security	S2-NF-003	Controller Area Network (CAN) bus communication should be protected from external attacks.		See document D6.5 [49], chapter 2.1 and the chapter 2.2 of this document.
	S2-NF-004	The Data stored on multicore ECU must be protected against adversaries.		See Chapter 2.2.
	S2-NF-005	The automotive use case should provide at least one example of communication or interaction with security concerns/issues that can be expressed in a quantitative and formal way.		Security mechanisms on the Ethernet MAC layer which are the basis for future automotive Ethernet applications are implemented in WP5. Latency and jitter measurements of encrypted communication give the quantitative results.
	S2-NF-006	There should be a mechanism to prevent/limit unknown/unexpected task activations (e.g. Interrupt Request (IRQ) limiting).		See Chapter 2.3 and D6.5 [49].
	S2-NF-007	A security mechanism for authentication during flashing phase must be provided.	Currently There is not a dedicated UC for this requirement, but it is important for security aspects.	Not part of the demonstrator, but part of the Secure Update mechanism described in D4.3 [47], Chapter 4.3.
	S2-NF-008	Internal memory access from not authorized devices must be blocked and refuse.		See Chapter 2.2.
	S2-NF-009	All types of memory access from different cores must be arbitrated to provide freedom of		See document D4.3 [47] where the dedicated driver MPU is described and here the Chapter 2.1.

Type	ID	Description of Requirements	Comments	Coverage
		interference.		
Time analyses	S2-NF-010	Security SW Components should not exceed 10% CPU load globally.		See Lauterbach measurements for security code, reported in the Chapter 2.2.2.1.
	S2-NF-011	Total system should not exceed 80% CPU load for each core.	This requirement is mandatory to guarantee the correct scheduling to avoid the loss of task activation.	See Lauterbach measurements reported in the Chapter 2.5.
	S2-NF-012	The automotive use case should provide at least one example of timing constraints that need verification.		See Chapter 2.3 and D6.5 [49].
	S2-NF-013	Temporal overheads for accessing shared resources must be known (cache, on-chip memory, IO, etc.)		See Chapter 2.4.
Mixed-Critical	S2-NF-014	A mechanism for spatial and temporal isolation of the two cores must be guaranteed in order to protect from external attacks and meet safety goals.		See D4.1 [46] and D4.3 [47] for the firmware driver MPU and TRPOT.
	S2-NF-015	Engine Control Unit must be allocated on core 0, and a simulation of automatic transmission ECU must be allocated on core 1.		The engine control is running on core “0” and a dummy application of the AMT on core “1”. For more details refer to the D6.5 [49].
Hw Platform	S2-NF-016	The automatic transmission ECU output commands must be simulated on CAN message and showed on		The information is included in messages of Category C and visible on the CAN analyser.

Type	ID	Description of Requirements	Comments	Coverage
		external terminal.		
Time Analyses	S1-NF-003	One of the HW platforms must include a COTS multicore with at least 4 cores (e.g. Freescale iMX6q, Freescale P4080)	*(*) This requirement was part of the telecom use case, but the corresponding technology has been finally integrated in the automotive multicore use case*	The AURIX TC275 multicore platform has only 3 cores, but it allowed to test those aspects relevant for 4 or more cores. Moreover, solutions apply to forthcoming AURIX processors, which have 6 cores (TC3xx)
	S1-NF-004	The COTS multicore in the previous requirement must include some on-chip shared resources across cores: at least (1) a shared interconnection network between the cores and a shared cache or shared memory, and (2) a shared memory controller. It is also valuable if such multicore includes a cache memory shared across cores.	*(*) This requirement was part of the telecom use case, but the corresponding technology has been finally integrated in the automotive multicore use case*	It includes a shared interconnection network between cores and several memories, as well as several shared memories
	S1-NF-005	Performance monitoring counters (PMCs) must be abundant and allow tracking activities occurring in the on-chip shared resources such as the number (and preferably also the type) of accesses to the on-chip interconnection network and the memory controller indicated in the previous requirement.	*(*) This requirement was part of the telecom use case, but the corresponding technology has been finally integrated in the automotive multicore use case*	AURIX TC275 processors include sufficient PMCs, which allowed to develop and integrate the corresponding technology successfully

