D3.7
Evaluation results, conclusions and further recommendations, including derived requirement recommendations for drive-by-data and embedded platform

Project number:	730830
Project acronym:	Safe4RAIL
Project title:	Safe4RAIL: SAFE architecture for Robust distributed Application Integration in roLling stock
Start date of the project:	1st of October, 2016
Duration:	24 months
Programme:	H2020-S2RJU-OC-2016-01-2

Deliverable type:	Report
Deliverable reference number:	ICT-730830 / D3.7 / 1.0
Work package	WP 3
Due date:	September 2018 – M24
Actual submission date:	3rd of October, 2018

Responsible organisation:	SIE
Editor:	Tobias Pieper
Dissemination level:	Public
Revision:	1.0

Abstract:	Contains evaluation results, conclusions and further recommendation, including derived requirements recommendations for drive-by-data and embedded platform.
Keywords:	Evaluation results, conclusions, recommendations, distributed simulation framework, Train-to-Ground test environment

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 730830.

Editor

Tobias Pieper (SIE)

Contributors (ordered according to beneficiary numbers)

Cristina Cruces, Rafael Priego (IKL)

Richard Pecl (UNI)

Peter Lehmann (IAV)

Marion Berbineau, Maha Bouaziz, Mohamed Kassab (IFS)

José Soler, Ying Yan (DTU)

Executive Summary

To simplify the integration and testing of railway components, the European research project Safe4RAIL develops two test environments. The first is a distributed simulation and validation framework which enables the connection of geographically distributed simulations, software applications and real devices. This way their interactions can be tested on a network-centric abstraction level. The second test environment is used to test Train-to-Ground (T2G) communication which defines the communication between an on-board Mobile Communication Gateway (MCG) and a track-side Ground Communication Gateway (GCG).

Both environments are developed in cooperation with the European CFM research project CONNECTA (project reference 730539). Regarding the simulation framework, the Safe4RAIL part focuses on the communication between the devices on a network-centric abstraction level. It is called Communication Emulator (CE). The CONNECTA framework instead concentrates on functional and electromechanical simulations. In case of the T2G Test Environment (TE), Safe4RAIL's objective is the implementation of a GCG as well as the test environment which connects the GCG with an MCG. To simulate the wireless communication, an Access Network Simulator (ANS) is developed using LTE (Long Term Evolution) or WiFi as communication media. The CONNECTA project instead develops four MCGs. All parts of the T2G TE follow the IEC 61375-2-6 standard.

This deliverable presents the results of evaluating both environments. Tests with the Communication Emulator show its usability as long as no temporal constraints are imposed and the communication network used to connect its subsystems (simulation bridges) do not introduce large delays. Furthermore, it is possible to run real-time simulations as long as the temporal constraints are not too strict and faults can be injected according to the EN 50159 standard [8]. However, the time synchronization of the simulation bridges invokes requirements on real devices. The most important ones are (I) a synchronization mechanism between the simulation bridge and the device and (II) communication based on a defined temporal schedule or dedicated communication periods.

Tests with the T2G TE in cooperation with CONNECTA show issues in the present version of the IEC 61375-2-6 standard. For example, there are inconsistencies in the text, undefined communication behaviour in some situations or errors in telegram content definitions. The deliverable provides proposed corrections in the text and further suggests the implementation of real hardware before the standardization process finishes. This hardware could be used to find further issues. Besides the MCG/GCG tests, LTE simulations show that the technology is suitable to be used for T2G communication. It is also possible to handover the communication between several WiFi Access Points (AP).

Integration and testing of railway components is a major topic during the development of railway components. Thus, a proper tool qualification must be ensured for the tools which verify both aspects. The deliverable provides comments and recommendations to reach such a qualification. Among others, such a qualification depends on the tools classification, its usage or the standard the tool qualification shall follow. These aspects shall be defined in the near future.

Contents

Chapter 1 Introduction 1
Chapter 2 Evaluation of the distributed simulation framework 2
2.1 Evaluation of communication delays between two hosts 2
2.1.1 Description of test setup 2
2.1.1.1 Test setup to measure delays when data flows are lower than the maximum throughput allowed by the CE 3
2.1.1.2 Test setup to measure delays when data flows are higher than the maximum throughput allowed by the CE 4
2.1.2 Tests without CE 6
2.1.3 Tests with local RTI 7
2.1.4 Tests with remote RTI 14
2.2 Evaluation and temporal characteristics of simulation bridges 16
2.2.1 Description of test setup 17
2.2.2 Evaluation FMI 19
2.2.3 Evaluation PCAP 26
2.2.4 Evaluation of fault-injection 31
2.2.5 Evaluation of state-estimation and delay-management 34
2.3 Framework evaluation with real train applications 39
2.3.1 CAF-setup 39
2.3.1.1 Description 39
2.3.1.2 Results 41
2.3.2 UNI-setup 42
2.3.2.1 Description of test setup 42
2.3.2.2 Evaluation results 44
2.4 Evaluation of test automation 45
2.4.1 Description of test setup 45
2.4.2 Test cases 46
2.4.3 Evaluation results 48
Chapter 3 Evaluation of the Train-to-Ground Test Environment 53
3.1 Evaluation of using the Test Tools for implementation of the T2G tests 54
3.1.1 Execution environment 54
3.1.2 Additional requirements on devices under test 55

3.2 Evaluation of performing the T2G tests implemented based on CONNECTA test specification 55
3.3 Evaluation of Wi-Fi Access Network Simulation using the Channel Emulator 57
3.4 Evaluation of the LTE Access Network Simulation models 60
3.4.1 Results for the pure simulation using LTE technology 61
3.4.1.1 Simulation parameters 61
3.4.1.2 Test scenario for the train speed and network load factors study 65
3.4.1.3 Different Test scenarios for the network load factor 66
3.4.1.4 Other TCMS traffic evaluation for the network load factor 68
3.4.1.5 Test scenario for the existence of jammer node 69
3.4.2 Results for the pure simulation using LTE and WiFi 71
3.4.2.1 Network architecture 71
3.4.2.2 Simulation context and parameters 73
3.4.2.3 Evaluation results 74
3.4.2.3.1 Connectivity scenario 74
3.4.2.3.2 Loaded WiFi scenario 77
3.4.3 Results for the Co-simulation using LTE technology (IFS) 78
3.4.3.1 Test Equipment 78
3.4.3.2 Test 1. Validation of GCGSim Environment 80
3.4.3.3 Test 2. Performance testing 85
3.4.3.4 Test 3: Redundancy test in backhaul network 93
Chapter 4 Conclusions and further recommendations 96
4.1 Conclusions and recommendations for the distributed simulation framework (IKL/SIE/IAV) 96
4.1.1 Conclusions and recommendations 96
4.1.2 Derived requirement recommendations for drive-by-data and the embedded platform 99
4.2 Conclusions and recommendations for the Train-to-Ground test environment 99
4.2.1 Comments on T2G standard 100
4.2.2 Conclusions and recommendations for the T2G TE regarding the Access Network Simulator 103
4.3 Conclusions and recommendations regarding certification aspects related to the test environment 104
4.3.1 General points about the needs caused by certification 104
4.3.2 Conclusions and recommendations 105
Chapter 5 Summary and Conclusion 106
Chapter 6 List of Abbreviations 108
Chapter 7 Bibliography 111

List of Figures

Figure 1. Basic topologies without CE 3
Figure 2. Oscilloscope delay capture 4
Figure 3. CE transmitting packets in burst 5
Figure 4. Basic topologies with BBBs and CE connected locally. 8
Figure 5. Latency in millisecond range when the cycle time of the transmitter BBB is 500µs. 11
Figure 6. Latency in millisecond range when the cycle time of the transmitter BBB is 250µs. 11
Figure 7. Latency in millisecond range when the cycle time of the transmitter BBB is 200µs. 12
Figure 8. Latency in millisecond range when the cycle time of the transmitter BBB is 150µs. 12
Figure 9. Basic topologies without CE. 14
Figure 10. Fan control with triple modular redundancy [10]. 17
Figure 11. Local setup [10]. 18
Figure 12. Communication via LAN or the Internet [10]. 18
Figure 13. Example for hierarchical RTI [10]. 18
Figure 14. Simulation durations for simulated times until 100s [10]. 19
Figure 15. Simulation durations for simulated times until 1000s [10]. 20
Figure 16. Simulation durations for simulated times until 100s if a VPN is used [10]. 20
Figure 17. Simulation durations for connection via PCAP. 26
Figure 18. Fan speed without fault injected. 31
Figure 19. Manipulation fault, destination of 1600 rpm at 10s changed to 1350 rpm. 32
Figure 20. Delay fault, destination of 1600 rpm at 10s is delayed by 8s. 32
Figure 21. Resequencing fault, destinations of 1600 rpm (10s) and 1200 rpm (20s) interchanged. 33
Figure 22. Insertion fault, new destination of 1700 rpm inserted after 15s. 33
Figure 23. Omission fault, destination of 1600 rpm at 10s omitted. 34
Figure 24. Replay fault, destination of 1600 rpm replayed at 25s. 34
Figure 25. Outputs of the system model compared to the PID controller outputs. 35
Figure 26. Simulation durations if the state-estimation is used. 36
Figure 27. Equipment provided by CAF. 39
Figure 28. CAF provided equipment connected via CE locally. 41
Figure 29. CAF provided equipment connected via CE remotely. 41
Figure 30. Use-case for evaluation of time management with real train hardware. 43

Figure 31. Distribution of test setup using the CE. 43
Figure 32. Relation of the UI client and server. 45
Figure 33. Example of a test automation. 46
Figure 34. Configuration subsystem. 48
Figure 35. Monitoring subsystem. 49
Figure 36. Example of a positive scenario (fault injection start – stop). 50
Figure 37. Example of a failure scenario (monitoring stop before start). 51
Figure 38. Example of wrong execution order (monitoring start before configuration). 52
Figure 39. T2G Test Environment Scope. 54
Figure 40. T2G test session setup. 56
Figure 41. Scenario 2.7. 58
Figure 42. Topology architecture of the railway trajectory [19]. 61
Figure 43. Sent/received MCG signalling data. 65
Figure 44. Packet lost number in the LTE PHY. 66
Figure 45. Packet delay for different TCMS applications. 66
Figure 46. Sent/received MCG signalling data for not-congested network. 67
Figure 47. Packet delay for different TCMS applications for not congested network. 67
Figure 48. Results for congested network. 68
Figure 49. Sent/received MCG message data for congested network. 69
Figure 50. Packet delay for message data. 69
Figure 51. Signalling data transferring without and with jammer node existing. 70
Figure 52. Voice data transferring without and with jammer node existing. 70
Figure 53. Video data transferring without and with jammer node existing. 70
Figure 54. File data transferring without and with jammer node existing. 71
Figure 55. WiFi-LTE network architecture. 72
Figure 56. WiFi-LTE MCG architecture. 72
Figure 57. Train association to the LTE eNodeB and WiFi Access Points. 74
Figure 58. Statistics for signalling (connectivity scenario). 75
Figure 59. Statistics for voice (connectivity scenario). 76
Figure 60. Statistics for signalling of stationary trains (Loaded WiFi). 77
Figure 61. Statistics for voice of stationary trains (Loaded WiFi). 78
Figure 62. System layout in test setup. 79
Figure 63. UE connection with GCG host, with the announce command. 81
Figure 64. UE connection with GCG host with location info and train info command. 81
Figure 65. Test tool result of announce command. 81
Figure 66. Test tool result of list MCG command. 82

Figure 67. Test tool result of sent location notification command. 82
Figure 68. Test tool result of sent train info notification command..83
Figure 69. Test tool result of upload request and upload command. 84
Figure 70. Test tool result of sent upload result command. 84
Figure 71. Test tool result of train info MCG command. 85
Figure 72. Test tool result of location MCG command. 85
Figure 73. Backhaul network in OPNET and link delay settings. 86
Figure 74. Ping from EPC without OPNET delay (average delay = 16.735ms). 87
Figure 75. Ping from EPC with tx_delay=1s for all packets (both directions, average delay = 3628ms). 87
Figure 76. Ping from UE to server without OPNET delay (average delay = 56ms)................88
Figure 77. Ping from UE to server with OPNET delay of 0.05s (average delay = 151ms). 88
Figure 78. RTT Result with RTT = 58 ms when no delay is introduced. 89
Figure 79. RTT = 178 ms when the OPNET delay is enabled. 89
Figure 80. Ping from EPC without delay (average delay = 14.558ms). 90
Figure 81. Ping from EPC with 100ms delay (average delay = 115.039ms)..........................91
Figure 82. Ping from eNodeB without delay (average delay = 0.291ms)..............................91
Figure 83. Ping from eNodeB with 5ms delay (average delay = 5.430ms). 91
Figure 84. Ping from UE to server without delay (average delay = 63ms)............................92
Figure 85. Ping from UE to server with in total 110ms delay (average delay = 158ms).........92
Figure 86. TCP RTT between UE and GCG host (48ms). 92
Figure 87. TCP RTT between UE and GCG host with 110ms delay (163ms). 93
Figure 88. Ping from UE to server without loss. 93
Figure 89. Ping from UE to server with 5% loss configured. 3% of packets are lost in fact. ..93
Figure 90. Overall test scenario with simulated backbone network. 94
Figure 91. Simulated Backhaul Network – with failure at link switch_2 <-> switch_4. 94
Figure 92. Simulated Backhaul Network – with failure at link switch_2 <-> switch_4. 95
Figure 93. Ping from EPC without failure (average delay = 115.6ms). 95
Figure 94. Ping from EPC with failure (average delay = 135.8ms). 95
Figure 95. Current proposal SIL with SB with VPN and HLA. 97
Figure 96. New proposal SIL with SB without VPN and HLA. 98

List of Tables

Table 1. Delay results without CE. 6
Table 2. Delay results with CE and high transmission cycle times. 8
Table 3.Delay results with CE and low transmission cycle times. 9
Table 4. Delay results with CE, high transmission cycle times, and bidirectional traffic. 13
Table 5. Delay results with CE, low transmission cycle times, and bidirectional traffic. 13
Table 6. Delay results with CE Central PC remotely located and high transmission cycle times. 15
Table 7. Delay results with CE Central PC remotely located and low transmission cycle times. 15
Table 8. Regression lines for each setup [10]. 21
Table 9. Interaction delays between the simulation bridges in the different topologies [10]. 21
Table 10. Average pass-through time of the simulation bridges for receiving in µs [10]. 23
Table 11. Average pass-through time of the simulation bridges for sending in µs [10]. 23
Table 12. Fractions of simulation bridge delays compared to the simulation duration in the local PC-setup [10]. 24
Table 13. Fractions of simulation bridge delays compared to the simulation duration in the LAN-setups [10]. 24
Table 14. Fractions of simulation bridge delays compared to the simulation duration in the Internet setups [10]. 25
Table 15. Regression lines for the setups using PCAP. 27
Table 16. Interaction delays between the simulation bridges in the different test setups using PCAP. 27
Table 17. Average pass-through time of the simulation bridges (PCAP) for receiving in µs. 28
Table 18. Average pass-through time of the simulation bridges (PCAP) for sending in µs. 29
Table 19. Fractions of simulation bridge delays compared to the simulation duration if the fan is connected via PCAP. 29
Table 20. Fractions of simulation bridge delays compared to the simulation duration if the voter is connected via PCAP. 30
Table 21. Simulation steps and missed events in sum for every simulation duration (upper line) and in average (lower line). 36
Table 22. Packets in sum for every simulation duration (upper line) and in average (lower line). 37
Table 23. Variable values sent by CCU1 when it is configured to send message every 32 ms. 40
Table 24. Variable values sent by CCU1 when it is configured to send message every 64 ms. 40
Table 25. Delay results with CE and real railway equipment. 41

Table 26. LTE configuration parameters. 62
Table 27. TCMS application modelling. 62
Table 28. Passenger UE profiles and their traffics. 63
Table 29. Standardized QCI characteristics. 63
Table 30. LTE bearer setting. 64
Table 31. Data delivery ratio for different TCMS applications. 65
Table 32. LTE bearer setting. 67
Table 33. LTE bearer setting. 68
Table 34. WiFi configuration parameters. 73
Table 35. WiFi Access Categories. 74
Table 36. Comments on the T2G standard. 100
Table 37. List of Abbreviations. 108

Chapter 1 Introduction

The integration and testing of railway components are important development steps as they ensure the correct interaction of components developed by different manufacturers. One objective in the Safe4RAIL project is the development of a distributed simulation and validation framework. The second objective in this project is the development of a Train-to-Ground (T2G) Test Environment (TE).

The distributed simulation framework enables early integration tests between geographically distributed simulations, software applications (Software-In-The-Loop, SIL) as well as real train devices (Hardware-In-The-Loop, HIL). These systems can be coupled securely via heterogeneous communication networks such as Local Area Networks (LANs) or the Internet. The framework is developed in close interaction with the CFM project CONNECTA. Safe4RAIL focuses on the communication of the devices on a network centric abstraction level whereas CONNECTA concentrates on functional and electromechanical simulations. To avoid naming conflicts, the Safe4RAIL simulation framework is called Communication Emulator (CE) in this deliverable.

T2G communication enables data exchange between an on-board Mobile Communication Gateway (MCG) and a Ground Communication Gateway (GCG) located on the railway track. The T2G TE developed in Safe4RAIL is capable of validating the communication according to the IEC standard 61375-2-6 using different test tools. These test tools include MCG/GCG simulators, real MCG/GCG implementations and controllable wireless data links using WiFi or LTE (Long Term Evolution). Similar to the CE, the work is done in close cooperation with CONNECTA which provide real MCG implementations while the test environment including a real GCG is implemented by Safe4RAIL.

This deliverable presents the evaluation results for the CE as well as the T2G test environment. Those evaluations are based on different test setups defined. The deliverable is organized as follows.

- Chapter 2 presents evaluation results regarding the CE. It starts with the evaluation of communication delays introduced by the CE between two hosts not considering the time management functionalities of the simulation bridges (Section 2.1). In Section 2.2, an evaluation of the simulation bridges and their temporal characteristics is presented including the time management. Section 2.3 follows with an evaluation according to the previous tests using real train hardware. Finally, Section 2.4 closes the chapter with an evaluation of the test automation API (Application Programming Interface).
- In Chapter 3, evaluation results of the T2G TE are depicted. Section 3.1 presents results using test tools for implementing the T2G tests. Afterwards, T2G tests according to CONNECTA's test specification follow (Section 3.2). The next two sections describe the evaluation results of the Access Network Simulators (ANS) using WiFi (Section 3.3) and LTE (Section 3.4).
- Chapter 4 outlines conclusions and further recommendations for future projects working with the outcomes of the CE (Section 4.1) and the T2G TE (Section 4.2). Furthermore, derived requirement recommendations regarding the CE are defined for drive-by-data and the embedded platform developed in other work packages from the Safe4RAIL project. The recommendations related to the T2G TE include a revision of the T2G standard according to the issues found during the T2G TE development.

Chapter 2 Evaluation of the distributed simulation framework

This chapter presents evaluation results of the CE. Section 2.1 starts with the evaluation of communication delays introduced by the CE between two hosts. In these tests, the time management functionalities of the simulation bridges are disabled. In Section 2.2, an evaluation of the simulation bridges and their temporal characteristics is presented using the time management. According to the previous tests, Section 2.3 describes use-cases including real train hardware. The chapter closes with an evaluation of the test automation API (Section 2.4).

2.1 Evaluation of communication delays between two hosts

This section presents the results of a set of measurements performed with the objective of determining which latency is introduced by the CE considering different topologies and data flows. Moreover, the results of the measures also show the maximum admissible throughput by the CE before it starts to drop messages.

The first evaluation of the communication delays introduced by the CE is tested without considering the Simulation Framework developed by CONNECTA nor real railway end devices.

2.1.1 Description of test setup

The three basic topologies proposed for this setup are represented in Figure 1. As it has been mention before, in this setup no real railway end devices are used. Instead we use Beagle Bone Black (BBB) development platforms. The BBB is a low-cost credit-card-sized development platform perfect for physical computing and smaller embedded applications. The BBB runs a Linux Operative System that has been modified to include a real-time patch. This patch allows soft-real time behaviour.

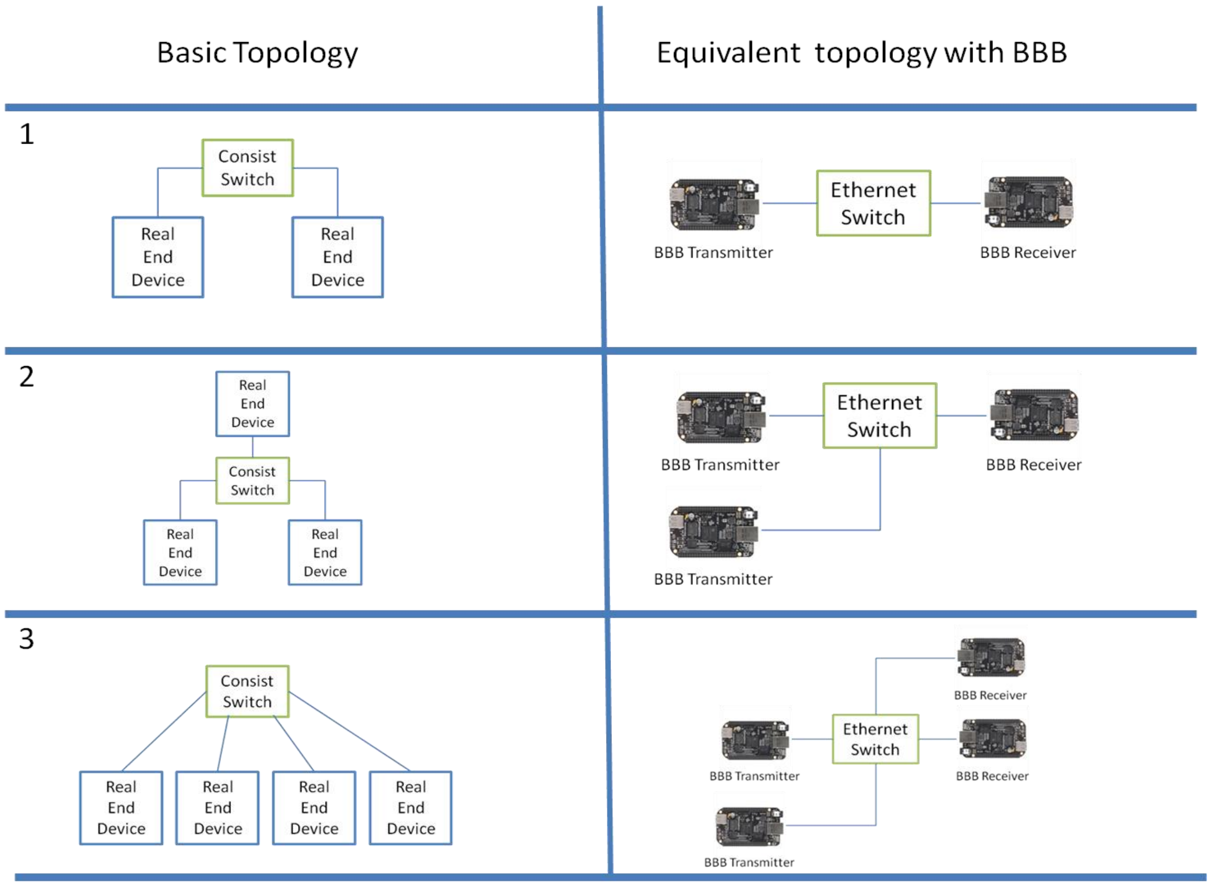


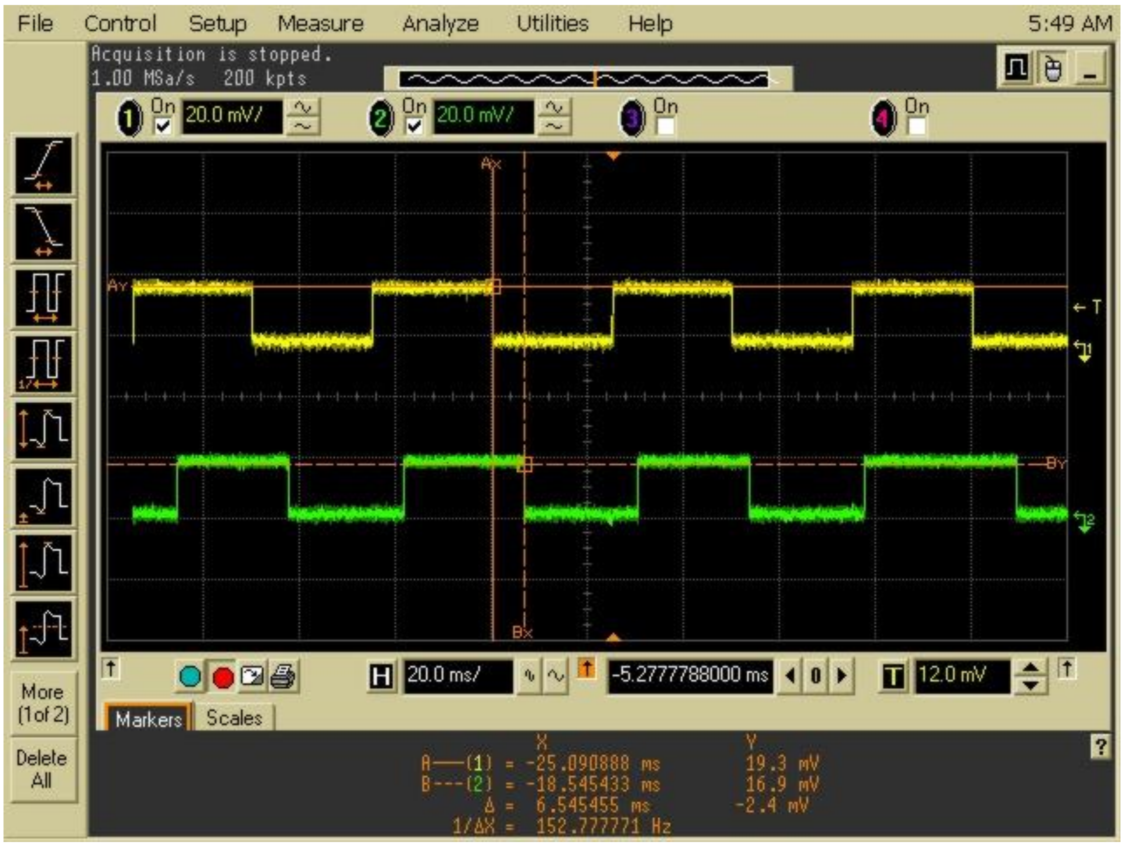
Figure 1. Basic topologies without CE

In order to simplify the measurements in this setup, the communication between the BBBs is considered unidirectional at a first stage. Therefore, two types of BBBs are distinguished in this setup, those that transmit UDP messages and those that receives those messages.

Two different test setups are used, the first one explained in Subsection 2.1.1.1 focuses on measuring the latency introduced by the CE. The second one, explained in Subsection 2.1.1.2, focuses on analysing the throughput limits of the CE. It also analyses what happens with the latency of the system when the CE is managing traffic that is near to it maximum throughput.

2.1.1.1 Test setup to measure delays when data flows are lower than the maximum throughput allowed by the CE

In order to measure the latency introduced by the CE at application level, the following method is used: the transmitter BBBs are programmed to transmit a UDP packet with fixed time intervals (down to 1 millisecond). Every time they send a packet, they toggle an LED to show the instant in which the packet is commanded to be sent at application level. In a similar way the receiver BBBs are programmed to toggle an LED as soon as they receive a packet at application level. The GPIO outputs of the BBB (transmitter and receiver) associated to the LEDs are connected to an oscilloscope in order to measure the delay between the toggle instant of both signal as it can be observed in Figure 2.



```
Acquisition is stopped.
1.00 MSa/s   200 kpts
```

```
                   X                       Y
A——(1) = -25.090888 ms              19.3 mV
B---(2) = -18.545433 ms             16.9 mV
      Δ =   6.545455 ms             -2.4 mV
   1/ΔX =  152.777771 Hz
```

Figure 2. Oscilloscope delay capture

It must be pointed out that the messages sent by the transmitter BBB are UDP packets with 6 bytes of payload. Two of them are a sequence number that is used by the receiver BBB in order to detect messages dropped or duplicated.

2.1.1.2 Test setup to measure delays when data flows are higher than the maximum throughput allowed by the CE

Once the traffic managed by the CE is approaching to its throughput limit the measurement method presented in the previous subsection is not valid. The reason is that the messages are transmitted in a burst by the CE as it can be observed in Figure 3.

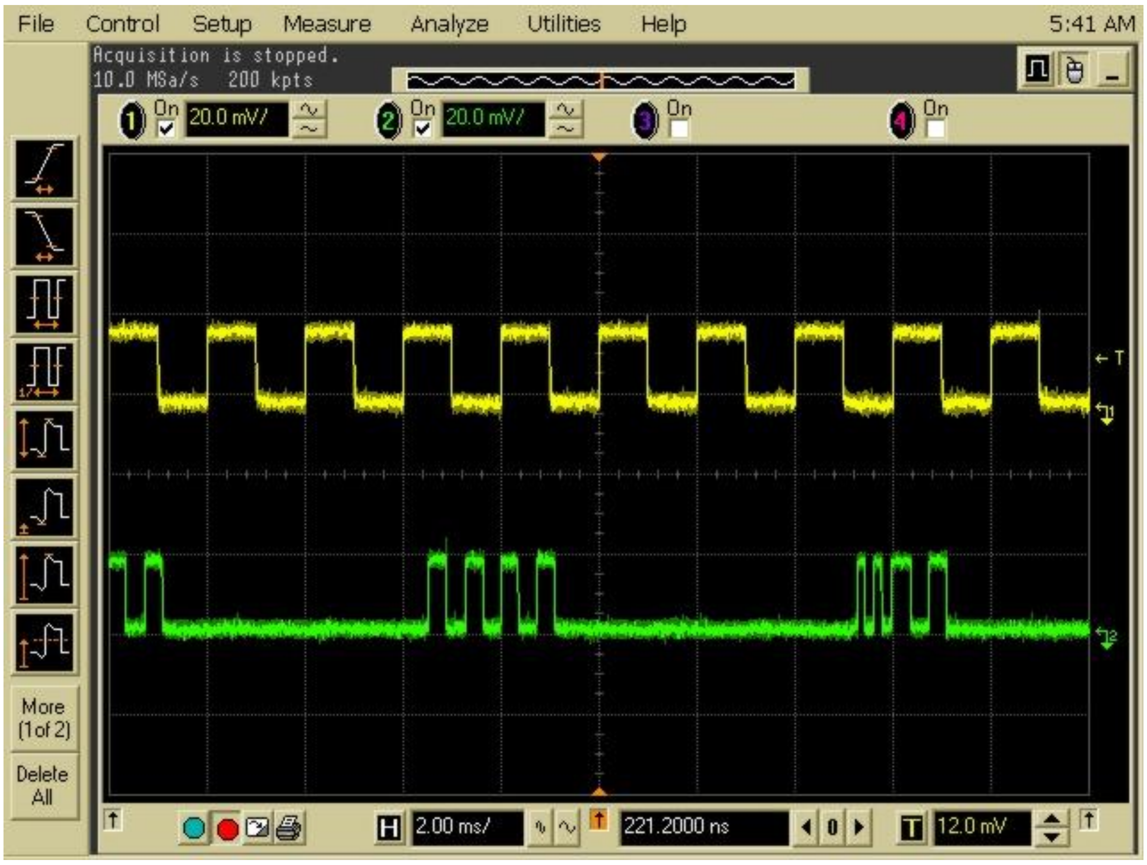


Figure 3. CE transmitting packets in burst

In this situation the delay between the transmitted and received packets is measured comparing two Wireshark[1] captures. One capture contains the packets sent by the transmitter BBB and the other one the packets received by the receiver BBB.

To capture these frames a switch with port mirroring capabilities is connected between the BBBs and the SB. The mirroring port of each switch is connected to a PC with the Wireshark program. Furthermore, to synchronize both captures, a well-known broadcast frame is sent to the PCs that take the frame capture, before the transmitter BBB starts to send packets. Both captures are post-processed in order to calculate the latency of each packet, and to determine the number of lost and duplicated packets that arrive at the receiver BBB.

As pointed out in the previous subsection, the messages sent by the transmitter BBB have a 2 byte sequence number that is used by the receiver BBB in order to detect messages dropped or duplicated.

This measurement setup does not measure the latencies with a precision higher than 5 ms. In contrast with the setup described in Subsection 2.1.1.1 it allows the analysis of the system when the CE is sending the packets in burst due to being close to its throughput limits.

[1] **Wireshark** is a free and open source packet analyzer. It is used for network troubleshooting, analysis, software and communication protocol development, and education

2.1.2 Tests without CE

The first measure is carried over the original topologies shown in Figure 1 in order to measure the latencies without the influence of the CE. These measures are done with the setup described in Subsection 2.1.1.1. In Table 1, the latencies measured for different cycle times are shown.

Table 1. Delay results without CE.

Cycle time	Topology	Latency [seconds]	Jitter [seconds]	Standard deviation [seconds]	Lost packets	Duplicated packets
200ms	1	2.132×10^{-4}	1.6×10^{-4}	3.9×10^{-5}	0	0
	2 (BBB1-BBB 3) (BBB2-BBB 3)	1.90×10^{-4}	1.6×10^{-4}	3.55×10^{-5}	0	0
		1.37×10^{-4}	1×10^{-4}	2.71×10^{-5}	0	0
	3 (BBB1-BBB 3) (BBB2-BBB 4)	2.23×10^{-4}	7.2×10^{-4}	6.01×10^{-5}	0	0
		1.44×10^{-4}	3.6×10^{-4}	2.93×10^{-5}	0	0
100ms	1	1.94×10^{-4}	1.6×10^{-4}	3.77×10^{-5}	0	0
	2	1.86×10^{-4}	2.2×10^{-4}	3.19×10^{-5}	0	0
		1.21×10^{-4}	2.2×10^{-4}	2.58×10^{-5}	0	0
	3	1.99×10^{-4}	1.4×10^{-4}	4.27×10^{-5}	0	0
		1.25×10^{-4}	1×10^{-4}	2.62×10^{-5}	0	0
50ms	1	1.89×10^{-4}	1.2×10^{-4}	2.94×10^{-5}	0	0
	2	1.68×10^{-4}	1.1×10^{-4}	3.12×10^{-5}	0	0
		1.13×10^{-4}	1×10^{-4}	2.68×10^{-5}	0	0
	3	1.87×10^{-4}	1.4×10^{-4}	7.21×10^{-5}	0	0
		1.12×10^{-4}	1.4×10^{-4}	2.55×10^{-5}	0	0
25ms	1	1.90×10^{-4}	1×10^{-4}	2.38×10^{-5}	0	0
	2	1.64×10^{-4}	4.5×10^{-4}	3.02×10^{-5}	0	0
		1.02×10^{-4}	1.2×10^{-4}	2.40×10^{-5}	0	0
	3	1.75×10^{-4}	5.4×10^{-4}	4.66×10^{-5}	0	0
		1.07×10^{-4}	1.2×10^{-4}	2.45×10^{-5}	0	0
10ms	1	1.87×10^{-4}	2×10^{-5}	1.85×10^{-5}	0	0
	2	1.61×10^{-4}	2×10^{-4}	3.11×10^{-5}	0	0
		1.02×10^{-4}	2×10^{-4}	3.40×10^{-5}	0	0
	3	1.72×10^{-4}	1.9×10^{-3}	1.56×10^{-4}	0	0

Cycle time	Topology	Latency [seconds]	Jitter [seconds]	Standard deviation [seconds]	Lost packets	Duplicated packets
		1.02×10^{-4}	1.6×10^{-3}	1.01×10^{-4}	0	0
5ms	1	1.76×10^{-4}	2×10^{-5}	1.97×10^{-5}	0	0
	2	1.67×10^{-4}	1.2×10^{-4}	2.76×10^{-5}	0	0
		9.72×10^{-5}	2.2×10^{-4}	3.64×10^{-5}	0	0
	3	1.79×10^{-4}	1.6×10^{-3}	1.13×10^{-4}	0	0
		9.50×10^{-5}	6×10^{-5}	2.06×10^{-5}	0	0
1ms	1	1.61×10^{-4}	2×10^{-5}	6.92×10^{-6}	0	0
	2	1.85×10^{-4}	1.1×10^{-3}	1.14×10^{-4}	0	0
		1.13×10^{-4}	1.2×10^{-3}	1.25×10^{-4}	0	0
	3	2.05×10^{-4}	1.5×10^{-3}	1.87×10^{-4}	0	0
		9.40×10^{-5}	2×10^{-5}	1.91×10^{-5}	0	0
500µs	1	1.56×10^{-4}	3.4×10^{-4}	2.01×10^{-5}	0	0
	2	1.75×10^{-4}	3.2×10^{-4}	3.07×10^{-5}	0	0
		8.72×10^{-5}	2.6×10^{-4}	2.52×10^{-5}	0	0
	3	1.70×10^{-4}	1.2×10^{-4}	1.93×10^{-5}	0	0
		8.87×10^{-5}	4×10^{-5}	1.66×10^{-5}	0	0

From these results, several conclusions can be drawn. On the one hand the mean delay measured is about 1-2ms. It can be observed that in some cases the jitter of this delay is as high as almost 2ms, this is due to the soft real-time behaviour of the BBB. Furthermore, it has been observed that in few occasions (shown in the standard deviation) the receiver BBB suffers a delay of about 2ms when it has to process a packet. That means that the jitter measured in the different tests using the setup described in Subsection 2.1.1.1 can be affected by an increment of 2ms. Finally, as it was expected, no packets are lost or duplicated.

2.1.3 Tests with local RTI

While the results of the latencies of the basic topologies without the presence of the CE are presented in 2.1.2, in this chapter the results of the latency and jitter measurements with the topologies shown in Figure 4 follow. In this case, the latencies include the delay introduced by the CE when all the elements (BBB, CE SBs, switches and CE central) are connected directly, without the presence of an heterogeneous network.

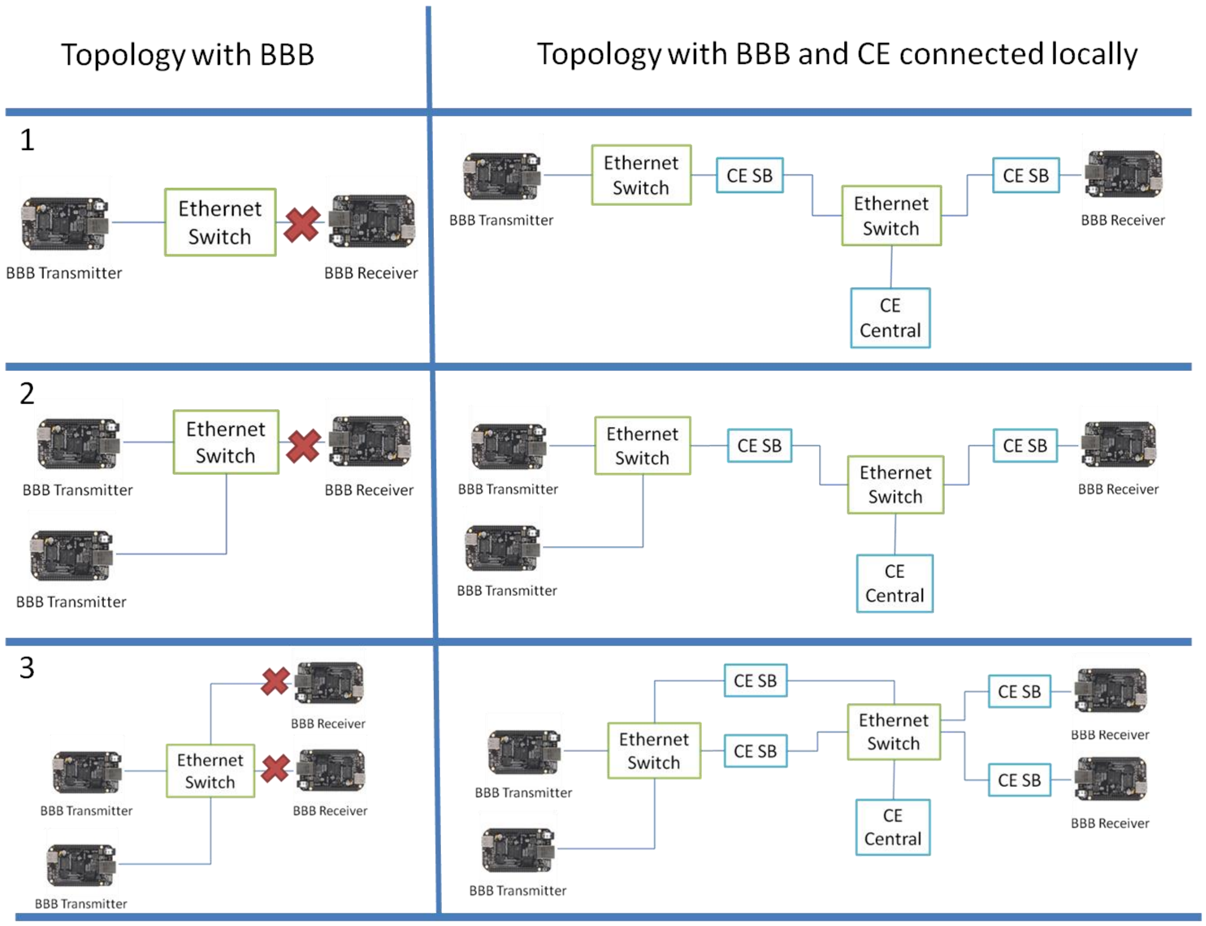


Figure 4. Basic topologies with BBBs and CE connected locally.

At a first stage, the measures were carried out with high transmission cycle times and with the setup described in subchapter 2.1.1.1. In Table 2 the latencies measured for different cycle times are shown.

Table 2. Delay results with CE and high transmission cycle times.

Cycle time	Topology	Latency [seconds]	Jitter [seconds]	Standard deviation [seconds]	Lost packets	Duplicated packets
200ms	1	6.3×10^{-3}	8.2×10^{-3}	1.9×10^{-3}	0	0
	2	6.7×10^{-3}	5.8×10^{-3}	1.7×10^{-3}	0	0
		6.6×10^{-3}	12.4×10^{-3}	2.1×10^{-3}	0	0
	3	5.7×10^{-3}	4.05×10^{-3}	1.9×10^{-3}	0	0
		6.7 × 10-3	5.4 × 10-3	2.1×10^{-3}	0	0
100ms	1	6.1×10^{-3}	6.1×10^{-3}	2.0×10^{-3}	0	0
	2	6.1×10^{-3}	4.2×10^{-3}	2.1×10^{-3}	0	0
		5.8×10^{-3}	8.4×10^{-3}	2.1×10^{-3}	0	0

Cycle time	Topology	Latency [seconds]	Jitter [seconds]	Standard deviation [seconds]	Lost packets	Duplicated packets
	3	6.0×10^{-3}	6.25×10^{-3}	2.2×10^{-3}	0	0
		6.2×10^{-3}	4.5×10^{-3}	2.4×10^{-3}	0	0
50ms	1	6.1×10^{-3}	5.8×10^{-3}	2.4×10^{-3}	0	0
	2	7.1×10^{-3}	5.9×10^{-3}	2.3×10^{-3}	0	0
		6.9×10^{-3}	3.5×10^{-3}	2.2×10^{-3}	0	0
	3	6.6×10^{-3}	5.6×10^{-3}	2.4×10^{-3}	0	0
		6.1×10^{-3}	5.65×10^{-3}	2.6×10^{-3}	0	0
25ms	1	6.2×10^{-3}	5.9×10^{-3}	2.4×10^{-3}	0	0
	2	6.4×10^{-3}	9.1×10^{-3}	2.9×10^{-3}	0	0
		6.3×10^{-3}	5.6×10^{-3}	2.8×10^{-3}	0	0
	3	6.1×10^{-3}	5.9×10^{-3}	2.3×10^{-3}	0	0
		6.2×10^{-3}	13.55×10^{-3}	2.4×10^{-3}	0	0
10ms	1	6.6×10^{-3}	11.2×10^{-3}	3.2×10^{-3}	0	0
	2	7.4×10^{-3}	5.1×10^{-3}	2.0×10^{-3}	0	0
		6.3×10^{-3}	4.1×10^{-3}	2.3×10^{-3}	0	0
	3	5.8×10^{-3}	11.3×10^{-3}	2.6×10^{-3}	0	0
		6.7×10^{-3}	6.1×10^{-3}	2.4×10^{-3}	0	0

The mean delay measured for the 3 topologies is about 6ms and with a standard deviation about 2ms. The higher jitter measured was 13.55ms. It can be observed that the delays are always higher for topology 2. It seems that the delay increases in case a simulation bridge has to receive traffic from two different sources. At a second stage the measures are done with low transmission cycle times and with the setup described in Section 2.1.1.2. In Table 3 the latencies measured for different cycle times are shown.

Table 3.Delay results with CE and low transmission cycle times.

Cycle time	Topology	Latency (seconds)	Jitter (seconds)	Standard deviation (seconds)	Lost packets	Duplicated packets
25ms	1	6.69×10^{-3}	17×10^{-3}	3.39×10^{-3}	0	0
	2	8.38×10^{-3}	23.5×10^{-3}	4.58×10^{-3}	0	0
		10.85×10^{-3}	27.5×10^{-3}	9.40×10^{-3}	0	0
	3	4.12×10^{-3}	16×10^{-3}	2.77×10^{-3}	0	0
		8.59×10^{-3}	17.5×10^{-3}	2.87×10^{-3}	0	0
10ms	1	6.50×10^{-3}	20×10^{-3}	2.46×10^{-3}	0	0

Cycle time	Topology	Latency (seconds)	Jitter (seconds)	Standard deviation (seconds)	Lost packets	Duplicated packets
	2	9.54×10^{-3}	23.5×10^{-3}	8.08×10^{-3}	0	0
		11.33×10^{-3}	26.5×10^{-3}	9.63×10^{-3}	0	0
	3	5.97×10^{-3}	19×10^{-3}	2.93×10^{-3}	0	0
		8.64×10^{-3}	14.3×10^{-3}	2.53×10^{-3}	0	0
5ms	1	10.04×10^{-3}	24×10^{-3}	5.1×10^{-3}	0	0
	2	15.25×10^{-3}	20×10^{-3}	9.31×10^{-3}	0	1
		17.99×10^{-3}	20×10^{-3}	10.21×10^{-3}	0	2
	3	9.41×10^{-3}	26×10^{-3}	5.69×10^{-3}	1	0
		11.08×10^{-3}	17.5×10^{-3}	5.01×10^{-3}	0	0
1ms	1	11.66×10^{-3}	23×10^{-3}	3.68×10^{-3}	3	0
	2	8.60×10^{-3}	21×10^{-3}	1.89×10^{-3}	8	0
		9.00×10^{-3}	24×10^{-3}	2.15×10^{-3}	0	0
	3	10.49×10^{-3}	24×10^{-3}	3.9×10^{-3}	0	0
		12.29×10^{-3}	23×10^{-3}	3.8×10^{-3}	1	2
500µs	1	9.01×10^{-3}	9×10^{-3}	1.65×10^{-3}	0	0
	2	9.92×10^{-3}	5.5×10^{-3}	1.14×10^{-3}	0	0
		9.96×10^{-3}	24×10^{-3}	1.35×10^{-3}	0	0
	3	7.60×10^{-3}	9.5×10^{-3}	1.7×10^{-3}	0	0
		10.01×10^{-3}	9×10^{-3}	1.55×10^{-3}	0	0

As mentioned before, this measurement setup does not measure the latencies with a precision higher than 5ms due to the error introduced by the port mirroring. That is why the jitter and in some cases the delay and the standard deviation are higher than the ones measured with the setup described in Section 2.1.1.1. It is worth to mention that in these tests some packets were lost (worse case: 8 of 30.000 packets) or duplicated (worse case: 2 of 30.000 packets).

In Table 3, it can be observed that when the period of the traffic is less than 10ms, the SB manages the traffic in burst. That is why the delay, jitter and standard deviation goes up a few milliseconds. In order to find the maximum throughput of the system, taking into account only topology 1, the transmission cycle time of the transmitter BBB has been reduced to 500µs, 250µs, 200µs and 150µs.

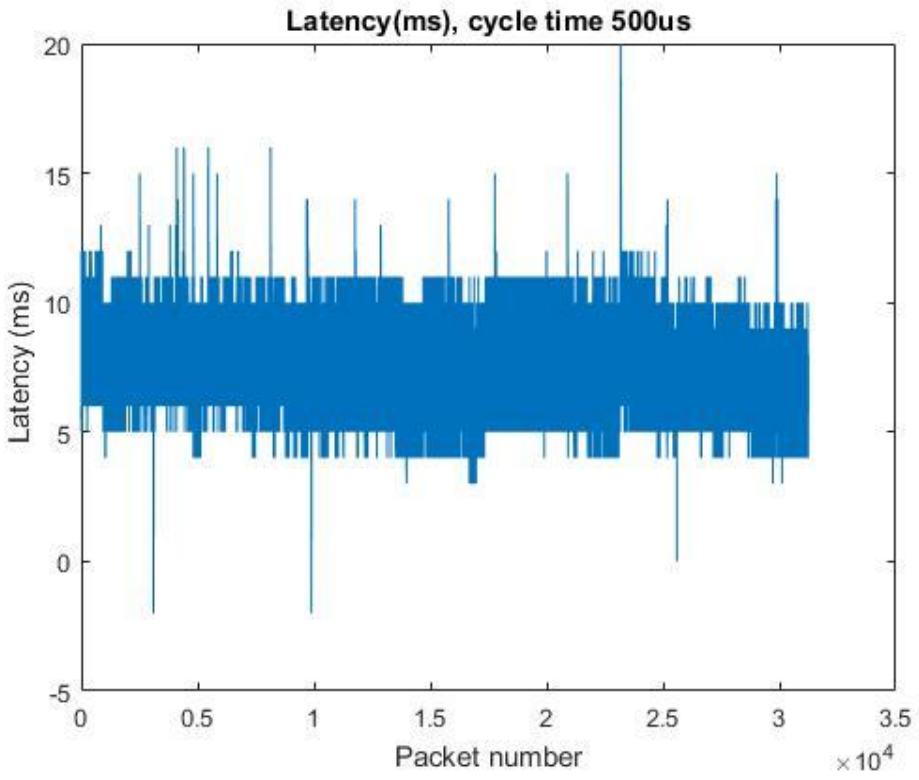


Figure 5. Latency in millisecond range when the cycle time of the transmitter BBB is 500µs.

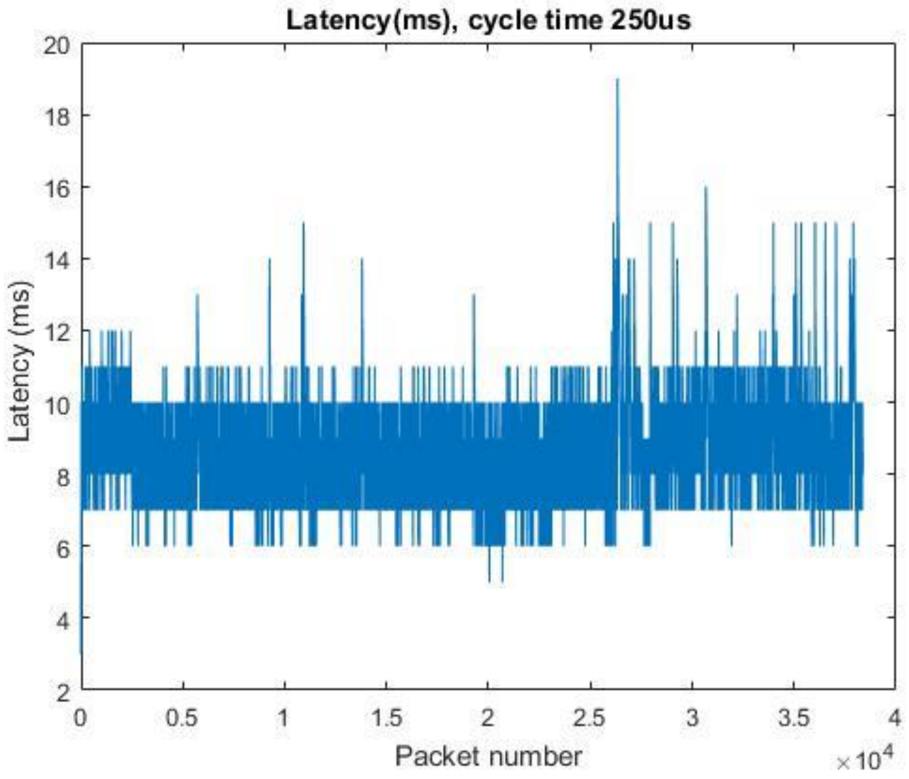


Figure 6. Latency in millisecond range when the cycle time of the transmitter BBB is 250µs.

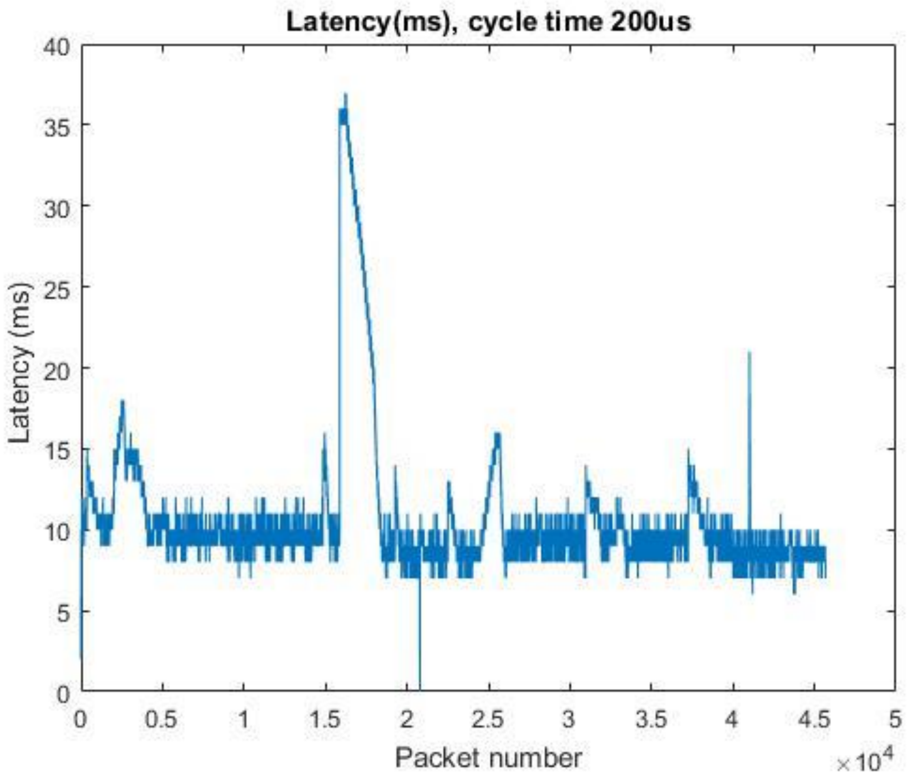


Figure 7. Latency in millisecond range when the cycle time of the transmitter BBB is 200µs.

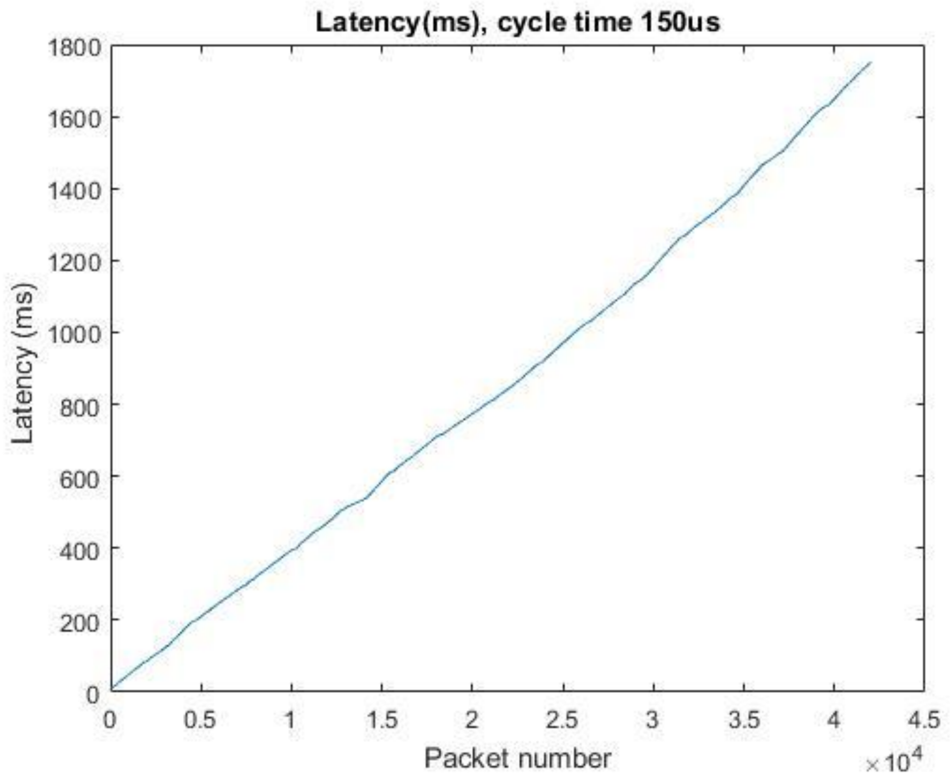


Figure 8. Latency in millisecond range when the cycle time of the transmitter BBB is 150µs.

As it can be observed in the images shown in Figure 5 and Figure 6, the latency for cycle times of 500µs and 250µs keep a stable value of the latency over the test.

Figure 7 represents the latency when the cycle time is 200µs. In the middle of the test the value of the latency started growing up in a peak although later the latency goes back to its normal value. Moreover, it can be observed in Figure 8 that the value of the latency increased over the test. This figure represents the latency when the cycle time is 150µs. The behaviour is produced because the number of packets received by the SB reached its throughput limit. No frames are lost due to the big size of the PCAP buffer of the SB, however if the duration of the test is increased, the SB would start to loose packets.

Summarizing it can be said that the throughput limit of the SB is between 200 packets/µs and 150 packets/µs.

In order to analyse the effect of sending and receiving packets in the same SB, the BBBs are modified in order to send and receive traffic. The measures are carried out only for topology 1.

Table 4. Delay results with CE, high transmission cycle times, and bidirectional traffic.

Cycle time	Topology 1 Direction	Latency [seconds]	Jitter [seconds]	Standard deviation [seconds]	Lost packets	Duplicated packets
200ms	BBB1→BBB2	7.9×10^{-3}	4.6×10^{-3}	1.5×10^{-3}	0	0
	BBB1←BBB2	7.0×10^{-3}	5.4×10^{-3}	1.6×10^{-3}	0	0
100ms	BBB1→BBB2	8.2×10^{-3}	5.75×10^{-3}	2×10^{-3}	0	0
	BBB1←BBB2	8.2×10^{-3}	6.45×10^{-3}	2.2×10^{-3}	0	0
50ms	BBB1→BBB2	7.7×10^{-3}	5.5×10^{-3}	2.5×10^{-3}	0	0
	BBB1←BBB2	7.9×10^{-3}	5×10^{-3}	2.3×10^{-3}	0	0
25ms	BBB1→BBB2	7.3×10^{-3}	9.15×10^{-3}	2.7×10^{-3}	0	0
	BBB1←BBB2	7.0×10^{-3}	7.9×10^{-3}	2.7×10^{-3}	0	0
10ms	BBB1→BBB2	7.8×10^{-3}	6.9×10^{-3}	2.5×10^{-3}	0	0
	BBB1←BBB2	7.0×10^{-3}	7.8×10^{-3}	2.5×10^{-3}	0	0

Table 5. Delay results with CE, low transmission cycle times, and bidirectional traffic.

Cycle time	Topology 1 Direction	Latency [seconds]	Jitter [seconds]	Standard deviation [seconds]	Lost packets	Duplicated packets
25ms	BBB1→BBB2	8.19×10^{-3}	13×10^{-3}	2.5×10^{-3}	0	0
	BBB1←BBB2	4.83×10^{-3}	7×10^{-3}	2.45×10^{-3}	0	0
10ms	BBB1→BBB2	7.69×10^{-3}	13.5×10^{-3}	3×10^{-3}	0	0
	BBB1←BBB2	5.54×10^{-3}	13.5×10^{-3}	2.80×10^{-3}	0	0
5ms	BBB1→BBB2	8.56×10^{-3}	13.5×10^{-3}	2.92×10^{-3}	0	0
	BBB1←BBB2	7.41×10^{-3}	9×10^{-3}	2.53×10^{-3}	0	0

Cycle time	Topology 1 Direction	Latency [seconds]	Jitter [seconds]	Standard deviation [seconds]	Lost packets	Duplicated packets
1ms	BBB1→BBB2	10.04×10^{-3}	8.5×10^{-3}	2.94×10^{-3}	0	0
	BBB1←BBB2	8.81×10^{-3}	26.5×10^{-3}	3.4×10^{-3}	4	0
500µs	BBB1→BBB2	8.31×10^{-3}	6.5×10^{-3}	1.63×10^{-3}	0	0
	BBB1←BBB2	7.69×10^{-3}	28×10^{-3}	1.86×10^{-3}	17	0

It can be observed that the delay increases (1-2ms) when the SBs have to manage sent and received traffic. It is worth to mention that in these tests no packet is duplicated and for 500µs packet cycle time some packets are lost (17 of 30.000 packets).

2.1.4 Tests with remote RTI

In the previous Section 2.1.3 the latency introduced by the CE is analysed in scenarios in which all the elements, BBB, SB and the CE central, are placed locally and connected without the presence of a heterogeneous network. In this section the same measurements are repeated considering a scenario in which the CE central is located remotely (Siegen, Germany) with respect to the BBBs and SBs (located in Mondragon, Spain) as shown in Figure 9.