5.3 Functional and Non-functional Requirements for Automotive Network Use Case

5.3.1 Functional Requirements

Table 11: Functional Requirements for Automotive Network UC

ID	Description of Requirements	Comments	Coverage
S3-F-002	The protocol for securely updating software makes use of the PUF feature to secure a hardware fingerprint.	PUF topic was discussed with the consortium and it was concluded that the PUF technology is in a too early stage for standardised application in the SAFURE relevant UCs. Further, the selected platform does not provide a PUF.	N/A

5.3.2 Non-functional Requirements

Table 12: Non-functional Requirements for Automotive Network UC

Type	ID	Description of Requirements	Comments	Coverage
Security	S3-NF-001	The cryptographic services, such as the management of cryptographic keys and certificates, shall be applied to meet the needs of secure communication in Ethernet-based real-time networks.	It is required for secure communication for ethernet-based realtime network.	In WP5 (D5.1 [54], D5.2 [52]), cryptographic services on MAC Ethernet layer were described (to serve as a basis for future automotive Ethernet communication). In this case, cryptographic keys are static and are not exchanged during runtime.
	S3-NF-002	The network admission controller must have an authorization mechanism which allows only the authorized entities to	Authenticity is required.	Refer to D3.2 [55], Chapter 3, and D5.2 [52], Chapter 2.3.2.

Type	ID	Description of Requirements	Comments	Coverage
		send requests.		
	S3-NF-003	There should be a mechanism to prevent/limit unknown/unexpected traffic (e.g. admission control, shaping).		The admission control use case has been evaluated in the formal analysis framework pyCPA, while different traffic shaping mechanisms have been evaluated both in formal analysis, as well as simulation, see D3.2 [55], D5.3 [53], D6.5 [49].
	S3-NF-004	The support for trust anchors and secure storage of keys should be provided for secure authentication and communication	Generic from security definition.	A Hardware Security Module (HSM) could be used to provide secure storage. For keys that are generated and used at run-time, a strong isolation mechanism is also sufficient against online attacks (cf. D4.3 [47], Chapter 4.1.2).
	S3-NF-005	Information collected within a vehicle should be authentic with respect to origin and time if the vehicle performs actions based on that information.	Generic from security definition.	Refer to D3.2 [55], Chapter 3.
	S3-NF-006	The mechanism is required to ensure integrity for information collected within a vehicle. Especially the pieces of information the	Generic from security definition.	Refer to D3.2 [55], Chapter 3.

Type	ID	Description of Requirements	Comments	Coverage
		vehicle performs actions on.		
	S3-NF-007	The mechanism is required to ensure availability of ECUs for safety critical applications (robustness to denial of service attacks).	Generic from security definition.	The tests made on the combined automotive demonstrator are reported in the Chapter 4.
	S3-NF-008	Implementation of security algorithms must not violate timing constraints.	Generic from security definition	See Chapter 2.3 and D6.5 [49].
	S3-NF-009	Communication in Ethernet-based real-time network shall be secured with regards to confidentiality, authenticity and integrity whilst respecting real-time constraints (i.e. predictable latency and low jitter).	This requirement is required if SAFURE aims to support secure real-time system applications.	See D5.1 [54] and D5.2 [52].
	S3-NF-010	For the initial demonstrator, a simple level of verification and validation of the security measures should be ensured.	This is an implementation requirement. The verification and validation of the security measures will be provided by the SAFURE platform in the sense of a man-in-the-middle attack, timing analysis and worst case performance analysis.	Refer to the Chapter 4 for the man-in-the-middle attacks implemented and to the Chapters 2.5 and 3.2 for the WCET analysis made with SymTA/S.
	S3-NF-011	Network-related security applications should allow for global network ow control, increase network dynamics and permit on-the-y reconfiguration for all	In SAFURE, the inclusion of the newly developed security mechanisms should not have a negative	See comment for S3-F-001