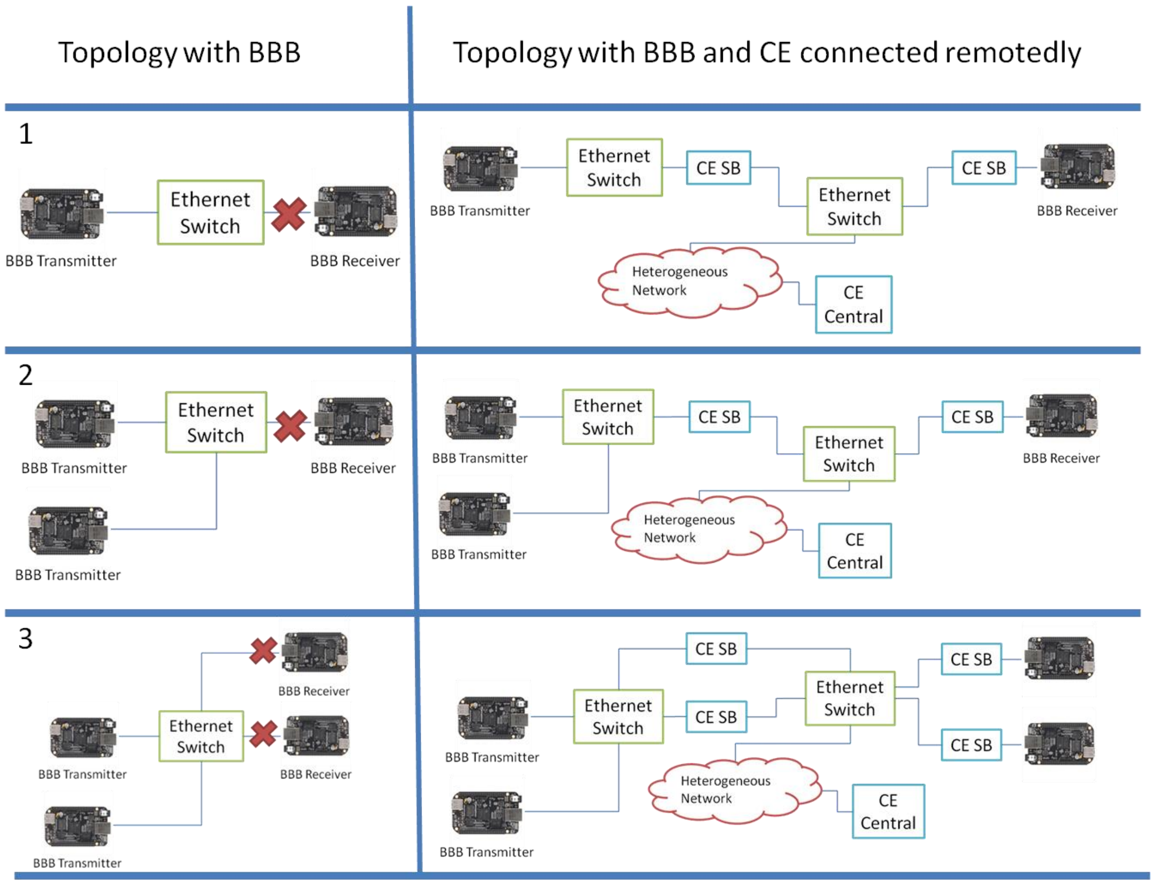


Figure 9. Basic topologies without CE.

At a first stage, the measures are carried out with high transmission cycle times and with the setup described in Section 2.1.1.1. In Table 6, the latencies measured for different cycle times are shown.

Table 6. Delay results with CE Central PC remotely located and high transmission cycle times.

Cycle time	Topology	Latency (seconds)	Jitter (sec-onds)	Standard deviation (seconds)	Lost packets	Duplicated packets
200ms	1	59.6×10^{-3}	10.45×10^{-3}	2.1×10^{-3}	0	0
	2	59.5×10^{-3}	15.4×10^{-3}	2.6×10^{-3}	0	0
		59.7×10^{-3}	7.05×10^{-3}	2.3×10^{-3}	0	0
	3	60×10^{-3}	29.4×10^{-3}	3.9×10^{-3}	0	0
		60.6×10^{-3}	7.15×10^{-3}	2.0×10^{-3}	0	0
100ms	1	59.8×10^{-3}	14.5×10^{-3}	2.3×10^{-3}	0	0
	2	61×10^{-3}	26.6×10^{-3}	4.2×10^{-3}	0	0
		88.2×10^{-3}	48.35×10^{-3}	24.4×10^{-3}	0	0
	3	61.5×10^{-3}	47.55×10^{-3}	4.5×10^{-3}	0	0
		61.0×10^{-3}	47.5×10^{-3}	3.9×10^{-3}	0	0

At a second stage the measures are performed with low transmission cycle times and with the setup described in Section 2.1.1.2. In Table 7, the latencies measured for different cycle times are shown.

Table 7. Delay results with CE Central PC remotely located and low transmission cycle times.

Cycle time	Topology	Latency (seconds)	Jitter (sec-onds)	Standard deviation (seconds)	Lost packets	Duplicated packets
50ms	1	62.61×10^{-3}	35.50×10^{-3}	3.85×10^{-3}	0	0
	2	65.95×10^{-3}	66.5×10^{-3}	9.29×10^{-3}	0	0
		65.88×10^{-3}	37.5×10^{-3}	7.17×10^{-3}	0	0
	3	54.14×10^{-3}	53×10^{-3}	4.52×10^{-3}	0	0
		56.37×10^{-3}	52.5×10^{-3}	3.96×10^{-3}	0	0
25ms	1	61.25×10^{-3}	62.50×10^{-3}	4.26×10^{-3}	0	0
	2	71.67×10^{-3}	56.55×10^{-3}	6.47×10^{-3}	0	0
		71.45×10^{-3}	56.4×10^{-3}	6.37×10^{-3}	0	0
	3	56.12×10^{-3}	49×10^{-3}	4.64×10^{-3}	0	0
		57.37×10^{-3}	48.5×10^{-3}	4.36×10^{-3}	0	0

Cycle time	Topology	Latency (seconds)	Jitter (sec-onds)	Standard deviation (seconds)	Lost packets	Duplicated packets
10ms	1	60.57×10^{-3}	42.50×10^{-3}	3.36×10^{-3}	0	0
	2	65.25×10^{-3}	40×10^{-3}	7.18×10^{-3}	0	0
		65.29×10^{-3}	44×10^{-3}	6.26×10^{-3}	0	0
	3	57.82×10^{-3}	38×10^{-3}	4.31×10^{-3}	0	0
		59.18×10^{-3}	42.5×10^{-3}	4.08×10^{-3}	0	0
5ms	1	66.25×10^{-3}	42.50×10^{-3}	7.41×10^{-3}	0	0
	2	64.39×10^{-3}	41.5×10^{-3}	5.92×10^{-3}	0	0
		65.34×10^{-3}	40.5×10^{-3}	5.74×10^{-3}	0	0
	3	62.71×10^{-3}	37×10^{-3}	6×10^{-3}	0	0
		62.59×10^{-3}	36.5×10^{-3}	5.7×10^{-3}	0	0
1ms	1	61.58×10^{-3}	45.50×10^{-3}	5.37×10^{-3}	0	0
	2	61.36×10^{-3}	13×10^{-3}	2.39×10^{-3}	0	0
		62.25×10^{-3}	14×10^{-3}	2.65×10^{-3}	0	0
	3	59.42×10^{-3}	15×10^{-3}	2.47×10^{-3}	0	0
		59.62×10^{-3}	19×10^{-3}	2.14×10^{-3}	0	0
500µs	1	61.97×10^{-3}	22.50×10^{-3}	4.35×10^{-3}	0	0
	2	63.19×10^{-3}	15.5×10^{-3}	3.14×10^{-3}	0	0
		63.00×10^{-3}	21.5×10^{-3}	3.52×10^{-3}	0	0
	3	59.82×10^{-3}	23×10^{-3}	4.89×10^{-3}	0	0
		61.19×10^{-3}	53.5×10^{-3}	4.46×10^{-3}	0	0

It can be observed that the delay introduced by the CE is about 60ms when CE central is located remotely and connected through and heterogeneous network. It is worth to mention that no packet is lost or duplicated in these tests.

2.2 Evaluation and temporal characteristics of simulation bridges

The evaluation presented in this Section is based on a fan example [10] as controllable, well-known test setup. The focus lies on the simulation bridges and the CESB library. First we consider the Functional Mock-Up Interface (FMI) as interface to the subsystems of the test setup (Section 2.2.2, taken from [10]). Afterwards, the usage of PCAP follows. PCAP is a library for Windows and Linux to capture packets on a network interface of a PC. In this set-up the fan model is executed on a real hardware device and the voter is executed as software (Section 2.2.3). The fault-injection based on the EN 50159 standard [8] (Section 2.2.4) and the management of delays based on state-estimation (Section 2.2.5) are evaluated in the subsequent sections.

2.2.1 Description of test setup

In a first step, the temporal characteristics of the simulation bridges are evaluated using a fault-tolerant closed-loop control algorithm of a fan as shown in Figure 10. Later the evaluation is also performed on a real train application. A PID controller (Proportional, Integral and Derivative controller) receives the current speed of a fan and a reference speed set by a user. It calculates the difference between both values and uses it as input for the PID algorithm. A triple modular redundancy concept [9] is used to reach fault-tolerance. In such a setup, the PID controller is triplicated and connected to a voter. This subsystem receives the PID outputs from all three PID controllers. Based on a majority election, the setup is able to detect a failure in one PID controller. If two controllers fail, all inputs differ and the voter puts the fan into a safe state by running it at the maximum speed.

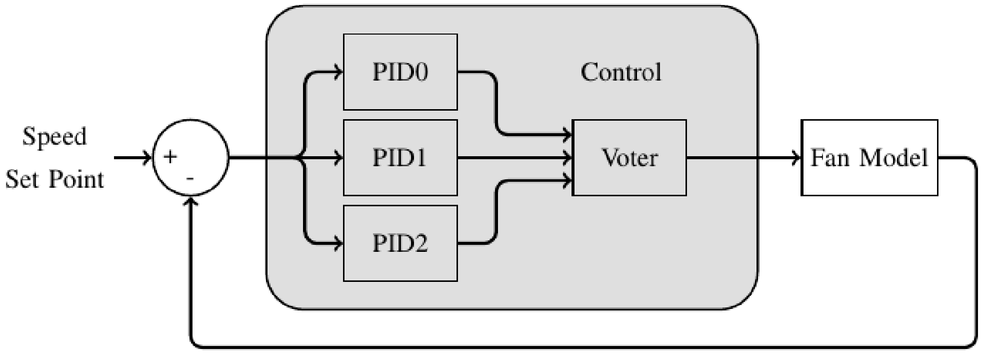


Figure 10. Fan control with triple modular redundancy [10].

The fan speed is calculated based on the model of an Intel E97379-001 fan. A basic speed is increased by a controllable portion. The two portions are modelled as first-order delay elements using the following equations. While the first equation represents the base speed, the second one denotes the controllable portion. At runtime, both equations are added resulting in the final speed.

$$a(t) = 10 * \left(1 - e^{\frac{-t}{0.46}}\right) \; and \; G(s) = \frac{10}{1 + 0.46s}$$

$$a(t) = 10 * \left(1 - e^{\frac{-t}{0.87}}\right) \; and \; G(s) = \frac{10}{1 + 0.87s}$$

The evaluation is performed using three different topologies. Figure 11 shows the first topology where all simulations, simulation bridges and the RTI are running on the same PC. In Figure 12, the RTI is moved to a remote PC which is connected either via a LAN or the Internet. In Figure 13, the RTI is distributed in a hierarchical manner. The PID controllers and the fan model are connected to a local RTI instance. This instance communicates with its remote, parent RTI via a network similar to the voter.

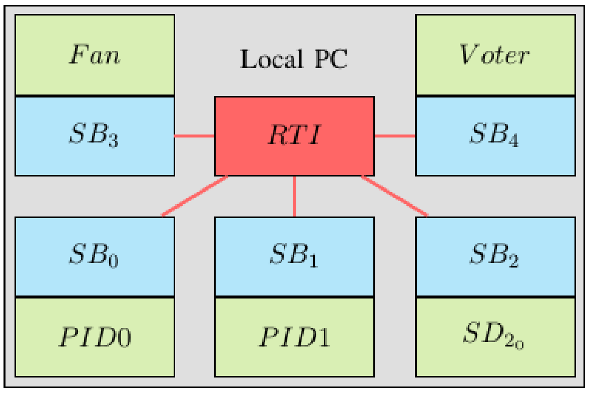


Figure 11. Local setup [10].

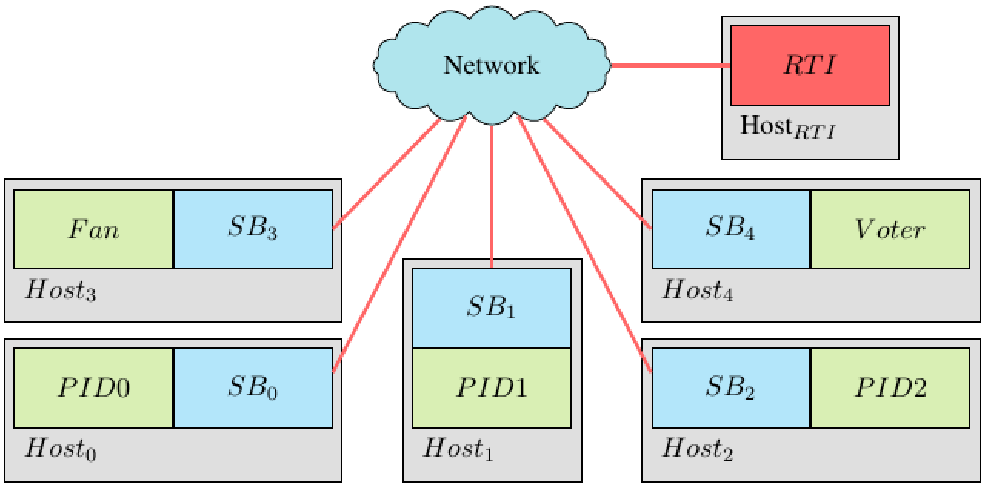


Figure 12. Communication via LAN or the Internet [10].

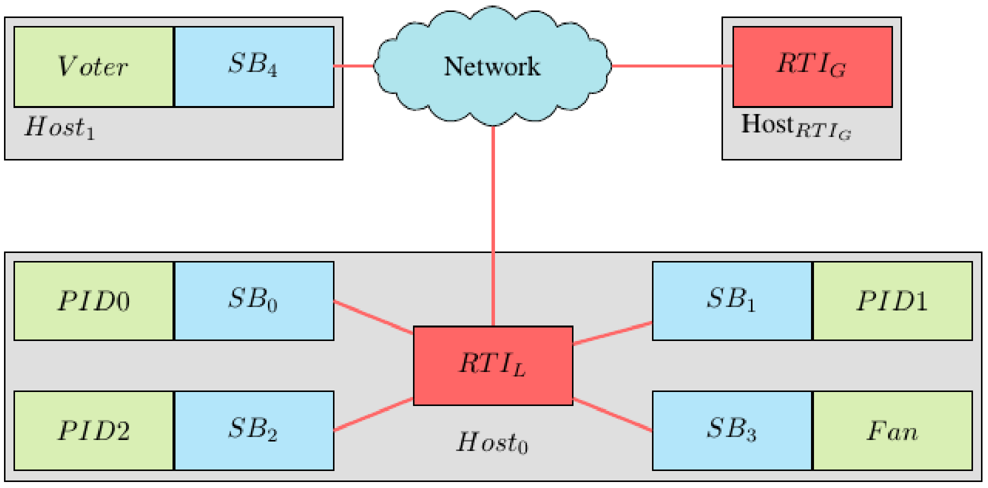


Figure 13. Example for hierarchical RTI [10].

Each subsystem of the fan communicates with the other ones based on a periodic communication schedule. The period is set to 10ms and there is one offset in the period assigned to each message when it has to be sent. The fan model calculates the new speed at offset 1ms and sends it at 2ms as multicast. Afterwards, the PID controllers calculate the new set point at 3ms (PID0), 4ms (PID1) and 5ms (PID2) and send their messages 1ms later. At offset 6ms, the voter compares the inputs and sends the result to the fan model at 8ms. A real fan would set the new speed now. Besides checking the PID outputs, the voter is further responsible to send a new destination. The period of this message is 10s and the message itself is sent at offset 9ms. This way, the message does not conflict with the ones sent in the control loop.

2.2.2 Evaluation FMI

FMI is a tool independent standard for co-simulation and model exchange [7], [29]. In our case, the capabilities of co-simulation are used where simulation tools or software applications can be connected to a master algorithm. This master algorithm realizes synchronization and data exchange which is performed by the simulation bridges. The connection is described in detail in Section 7.10 in D3.2, *Report on design of TCMS Distributed Simulation Framework Concept* [2] and Section 4.13.1 of D3.4, *Proof-of-concept implementation of distributed simulation framework concept* [4].

During the evaluation, the fan use-case is executed for 13 different simulation durations. They start with 1 second and continue from 10 seconds to 100 seconds in 10s-steps. In case of the local PC- and LAN-setups, each duration was executed 100 times. In the other cases, we decreased the number of execution runs to 20. To provide a tendency for longer simulation durations, we added two durations of 500 seconds and 1000 seconds for the local PC-setup. In the following, the simulation durations, the delays introduced by the simulation bridges as well as the influence of the communication network on the communication delay are analysed.

Figure 14, Figure 15 and Figure 16 show the simulation durations (placed on the y-axis) compared to the selected simulated time. The latter is shown on the x-axis. Figure 14 depicts the durations from 1s to 100s while Figure 15 denotes a tendency for longer simulation durations until 1000s. Figure 16 presents both Internet-cases in a scaled coordinate system, since using the Internet requires the application of a VPN to secure the communication. The application of security mechanisms in this combination introduces large communication delays.

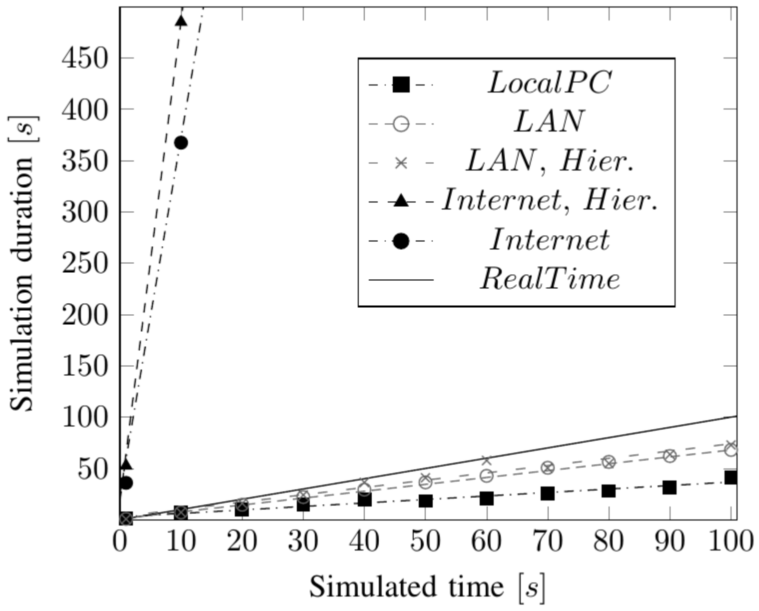


Figure 14. Simulation durations for simulated times until 100s [10].

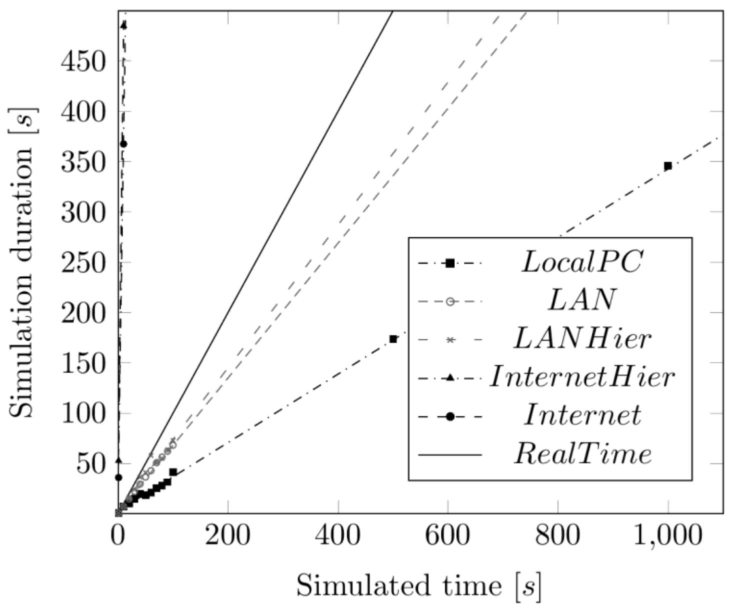


Figure 15. Simulation durations for simulated times until 1000s [10].

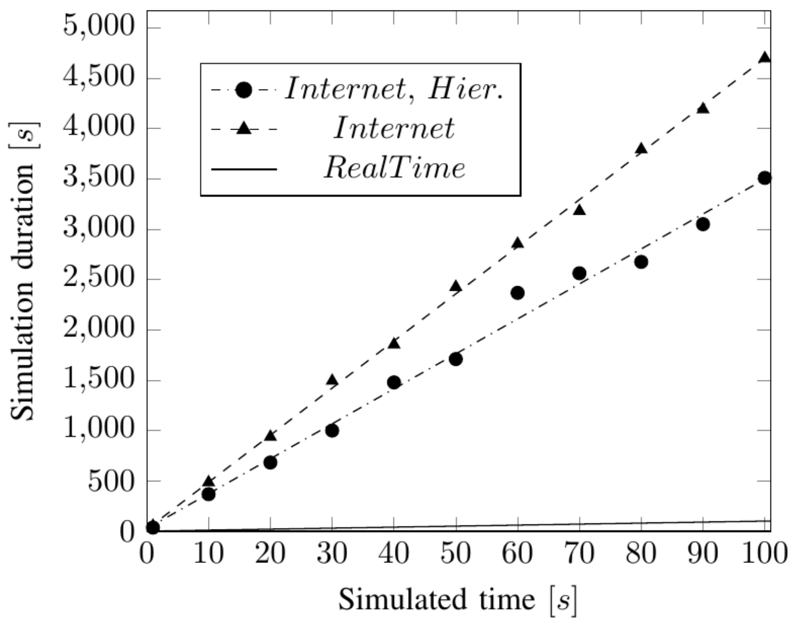


Figure 16. Simulation durations for simulated times until 100s if a VPN is used [10].

In each figure, the black dash-dotted graphs with filled square marks represent the local set-up, the grey ones the second topology in which the RTI is executed on a host in the same LAN (densely dashed lines, circle marks) and the hierarchical LAN-setup (loosely dashed lines, x-marks). A densely dashed line with filled triangle mark illustrates the hierarchical Internet-setups while the Internet-hierarchy-cases are shown as black, dash-dotted line with filled circle marks. Each mark represents the average of all simulation durations for the related simulated time. The black solid line denotes the case when the simulation duration equals the simulated time, therefore we call it *real-time*. Graphs below this line are executed faster than real-time, otherwise the simulation is executed slower.

As shown in the figures, there is a linear correlation between the simulation duration and the simulated time. The five linear regression lines shown in Table 8 represent this linearity.

Table 8. Regression lines for each setup [10].

Setup	Regression line
Local setup	$0.34t + 2.746$
RTI in LAN	$0.67t + 0.988$
Hierarchical RTI in LAN	$0.71t + 3.055$
Hierarchical RTI via Internet	$34.78t + 24.56$
Internet	$46.79t + 18.21$

The first three equations represent the local PC- and the LAN-setups. All of them are executed faster than real-time. Determining the round trip times of the communication using a ping the related network interface results in 0.05ms to 0.1ms for the local PC, while the LAN introduces delays of about 0.5ms. The influence of the Internet and the application of security services is shown in the Internet-cases where round trip times of 15ms to 30ms were measured. Compared to pinging the local interface, the delays are 300 times larger which is why the simulation durations in the Internet-cases are much longer. Distributing the RTI in a hierarchical manner provides benefits if the communication delays are large. This effect is shown in the equations for the hierarchical setups. In theory, there is less network traffic due to a local RTI instance. However, the performance is limited by the data exchange between the RTI instances which results in a high overhead. As the communication delays are small in a LAN, there is no benefit of a hierarchical solution.

Table 9 to Table 14 show the influence of the communication delay on the simulation duration. These delays are introduced by the heterogeneous communication network connecting the simulation bridges.

Table 9. Interaction delays between the simulation bridges in the different topologies [10].

Interaction	SensorRPM Fan - PID0	SensorRPM Fan – PID1	SensorRPM Fan – PID2	PWM0 PID0 - Voter	PWM1 PID1 - Voter	PWM2 PID2 - Voter	PWM Voter - Fan
Delay PC [ms]	0.395	0.387	0.387	0.527	0.832	1.219	0.630
Delay LAN [ms]	0.787	0.780	0.777	0.877	1.466	2.231	1.324
Delay Hier. LAN [ms]	0.867	0.923	0.907	1.109	1.557	2.443	1.041

Interaction	SensorRPM Fan - PID0	SensorRPM Fan – PID1	SensorRPM Fan – PID2	PWM0 PID0 - Voter	PWM1 PID1 - Voter	PWM2 PID2 - Voter	PWM Voter - Fan
Delay Hier. Internet [ms]	11.631	11.577	11.609	19.231	31.895	51.926	28.327
Delay Internet [ms]	47.892	47.224	46.641	48.838	94.327	143.990	91.065

In Table 9, the interaction delays are compared between the simulation bridges in milliseconds. There are five interactions: SensorRPM which is the multicast from the fan to the three PID-controllers, three interactions PWMi send from the PID-controllers to the voter. Finally, the interaction PWM (Pulse Width Modulation) closes the loop between voter and fan. The delays between the simulation bridges are depicted for each interaction and topology. The timestamps used for the calculation are taken in the communicating simulation bridges before sending an interaction and after receiving it. As a consequence, the delays include the delays introduced by the communication network as well as the time required for the HLA time management which synchronizes the bridges. This effect can be seen in case of the PWMi-interactions. Although the PID-controllers calculate the control values at the same time, the delay for PWM0 is much smaller than the one for PWM2. The reason is the reception of the interactions by the voter according to the schedule described in Section 2.2.1. Between each offset, the HLA time management synchronizes the time advance of the simulation bridges.

Comparing the equations with the results in Table 9, there is a similar relationship between the interaction delays and the simulation durations for each topology. The average fraction of the Internet delay divided by the delay of the LAN is 62.111. Using the delay of the local setup as divisor, the fractions are 1.911 (LAN), 2.080 (hierarchical RTI in LAN), 35.964 (hierarchical RTI in Internet) and 118.929 (Internet). The related fractions between the slopes of Equations 3 to 7 are 69.836 (Internet / LAN) 1.971 (LAN / Local), 2.088 (hierarchical RTI in LAN / Local), 102, 29 (hierarchical RTI in Internet / Local) and 137.618 (Internet / Local). The dimensions show the main influence of the communication even if the values are not exactly the same. This assumption is further proved by the analysis of the remaining tables.

The average pass-through times of the simulation bridges in µs are shown in Table 10 (receiving) and Table 11 (sending). Again, the times are depicted for each topology. On the incoming side of the simulation bridges, the packets have to traverse more modules than for sending. Hence, the pass-through times are longer. Compared to the local setup and the LAN-case, the VPN has a strong influence. Using it, all traffic is secured and routed to the related VPN server. Applying the security services requires processing power which affects the execution times of the bridges. However, the order of magnitude of the pass-through times is still much smaller than the interaction delays. The decreasing time for longer simulation times can be constituted with a smaller effect of outliers due to more inputs. In the monitoring files, there is a few number of pass-through times in the dimension of milliseconds instead of microseconds.

Table 10. Average pass-through time of the simulation bridges for receiving in µs [10].

Simulated time [s]	Local PC [µs]	LAN [µs]	Hier. LAN [µs]	Hier. Internet [µs]	Internet [µs]
1	67.848	66.199	56.711	129.960	164.236
10	63.189	57.075	53.810	173.459	147.863
20	61.913	61.554	55.204	79.968	161.994
30	62.076	61.712	54.002	80.723	145.340
40	61.422	61.655	57.185	80.238	129.933
50	50.216	54.760	54.001	81.562	149.553
60	48.429	56.431	58.601	81.740	130.794
70	49.701	55.343	57.080	195.886	149.933
80	48.458	54.598	56.671	80.954	150.600
90	48.489	52.858	55.603	82.375	134.215
100	47.698	53.420	57.244	82.040	150.495

Table 11. Average pass-through time of the simulation bridges for sending in µs [10].

Simulated time [s]	Local PC [µs]	LAN [µs]	Hier. LAN [µs]	Hier. Internet [µs]	Internet [µs]
1	61.410	53.160	49.118	119.318	146.618
10	54.515	44.967	44.514	146.233	125.755
20	47.747	47.739	45.779	73.849	135.102
30	47.450	47.441	45.207	74.455	120.380
40	46.803	47.092	49.192	73.711	107.733
50	39.498	41.931	45.371	74.524	124.002
60	38.261	42.912	50.877	74.426	107.480
70	38.939	42.038	45.651	161.078	123.126
80	38.150	41.215	44.923	73.845	123.033
90	38.224	40.339	44.988	74.041	109.842
100	38.757	40.651	46.336	74.400	122.018

To analyse the general influence of the simulation bridges pass-through times on the simulation durations, we calculated the delay sum and divided it by the duration. Table 12, Table 11Table 13 and Table 14 show the resulting fractions. The pass-through times in the local setup account for 13.6% (in average) of the overall duration. This fraction is reduced to 8.8% (LAN), 8.3% (Hierarchical LAN) and 0.3% (both Internet cases). Together with the previous results, these fractions elucidate the strong influence of the communication delays on the simulation duration.

Table 12. Fractions of simulation bridge delays compared to the simulation duration in the local PC-setup [10].

Simulated time [s]	Local PC		
	Delay [s]	Duration [s]	Fraction
1	0.079	0.826	0.096
10	0.6825	6.828	0.100
20	1.333	9.842	0.135
30	2.022	14.588	0.139
40	2.661	19.522	0.136
50	3.065	18.280	0.168
60	3.186	20.987	0.152
70	3.828	25.520	0.152
80	4.258	27.996	0.135
90	4.835	31.499	0.153
100	5.377	41.030	0.131

Table 13. Fractions of simulation bridge delays compared to the simulation duration in the LAN-setups [10].

Simulated time [s]	LAN			Hierarchical RTI in LAN		
	Delay [s]	Duration [s]	Fraction	Delay [s]	Duration [s]	Fraction
1	0.075	0.788	0.095	0.064	0.680	0.094
10	0.631	7.155	0.088	0.599	7.223	0.083

Simulated time [s]	LAN			Hierarchical RTI in LAN		
	Delay [s]	Duration [s]	Fraction	Delay [s]	Duration [s]	Fraction
20	1.343	14.741	0.091	1.231	15.919	0.077
30	1.999	21.906	0.091	1.812	24.072	0.075
40	2.684	29.659	0.091	2.585	36.358	0.071
50	3.008	33.653	0.089	3.024	40.919	0.074
60	3.698	41.248	0.090	3.988	57.748	0.069
70	4.113	47.469	0.087	4.395	50.434	0.087
80	4.739	55.736	0.085	4.971	54.885	0.091
90	4.878	62.550	0.078	5.528	63.634	0.087
100	5.877	66.706	0.088	6.324	72.872	0.087

Table 14. Fractions of simulation bridge delays compared to the simulation duration in the Internet setups [10].

Simulated time [s]	Internet			Hierarchical RTI in Internet		
	Delay [s]	Duration [s]	Fraction	Delay [s]	Duration [s]	Fraction
1	0.153	35.650	0.0043	0.188	52.633	0.0036
10	1.945	356.987	0.0040	1.664	484.952	0.0034
20	1.858	744.058	0.0025	3.619	936.809	0.0039
30	2.812	1030.94	0.0027	4.858	1492.54	0.0033
40	3.721	1309.71	0.0028	5.793	1853.26	0.0031
50	4.718	1608.41	0.0029	8.334	2426.03	0.0034
60	5.666	1967.28	0.0029	8.718	2853.82	0.0031
70	15.236	2684.17	0.0077	11.656	3179.66	0.0037
80	7.489	2892.25	0.0026	13.355	3791.84	0.0035
90	8.521	2882.68	0.0030	13.398	4190.04	0.0032
100	9.463	3528.94	0.0027	16.636	4697.01	0.0035

2.2.3 Evaluation PCAP

PCAP is a library to capture packets on a network interface of Linux or Windows PCs. It provides an API which is used in the simulation bridge as described in Section 4.13.2 of D3.4, "Proof-of-concept implementation of distributed simulation framework concept" [4]. In the PCAP evaluation, we executed the fan model on a ZYBO Zynq-7000 development board and the voter software on a PC. The ZYBO Zynq-7000 is a development board which hosts a Xilinx All Programmable System-on-Chip (AP SoC) architecture. This architecture integrates a dual-core ARM Cortex-A9 processor with a Xilinx Z-7010 field programmable gate array (FPGA). The fan is executed on the ARM processor and uses the on-board Ethernet interface for communication. As a first step, the fan model is connected to a simulation bridge using the PCAP library while the voter and the remaining subsystems are connected via FMI. In a second test, the voter is connected via PCAP instead. Furthermore, we distinguished between running the RTI on the local host and on a server in a LAN. In the following, we analyse the same aspects as in the previous section.

Figure 17 shows the simulation durations compared to the simulated times. Again, the simulated time is shown on the x-axis while the simulation duration is placed on the y-axis. There are four graphs. The black graphs represent the durations for connecting the voter via PCAP, the grey graphs the ones for connecting the fan. In both cases, the local setup is depicted using circle marks and dash-dotted lines while triangle marks and dashed lines constitute the LAN-cases.

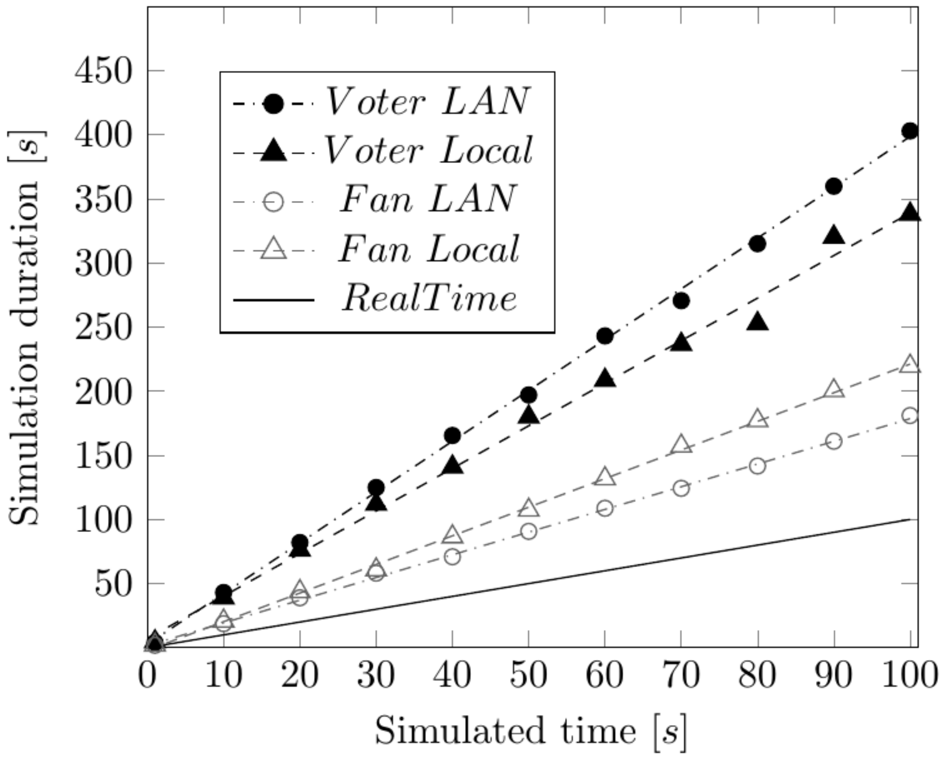


Figure 17. Simulation durations for connection via PCAP.

Similar to the connection via FMI, there is a linear relation between the simulated time and the simulation duration. However, neither in the local nor in the LAN-setups it is possible to achieve real-time requirements. The regression lines are calculated in Table 15.

Table 15. Regression lines for the setups using PCAP.

Setup	Regression line
Fan via PCAP, RTI on local host	$2.23t - 2.040$
Fan via PCAP, RTI in LAN	$1.77t + 1.652$
Voter via PCAP, RTI on local host	$3.32t + 7.053$
Voter via PCAP, RTI in LAN	$3.96t + 2.644$

In contrast to the results when the fan is connected to a simulation bridge via FMI, the simulation durations in case of PCAP and a local RTI are larger than the ones for an RTI in a LAN. This is not the case if the voter is connected via PCAP, here a local RTI leads to smaller simulation durations. The main goal of this test was to show the influence of the PCAP library on the simulation duration. The regression slopes in the local-setups are 6.5 (fan) and 9.5 (voter) times larger than using FMI. In the LAN-setups the factors are 2.6 (fan) and 5.9 (voter). These results show that real-time simulations are not feasible anymore if at least one device is connected via PCAP and the communication cycles are in the same order of magnitude as before.

If FMI is used to connect the devices, the main influence on the simulation duration are the communication delays introduced by the heterogeneous communication network. Comparing the simulation durations using FMI with those using PCAP, the library introduces significant delays which negatively influence the simulation duration. In the following we analyse in how far the interaction delays and pass-through times are affected.

Table 16. Interaction delays between the simulation bridges in the different test setups using PCAP.

Interaction	SensorRPM Fan - PID0	SensorRPM Fan – PID1	SensorRPM Fan – PID2	PWM0 PID0 - Voter	PWM1 PID1 - Voter	PWM2 PID2 - Voter	PWM Voter - Fan
Delay Fan Local [ms]	0.327	0.346	0.313	0.251	0.781	1.097	0.225
Delay Fan LAN [ms]	0.896	0.905	0.852	0.656	1.598	2.300	0.700
Delay Voter Local [ms]	0.716	0.661	0.739	0.655	7.054	13.884	0.784
Delay Voter LAN [ms]	0.574	0.685	0.575	0.374	7.502	14.439	0.471

Table 16 illustrates the interaction delays between the simulation bridges in the different test setups if the PCAP library is used to connect a real device or a software application. The interactions are the same like in Table 9. Furthermore, the timestamps used for the calculation are still taken in the communicating simulation bridges before sending an interaction and after receiving it.

The interaction delays in the case where the fan is connected and the RTI is executed on the same PC like the simulation bridges are shown in the first line. Compared to the local setup in Table 9, the delays of the SensorRPM-interaction are in the same order of magnitude, the PWM0 and PWM-interaction delays are even smaller. The second line in Table 16 shows the delays in the fan case if the RTI is running on another server in the LAN. Similar to the previous case, the delays differ less than 0.15ms compared to FMI. The only outlier is the PWM-interaction, here the delay is even almost half of the related one in Table 9. Line three in Table 16 illustrates the voter-case where the RTI is running on the local PC. The differences in the SensorRPM interaction-delays are analogical compared to the first two lines. However, the PWMi-interactions show the influence of the PCAP library on the simulation duration. As explained before, the PWMi-delays include the delays required to synchronize the simulation bridges and therefore they are also affected by the PCAP library.

The average pass-through times of the simulation bridges if PCAP is used are shown in µs in Table 17 (receiving) and Table 18 (sending). While the times were quite similar for all simulation bridges if FMI was used, we have to differentiate between the bridges in this case. Only one device (fan) and one application (voter) are connected via PCAP at one time, the other models are connected via FMI. Hence, the pass-through times differ a lot. Furthermore, we calculate the average of these times independent from the selected simulated time.

It is obvious that PCAP influences the pass-through times is quite strong, while the sending times are much more affected than the receiving ones. One reason is the inclusion of another thread which captures and handles the received packets. This thread is scheduled every 100µs wherefore the Operating System has to perform a context switch quite often. Since all parts of the simulation execution run on the same PC in these tests, the context switches are one reason for the increased pass-through times. Another one is the usage of the PCAP functions which can be seen best on the sending side. Capturing a packet and handling it requires up to 6ms and more. The PID-models attached via FMI show that handling a captured packet in the simulation bridge is still much faster than forwarding a received one. This implies that the large pass-through times for the fan and voter are caused by PCAP. In future works, these delays must be reduced by speeding up the packet capturing or by using a faster mechanism.

One point which also needs further attention are the pass-through times of the fan model if it is attached via FMI while the voter communicates with its simulation bridge using the PCAP library. Although the fan uses FMI, the pass-through times are up to 6ms. During our tests we could not find a proper reason for this behaviour and re-running the tests also resulted in similar times. If the voter application had any effect on another application, also the pass-through times of the simulation bridges attached to the PID-models should show an impact. However, these times are in the same order of magnitude as the related times if only FMI is used.

Table 17. Average pass-through time of the simulation bridges (PCAP) for receiving in µs.

Simulation Bridge	Fan Local PC [µs]	Fan LAN [µs]	Voter Local PC [µs]	Voter LAN [µs]
Fan	921.322	759.326	415.393	795.438
Voter	698.372	518.112	360.486	302.425

Simulation Bridge	Fan Local PC [µs]	Fan LAN [µs]	Voter Local PC [µs]	Voter LAN [µs]
PID0	619.081	360.827	368.683	320.636
PID1	616.886	358.087	371.301	320.640
PID2	616.045	358.841	368.506	317.948

Table 18. Average pass-through time of the simulation bridges (PCAP) for sending in µs.

Simulation Bridge	Fan Local PC [µs]	Fan LAN [µs]	Voter Local PC [µs]	Voter LAN [µs]
Fan	6045.299	6521.870	6082.977	5911.976
Voter	105.471	65.098	5849.792	5721.989
PID0	130.555	69.548	59.647	46.467
PID1	130.603	69.475	59.205	46.389
PID2	132.221	69.322	60.247	46.259

Similar to Table 12, Table 13 and Table 14, Table 19 and Table 20 show the general influence of the simulation bridge pass-through times on the simulation durations. Again, the delay sum is calculated and divided by the duration. The amount of processing time in the local setups account for 50.5% if the fan is connected by PCAP and 23.7% in case of the voter. Although there are larger average pass-through times in the voter-case, the fraction is half of the one in the fan-cases. As shown in the schedule of the fan control application, there are three packets sent to the voter while the fan hardware only receives one packet. Hence, the amount of the communication is larger in case of the voter. If the RTI is running on a host in the same LAN, the fractions are 54.0% (fan) and 21.2% (voter). These results are proportional to the simulation durations shown in Figure 17 and the results in the local case.

Compared to the connection of the devices and applications using FMI, the simulation bridge pass-through times are much larger if PCAP is used. The reason is the usage of function calls to send and receive data via a network interface. Using FMI, the model is connected as dynamically shared library which is much faster since no system calls are used.

Table 19. Fractions of simulation bridge delays compared to the simulation duration if the fan is connected via PCAP.

Simulated time	RTI on local host			RTI in LAN		
	Delay [s]	Duration [s]	Fraction	Delay [s]	Duration [s]	Fraction
1	1.081	2.085	0.518	1.016	1.884	0.539

Simulated time	RTI on local host			RTI in LAN		
	Delay [s]	Duration [s]	Fraction	Delay [s]	Duration [s]	Fraction
10	10.672	20.868	0.511	9.948	18.715	0.532
20	23.624	43.756	0.540	20.482	38.923	0.526
30	31.610	60.766	0.520	30.907	58.139	0.532
40	43.562	86.530	0.503	38.900	70.923	0.548
50	53.796	107.286	0.501	49.246	90.714	0.543
60	64.742	131.852	0.491	59.564	108.764	0.548
70	76.871	157.256	0.489	68.172	124.710	0.547
80	87.326	177.129	0.493	77.724	141.710	0.548
90	98.618	200.426	0.492	87.167	161.035	0.541
100	109.238	219.580	0.497	97.486	181.072	0.538

Table 20. Fractions of simulation bridge delays compared to the simulation duration if the voter is connected via PCAP.

Simulated time	RTI on local PC			RTI in LAN		
	Delay [s]	Duration [s]	Fraction	Delay [s]	Duration [s]	Fraction
1	1.239	4.618	0.268	0.881	4.017	0.2193179
10	9.692	38.975	0.249	9.260	43.016	0.2152687
20	17.445	76.100	0.229	17.303	81.982	0.2110585
30	25.690	111.956	0.229	26.107	124.908	0.2090098
40	32.707	140.951	0.232	34.971	165.502	0.2113026
50	41.612	179.978	0.231	41.505	197.086	0.2105933
60	48.515	208.583	0.233	51.691	243.156	0.2125837
70	55.496	236.602	0.235	57.455	270.657	0.2122797
80	60.131	252.778	0.238	65.985	315.070	0.2094297

Simulated time	RTI on local PC			RTI in LAN		
	Delay [s]	Duration [s]	Fraction	Delay [s]	Duration [s]	Fraction
90	74.127	320.388	0.231	74.931	359.912	0.2081926
100	79.358	338.16	0.235	84.522	403.033	0.2097148

2.2.4 Evaluation of fault-injection

The communication emulator supports the injection of faults according to the EN 50159 standard for safety in railway applications (EN 50159 - Railway applications - Communication, signalling and processing systems - Safety-related communication in transmission systems). This standard describes the injection of the following seven failure types:

- Corruption
- Manipulation
- Delay
- Resequencing
- Insertion
- Omission
- Replay

As explained in Section 2.2.1, the voter sends a new destination speed to the PID controllers every 10s. In the following, the faults described by the standard are injected into these destination speed messages. Each figure shows the simulated time on the x-axis in seconds and the fan speed in revolutions per minute (rpm) on the y-axis. The fan speed without any fault is printed in black while a grey graph denotes the speed if a fault is injected.

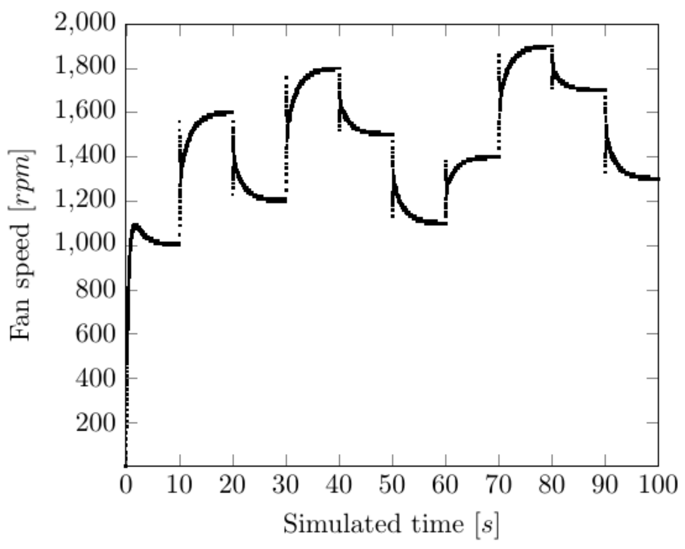


Figure 18. Fan speed without fault injected.

Figure 18 shows the fan speed without any fault. Initially, the fan runs at a speed of 1000 rpm. After 10s, the voter sends a new destination of 1600 rpm which is reduced to 1200 rpm at 20s and increased to 1800 rpm at 30s. Afterwards, a sequence of 1500 rpm (40s), 1100 rpm (50s), 1400 rpm (60s), 1900 rpm (70s), 1700 rpm (80s) and 1300 rpm (90s) follows.

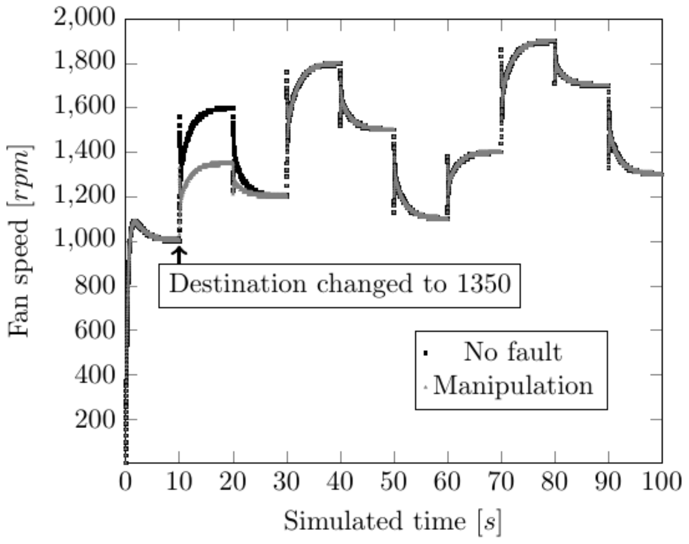


Figure 19. Manipulation fault, destination of 1600 rpm at 10s changed to 1350 rpm.

In Figure 19, the first destination sent by the voter (1600 rpm at 10s) is manipulated and changed to 1350 rpm. To configure this fault, the user has to specify the indexes of the bytes to manipulate and their new values. Since the corruption fault is implemented the same way, it is not shown further. After 20s, the voter sends a new, correct speed. Hence, there is no difference between the initial and the faulty graphs afterwards.

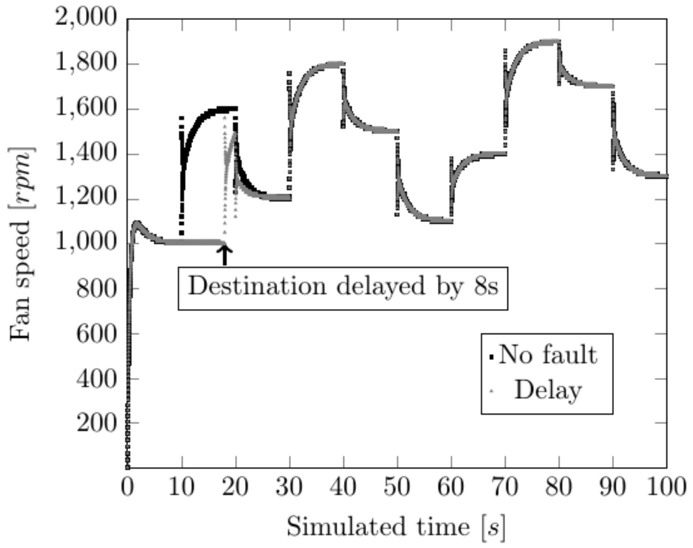


Figure 20. Delay fault, destination of 1600 rpm at 10s is delayed by 8s.

The effect of a delay fault on the fan speed is shown in Figure 20. In this test, the destination of 1600 rpm is delayed by 8s. As a consequence, the fan does not reach its designed speed of 1600 rpm before the next destination message is sent after 20s. The approximate speed at 20s is 1500 rpm.

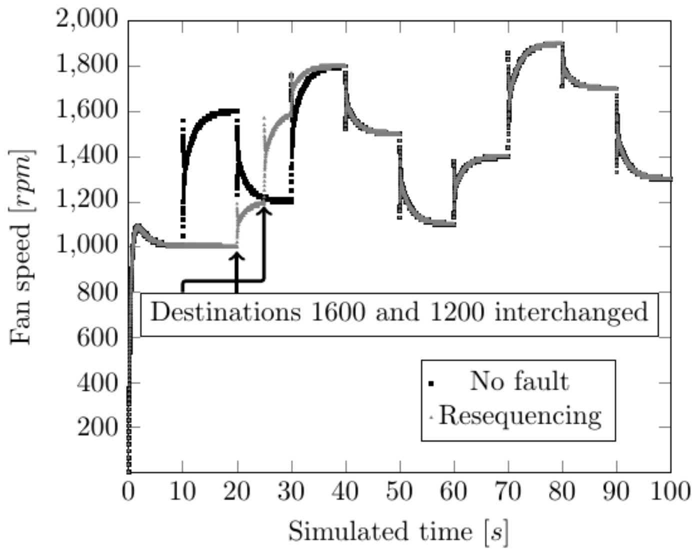


Figure 21. Resequencing fault, destinations of 1600 rpm (10s) and 1200 rpm (20s) interchanged.

Changing the order of two or more packets (resequencing) can be reached by delaying the related packets. In Figure 21, the destination of 1600 rpm (10s) is delayed by 15s, hence it is interchanged with the destination of 1200 rpm (20s). Although the sequence is changed, the interval of 5s is large enough so that both destinations are reached when a new value is sent by the voter. Furthermore, the destination of 1800 rpm sent at 30s can be reached faster.

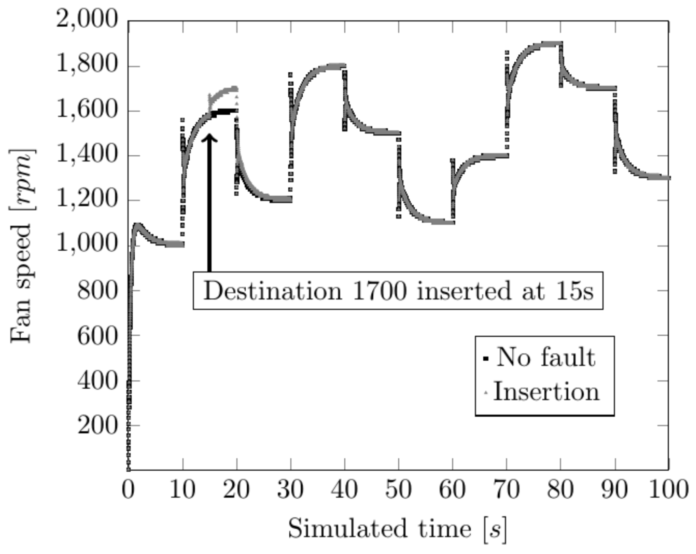


Figure 22. Insertion fault, new destination of 1700 rpm inserted after 15s.

The fourth fault type shown is the insertion fault (Figure 22). In the simulation execution, a new destination message of 1700 rpm is inserted after 15s. Again, the destination of 1600 rpm cannot be reached exactly and there is a difference of 20 rpm. Although the difference between the current speed of 1700 rpm at 20s is 100 rpm larger than desired, the fan can reach the destination speed of 1200 rpm in approximately the same time compared to the run without any fault.

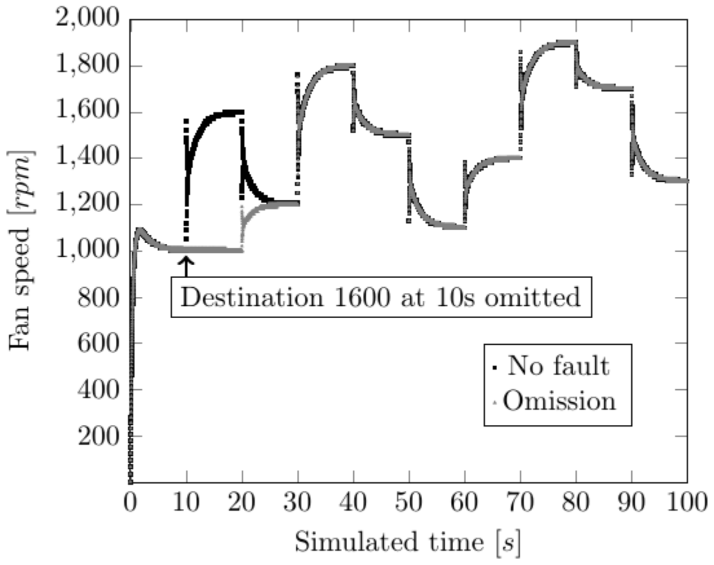


Figure 23. Omission fault, destination of 1600 rpm at 10s omitted.

Figure 23 illustrates an omission fault by reference to dropping the destination of 1600 rpm at 10s. In this example, the PID controllers have 10s in addition to reach the destination speed. The figure shows that the initial interval of 10s is not large enough so that the fan speed can stabilize before a new destination is reached. However, the difference to the destination speed is negligible.

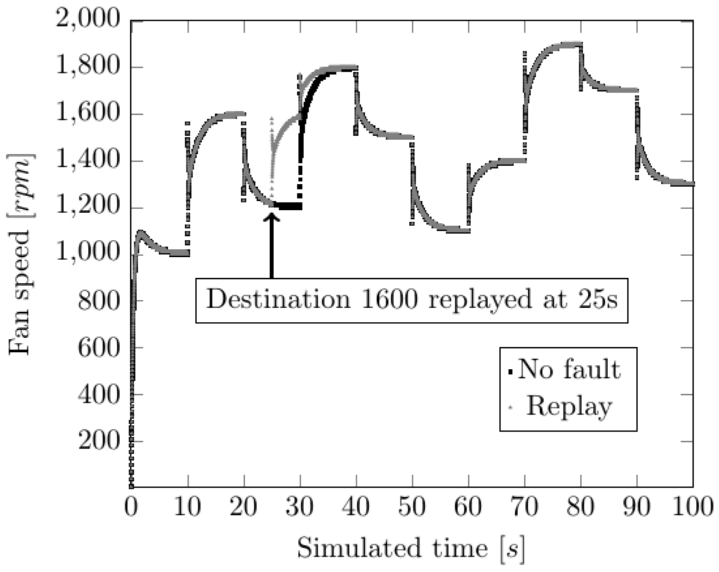


Figure 24. Replay fault, destination of 1600 rpm replayed at 25s.

The replay fault is shown in Figure 24. Since the destination of 1600 rpm is replayed at 25s, the behaviour is similar to the resequencing fault in Figure 21. There is a difference to the destinations of 1200 rpm and the inserted 1600 rpm remaining and the destination of 1800 rpm at 30s can be reached faster.

2.2.5 Evaluation of state-estimation and delay-management

Since the Simulation Bridges can be connected via heterogeneous communication networks including the Internet, a mechanism is required to handle too large communication delays.

While the delay-management module checks the delays and terminates the simulation if the interaction delay exceeds a preconfigured maximum, the state-estimation module estimates future inputs to provide the correct inputs in time.

As explained in D3.4, Section 4.1.4, the state-estimation module contains two functionalities. One function estimates the future inputs of the connected based on its previous outputs and a model of the remaining system under test. The other one compares the estimated inputs with the received ones and terminates the simulation if the difference between them is too large. In this section, we show first that we use a system model which is appropriate to estimate the inputs. Afterwards, we evaluate the estimation proving that it is capable to maintain real-time behaviour with limitations. Similar to the previous sections, we use the fan-application described in Section 2.2.1 and enable the state-estimation in the simulation bridge connected to the fan.

The system model to estimate the future inputs is an extension of the PID model. It uses the SensorRPM-message sent by the fan as input and calculates the new PID control value based on the current speed. Instead of sending it to the voter, the model encapsulates the output directly into a PWM-message which is the input for the fan.

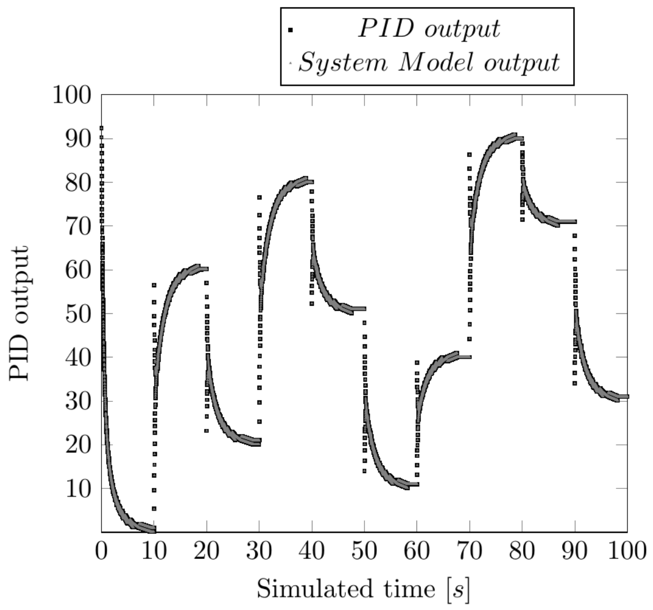


Figure 25. Outputs of the system model compared to the PID controller outputs.

Figure 25 shows the outputs of the system model (grey, triangle marks) and those of the PID-controllers (black, square marks). Since the same algorithm is used in both implementations, the outputs are exactly the same and valuable to be used in the state-estimation.

To prove the real-time behaviour of the state-estimation module, we ran the same tests like in the evaluation of the simulation bridge using FMI. Since the simulation durations in this setup are already faster than real-time we added additional delays by printing debugging information in the terminal running the tests. This way, we increased the simulation duration up to twice the simulated time. In the following, we show the possibility to reach real-time simulations even if the simulation requires more time. Furthermore, we analyse the accuracy of the thread which triggers the injection of packets in real-time.

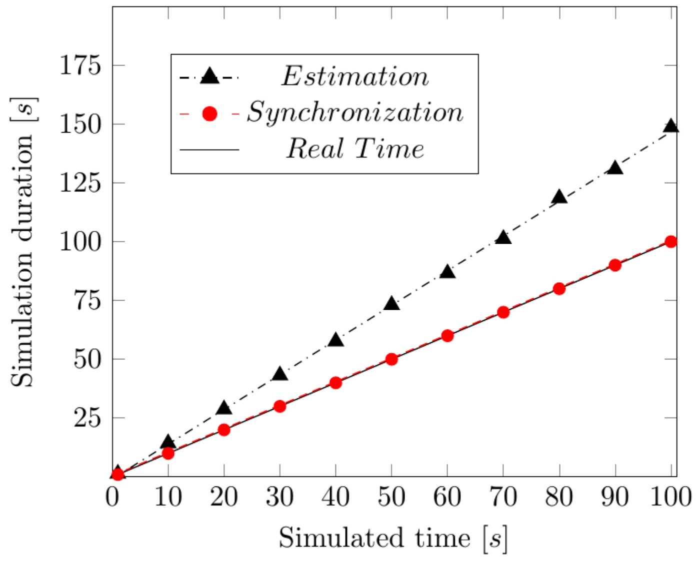


Figure 26. Simulation durations if the state-estimation is used.

In Figure 26, two graphs are depicted. The first graph (red, circle marks) illustrates the time after which the state-estimation finished to estimate all future outputs. The second graph (black, triangle marks) illustrates the duration all subsystems need to finish their execution after the estimation has finished already. It is necessary to continue the simulation after the estimation has finished to ensure all calculated inputs are received and compared to the estimated ones. As the figure shows, it is possible to run simulation with real-time characteristics if the state-estimation functionality is enabled.

Table 21. Simulation steps and missed events in sum for every simulation duration (upper line) and in average (lower line).

Simulated time [s]	Scheduled steps	Simulated steps	Estimated steps	Missed steps message (fraction of scheduled steps)		Missed steps task (fraction of scheduled steps)	
1	4000 200	85 4.25	3897 194.85	18 0.90	0.0045	0 0	0
10	40000 2000	57 2.85	39799 1989.95	139 6.95	0.0035	5 0.25	0.0001
20	80000 4000	85 4.25	79614 3980.7	297 14.85	0.0037	4 0.20	0.0001
30	120000 6000	98 4.90	119282 5964.1	609 30.45	0.0051	11 0.55	0.0001
40	160000 8000	78 3.90	159041 7952.05	852 42.60	0.0053	29 1.45	0.0002
50	200000 10000	77 3.85	198817 9940.85	1055 52.75	0.0053	51 2.55	0.0003

Simulated time [s]	Scheduled steps	Simulated steps	Estimated steps	Missed steps message (fraction of scheduled steps)		Missed steps task (fraction of scheduled steps)	
60	240000 12000	50 2.50	238580 11929.00	1322 66.10	0.0055	48 2.40	0.0002
70	280000 14000	68 3.40	278401 13920.05	1482 74.10	0.0053	49 2.45	0.0002
80	320000 16000	76 3.80	318461 15923.05	1429 71.45	0.0045	34 1.70	0.0001
90	360000 18000	27 1.35	357997 17899.85	1906 95.30	0.0053	70 3.50	0.0002
100	400000 20000	58 2.90	398060 19903.00	1815 90.75	0.0045	67 3.35	0.0002

Table 21 presents the scheduled simulation steps for each simulated time and whether these steps are simulated (executed without estimation) or estimated. Furthermore it shows the missed steps for message reception/injection and task execution (calculate the sensor speed). For each simulation duration, a large amount of events is estimated. However, a few number of events is executed normally. This mainly occurs at the beginning of the simulation when the offset between simulated time and the estimated (real) time is negative so that the simulation is faster than real-time. Since the simulation is slowed down to show the functionality of the estimation, it is executed slower than real-time after a few events. However, if the offset remains almost zero or the simulation starts running faster than real-time, it is possible that the execution returns to simulation instead of estimation. As shown in the table, the number of missed events is quite small compared to the number of scheduled events. For message reception/injection it accounts for about 0.5%, the number of missed task executions is even much smaller (about 0.02%).

Table 22. Packets in sum for every simulation duration (upper line) and in average (lower line).