Type	ID	Description of Requirements	Comments	Coverage
		types of traffic classes.	impact on the network behaviour.	
Time analyses	S3-NF-012	Time and safety critical traffic must state their special requirements (e.g. deadlines, redundancy, weakly hard constraints for typical case analysis) in a way which can serve as in input description to our analysis tools.	Must be provided by the network designer.	N/A
	S3-NF-013	If a traffic stream uses Typical Case Analysis (TCA), its description must provide enough information for a TCA analysis. TCA gives \m-out-of-k" guarantees (e.g. m out of k frames will meet their deadline). Hence, the parameters m and k must be provided along with a deadline.	Must be provided by the network designer.	N/A
	S3-NF-015	Network re-configuration must be performed in a bounded time.	Must be provided by the network designer.	N/A
	S3-NF-016	Each traffic stream must specify whether it requires special fault/failure tolerance, e.g. Automatic Repeat Request (ARQ), TCA, redundant paths.	Must be provided by the network designer.	N/A
	S3-NF-017	If a traffic stream uses ARQ, its description must provide enough information for the selected ARQ scheme, i.e. the ARQ	Must be provided by the network designer.	N/A

Type	ID	Description of Requirements	Comments	Coverage
		scheme, the retransmission timeout, and the number of expected retransmissions (e.g. errors).		
	S3-NF-018	Redundant paths must be specified at design time.	Must be provided by the network designer.	N/A
	S3-NF-020	Each traffic stream must be categorized into critical (e.g. time- and/or safety-critical) or non-critical traffic (e.g. best effort).	Must be provided by the network designer.	N/A
	S3-NF-021	The arbitration scheme in the switches must support mechanisms to distinguish critical (e.g. timing, safety) from non-critical traffic streams to guarantee freedom from interference/sufficient independence for critical traffic streams.	Must be provided by the switch manufactorer.	N/A
Safety	S3-NF-022	There must be some kind of admission control in the (virtual) network to ensure robustness to denial of service attacks.	Must be provided by the switch manufactorer.	N/A
	S3-NF-023	Switches and/or end stations (in the virtual network) must support the detection of hardware failures, e.g. broken links or switches.	Must be provided by the switch manufactorer.	N/A
	S3-NF-024	Switches and/or end stations (in the virtual network) must support monitoring schemes capable of timely detecting attacks and misbehaving traffic.	Must be provided by the switch manufactorer.	N/A

Type	ID	Description of Requirements	Comments	Coverage
		The monitoring scheme must be configurable, e.g. via SDN, and their parameters should be provided, e.g. number of replenishment tokens and replenishment interval for leaky bucket shapers or l-repetitive arrival functions for advanced monitoring.		
	S3-NF-025	Switches and/or end stations (in the virtual network) must support mechanisms to shape/block attacking/misbehaving traffic in a timely and appropriate way. These mechanisms must be configurable, e.g. via SDN.	Must be provided by the switch manufactorer.	N/A
Hw Platform	S3-NF-026	The SDN mechanisms together with the (virtual) network equipment (e.g. switches) must support the reconfiguration of the network.	Must be provided by the HW manufactorer.	N/A
	S3-NF-027	SAFURE platform should provide Non-Volatile Memory (NVM) and a Physical Unclonable Function (PUF) feature.	PUF topic was discussed with the consortium and it was concluded that the PUF technology is in a too early stage for standardized application in the SAFURE relevant UCs. Further, the selected platform does not provide a PUF	N/A

Chapter 6 Potential evolution

Concerning the security topic on the Automotive Multicore Demonstrator, in the current version of the prototype we have used static (hard-coded) keys for AES-GCM.