Simulated time [s]	Scheduled packets	Received packets	Estimated packets	Dropped packets	Fraction of dropped packets
1	2000 100	34 1.70	1949 97.45	18 0.90	0.009 0.009
10	20000 1000	23 1.15	19546 977.30	139 6.95	0.007 0.007
20	40000 2000	35 1.75	39482 1974.10	297 14.85	0.007 0.007

Simulated time [s]	Scheduled packets	Received packets	Estimated packets	Dropped packets	Fraction of dropped packets
30	60000 3000	45 2.25	58960 2948.00	609 30.45	0.010 0.010
40	80000 4000	31 1.55	77376 3868.80	852 42.60	0.011 0.011
50	100000 5000	34 1.70	98388 4919.40	1055 52.75	0.011 0.011
60	120000 6000	21 1.05	118417 5920.85	1322 66.10	0.011 0.011
70	140000 7000	26 1.30	133475 6673.75	1482 74.10	0.011 0.011
80	160000 8000	32 1.60	157104 7855.20	1429 71.45	0.009 0.009
90	180000 9000	11 0.55	177660 8883.00	1906 95.30	0.011 0.011
100	200000 10000	24 1.20	196987 9849.35	1815 90.75	0.009 0.009

In Table 22, the number of scheduled packet receptions is depicted. It further shows whether the packets are received from other simulation bridges in time or if they are estimated. Compliant with Table 22, most of the packets are estimated and for each missed message reception/injection, the related packet is dropped and not received by the fan model. The amount of dropped packets accounts for about 1% which is small, but should be improved for guaranteeing a valuable simulation execution using the state-estimation.

To figure out the reason for the missed events we monitored the debug outputs of the simulation bridges. Although some packets are not forwarded to the model, they are inserted into the forward packet buffer of the responsible simulation bridge. The debugging information about the time-stamps when the injection of a message is triggered shows that the thread which is responsible for the injection is not scheduled in time every time a message is dropped. One possible reason for this behaviour is the usage of a non-deterministic operating system. There are several applications executed in parallel on the same PC and neither processor affinity is used nor the real-time task has a higher priority assigned than the other ones. In another topology where the fan model is executed on a separate PC, no misses of events could be observed. Hence, a dedicated PC should be used for running a simulation bridge where the state-estimation is enabled.

In a second test, we replaced the fan-model by a real device. This device hosts an application which uses the FreeRTOS real-time operating system to execute the schedule described in Section 2.2.1 in real-time. This test was not successful. However, due to the setup used to run the tests, the malfunctionality was expected. First, we did not use a real-time operating system on the simulation host. Therefore, the task triggering the packet injection is not scheduled timely so that the reception deadlines of the application can be met. The second reason is a missing temporal synchronization between the device and the simulation host. As soon as the simulation starts, the simulation bridge sends a start-message to the device which initiates the execution of the application. Both, the simulation bridge and the device use the current start-time as a relative reference to calculate future events. However, the communication delay of the start-message is not considered in the calculation which leads to an offset between the schedule in the simulation bridge and the device. Due to this offset, the messages are not sent timely correct.

As a conclusion, this section shows that it is possible to run simulations with real-time requirements as long as the simulation bridge and the connected application are timely synchronized. However, a real-time operating system must be used to evaluate the determinism of the state-estimation in future projects. Furthermore, a better synchronization mechanism must be included between the simulation bridge and a connected real-time hardware which enables forwarding the required data in time.

2.3 Framework evaluation with real train applications

The previous sections evaluate the CE using simple test applications which support all capabilities of the simulation bridges. However, the CE shall be used with real train equipment which is why the previous tests are repeated with two setups of real hardware. The first is a setup (called “CAF-setup”) provided by CAF (CONNECTA member) with one Consist Switch and two Car Control Units (CCU). The second one uses hardware provided by UniControls (called “UNI-setup”) with a more complex topology. It consists of two end devices, two consist switches and two Ethernet Train Backbone Nodes (ETBN).

2.3.1 CAF-setup

In order to test the CE with real train applications, CAF has provided real train equipment to Ikerlan.

2.3.1.1 Description

The equipment provided by CAF consists of two Car Control Units (CCU) and a Consist Switch. The CCUs are connected via the Consist Switch as shown in Figure 27.

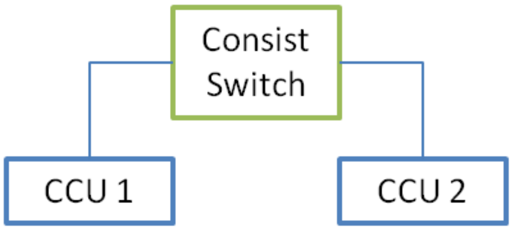


Figure 27. Equipment provided by CAF.

The CCUs are programmed with simple applications that make the measure of the delays suffered by the messages exchanged between both CCUs easy. The information included in the packets exchanged are composed of a set of variables that inform about the packets’ cycle times. For example, the CCU1 is programmed originally to transmit packets with a period of 32ms. In this case the payload of the message includes the following information:

Table 23. Variable values sent by CCU1 when it is configured to send message every 32 ms.

Variable_32ms	1
Variable_64ms	0
Variable_128ms	0
Variable_256ms	0
Variable_512ms	0

In the beginning, the CCU 1 announces the publication of this type of traffic (traffic with a cycle time of 32ms) and the CCU 2 subscribes to all time of traffics (traffic with a cycle time of 32ms, 64ms, 128ms, 256ms and 512ms).

Once the CCU2 receives the traffic of CCU1, it extracts the cycle time of this type of traffic from the value of the Variable_32ms and switches on an LED that correspond to this type of traffic. If the CCU2 does not receive a new message of this same type in 32ms the LED is switched off.

The program loaded in CCU 1 could be modified in order to change the type of traffic, understanding the type of traffic as the cycle time of the messages published. If the CCU1 was programmed to send frames every 64ms payload of the message would include the following information:

Table 24. Variable values sent by CCU1 when it is configured to send message every 64 ms.

Variable_32ms	0
Variable_64ms	1
Variable_128ms	0
Variable_256ms	0
Variable_512ms	0

In this case, the CCU 2 extracts the cycle time of this type of traffic from the value of the *Variable_64ms* and switches on another led if traffic is received from CCU1. This LED correspond with the configured type of traffic and CCU2 subscribed to all traffic types as mentioned before. If the CCU2 does not receive a new message of this same type in 64ms the LED would be switched off.

The CE is introduced in the setup as shown in Figure 29. In a first step, the CE simply substitutes the cable between the CCU 1 and the CS. All SBs and the Central PC are situated in the same location as the CCUs and CS (Figure 28).

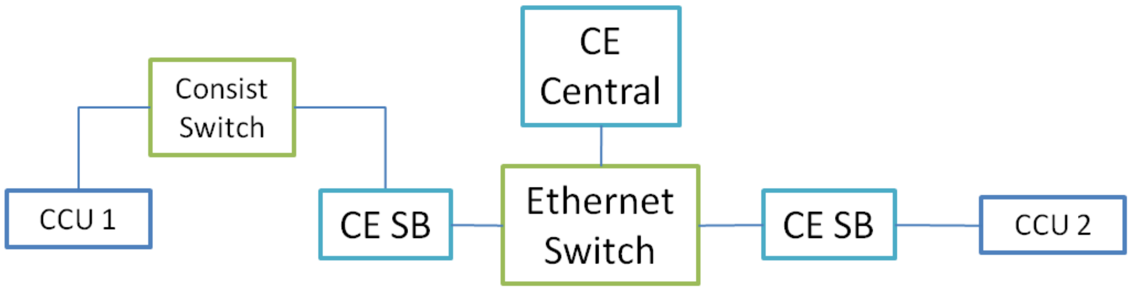


Figure 28. CAF provided equipment connected via CE locally.

In a second step, the CE central is located in a remote location (Siegen, Germany) in order to measure the effect of connecting a CCU in a remote location via a heterogeneous network (Figure 29).

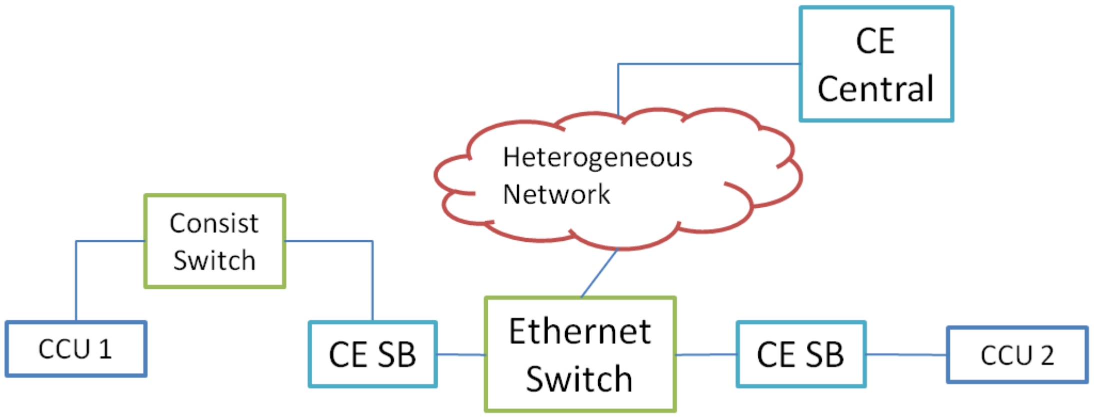


Figure 29. CAF provided equipment connected via CE remotely.

As the LEDs of the CCU2 are not accessible, it is not possible to use the setup described in Subsection 2.1.1.1. Hence, the method described in Subsection 2.1.1.2 is used directly.

2.3.1.2 Results

In Table 25 the latencies measured for different cycle times in a scenario with real railway equipment are shown.

Table 25. Delay results with CE and real railway equipment.

Cycle time	Local/ Remote	Latency [seconds]	Jitter [seconds]	Standard deviation [seconds]	Lost packets	Duplicated packets
512ms	Local	4.54×10^{-3}	6.5×10^{-3}	3.35×10^{-3}	0	0
	Remote	38.53×10^{-3}	39.5×10^{-3}	12.76×10^{-3}	0	0
256ms	Local	6.43×10^{-3}	5.5×10^{-3}	2.2×10^{-3}	0	0
	Remote	53.56×10^{-3}	24.5×10^{-3}	6.57×10^{-3}	0	0

Cycle time	Local/ Remote	Latency [seconds]	Jitter [seconds]	Standard deviation [seconds]	Lost packets	Duplicated packets
128ms	Local	5.85×10^{-3}	5.5×10^{-3}	2.49×10^{-3}	0	0
	Remote	59.85×10^{-3}	17×10^{-3}	3.65×10^{-3}	0	0
64ms	Local	5.28×10^{-3}	6×10^{-3}	2.73×10^{-3}	0	0
	Remote	64.93×10^{-3}	165.5×10^{-3}	22.02×10^{-3}	2	0
32ms	Local	5.64×10^{-3}	4.5×10^{-3}	1.99×10^{-3}	0	0
	Remote	62.16×10^{-3}	96.5×10^{-3}	7.63×10^{-3}	0	0

The results are similar to those obtained in the previous Section 2.1. The delay is about 6ms when all the equipment is located at the same place and connected directly. If the Central PC is located in Germany and the rest of the equipment is in Spain, it is about 60ms. It is worth to mention that in these tests no packet is duplicated and some packets are lost (worse case: 2 of 30.000 packets).

2.3.2 *UNI-setup*

The goal of the previous use-case was to test the CE using real train equipment. Similar to the tests in Section 2.1, the time synchronization is not used. The evaluation of this mechanism with real train hardware follows in this section.

2.3.2.1 Description of test setup

Figure 30 shows the setup used to evaluate the simulation bridges including the time management services of the High Level Architecture (HLA). It consists of a Display (DIS) and an HVAC (Heating, Ventilation and Air Conditioning) Control Unit (HCU) as end devices and four network nodes. Those nodes are two Consist Switches (CS) connected via two ETBNs (Ethernet Train Backbone Nodes). All devices are provided by UniControls (UNI) and connected via standard Ethernet cables (orange lines).

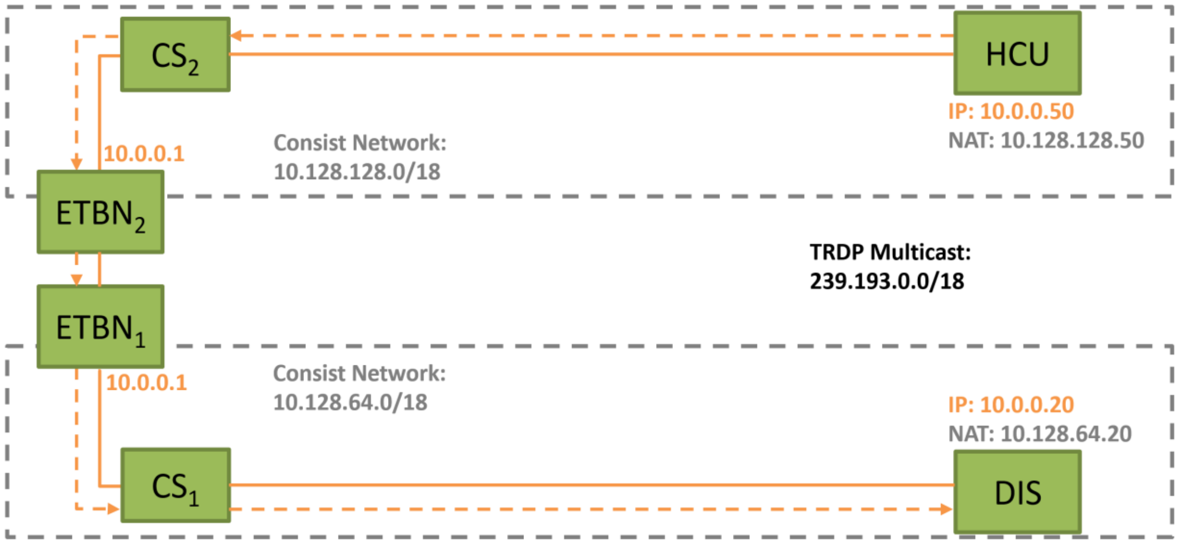


Figure 30. Use-case for evaluation of time management with real train hardware.

In this setup, the HCU sends the status of its digital inputs to the display as TRDP multicast with a period of 100ms. Since both end devices are located in different consist networks, the data transfer has to be routed via the ETBNs connecting the networks. In each consist network, there might be devices with the same IP address which prevents routing between the networks. Using Network Address Translation (NAT), the IP addresses of the devices are mapped to addresses in the related consist networks. For each device, the host part of the IP address (the last 14 bits) is added to the consist network address. In Figure 30, the data transfer is illustrated using orange arrows. To test the communication via the simulation bridges using the HLA time management, the setup is distributed as shown in Figure 31.

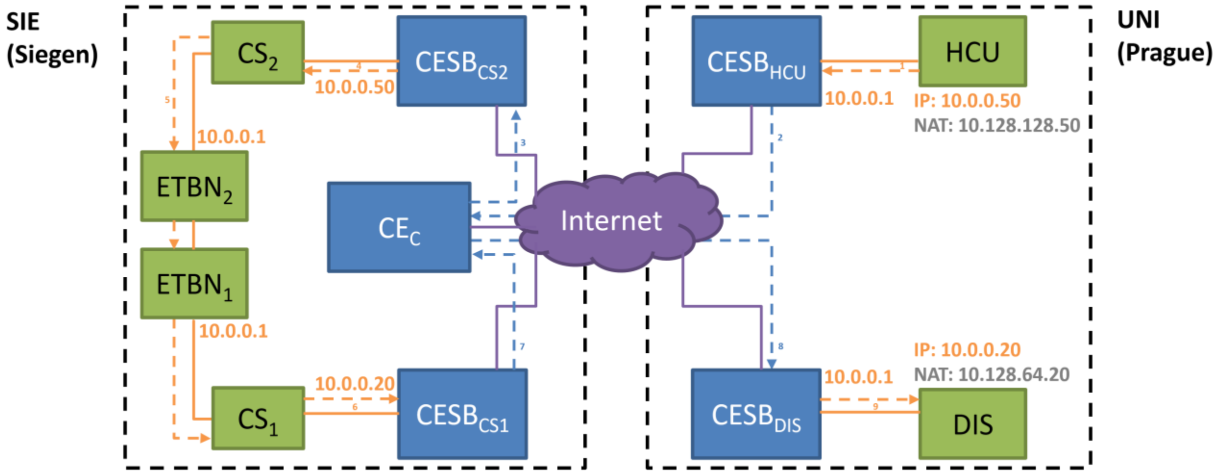


Figure 31. Distribution of test setup using the CE.

After removing the network cables between the end devices and the consist switches (see Figure 31), the end devices remain located at UniControls in Prague while the network devices were shipped to Siegen (SIE). To connect the devices with the switches, the CE has to be used by including four simulation bridges. Each of the switches and devices are attached to one of the four simulation bridges using Ethernet cables. Furthermore, the simulation bridges are connected to the Internet. To provide the synchronization and data exchange between the simulation bridges, an RTI component is added as well. It is executed on the Central CE (CE_C). The simulation bridges are transparent for the connected devices. Hence, they have to use the IP addresses of the devices they represent. $CESB_{HCU}$ for example acts as a gateway for the HCU so it uses the same IP address as $ETBN_2$. In contrast, $CESB_{CS2}$

represents the HCU for $ETBN_2$ and has the end device's IP address. The HCU now sends its data via $CESB_{HCU}$ and the RTI to $CESB_{CS2}$. Since the data transfer is realized using the CE, the arrows are printed in blue. The simulation bridge injects the packets into the real network from where they arrive at $CESB_{CS1}$. This simulation bridge captures the packets to forward them via the RTI to $CESB_{DIS}$ which communicates with the display.

2.3.2.2 Evaluation results

The HLA and its time management are described quite detailed in Chapter 4.2.1.1 of deliverable D3.2, *Report on Design of TCMS Distributed Simulation Framework Concept* [2]. Its realization as well as the handling of packets in the simulation is depicted in Chapters 4.12 and 4.13 of deliverable D3.4, *Proof-of-concept implementation of distributed simulation framework* [4]. As explained in those chapters, the time management follows a step-wise algorithm in which the simulation bridge requires knowledge about the packets sent in a step to advance in time. The benefit of using the time management is the transmission of dedicated packets in the correct temporal order. Without this mechanism, those temporal guarantees cannot be assured.

There are several ways of notifying the bridges that a request to advance can be sent. As explained in Section 4.13 of D3.4, the mechanism implemented in the proof-of-concept includes an additional protocol between the transport layer (e.g., TCP or UDP) and the application layer (e.g., TRDP (Train Real-Time Data Protocol)). In this protocol, one byte denotes if the packet is the last one sent in this step. In this case, the simulation bridge can request the time advance. If no packet is sent in a step, an additional packet is sent by the device which signals this case.

During this project, only a proof-of-concept of the CE could be implemented. The definition of the synchronization mechanism between a real device and the simulation bridges is quite challenging since this mechanism must be generic to be supported by various hardware suppliers. Hence, we decided to implement a simple mechanism first which shows the concepts explained in D3.2 work fine. Later, this simple mechanism should be replaced by another one which is also supported by real hardware. However, we encountered several integration problems between the simulation bridge library and the other parts of the CE. Solving these issues introduced delays in the evaluation due to which a proper synchronization mechanism could not be added in time. Hence, it would only be possible to test the UNI-setup without time management. Such a test is already executed using the CAF-setup as explained in Section 2.3.1. Although the time management cannot be used at the moment with existing hardware, the tests performed in Section 2.2 show that the mechanism works if it is supported by the devices and models.

In future projects, a mechanism must be implemented which provides information that the simulation bridge can use to figure out when it can advance in time. In the following, three ideas are presented.

- The first idea is to perform a time advance to the send time every time a packet is captured. However, there are two disadvantages. First, the simulation bridge must know the send time for which the time of the device and the host the bridge is executed on must be synchronized. Second, the mechanism is quite inefficient if multiple packets are sent which could be considered as he outputs of one step since. In this case multiple additional synchronization messages must be exchanged between the simulation bridges and the RTI.
- The second idea is to establish a global time-base between the simulation bridges and the devices. One possibility is to synchronize the bridges via NTP (Network Time Protocol) and the bridge with the device using PTP (Precision Time Protocol). In this solution, the send times of each packet can be detected easily and the simulation bridge can gather the required information for a time advance based on a time-

triggered schedule. However, a time synchronization mechanism and time-triggered schedule are not used in many available train devices.

- Finally, as third solution, all packets sent in one step can be collected in the configuration file. The simulation bridge can advance in time if all packets of the step are collected. To support this solution, the packets must contain information which allows the simulation bridges to identify the packet's affiliation to a step. Such information could be included in one of the protocol headers, e.g. the ComId in the TRDP protocol. This ID is used to define data content, the interval and the timeout of the packet sent.

All three solutions have their advantages and disadvantages. They must be evaluated and the best mechanism must be implemented in both, the simulation bridge as well as the devices and models used. If only TRDP data must be sent using the time management, the most promising solution is the last one exploiting the ComId.

2.4 Evaluation of test automation

According to deliverable D3.6, *Report on final requirements* [6], the CE shall provide a mechanism to connect a test automation tool. In this chapter, the evaluation of the test automation API used for the connection is presented. It covers a description of the test setup, test cases and evaluation results.

2.4.1 Description of test setup

The User Interface (UI) consists of a server and a client, which are connecting the Simulation Framework Toolset (SFTS) with the Communication Emulator Toolset (CETS). Their location and relation is shown in Figure 32Figure 30.

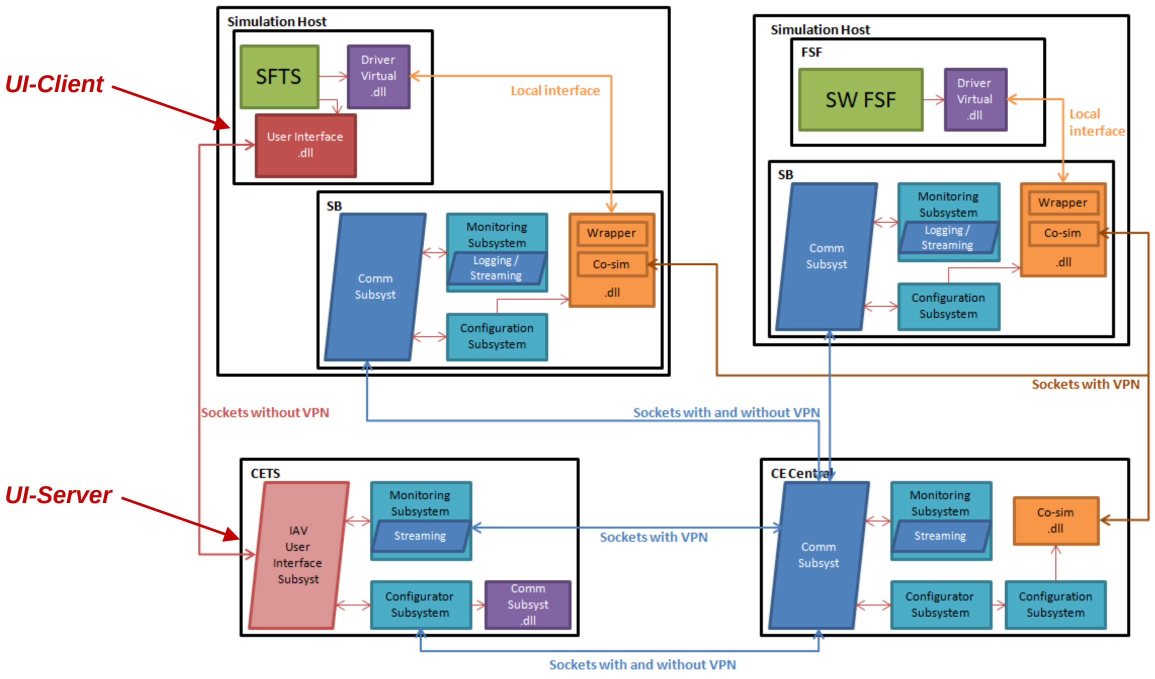
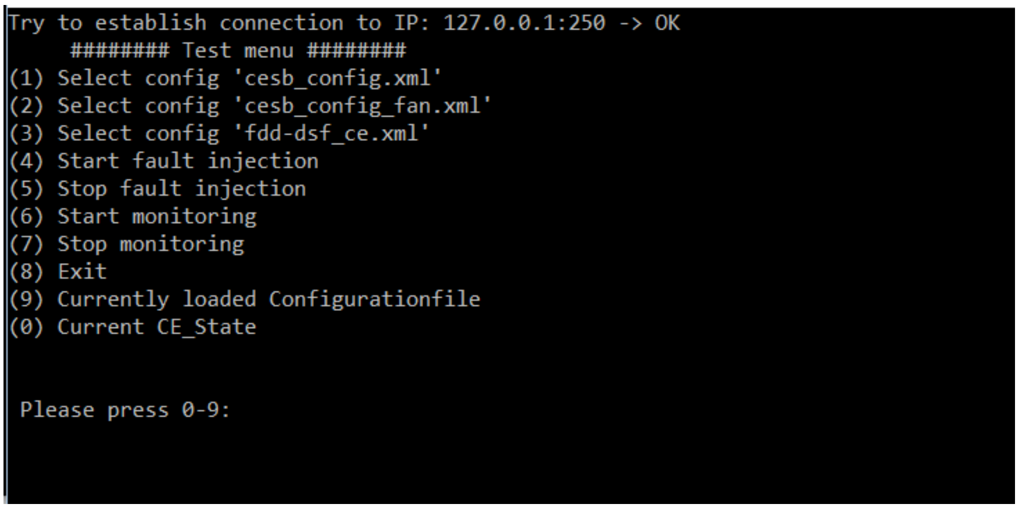


Figure 32. Relation of the UI client and server.

The UI client library (UI_Client.dll) serves a command interface via API. It sends the commands to the server (UI_Server.exe) which is processing them, calls the configuration and monitoring subsystem and takes care of the correct command order. For simplified usage in test automations, all commands are capsuled in standardized function calls as required in D3.2, *Report on Design of TCMS distributed simulation framework concept* [2] and further described in D3.4, *Proof-of-concept implementation of distributed simulation framework* [4].

To evaluate the functionality of the user interface, two Windows computers connected via TCP/IP were serving as SFTS client and CETS server. On the server side, a CETS subsystem environment was installed. On the client system several simplified test automations (one example shown in Figure 33) were used to execute the test cases.



```
Try to establish connection to IP: 127.0.0.1:250 -> OK
     ######## Test menu ########
(1) Select config 'cesb_config.xml'
(2) Select config 'cesb_config_fan.xml'
(3) Select config 'fdd-dsf_ce.xml'
(4) Start fault injection
(5) Stop fault injection
(6) Start monitoring
(7) Stop monitoring
(8) Exit
(9) Currently loaded Configurationfile
(0) Current CE_State


 Please press 0-9:
```

Figure 33. Example of a test automation.

2.4.2 Test cases

Corresponding to the requirements in D3.2 [2], Section 5.1.1 ff. a set of test cases has been executed.

For the implemented function calls / commands listed below (as described in D3.4 [4]) the behaviour and return values were evaluated:

- ConnectToServer((char* Server_IP, int Port);
- Configure(char* XMLFilename)
- Monitoring_Start(char* CESB_Id, int flagFile)
- Monitoring_Stop(char* CESB_Id)
- FaultInjection_Start(char* CESB_Id, char* faultType, char* parameters)
- FaultInjection_Stop(char* CESB_Id, char* faultType, char* parameters)
- Stop(int flagFile)
- CE_State()
- Configuration_Request()
- CloseConnection();

Special command for super user:

- SendCommandToServer(“[Exit]”);

Test of connections to subsystems and data exchange

Testing a proper connection to the configuration and monitoring subsystem, a correct transfer of function parameters and residue-free disconnection/finalization of the server and client has to be shown.

Positive / negative tests

Every function call has to return the current CE state ID (value > 0) if the command was executed successfully.

If the command failed, the return value must indicate an error (value < 0).

Example of a positive test case scenario:

```
#include "UIClient.h"
cestatus = ConnectToServer(172.16.8.29, 5555);
cestatus = Configure(„C:\ConfigurationFile.xml“);
cestatus = Monitoring_Start(PC1, 0);
printf(“Current CE state: ”, cestatus);   --> cestatus always greater 0
```

Tests of command order handling

Every function call is allowed only in predefined states, as described in Deliverable 3.4, Fig.12. In case a function is called at the wrong time, the execution must be prohibited and the function must return an error (value < 0).

Example of an illegal command order test case scenario:

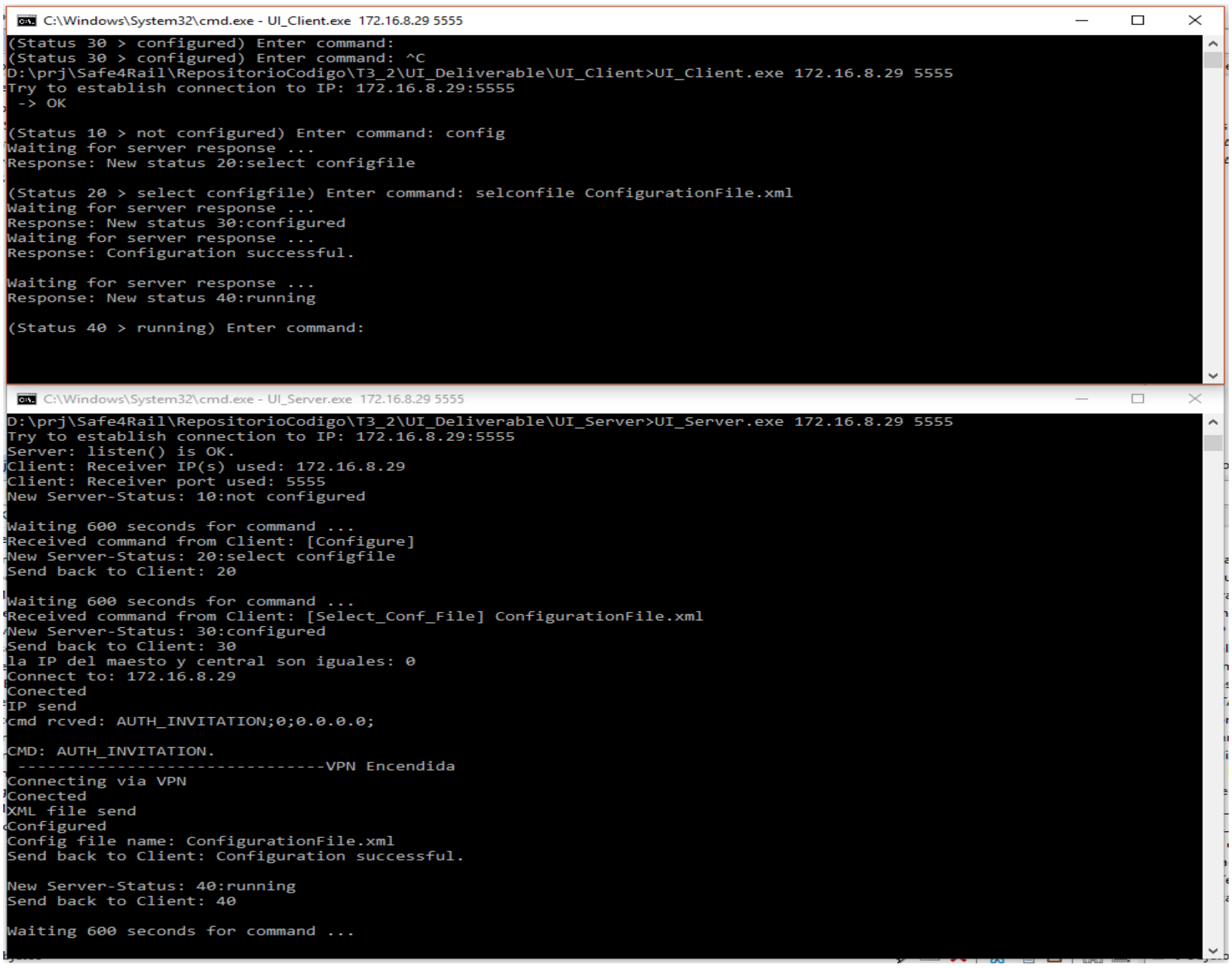
```
#include "UIClient.h"

cestatus = ConnectToServer(172.16.8.29, 5555);

cestatus = Monitoring_Start(PC1, 0);    --> cestatus is negative = not allowed

                    ↕ switched command order

cestatus = Configure(„C:\ConfigurationFile.xml“); );

printf(“Current CE state: ”, cestatus);
```

2.4.3 Evaluation results

Test results of connection to subsystems and data exchange

The user interface calls the subsystems and transfers the given parameters correctly.

With the optional command *SendCommandToServer(“[Exit]”)* the UI-server can be finalized anytime and the *CloseConnection* – command closes the communication.



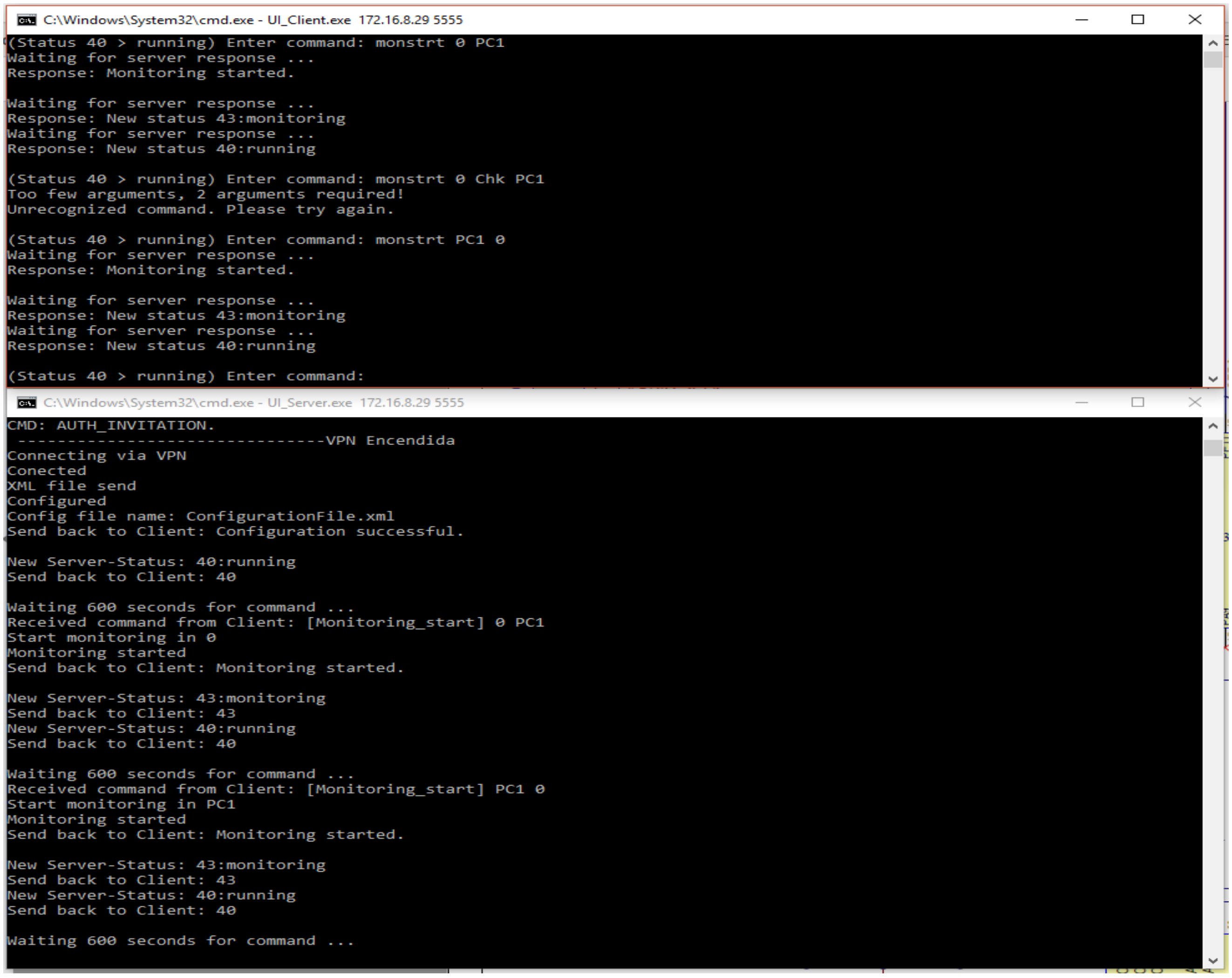
C:\Windows\System32\cmd.exe - UI_Client.exe 172.16.8.29 5555

```
(Status 30 > configured) Enter command:
(Status 30 > configured) Enter command: ^C
D:\prj\Safe4Rail\RepositorioCodigo\T3_2\UI_Deliverable\UI_Client>UI_Client.exe 172.16.8.29 5555
Try to establish connection to IP: 172.16.8.29:5555
 -> OK

(Status 10 > not configured) Enter command: config
Waiting for server response ...
Response: New status 20:select configfile

(Status 20 > select configfile) Enter command: selconfile ConfigurationFile.xml
Waiting for server response ...
Response: New status 30:configured
Waiting for server response ...
Response: Configuration successful.

Waiting for server response ...
Response: New status 40:running

(Status 40 > running) Enter command:
```

C:\Windows\System32\cmd.exe - UI_Server.exe 172.16.8.29 5555

```
D:\prj\Safe4Rail\RepositorioCodigo\T3_2\UI_Deliverable\UI_Server>UI_Server.exe 172.16.8.29 5555
Try to establish connection to IP: 172.16.8.29:5555
Server: listen() is OK.
Client: Receiver IP(s) used: 172.16.8.29
Client: Receiver port used: 5555
New Server-Status: 10:not configured

Waiting 600 seconds for command ...
Received command from Client: [Configure]
New Server-Status: 20:select configfile
Send back to Client: 20

Waiting 600 seconds for command ...
Received command from Client: [Select_Conf_File] ConfigurationFile.xml
New Server-Status: 30:configured
Send back to Client: 30
la IP del maesto y central son iguales: 0
Connect to: 172.16.8.29
Conected
IP send
cmd rcved: AUTH_INVITATION;0;0.0.0.0;

CMD: AUTH_INVITATION.
 -------------------------------VPN Encendida
Connecting via VPN
Conected
XML file send
Configured
Config file name: ConfigurationFile.xml
Send back to Client: Configuration successful.

New Server-Status: 40:running
Send back to Client: 40

Waiting 600 seconds for command ...
```

Figure 34. Configuration subsystem.


C:\Windows\System32\cmd.exe - UI_Client.exe 172.16.8.29 5555

```
(Status 40 > running) Enter command: monstrt 0 PC1
Waiting for server response ...
Response: Monitoring started.

Waiting for server response ...
Response: New status 43:monitoring
Waiting for server response ...
Response: New status 40:running

(Status 40 > running) Enter command: monstrt 0 Chk PC1
Too few arguments, 2 arguments required!
Unrecognized command. Please try again.

(Status 40 > running) Enter command: monstrt PC1 0
Waiting for server response ...
Response: Monitoring started.

Waiting for server response ...
Response: New status 43:monitoring
Waiting for server response ...
Response: New status 40:running

(Status 40 > running) Enter command:
```

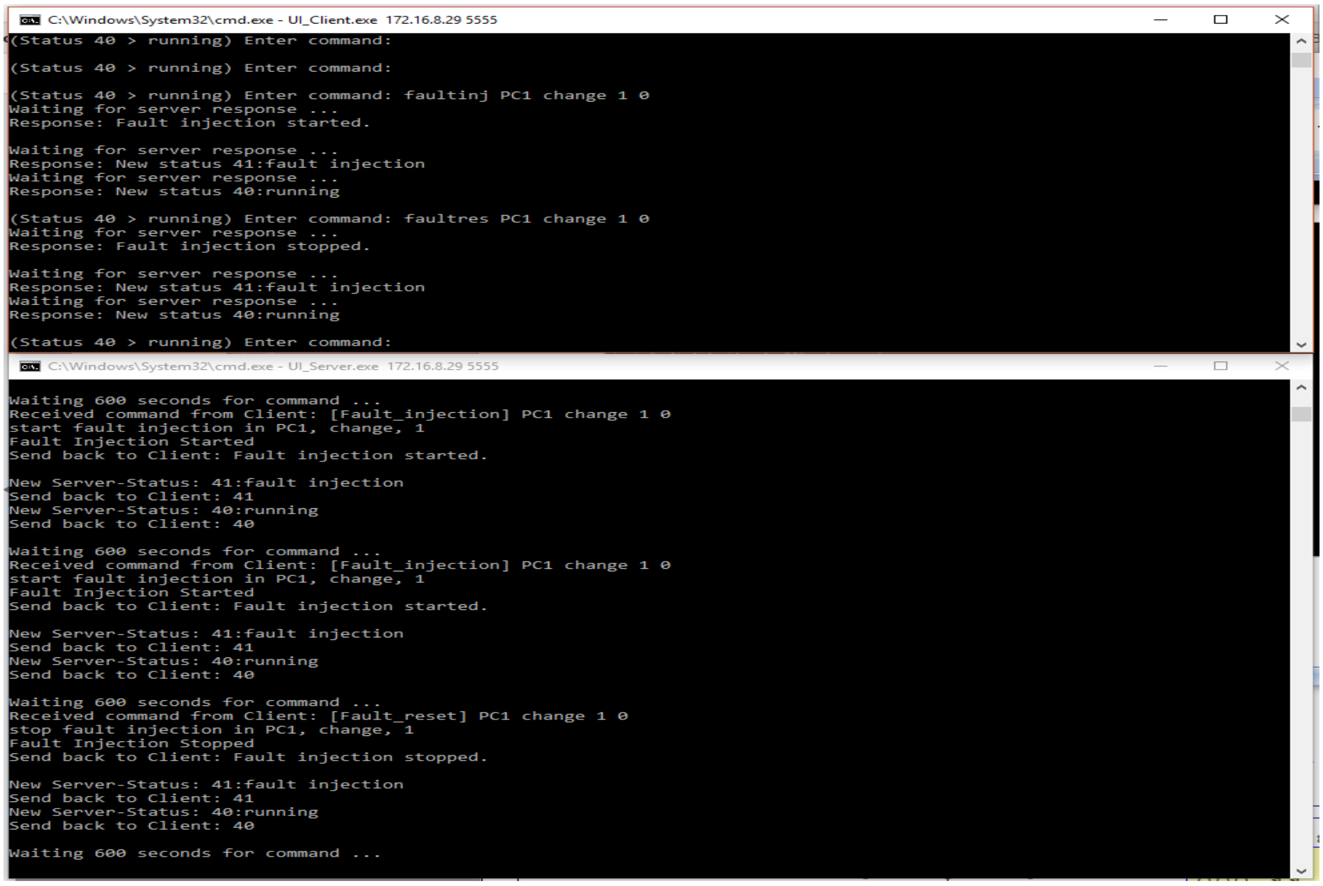
C:\Windows\System32\cmd.exe - UI_Server.exe 172.16.8.29 5555

```
CMD: AUTH_INVITATION.
 ------------------------------VPN Encendida
Connecting via VPN
Conected
XML file send
Configured
Config file name: ConfigurationFile.xml
Send back to Client: Configuration successful.

New Server-Status: 40:running
Send back to Client: 40

Waiting 600 seconds for command ...
Received command from Client: [Monitoring_start] 0 PC1
Start monitoring in 0
Monitoring started
Send back to Client: Monitoring started.

New Server-Status: 43:monitoring
Send back to Client: 43
New Server-Status: 40:running
Send back to Client: 40

Waiting 600 seconds for command ...
Received command from Client: [Monitoring_start] PC1 0
Start monitoring in PC1
Monitoring started
Send back to Client: Monitoring started.

New Server-Status: 43:monitoring
Send back to Client: 43
New Server-Status: 40:running
Send back to Client: 40

Waiting 600 seconds for command ...
```

Figure 35. Monitoring subsystem.

Results for positive / negative tests

All function calls passed the evaluation for the positive and negative test scenario and returned either the current server state in the positive case or a negative value in case of a failure.

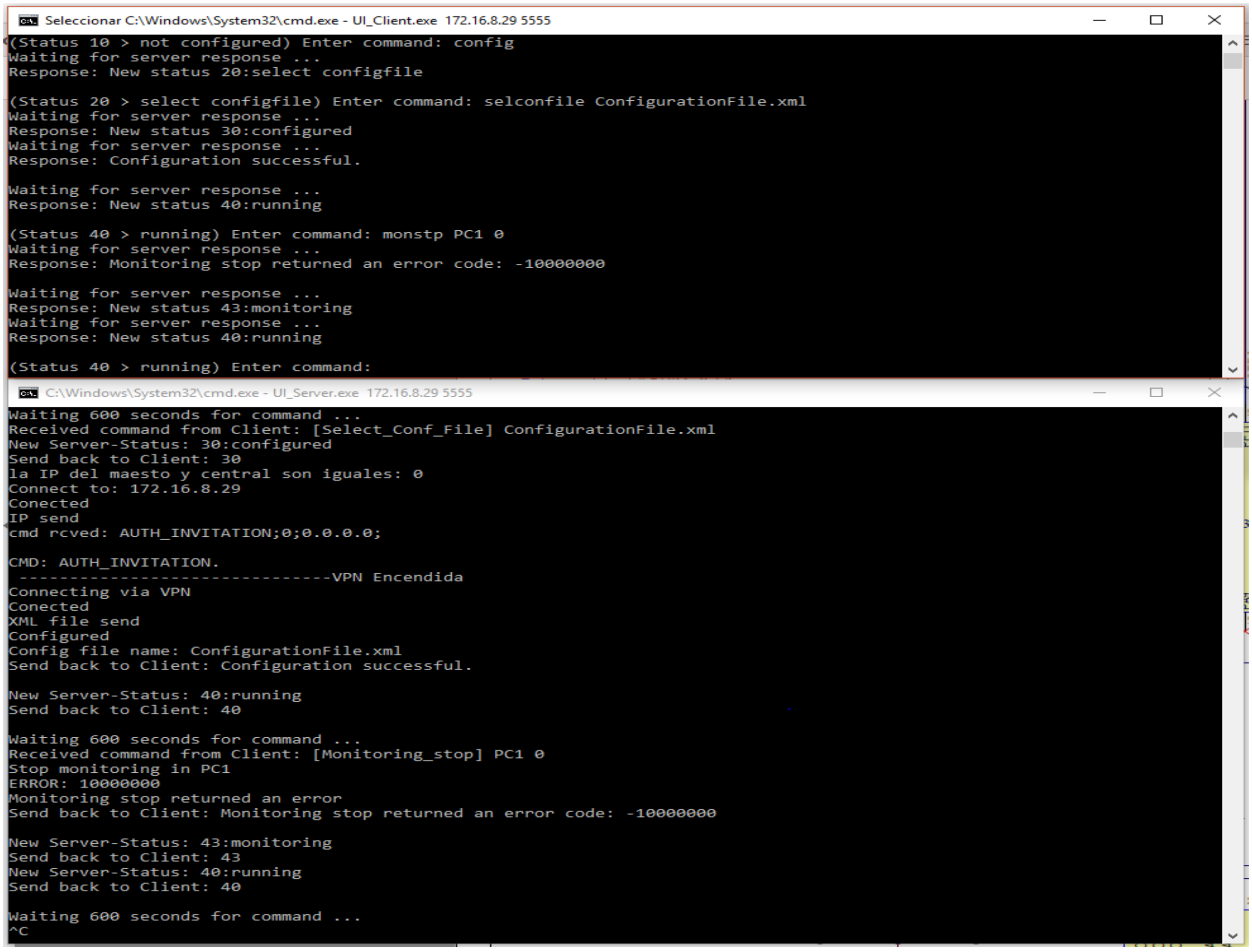

C:\Windows\System32\cmd.exe - UI_Client.exe 172.16.8.29 5555

```
(Status 40 > running) Enter command:

(Status 40 > running) Enter command:

(Status 40 > running) Enter command: faultinj PC1 change 1 0
Waiting for server response ...
Response: Fault injection started.

Waiting for server response ...
Response: New status 41:fault injection
Waiting for server response ...
Response: New status 40:running

(Status 40 > running) Enter command: faultres PC1 change 1 0
Waiting for server response ...
Response: Fault injection stopped.

Waiting for server response ...
Response: New status 41:fault injection
Waiting for server response ...
Response: New status 40:running

(Status 40 > running) Enter command:
```

C:\Windows\System32\cmd.exe - UI_Server.exe 172.16.8.29 5555

```
Waiting 600 seconds for command ...
Received command from Client: [Fault_injection] PC1 change 1 0
start fault injection in PC1, change, 1
Fault Injection Started
Send back to Client: Fault injection started.

New Server-Status: 41:fault injection
Send back to Client: 41
New Server-Status: 40:running
Send back to Client: 40

Waiting 600 seconds for command ...
Received command from Client: [Fault_injection] PC1 change 1 0
start fault injection in PC1, change, 1
Fault Injection Started
Send back to Client: Fault injection started.

New Server-Status: 41:fault injection
Send back to Client: 41
New Server-Status: 40:running
Send back to Client: 40

Waiting 600 seconds for command ...
Received command from Client: [Fault_reset] PC1 change 1 0
stop fault injection in PC1, change, 1
Fault Injection Stopped
Send back to Client: Fault injection stopped.

New Server-Status: 41:fault injection
Send back to Client: 41
New Server-Status: 40:running
Send back to Client: 40

Waiting 600 seconds for command ...
```

Figure 36. Example of a positive scenario (fault injection start – stop).



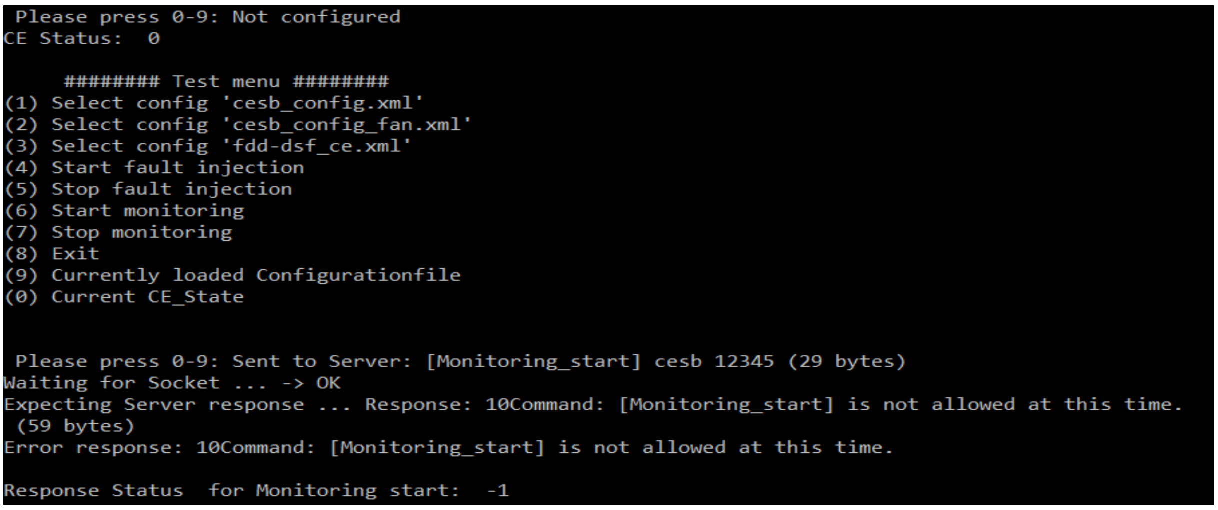
```
Seleccionar C:\Windows\System32\cmd.exe - UI_Client.exe  172.16.8.29 5555
(Status 10 > not configured) Enter command: config
Waiting for server response ...
Response: New status 20:select configfile

(Status 20 > select configfile) Enter command: selconfile ConfigurationFile.xml
Waiting for server response ...
Response: New status 30:configured
Waiting for server response ...
Response: Configuration successful.

Waiting for server response ...
Response: New status 40:running

(Status 40 > running) Enter command: monstp PC1 0
Waiting for server response ...
Response: Monitoring stop returned an error code: -10000000

Waiting for server response ...
Response: New status 43:monitoring
Waiting for server response ...
Response: New status 40:running

(Status 40 > running) Enter command:
```

```
C:\Windows\System32\cmd.exe - UI_Server.exe  172.16.8.29 5555
Waiting 600 seconds for command ...
Received command from Client: [Select_Conf_File] ConfigurationFile.xml
New Server-Status: 30:configured
Send back to Client: 30
la IP del maesto y central son iguales: 0
Connect to: 172.16.8.29
Conected
IP send
cmd rcved: AUTH_INVITATION;0;0.0.0.0;

CMD: AUTH_INVITATION.
 -------------------------------VPN Encendida
Connecting via VPN
Conected
XML file send
Configured
Config file name: ConfigurationFile.xml
Send back to Client: Configuration successful.

New Server-Status: 40:running
Send back to Client: 40

Waiting 600 seconds for command ...
Received command from Client: [Monitoring_stop] PC1 0
Stop monitoring in PC1
ERROR: 10000000
Monitoring stop returned an error
Send back to Client: Monitoring stop returned an error code: -10000000

New Server-Status: 43:monitoring
Send back to Client: 43
New Server-Status: 40:running
Send back to Client: 40

Waiting 600 seconds for command ...
^C
```

Figure 37. Example of a failure scenario (monitoring stop before start).

Test results of command order handling

The user interface ensures the correct execution order of function calls. It prevents reliably the execution of commands, not allowed at the current state.



```
 Please press 0-9: Not configured
CE Status:  0

     ######## Test menu ########
(1) Select config 'cesb_config.xml'
(2) Select config 'cesb_config_fan.xml'
(3) Select config 'fdd-dsf_ce.xml'
(4) Start fault injection
(5) Stop fault injection
(6) Start monitoring
(7) Stop monitoring
(8) Exit
(9) Currently loaded Configurationfile
(0) Current CE_State


 Please press 0-9: Sent to Server: [Monitoring_start] cesb 12345 (29 bytes)
Waiting for Socket ... -> OK
Expecting Server response ... Response: 10Command: [Monitoring_start] is not allowed at this time.
 (59 bytes)
Error response: 10Command: [Monitoring_start] is not allowed at this time.

Response Status  for Monitoring start:  -1
```

Figure 38. Example of wrong execution order (monitoring start before configuration).

Conclusion

Even if the user interface implemented all functions required in deliverable D3.2, there are currently some restrictions due to the proof-of-concept.

In the configuration process of the CE, basically the simulation bridges and the RTI receive information about all the simulation bridges connected to the simulation execution and the type of interactions exchanged between the bridges.

Once the CE is configured, the SFTS will send commands to configure the CE subsystems. Once the subsystems are configured, the SFTS test automation will command the different tests, including a possible reconfiguration of all subsystems to run a different test. In this re-configuration, as the number of simulation bridges does not change, there is no need to stop or reconfigure the CE, because its function (to be able to exchange transparently the different messages sent by the simulation bridges) is the same and unchanged. If at functional level a test has stopped, that does not affect the CE because it is handled in the same way as a message sent from one simulation bridge to another.

The only situation in which a reconfiguration of the CE is necessary is, when the setup of the test changes (i.e. the number of devices or the communication relations between them). In that case it is currently always necessary to stop the CE and to reconfigure the RTI again.

Therefore the “Reconfiguration” command is not available (but via the optional super user command for future implementations) and the *Configuration_Request* command returns always the last addressed configuration file, independent of its successful configuration. In this case the result of the *CE_State* function (0 – not configured, 1 – configured) can be used to keep track if a configuration file could be successfully mounted or not.

Further it is currently not possible to disconnect and reconnect to the user interface server without restarting the server process and CETS subsystems. It makes no difference if the connection is closed intentionally or lost through a disrupted network. That is because no multi-user safety and data management process has been defined yet. If currently a reconnect would be allowed, a different user or automation tool could capture the current session and make a running test process invalid. Therefore, a theoretically possible “echoing”, the parallel displaying of control messages in multiple CE’s or SFTS, is also not supported yet.

Chapter 3 Evaluation of the Train-to-Ground Test Environment

This chapter presents the experience with the implementation of the Train-to-Ground Test Environment (T2G TE) and the Ground Communication Gateway (GCG).

The GCG has been implemented so that it fulfils the selected requirements placed by the Train-to-Ground communication standard IEC 61375-2-6 [11] released to public in 2018. The requirements chosen for the implementation of the GCG have been identified in the CONNECTA project (Chapter 3.1 “Tested Features” in CONNECTA’s T2G System Test Plan [12]). They represent fundamental concepts of the communication between a GCG and a Mobile Communication Gateway (MCG).

The T2G TE is aimed for validation of the T2G communication, so that it can be proven that an information and a data exchange between a GCG and an MCG fulfils the requirements defined in the T2G standard [11].

As the goal of WP3 in Safe4RAIL with respect to T2G was to design and implement a T2G Test Environment, we describe the results of the usage of the T2G TE in the following sections.

The components of the T2G TE are:

- ANS (Access Network Simulator): a controlled wireless communication link between MCG & GCG (implemented for LTE and Wi-Fi)
- GCGSim: GCG simulator for testing the MCG under test
- MCGSim: MCG simulator for testing the GCG under test
- Test Tools: provide controlling & monitoring of the other subsystems

The components are depicted on Figure 39.

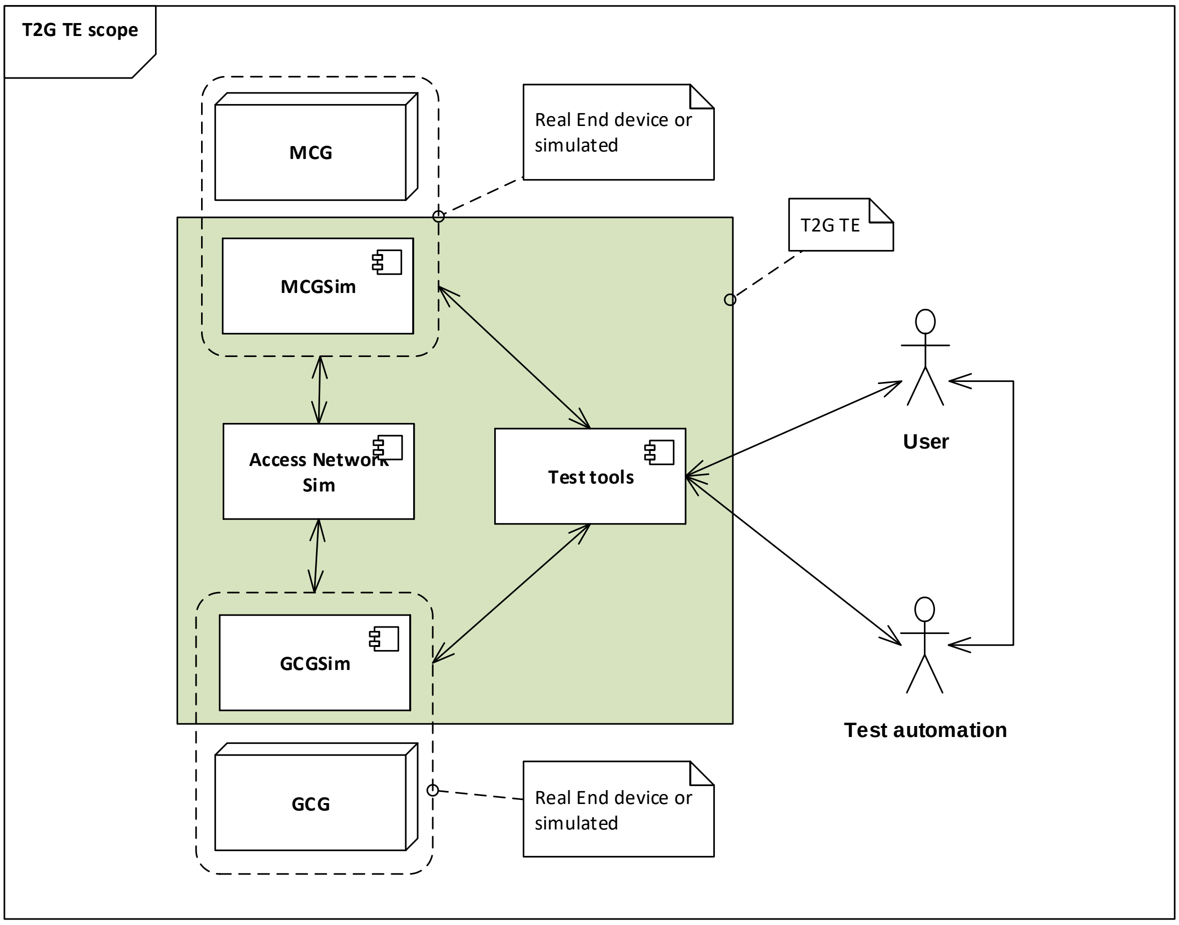


Figure 39. T2G Test Environment Scope.

3.1 Evaluation of using the Test Tools for implementation of the T2G tests

To control and monitor all the components of the T2G test environment, the Test Tools have been implemented. They provide means to write automated test scripts according to the test scenarios. The test cases including the test scenarios have been defined in the CONNECTA project and they are documented in [13].

In general, the functionality provided by the Test Tools is sufficient to implement all the required test scenarios. There could be more helper utility functions for manipulating JSON responses (parsing and checking JSON content). That would allow writing test scripts with a better arrangement.

3.1.1 Execution environment

The Test Tools are implemented as a set of shell script libraries for the Linux GNU BASH shell. It uses several command line utilities such as netcat, curl, jq JSON processor etc. which are usually available in common Linux distributions. The development has been done in a Linux operating system, but the utilities used are also available for Windows. Thus, the T2G TE can be executed on both, the Linux and Windows machines. This possibility proved to be very useful, since both operating system environments have been used for the real MCG validation tests (see Section 3.2).

3.1.2 Additional requirements on devices under test

The T2G TE requires the Test Interface to be implemented on both, a GCG and an MCG (devices under test). This interface is necessary for the test automation. It allows programmatic control of actions performed by the devices under test, checks their status and T2G communication inputs and output.

The Test Interface is quite simple as it consists of several services using a communication pattern very similar to the T2G protocol itself. However, it appeared during discussions with some CONNECTA partners who implement the MCGs that they are not willing to implement such an interface. The arguing for this was as follows. If the interface would be implemented in an MCG itself, it will modify a device under test, which is not acceptable for the validation. Either the validated device would be different from the device used in an operation service, or the device would contain functionality unnecessary for the operation. The Test Interface could be implemented outside of an MCG utilizing the MCG's interface on a train side – so that the Test interface would either mimic the Train Control and Management System (TCMS) or simulate inputs to a TCMS. Additional arguments were limited budget for the project, and an unimplemented interface on a train side of the MCG. We think that all these arguments are valid also for the usual commercial development path.

Based on that we decided to implement the Test Interface module also in the T2G TE. On one side, this module communicates with the T2G TE using the defined test protocol. On the other side it interacts with a human operator. The human operator is then responsible for the action to be performed on the MCG. For example, when there is a request for changing the train topology on the train in a test scenario (which in turn shall trigger sending a train information message from an MCG to a GCG), the request is printed on the user interface terminal. The module then waits for an acknowledgement from the operator who shall perform the action on the train side and acknowledge this on the terminal. The action can be performed (I) using a debug channel in a TCMS or in an MCG, (II) simulating a vehicle connection/disconnection manipulating train bus cables, or (III) in any other way.

With the extra module implemented it is up to an MCG or GCG supplier's decision if he decides to implement the Test Interface for the complete test automation or not.

3.2 Evaluation of performing the T2G tests implemented based on CONNECTA test specification

The test scenarios implemented according to [13] in the T2G Test Environment have been used for validation of the three real MCGs implemented by participants of the CONNECTA project (Bombardier, Siemens, CAF), and of the GCG implemented by UniControls in the Safe4RAIL project.