A new robust and secure extension for this demonstrator is to integrate a key distribution system.

This can be done using two modern algorithms based on Elliptic Curve Cryptography:

- EdDSA [39] for signature generation and verification
- X25519 [40] for Diffie-Hellman key exchange

The protocol consists of the following steps (assuming two parties called Alice and Bob):

1. Alice and Bob create long-term EdDSA/Ed25519 key pairs
2. Each public key is transferred to the other party
3. Alice and Bob create short-term X25519/Curve25519 key pairs, this includes:
    a. Generation of a random private key (32 random bytes)
    b. Computation of the public key
4. Using X25519, Alice and Bob agree on a common session key K, this includes:
    a. Sending the public key, signed with the EdDSA key, to the other party
    b. Verifying the received public key of the other party
    c. Computation of the common session key
5. The session key is used for AES-GCM encryption of the bulk data

Steps 1 and 2 will be performed only once at production time. The public/private keys can be hard-coded into the demonstrator code (in production use, there would be a public key infrastructure and the private key would be securely stored). Steps 3 to 5 are performed for every session (e.g., at boot time of the demonstrator).

The benefits for the demonstrator will be:

- Demonstration of start-of-the-art algorithms that are well suited for embedded systems,
- Addressing the key distribution problem,
- All algorithms offer security comparable to that of AES with 128 bits (i.e., not breakable in the foreseeable future [44]),
- EdDSA and X25519 can easily be implemented with protection against many side-channel attacks, especially timing attacks.

Chapter 7 Summary and conclusion

This document has illustrated the results, tests and the evaluation of one of the industrial SAFURE Demonstrators: the Automotive Demonstrator that is described in details in the D6.5 [49]. In particular, the D6.5 [49] explains how the Automotive Demonstrator has been realized integrating and applying the SAFURE framework that is summarized in the D6.7 [50]. As described in D6.5 [49], the Automotive Demonstrator consists of two prototypes: the Multicore and the Network demonstrators.
Chapter 2 reports the evaluations made on the Multicore Automotive demonstrator, according to the tests implemented and the results achieved on the main features of this demonstrator:

- the Secure Real-time CAN communication is stressed implementing specific tests in the environment presented in Chapter 2.2, where also the solution implemented in SAFURE is compared against other proposals introduced and published in last period;
- the multicore topics are considered in the Chapters 2.1, 2.3, 2.4 and 2.5 that take into account how the mandatory safety aspects are respected, how to achieve from them an RTE generation and how take advantages from the timing analysis evaluation.

These concepts are fundamentals for the new generations of ECUs in the Automotive Industry that is paying more attention to the safety and security aspects of their systems and the SAFURE framework is able to achieve all these aspects, considering the economic needs.
The Automotive Network demonstrator, in Chapter 3 focuses on safety measures required to enable mixed-critical communication in future in-vehicle Ethernet networks. This scenario also considers security aspects of Ethernet, e.g. sufficient isolation between traffic streams to protect against denial of service attacks.
Furthermore, in the Automotive Demonstrator the two prototypes are combined introducing the gateway which connects the multicore network prototype transmitting CAN-messages to an Ethernet network, as explained in the D6.5 [49]. Chapter 4 describes the tests build and implemented to evaluate it and the full equipment used to realize them.
Finally, the Automotive Demonstrator is built covering the requirements presented in WP1 [34] [35] and reported in this document at Chapter 5 and last but not least, Chapter 6 considers a possible evolution of this demonstrator in the Automotive industry road-map.

Chapter 8 List of Abbreviations

Table 13: List of Abbreviations

PWT	Powertrain
ECU	Engine Control Unit
AMT	Automated Manual Transmission
CAN	Controller Area Network
CAN-FD	Controller Area Network – Flexible Data Rate
SPP	Static Priority Preemptive
SPNP	Static Priority Non-Preemptive
CPA	Compositional Performance Analysis
CAN	Controller Area Network
LIN	Local Interconnect Network
CSM	Crypto Service Manager
OS	Operationg System
MPU	Memory Protection Unit
RTOS	Real-time Operating system
RPM	Revolution Per Minute
AES	Advanced Encryption Standard
HSM	Hardware Security Module
WCET	Worst Case Execution Time
WCRT	Worst Case Response Time
UDS	Unified Diagnostic Services
GCM	Galois/Counter Mode

Chapter 9 Bibliography

[1] Infineon. AURIX. http://www.infineon.com/cms/en/product/microcontroller/32-bit-tricore-tm-microcontroller/aurix-tm-family/channel.html?channel=db3a30433727a44301372b2eefbb48d9.

[2] International Organization for Standardization (ISO). ISO 26262: Road Vehicles - Functional Safety, 2011.

[3] Vector. ECU Analysis with CANalyzer. http://vector.com/vi_canalyzer_en.html.

[4] OMG UML (2011). Unified Modeling Language (UML), formal/2011-08-06, Version 2.4.1, August 2011.

[5] OMG SYML (2012). System Modeling Language (SysML), formal/2012-06-01, Version 1.3, June 2012.

[6] AUTomotive Open System ARchitecture (AUTOSAR). Specification of Operating System. http://www.autosar.org/fileadmin/files/releases/4-0/software-architecture/system-services/standard/AUTOSAR_SWS_OS.pdf, November 2011. R4.0 Rev 3 V5.0.0.

[7] AUTOSAR. AUTOSAR Software Component Template: AUTOSAR Release 4.2.2.

[8] AUTOSAR. AUTOSAR Specification of Crypto Service Manager: AUTOSAR Release 4.2.2.

[9] AUTOSAR. Specification of Security Extensions: AUTOSAR Release 4.2.1.

[10] AUTOSAR. Glossary: AUTOSAR Release 4.2.1.

[11] AUTOSAR. Specification of Timing Extensions: AUTOSAR Release 4.2.1.

[12] AUTOSAR. Timing Analysis: AUTOSAR Release 4.2.1.

[13] AUTOSAR. Specification of Safety Extensions: AUTOSAR Release 4.2.1.

[14] AUTOSAR. Overview of functional safety measures: AUTOSAR Release 4.2.2.

[15] ERIKA Enterprise - Evidence Sr is the main contributor of the ERIKA Enterprise RTOS (http://erika.tuxfamily.org), the first open-source RTOS that received the OSEK/VDX certification, which has been ported to various MCUs for automotive, including multicores.

[16] Evidence integrated ERIKA Enterprise together with Linux on a Multicore iMX6 without virtualization support to provide an open-source solution for automotive system including real-time support and HMI/Networking. http://www.evidence.eu.com/embedded-linux-osekvdx-erika-enterprise-dual-core-automotive-cpu-without-hypervisor.html

[17] Paolo Gai, Giuseppe Lipari and Marco Di Natale, Design Methodologies and Tools for Real-Time Embedded Systems, Special Issue of Design Automation for Embedded Systems, 2002.

[18] Bosch Controller Area Network - ver 2.0 http://www.kvaser.com/software/7330130980914/V1/can2spec.pdf

[19] ISO 11898-1:2015 Road vehicles -- Controller area network (CAN) -- Part 1: Data link layer and physical signaling

[20] ISO 11898-2:2003 Road vehicles -- Controller area network (CAN) -- Part 2: High-speed medium access unit

[21] ISO 11898-3:2006 Road vehicles -- Controller area network (CAN) -- Part 3: Low-speed, fault-tolerant, medium-dependent interface

[22] Zhang L, Gao H, Kaynak O (2013) Network-induced constraints in networked control systems—a survey. IEEE Trans Ind Inf 9(1):403–416

[23] Thiele L, Chakraborty S, Naedele M (2000) Real-time calculus for scheduling hard real-time systems. In: Proceedings of IEEE international symposium on circuits and systems (ISCAS), vol 4, pp 101–104. doi: 10.1109/ISCAS.2000.858698