There have been two test sessions arranged:

- On-site test at Bombardier facility in Mannheim, Germany
  - Two MCGs made by CAF and by Bombardier validated
  - The GCG made by UniControls validated
  - All the devices under test including the T2G TE have been connected locally
  - The T2G TE executed on a PC with Windows operating system
- Online test over the public Internet
  - The MCG made by Siemens validated, located in Budapest, Hungary
  - The GCG together with the T2G TE located in Prague, Czech

- The T2G TE executed on a server machine with Linux operating system

The setup of the test sessions is depicted on Figure 40.

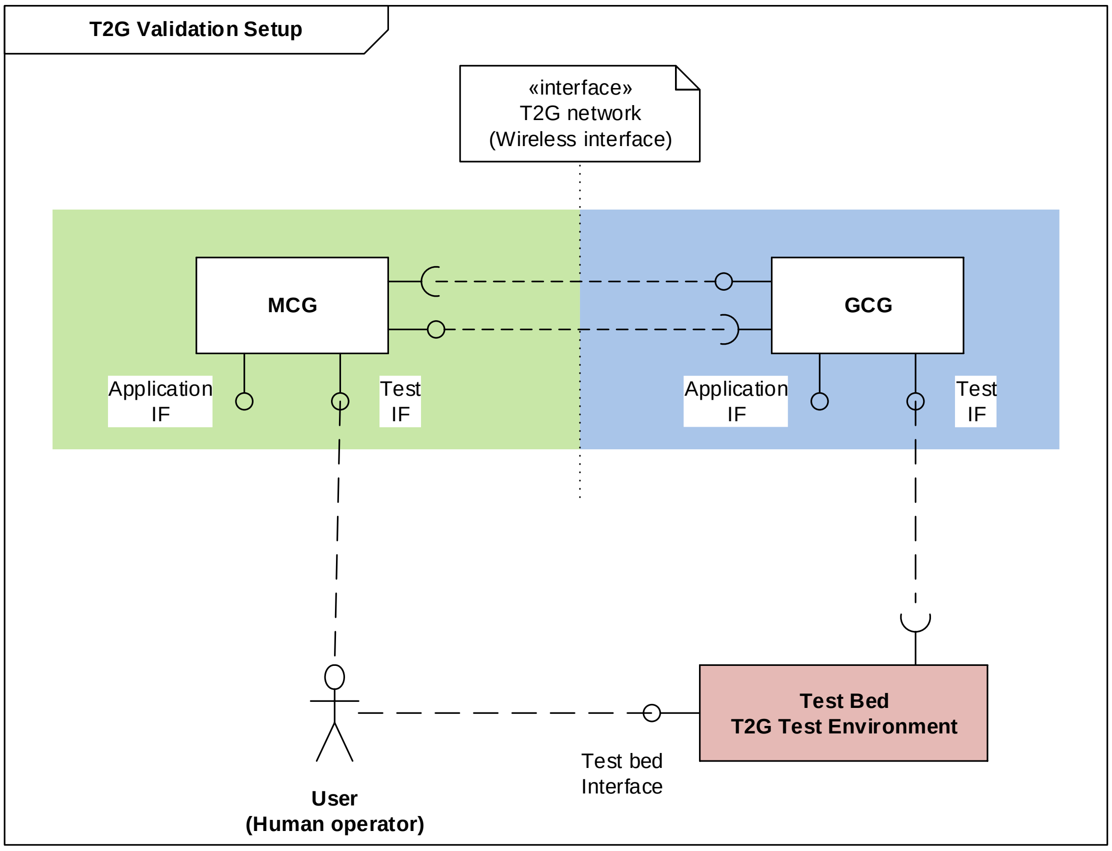


Figure 40. T2G test session setup.

During the first session, issues in the implementation of all the devices (MCGs, GCG) and also in the TG2 Test Environment itself have been detected. Some issues (obvious bugs in the software) in the MCGs, the GCG and the T2G TE could be resolved immediately. After the bugs had been resolved, all the tests performed smoothly.

The validation tests correctly detected bugs in devices under test, but also the conceptual issues in the implementations. These conceptual issues are the consequence of the ambiguities in the T2G standard [11], which allows various interpretations. The requirements and functionalities defined in the standard have been discussed in several meetings between CONNECTA and Safe4RAIL during the design and implementation phases. However, some differences have been found in the final implementations among the partners. Section 4.2 contains further discussions about next steps with the standard.

Here, for the illustration, we provide the results of the validation tests of the individual devices:

- MCG #1:
    - No problem found
- MCG #2:
    - comID 236 (Train Information Notification) sent without request from GCG (unexpected behaviour by T2G S4R TE)

- MCG does not respect comID 234 (Train Information Request) with onChange=disable (unexpected behaviour by T2G S4R TE)
- File upload does not work on MCG (curl missing on MCG)
- In comID 230, the MCG sends "kilometricPoint":-1 (according to CTA-T2.2-T-SIE-009-07, kilometricPoint is of type 'uint')
- TE expects the 'timeStamp' field changes in comID 230 between train location reads

- MCG #3:
    - Incorrect consistIDs format: [{"consistID1": "UIC001"}, {"consistID2": "UIC002"}]
    - File upload status sent with incorrect comID 202 instead of 206, invalid "storageURL": "asd"
    - Second file upload attempt uses duplicate "fileTransferUID": 12323 - already used in a previous attempt

3.3 Evaluation of Wi-Fi Access Network Simulation using the Channel Emulator

The WiFi Access Network Simulator was mainly validated in Task 3.4 and the test equipment, test setups, test description and test results are described in D3.5. In Task 3.5, the inclusion of the channel emulator in the Wi-Fi Access Network Simulation is implemented. In the following section, the test equipment, test setups, test description and test results of the channel emulator included in the Wi-Fi Access Network Simulation is explained.

The scenario is called Scenario 2.7 and uses the following elements:

- E1: Two Network Access Simulator Access Point.
- E2: Network Access Simulator Signal Attenuator.
- E3: Network Access Simulator Central PC.
- E6: Ethernet Packet Generator.
- E7: Two Packet Sniffers.
- E11: Test Automation PC
- E12: Channel Emulator

These elements are connected as presented in Figure 41.

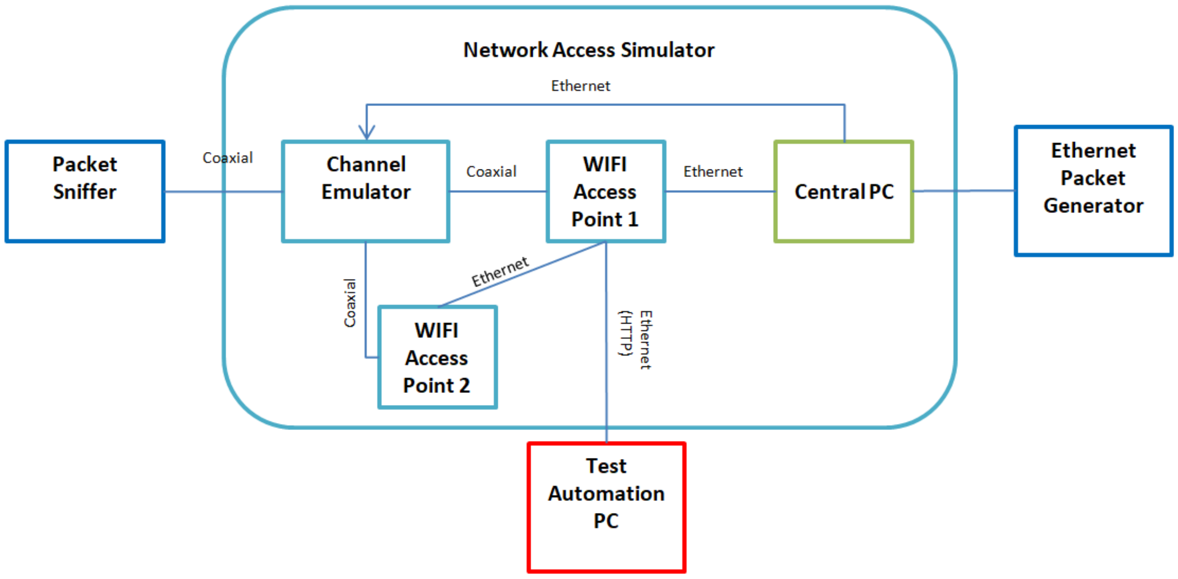


Figure 41. Scenario 2.7.

The previous scenarios presented in D3.5 test concrete and simple cases of the Network Access Simulator, but this last scenario gathers more realistic railway characteristics to validate the system. First of all, an average train station has been defined. From this ideal station, a wireless channel between a train and several APs of the train station have been modelled in order to emulate the departure of the train from the station with the channel emulator.

Therefore, depending on the positioning of the imaginary train, the system should be affected by the channel fading as well as the handover process during the test. Furthermore, several communication statistics are also monitored to evaluate the Quality of Service provided by the wireless network.

Model description:

An average train station condition is defined with two Access Points distanced for several hundred meters. First of all, the train is always connected to AP1 during the train's stop position. Then, a simplified train departure process is defined whose conditions are described in the following lines.

Two acceleration values have been considered for the departure phase, the first acceleration from static position to an initial speed and a second acceleration from that initial speed until the cruising speed is achieved.

As the emulation time goes by, the channel conditions to AP1 are deteriorated while the train moves away from the AP1. Simultaneously, the train is coming closer to AP2, so the channel conditions become more favourable. Therefore, it is expected to make a handover during the departure process to keep connectivity with the station.

Test description:

First of all, the MCG is automatically connected to AP1, consequently there is a ping response from both Access Points as well as from the GCG. Once the Packet Sniffer is activated, the LO of the channel to the AP2 is enabled. In this way, it is ensured that the connection is always established directly through AP1 and never through AP2.

Then, the emulation is commanded. The RSSI value is modified during the emulation process according to the evolution of the characteristics of the emulated channel. Suddenly, some packets are logged in the Packet Sniffer indicating the end of the handover process. Later, the ping is rechecked to verify the correct response from both APs.

Moreover, considering that currently the connection of the network is through AP2, the LO of the second channel is disabled in order to confirm the handover process. Consequently, the connection is lost and never recovered because of the demanding final channel conditions in the model for the channel to AP1.

Test Result 2.7: Channel emulator test

ID	T2.7		
Description	This test verifies that the reconnection and traffic recovery is possible in a realistic train station channel model		
Requirements	RWP3_11_6		
Scenario	Scenario 2.7		
Precondition	• Access Point 1 (E1) is activated • The Channel Emulator (E12) is activated and the signal is not modified • The Central PC (E3) is activated and configured • The Ethernet Packet Generator (E6) is loaded with SW1[2] • The Packet Sniffer (E7) is executed • Access Point 2 (E1) is configured to extend the network formed by AP1: -Access Point Mode (extends the coverage area of the wireless signal) -Same SSID as AP1 -Same authentication and key as AP1 • The Test Automation PC (E11) is connected to the AP		
Step	Step Description	Expected Step	Step Result
1	Configure the Packet Sniffer PC to connect to the SSID automatically	The PC is connected automatically	
2	Activate Packet Sniffer	The Packet Sniffer is started	
3	Start logging the packets on the Packet Sniffers	The packet logging processes has been started	
4	Activate AP2	AP2 is started	
5	Launch the emulation command	Emulation is started	
6	Wait for reconnection to AP2	A Router Solicitation packet is logged in the Packet Sniffer.	
7	Recover the network traffic	The Packet Sniffer restarts receiving packets	

[2] SW1: The Ostinato program is used to generate the network traffic data. The packet stream is formed of 100 packets with the following characteristics: A fixed packet format was specified to generate network traffic with the required MAC addresses on each test scenario. The EtherType field is constant to 0x8000 and the IP addresses are defined according to the addresses provided by the AP. The defined size for the data packets is 64 bytes long

Step	Step Description	Expected Step	Step Result
8	Stop the logging of the packets and save the packet capture in a PCAP file	A PCAP file is generated with the information of all the captured packets	
9	Check in the file the reconnection to the second AP	Evaluate the reconnection and recovery of the traffic	

Observations:

- The train station model has also been adapted considering the limitations of the channel emulator.

Description:

First of all, the train is connected to AP1 during the stop position. Two acceleration values have been considered for the departure phase, the first acceleration from the static position to an initial speed, the second acceleration from the initial speed to cruising speed.

Several APs are defined in different positions of the train station in order to extend the coverage area. Therefore, during the departure process it is expected to make a handover to keep connectivity with the station.

3.4 Evaluation of the LTE Access Network Simulation models

The Train Control Management System (TCMS) consists in providing different services to drivers and maintenance personal, in order to control and monitor the train remotely for the safety and comfort of passengers. For this reason, some data is required to be communicated between trains and ground systems [14].

Around the world, most railways use GSM-R (Global System for Mobile Communications – Rail) as the radio communications network, mainly for the voice communication needs and the data related to the ETCS (European Train Control System) [2]. GSM-R is a communication technology based on the standardised commercial GSM (Global System for Mobile Communications) equipment (2G). It is commonly used over the past decades, however, it cannot be efficient for the required and new railway services. For this reason, at the last years, the railway domain converged to the LTE (Long Term Evolution) communication technology, because its provided characteristics may fulfil the required performance for the new services [3, 5].

In this context, we provide a summary of simulation and co-simulation tests performed to validate the Train-to-Ground communication. The Train Control Management System (TCMS) consists in providing different services to drivers and maintenance personnel, in order to remotely control and monitor the train for the safety and comfort of passengers. For this reason, some data are required to be communicated between trains and ground systems [14].

3.4.1 Results for the pure simulation using LTE technology

This part focuses on the implementation and performance evaluation of the Train-to-Ground communication through LTE technology and using TCMS traffic exchanges. This communication is provided through the so-called Mobile Communication Gateway (MCG) and the Ground Communication Gateway (GCG), as specified in IEC 61375 [11]. A pure simulation is used to evaluate different scenarios based on the discrete-event network simulator Riverbed Modeler [17]. The LTE model deployed is based on the Riverbed LTE model (The Riverbed LTE Specialized Model is available for Riverbed Modeler Wireless Suite. It supports Release 8 of the 3GPP standard [18]). The trains are modelled as LTE User Equipment (UE). Evolved Nodes B (eNB) are connected to an Evolved Packet Core (EPC) node, which models the whole LTE backbone functionalities. The ground servers are directly connected to the EPC node (Figure 42).

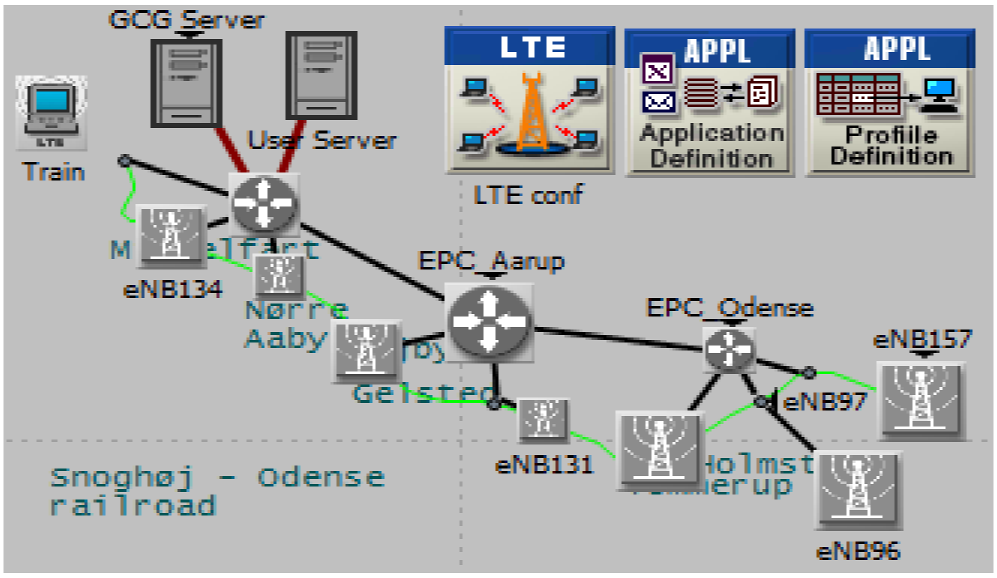


Figure 42. Topology architecture of the railway trajectory [19].

Based on the Roll2Rail project, the TCMS traffic is varied following different uses cases [20]. In the Safe4RAIL project, we select some traffic that represents the highest constraints, as described in deliverable D3.5 in Chapter 2.

As described in D3.5, we define some scenarios for the high-speed case using LTE macro cells. We study two factors that may affect performance: (I) the train speed where the train is required to move with a high speed exchanging data with the GCG through LTE and (II) the network load, where the LTE Network is shared between the train and passengers. In addition, we studied the jammer effect on performance of transferring TCMS traffic.

3.4.1.1 Simulation parameters

The used LTE parameters are summarized in Table 26. Regarding the multipath channel model, two model types can be used: The ITU Vehicular A and ITU Pedestrian A. Based on [21], the ITU Pedestrian A is a low-speed channel model and it is more suitable to be used for the train station case, where trains are stationed or move with a low speed. For this reason, the ITU Pedestrian A channel model is defined for micro cells, since they are used in the train station deployment. The ITU Vehicular A multipath channel model is suitable to be used for mobile and high speed trains. Thus, it is defined for macro cells, which are used in the high speed scenario deployment.

Table 26. LTE configuration parameters.

Parameter	Macro cells
Frequency band	920 MHz (BW: 5 MHz)
eNB Transmission power	36 dBm
eNB antenna height	50 meters
eNB antenna gain	15 dBi
UE antenna gain	1 dBi
Pathloss model	UMa[1]
Multipath channel model	ITU Vehicular A[3]

1: ITU-R M2135 Urban Macro (UMa) . The simulation randomly chooses between Line-of-Sight and Non-Line-of-Sight cases. This parameter was chosen for a preliminary configuration.

2: ITU-R M2135 Urban Micro (UMi)

3: The ITU Vehicular A multipath channel model is used for stationary and low speed trains.

4: The ITU Pedestrian A multipath channel model is used for mobile and high speed trains.

Based on the Roll2Rail project [20], the TCMS traffic is varied following different use-cases. In the Safe4RAIL project, we have selected traffic types that represent the highest constraints: signalling data, video data, file date update and voice data. These traffic types are considered for the evaluation. Table 27 describes parameters related to selected application traffic.

Table 27. TCMS application modelling.

Application	Datagram Size	Data Rate
Signalling data	128 bytes	UL 8 kbps / DL 4.6 kbps
Video data	6250 bytes	UL 1 Mbps
Voice data (live)	160 bytes	UL 64 kbps
Voice data (recorded)	160 bytes	DL 64 kbps
File data	1000 bytes	UL 2 Mbps

A set of basic user profiles predefined by Riverbed Modeler models the passenger traffic. Each profile defines a set of usual LTE user applications that exchange traffic with servers in the core network. The passenger UE profiles are presented in Table 28.

Table 28. Passenger UE profiles and their traffics.

User Profile	Traffics
Mobile User (MoU)	Instant messaging, Gaming, Interactive content Pull
Multimedia User (MuU)	VoIP and video conferencing
Engineer User (EnU)	Web browsing, Email, Telnet session, File transfer

LTE defines a class-based Quality of Service (QoS) provisioning based on the concept of bearers. These bearers are used to gather packets that have to receive a common QoS treatment. Two types of bearers are defined: Guaranteed Bit Rate (GBR) bearers and Non-GBR bearers:

- GBR (Guaranteed Bit Rate): represents the minimum rate guarantees. They are required to go through admission control when their radio bearers are created.
- Non-GBRs (Non- Guaranteed Bit Rate): represents the best effort bearers with no resource guarantees.

A bearer is associated to a QoS Class Identifier (QCI) characterized by priority level, packet delay budget and acceptable packet loss rate. In addition, a GBR bearer has fixed Uplink (UL) and Downlink (DL) data rates. Table 29 presents the standardized QCI characteristics.

Table 29. Standardized QCI characteristics.

QCI	Resource Type	Priority	Packet Delay Budget	Packet Error Loss Rate	Example Services
1	GBR	2	100ms	10^{-2}	Conversational Voice
2		4	150ms	10^{-3}	Conversational Video (live streaming)
3		3	50ms	10^{-3}	Real-Time Gaming
4		5	300ms	10^{-6}	Non-Conversational Video (buffered streaming)
5	Non-GBR	1	100ms	10^{-6}	IMS Signalling (IP Multimedia System)
6		6	300ms	10^{-6}	- Video (Buffered Streaming) - TCP-based (e.g., web, e-mail, chat, FTP, point-to-point file sharing, progressive video, etc.)

QCI	Resource Type	Priority	Packet Delay Budget	Packet Error Loss Rate	Example Services
7		7	100ms	10^{-3}	- Voice - Video (Live Streaming) - Interactive Gaming
8		8	300ms	10^{-6}	- Video (Buffered Streaming) - TCP-based (e.g., web, e-mail, chat, FTP, point-to-point file sharing, progressive video, etc.)
9		9			

We have selected a set of data traffic according to [20]: Signalling messages, voice data (recorded and live), video data and file transfer. The simulated LTE architecture considers 3 bearers according to the data type. The first one is a GBR bearer for TCMS signalling data with the highest priority and scheduling priority. The second one is also a GBR bearer for TCMS voice data with a lower priority and scheduling priority. The last one is a non-GBR bearer for TCMS video, file and all the Passenger UE data. Table 30 summarizes the bearers' setting.

Table 30. LTE bearer setting.

EPS bearer	Signalling	Voice	Default
Application(s)	Signalling data	Voice data	Other
QoS Class Identifier (QCI)	3 (GBR)	2 (GBR)	9 (Non-GBR)
Guaranteed bitrate (uplink)	16kbps	64kbps	-
Guaranteed bitrate (downlink)	16kbps	64kbps	-
Allocation retention priority	1	5	9
Scheduling priority	3	4	9
Delay budget	50ms	150ms	300ms
Packet error loss rate	10^{-3}	10^{-3}	10^{-6}

3.4.1.2 Test scenario for the train speed and network load factors study

In Europe, a train can move up to 350 km/h. However, the average is about 300 km/h. Since the train moves and exchanges TCMS traffic, it has to be connected to different LTE eNBs along its trajectory in order to provide its connectivity with the GCG. The goal of this factor study consists in evaluating the TCMS traffic transfer of a train moving with a high speed.

In addition, since trains use public LTE Networks to exchange TCMS traffic, it is required to share available bandwidth with other users such as passengers. The goal of this factor study consists in evaluating the TCMS traffic of a train when the networks is loaded by other uses, and evaluating the efficiency of LTE QoS management.

The data delivery ratio allows evaluating the capacity of LTE to transfer TCMS application data between MCG and GCG, when the connection is shared with passengers (8 passengers are used in this part). The Simulation results show that LTE provides good performance. As depicted in Table 31, all signalling data is successfully transmitted from the MCG to the GCG (Figure 43). The reason is its priority among others and parameters provided from LTE such as the Guaranteed Bearer Rate (GBR) that allows transmission integrity. Besides, this traffic application is transferred through the TCP protocol following the ARQ (Automatic Repeat reQuest) scheme. This scheme allows retransmission of lost and erroneous data. Some fluctuations exist in voice data transmission since it uses the UDP protocol without retransmission of lost packets. However, video application data is not well transmitted because there is no guarantee parameter and the UDP protocol used for this traffic kind.

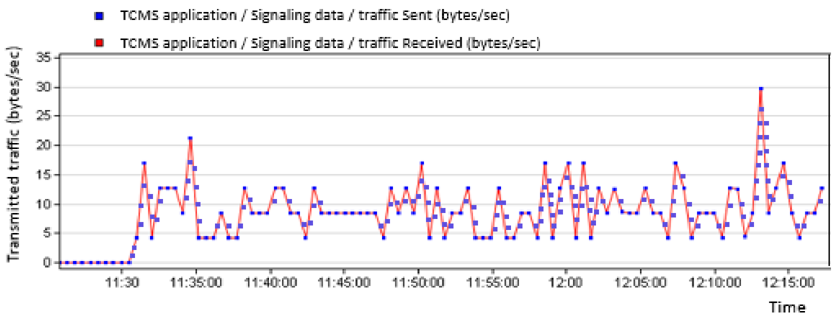


Figure 43. Sent/received MCG signalling data.

Table 31. Data delivery ratio for different TCMS applications.

	Signalling data	File update data	Video data	Voice data (live+ Recorded)
Transfer protocol	TCP	TCP	UDP	UDP
Data delivery Ratio	100%	100%	62.26%	97.4%

Figure 44 shows dropped packets of the different TCMS traffic detected in the physical LTE layer during the time. At 12:13:00, an important number of data lost is detected due to the scheduling conflicts of the different data types at this time. The conflicts lead to overloaded links which causes loss of data.

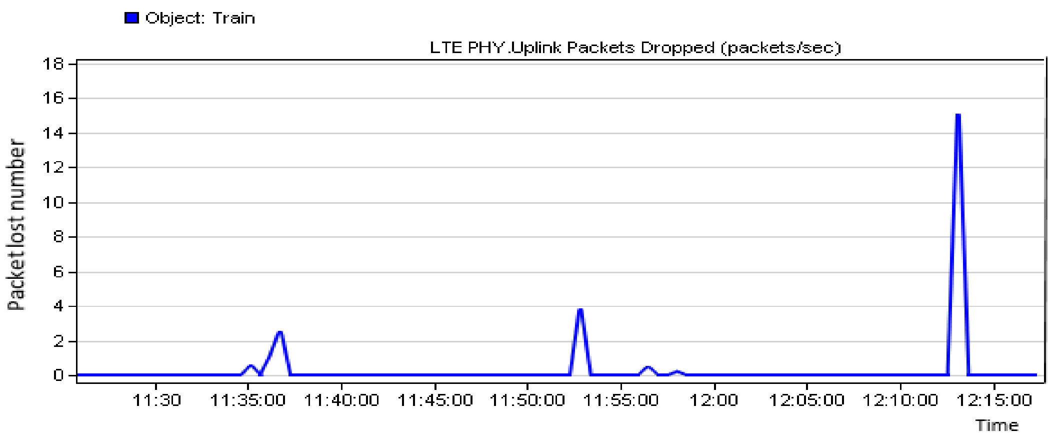


Figure 44. Packet lost number in the LTE PHY.

The data transfer delay allows evaluating the required delay to transfer traffic from the MCG to the GCG. Simulation results in Figure 45 show that all application traffic uses a stable delay during the simulation time. Almost all traffic delay is around 0.015 seconds and only voice data traffic required around 0.1 seconds.

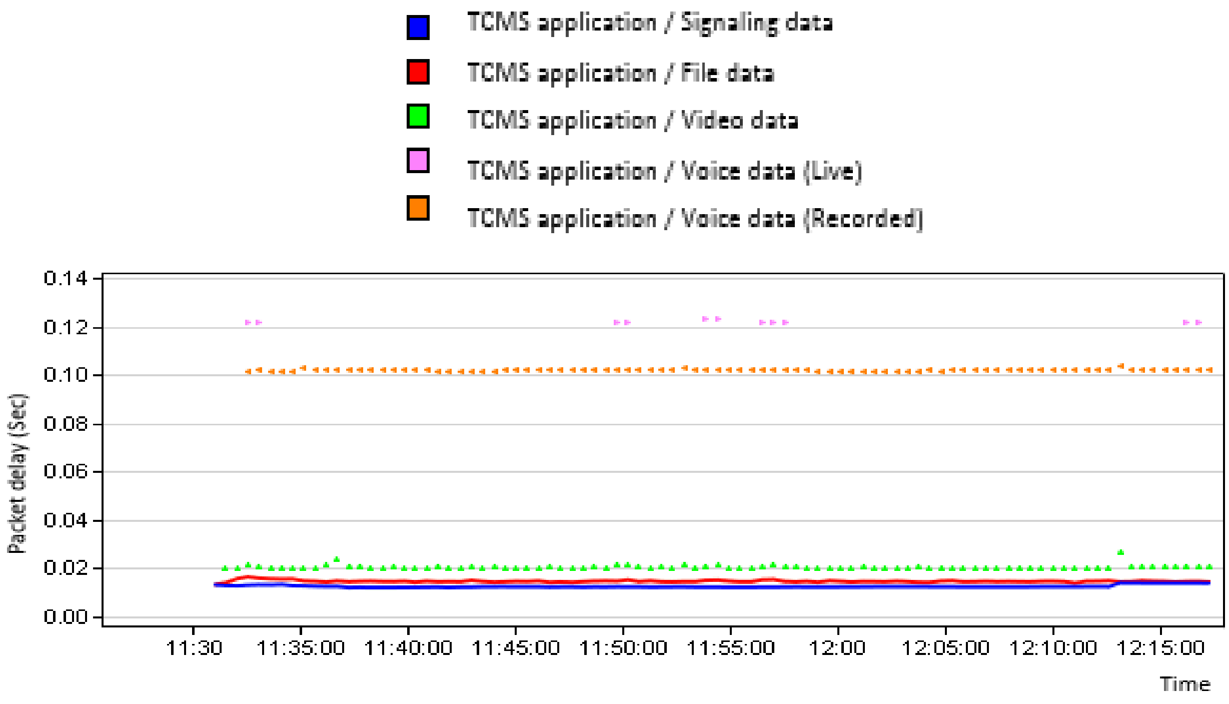


Figure 45. Packet delay for different TCMS applications.

3.4.1.3 Different Test scenarios for the network load factor

Following the standard parameters described in Table 29, we accord the appropriate QCI for each application traffic as presented in Table 32.

Table 32. LTE bearer setting.

EPS bearer	Signalling	Voice (live)	Default
Application(s)	Signalling data	Voice data	Other
QoS Class Identifier (QCI)	5 (Non-GBR)	1 (GBR)	9 (Non-GBR)
Priority	1	2	9

We first start with a topology where a non-congested network consists of one train and three users (1 Mobile User, 1 Multimedia User and 1 Engineer User). As shown in Figure 46, the signalling traffic is received completely by the destination (server in the ground) and there is no retransmission of packets since the communication follows the TCP protocol. The same results are found for the other traffic types.

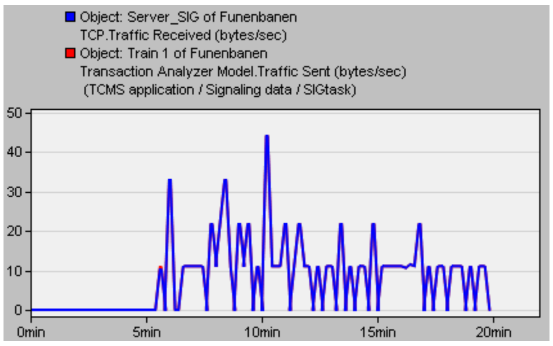
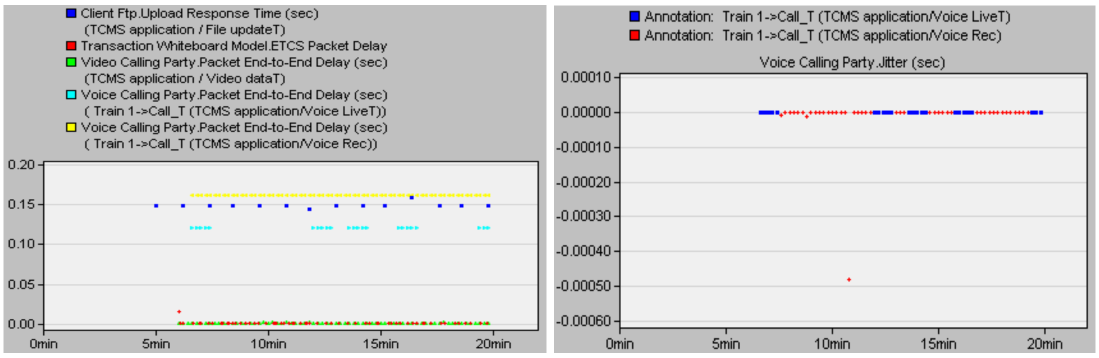


Figure 46. Sent/received MCG signalling data for not-congested network.