[24] Henia R, Hamann A, Jersak M, Racu R, Richter K, Ernst R (2005) System level performance analysis – the SymTA/S approach. IEE Proc Comput Digit Tech 152(2):148–166

[25] IEEE 802.1Q

[26] Schliecker S, Rox J, Ivers M, Ernst R (2008) Providing accurate event models for the analysis of heterogeneous multiprocessor systems. In: Proceedings of CODES-ISSS, pp 185–190

[27] Feiertag N, Richter K, Nordlander J, Jonsson J (2008) A compositional framework for end-to-end path delay calculation of automotive systems under different path semantics. In: Work on compositional theory and technology for real-time embedded systems CRTS

[28] Perathoner S, Rein T, Thiele L, Lampka K, Rox J (2010) Modeling structured event streams in system level performance analysis. In: ACM SIGPLAN/SIGBED conference on languages, compilers and tools for embedded systems (LCTES). ACM, Sweden, pp 37–46

[29] Kern A, Reinhard D, Streichert T, Teich J (2011) Gateway strategies for embedding of automotive CAN-frames into ethernet-packets and vice versa. In: Architecture of computing systems, pp 259–270

[30] Ayed H, Mifdaoui A, Fraboul C (2011) Gateway optimization for an heterogeneous avionics network afdx-can. In: IEEE real-time systems symposium (RTSS)

[31] Scharbarg J, Boyer M, Fraboul C (2005) Can-ethernet architectures for real-time applications. In: IEEE conference on emerging technologies and factory automation (ETFA), vol 2, pp 8–252

[32] Nacer A, Jaffres-Runser K, Scharbarg JL, Fraboul C (2013) Strategies for the interconnection of can buses through an ethernet switch. In: IEEE international symposium on industrial embedded systems (SIES), pp 77–80

[33] Diemer J, Axer P, Ernst R (2012a) Compositional performance analysis in python with pycpa. In: Proceedings of WATERS. http://retis.sssup.it/waters2012/WATERS-2012-Proceedings.pdf

[34] SAFURE D1.1: https://safure.eu/downloads/SAFURE-D1.1-PU-M06.pdf

[35] SAFURE D1.2: https://safure.eu/downloads/SAFURE-D1.2-PU-M06.pdf

[36] «OSEK/VDX – Operative System,» [Online]. Available: www.osek-vdx.org

[37] ISO 14229-1:2013 https://www.iso.org/standard/55283.html

[38] https://vector.com/vi_canalyzer_en.html

[39] https://en.wikipedia.org/wiki/EdDSA

[40] https://en.wikipedia.org/wiki/Curve25519

[41] https://auto.luxoft.com/uth/timing-analysis-tools/

[42] https://www.wireshark.org/

[43] https://www.lauterbach.com/frames.html?home.html

[44] https://www.keylength.com

[45] Radu A, Garcia F (2016) LeiA: A Lightweight Authentication Protocol for CAN, In: ESORICS 2016: 21st European Symposium on Research in Computer Security. https://www.cs.bham.ac.uk/~garciaf/publications/leia.pdf

[46] SAFURE D4.1: https://safure.eu/downloads/SAFURE-D4.1-DEM-PU-M18.pdf

[47] SAFURE D4.3: “Final OS & RTE prototypes”

[48] SAFURE D6.2: “Architecture of automotive prototype”

[49] SAFURE D6.5: “Final automotive prototype”

[50] SAFURE D6.7: “Final specifications of the SAFURE Framework and Methodology”

[51] SAFURE D6.4: “Evaluation of telecommunications demonstrator”

[52] SAFURE D5.2: “Final communication prototypes”

[53] SAFURE D5.3: “Communications safety and security analysis”

[54] SAFURE D5.1: “Alpha communication prototypes”

[55] SAFURE D3.2: https://safure.eu/downloads/SAFURE-D3.2-PU-M30.pdf