As presented in Figure 47, the end-to-end delay for all traffic is almost stable during the simulation, which allows stable use of the different applications.

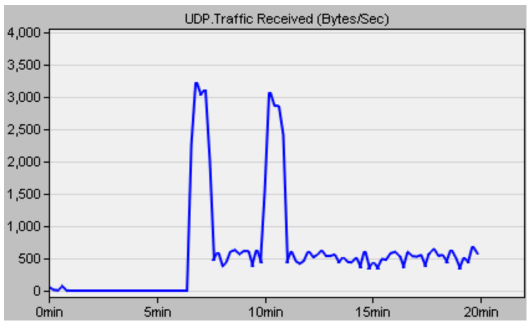


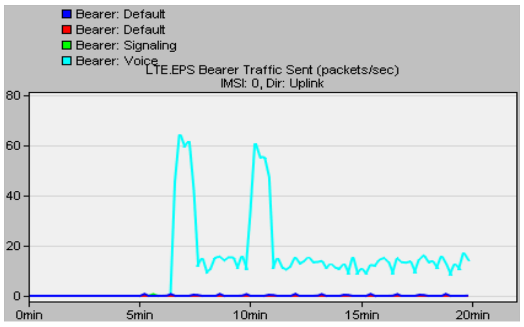
(a) End-to-End delay

(b) Jitter for the voice data

Figure 47. Packet delay for different TCMS applications for not congested network.

However, for a congested network and with these bearer parameters, simulation is aborted during its runtime. The reason consists in using the most bandwidth for a non-guaranteed traffic. For this reason, we change bearer parameters as presented in Table 30 in Subsection 3.4.1.1. Therefore, it is necessary to assign guarantees for traffic with high priorities in a congested network.





a) Received MCG voice data b) LTE EPS Bearer

Figure 48. Results for congested network.

For a scenario with more users (congestion increased), results provide that only voice data is sent as illustrated in Figure 48.a). Other data is not sent. This is can be explained by the fact that the remaining resources cannot be sufficient for the traffic with the highest priority (signalling data). Thus, voice data will take the allowed resources and is sent instead of signalling data (show Figure 48.b)). Indeed, in a congested cell, GBR radio bearers with low priority are preempted in case it is sufficient to accept a radio bearer request for an EPS bearer with a high priority.

3.4.1.4 Other TCMS traffic evaluation for the network load factor

In this part, we evaluate TCMS traffic exchange based on CONNECTA requirements defined in [30]. Two traffic types are used by the TCMS: best effort data and message data. We use the configuration illustrated in Table 33, where only message data uses the TCP protocol.

Table 33. LTE bearer setting.

EPS bearer	Gold	Default
Application(s)	Interactive msg	others
QoS Class Identifier (QCI)	3 (GBR)	9 (Non-GBR)
Priority	3	9

The scenario consists of one train and 39 users. The traffic message with the high priority is sent and received with almost 100 % (as shown in Figure 49). However, the best effort data is not sent. There is a lack of sufficient resources since the traffic with the highest priority following a guaranteed bit rate uses the existing resources. In addition to the complete transmission, LTE can provide a stable end-to-end delay for message data exchange. It amounts to around 0.02s as shown in Figure 50.

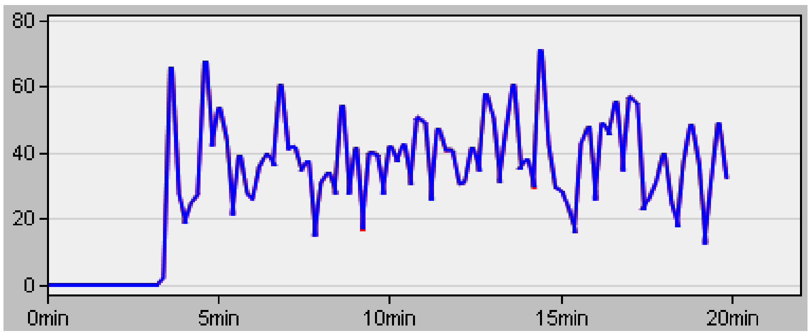


Figure 49. Sent/received MCG message data for congested network.

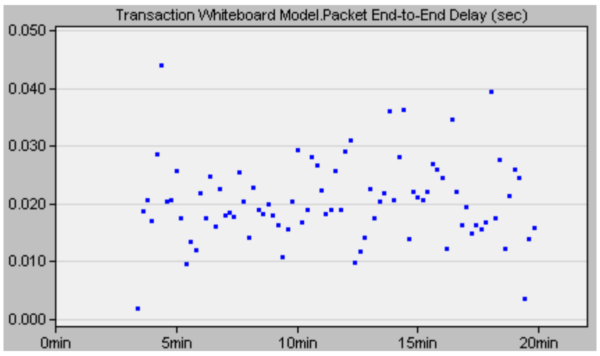


Figure 50. Packet delay for message data.

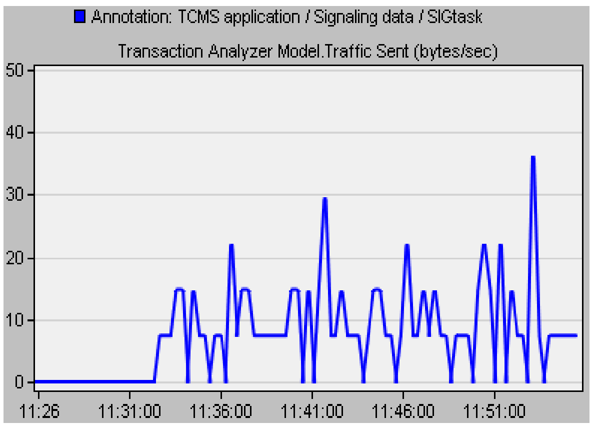
3.4.1.5 Test scenario for the existence of jammer node

TCMS traffic exchange between MCG and GCG is important in order to monitor and control trains and provide safety of passengers. For these reasons, LTE should provide continuous connectivity between these two entities. However, wireless links are easily exposed to attacks by jamming technology, which impacts connectivity and data exchange [22]. Jamming is the disruption of the wireless communication. It transmits an interfering wireless signals in order to decrease the signal-to-noise ratio at the receiver sides.

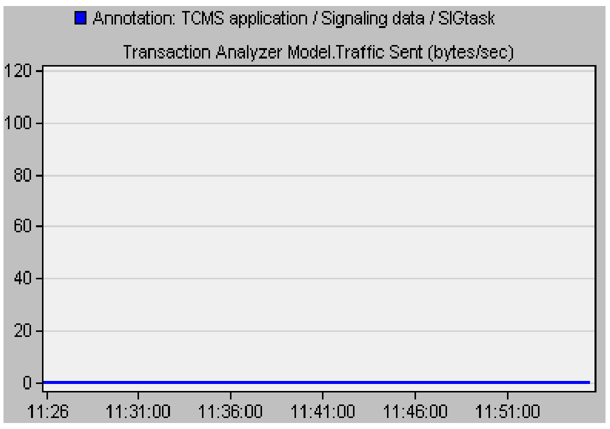
Riverbed Modeler provides three models of basic jammer presented as a jammer node:

- Pulsed jammer: is defined as a node that provides transmission of packets on a single fixed frequency. This transmission is masked by a periodic pulse which is controlled by the process model of the jammer.
- Frequency-swept jammer: is defined as a node that provides continuous transmission on a range of frequencies varied at a specific rate.
- Fixed-frequency single-band jammer: is defined as a node that provides continuous transmission of packets on a single fixed frequency band at the rate of 1 packet per second.

We propose some tests of scenarios with the existence of a jammer node around an eNodeB. The goal of these scenarios is to evaluate the reactivity of jamming in the data sent.

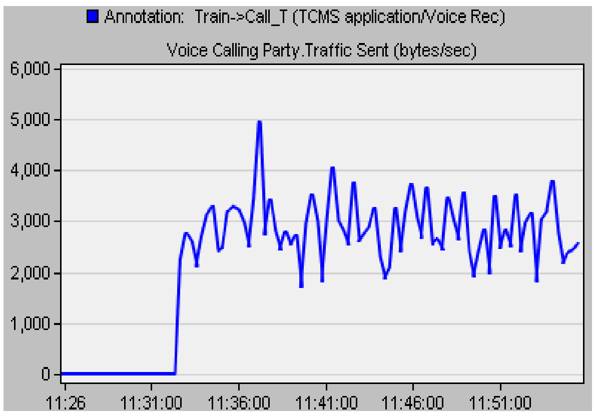


(a) Without jammer node

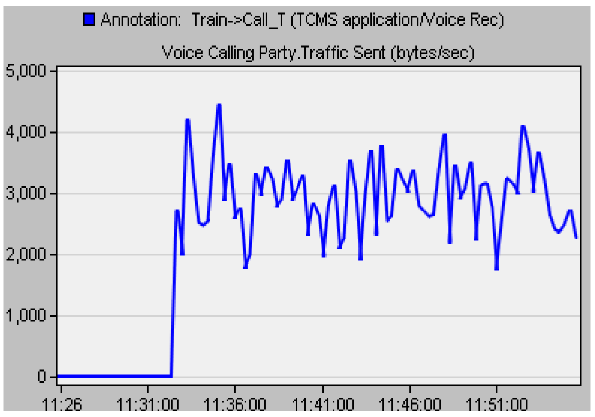


(b) With jammer node

Figure 51. Signalling data transferring without and with jammer node existing.

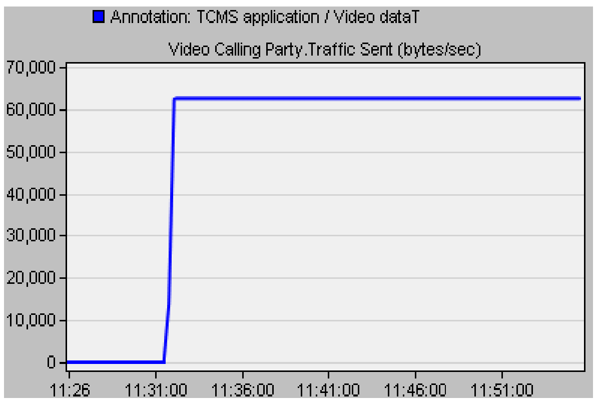


(a) Without jammer node

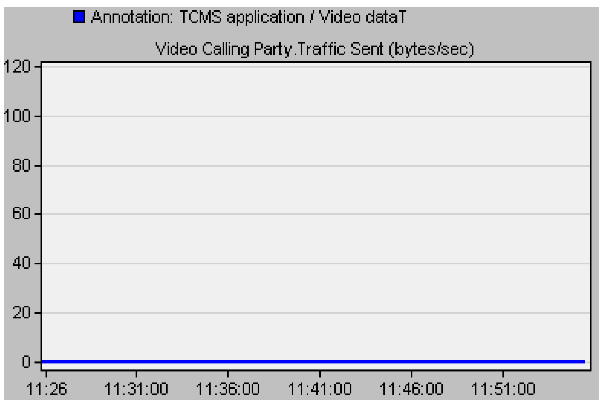


(b) With jammer node

Figure 52. Voice data transferring without and with jammer node existing.

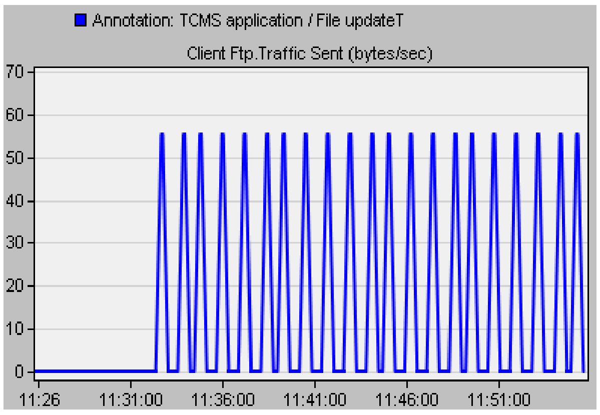


(a) Without jammer node

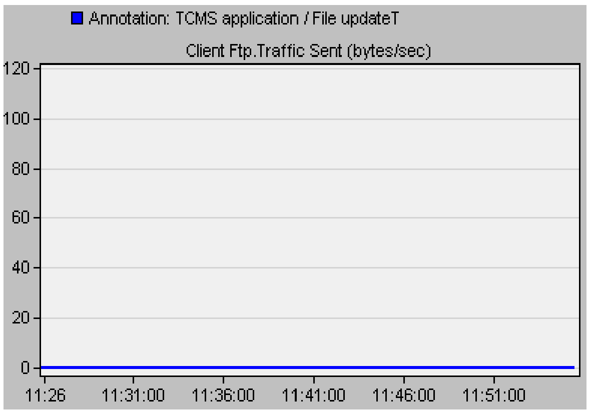


(b) With jammer node

Figure 53. Video data transferring without and with jammer node existing.



(a) Without jammer node



(b) With jammer node

Figure 54. File data transferring without and with jammer node existing.

The simulation results show that without jammer node the different TCMS traffic data is sent by the MCG in the train side to the GCG through the LTE Network, especially through its connectivity with the eNodeB. However, a jammer node around eNodeb avoids the transmission of some data. As illustrated in Figure 51, Figure 52, Figure 53 and Figure 54, the signalling, video and file data is not sent due to the interfering signal caused by the jammer node. Voice data is sent successfully with some fluctuations.

For this reason, additional techniques of anti-jamming are crucial. Wireless Networks are very exposed to some attacks that affect the Network security and the safety of trains and passengers in consequence.

3.4.2 Results for the pure simulation using LTE and WiFi

This section provides results for the pure simulation using LTE and WiFi. After explaining the network architecture and the simulation context/parameters, the results are analysed. We consider two scenarios, one testing connectivity and the other one evaluates the performance in case of a loaded WiFi.

3.4.2.1 Network architecture

The WiFi-LTE network offers a heterogeneous radio access for T2G communication:

– The LTE radio access is available on the entire network. The trains use this access for T2G communications on the sections where they travel at high speed.
– The WiFi radio access is available on specific areas like train stations, shunting zones, train shed, etc. The trains use this access for T2G communications when it is stationary or moving with low speed within these areas.

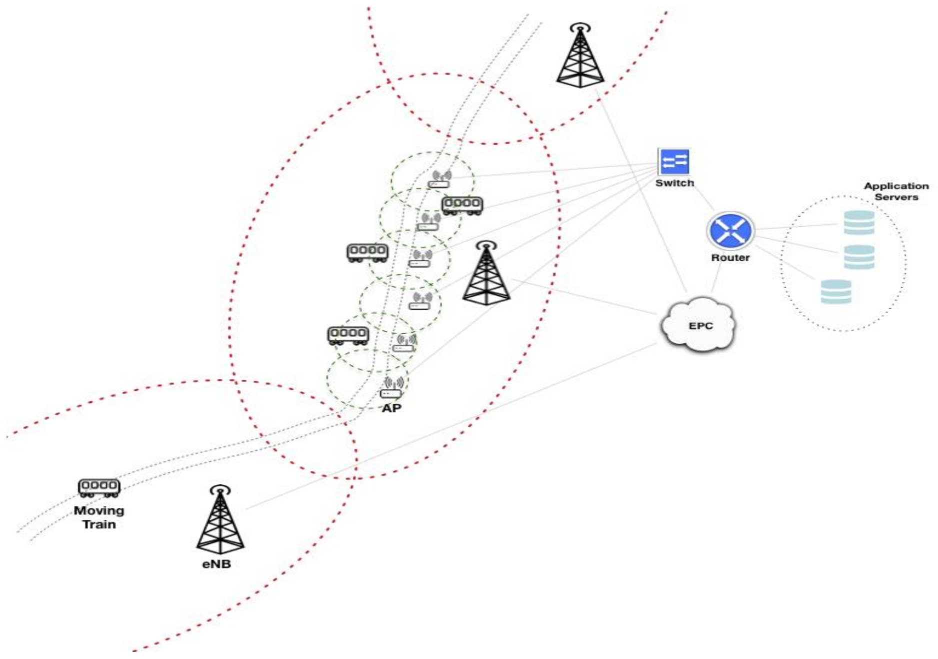


Figure 55. WiFi-LTE network architecture.

Figure 55 presents an example of a WiFi-LTE network. LTE Macro-cells are deployed along the railway line. At the train station, the main line (as well as the secondary sections of the station) is covered by WiFi cells. A train, which moves through the main line, is connected through the LTE radio access until it enters the station. In the station, the moving train communicates using WiFi radio access, as well as stationary trains at the station.

As shown in Figure 56, the MCGs have WiFi and LTE interfaces to communicate alternatively or simultaneously using the WiFi and LTE radio accesses.

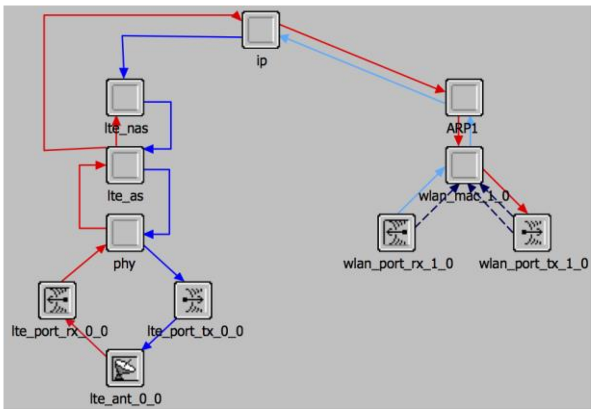


Figure 56. WiFi-LTE MCG architecture.

Each interface has its own IP address and is associated to its radio access network when available. To ensure the vertical handover between the LTE and WiFi, we configure the train routing table so that the routes using the LTE interface have higher costs than those using the WiFi interface. Thus, the WiFi interface will be used when both radios are available.

3.4.2.2 Simulation context and parameters

The goal of the simulations is the evaluation of performances offered by the WiFi-LTE network for T2G communications. The proposed scenarios consider a high-speed line crossing a train station. The LTE radio access network covers the overall high-speed line (including the station). The WiFi radio access covers the train station including the section of the high-speed line crossing it. A train, moving through the high-speed line, performs the T2G communications using the LTE access. When this train goes through the station, the T2G communication is switched to the WiFi radio access. In the following, we detail the parameters implementing this context in Riverbed Modeler simulator.

Figure 55 gives an overview on the modelled network architecture. The main components of the architecture are:

- An LTE network modelled by a set of eNBs connected to a core network node modelling the EPC,
- A WiFi network modelled by a set of Access Points (AP) gathered in an Ethernet based LAN,
- A router connecting the LTE network and the WiFi network to the application servers.

The network layer connectivity is based on IP and the routing is based on the OSPF protocol.

The train station is covered with a WiFi radio access though ten APs gathered in an Ethernet LAN. This WiFi network is connected to the core network through a router. For WiFi radio coverage, we consider the 5 GHz frequency band. Table 34 presents the WiFi deployment parameters.

Table 34. WiFi configuration parameters.

Frequency band	5 Ghz
WiFi version	IEEE 802.11n
Data Rate	65Mbps(base)/600 Mbps(Max)
APs and trains Transmission power	0.005 W
Train antenna height	3 meters
AP antenna height	5 meters
Inter-AP distance	108 meters

WiFi offers an optional QoS management mechanism named Enhanced distributed channel access (EDCA). EDCA uses the native contention based channel access of WiFi. It implements the QoS management based on Access Categories (AC) used to gather packets that have to receive a common QoS treatment. Every AC is associated to a queue for channel access with specific parameters. Four Access Categories are defined: AC_VO (AC3), AC_VI (AC2), AC_BE (AC1) and AC_BK (AC0).

We make use of EDCA to ensure the service differentiation between data exchange based on application requirements. As shown in Table 35, the signalling data is associated with the highest priority access category AC_VO, the AC_VI is assigned to voice data and finally the lowest priority access category AC_BE is associated with video and file transfer data.

Table 35. WiFi Access Categories.

WiFi Access Categories	Priority	TCMS Applications
AC_VO	3	Signalling data
AC_VI	2	Voice data (live and recorded)
AC_BE	1	Other

3.4.2.3 Evaluation results

To evaluate the pure network simulation using LTE and WiFi, two scenarios are defined. The first evaluates the connectivity while the second one tests considers performance for a loaded WiFi connection.

3.4.2.3.1 Connectivity scenario

This scenario is proposed to evaluate the effect of the vertical handover performed when a train switches its T2G communication from LTE access to WiFi and vice-versa.

We define a train moving within the high-speed line while implementing the TCMS applications. The train is performing a trajectory of 4 Km including 1 Km in the station area. The train speeds are about 50 km/h while approaching the station area and 10 km/h inside the station area.

Figure 57 shows the train association to the LTE eNodeB and WiFi Access Points. The train begins its trajectory outside the station. It is the exclusively in the coverage of the LTE radio access and associates to the first eNodeB (eNB_0). It makes handovers to next eNBs, while moving through the line. When the train enters to the station, it detects the WiFi radio access and connects to the first access point AP_0. While crossing the station, the train performs handovers between WiFi APs in its way.

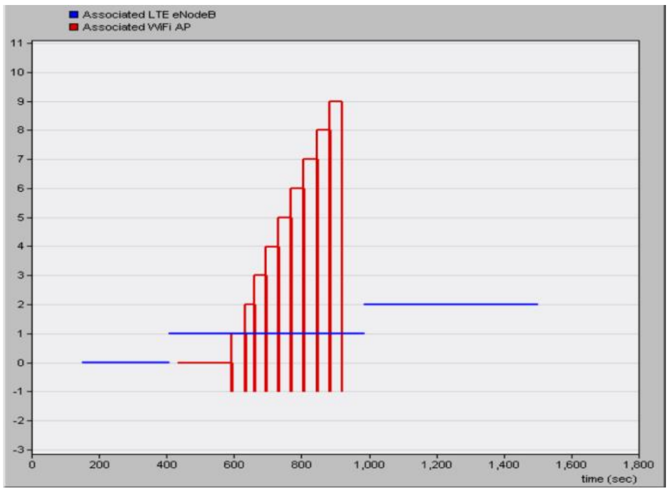
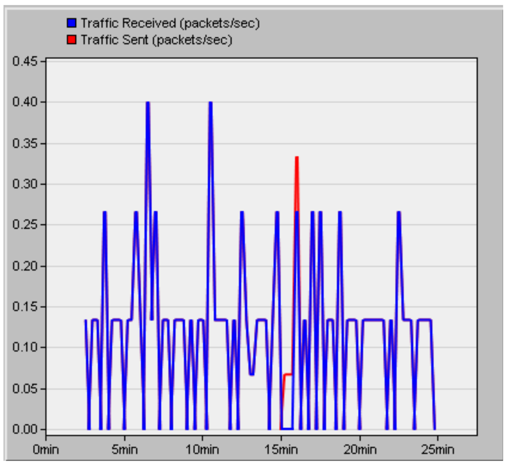


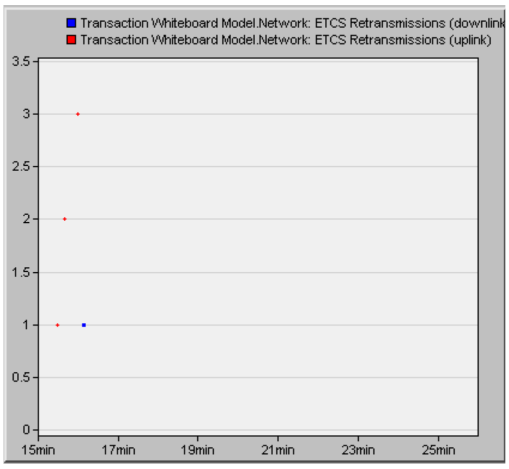
Figure 57. Train association to the LTE eNodeB and WiFi Access Points.

In the same scenario, we evaluate the effects of horizontal (intra-technology) and vertical handovers (inter-technology) on application performances.

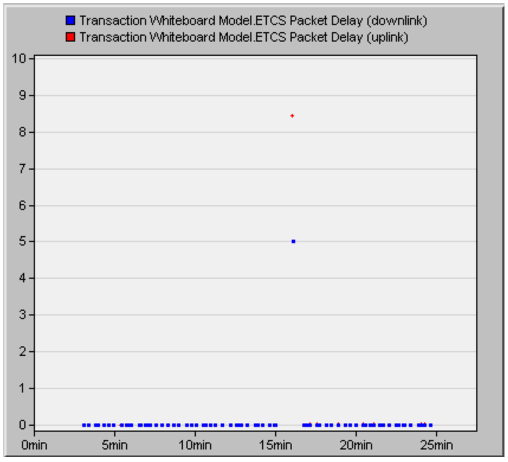
Figure 58 shows performances for the signalling data. We consider sent vs. received packets and End-to-End delay and packet retransmissions. The sent vs. received packets graph (Figure 58 a) shows a packet loss between minutes 15 and 16 of the simulation. Figure 58 b) and Figure 58 c) show three retransmissions and an increase of End-to-End delay at the same simulation time. This time corresponds to the exit of the station area (and consequently the exit of the WiFi coverage) by the moving train.



(a) Sent vs. received packets



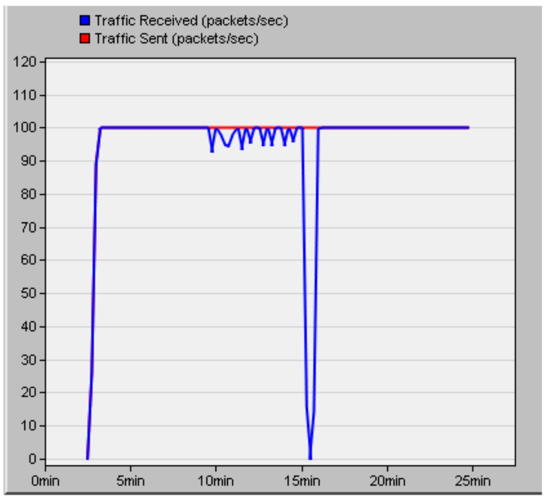
(b) Packet retransmissions



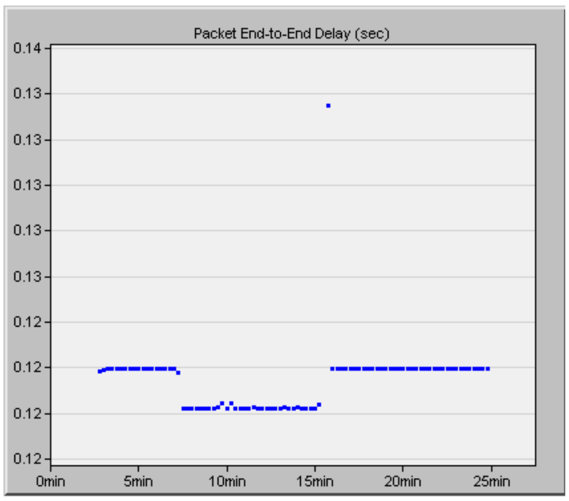
(c) Packet delay

Figure 58. Statistics for signalling (connectivity scenario).

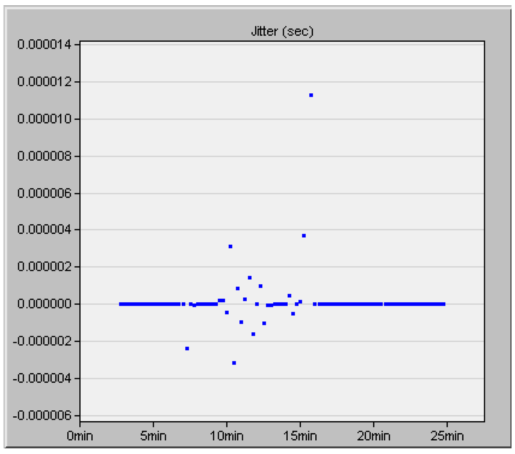
Figure 59 shows performances for the voice applications. We consider sent vs. received packets, end-to-end delay and jitter. The sent vs. received packets graph (Figure 59 a) shows several sequences of packet loss and a peak of loss between minutes 15 and 16 of the simulation time. Figure 59 b) shows an increase of end-to-end delay between minutes 15 and 16 of the simulation time. The minor packet loss sequences correspond to a change of WiFi access points inside the station area. The peak of packet loss and increase of end-to-end delay correspond to the exit of the station area and consequently the exit of the WiFi coverage by the moving train.



(a) Sent vs. received data



(b) Packet delay



(c) Jitter

Figure 59. Statistics for voice (connectivity scenario).

Overall, the results obtained show that the establishment of WiFi-LTE heterogeneous connectivity allows a relative continuity in the communications of a mobile train. However, the results of voice applications show that there are several disconnections that cause packet losses during the handovers between WiFi APs and during the handover from WiFi to LTE.

Regarding the critical signalling application, the observed packet losses and delay increase remain in compliance with the imposed limits. This is not the case for the voice and video applications for which the losses are likely to cause disturbances perceived by users.

These disconnections can be minimized with more optimization of the mobility management. For the intra-AP handovers, improvements have to be made in terms of handover triggers and also in the downlink traffic redirection at the switch that connects the APs.

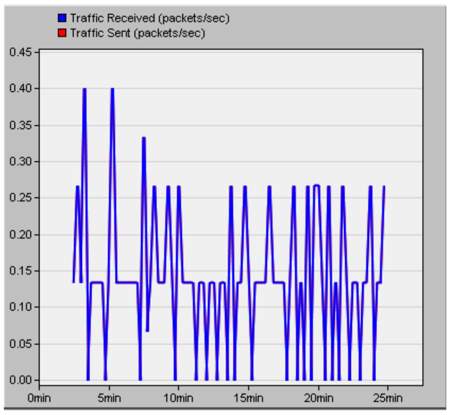
For WiFi to LTE handover, proactive redirection mechanisms must be implemented to transfer downlink traffic from the WiFi core network to the LTE core network.

3.4.2.3.2 Loaded WiFi scenario

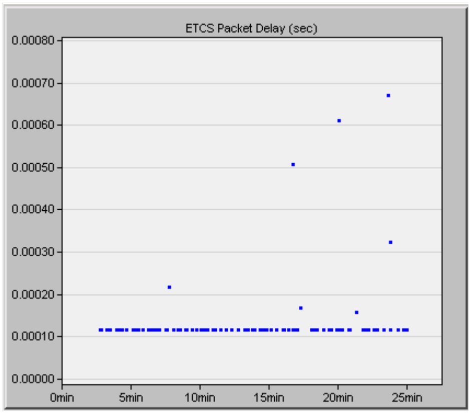
This scenario is proposed to evaluate the performances offered by the WiFi-LTE network to TCMS applications in network load conditions.

We consider the previous scenario while adding a set of stationary trains in the station implementing the same TCMS applications as the moving train. We consider 40 stationary trains each 4 associated to one WiFi access point. We evaluate the performances offered to TCMS applications of the mobile and the stationery trains.

Figure 60 and Figure 61 show the observed performances for the TCMS applications of the stationary trains. The obtained results show optimal performances for signalling data and voice applications. The same performances are observed for the video application.

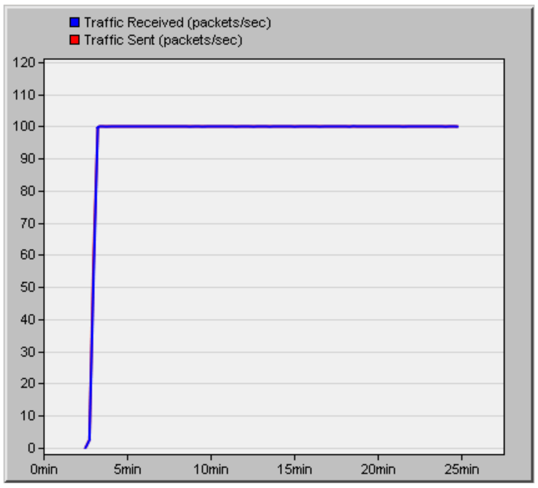


(a) Sent vs. received data

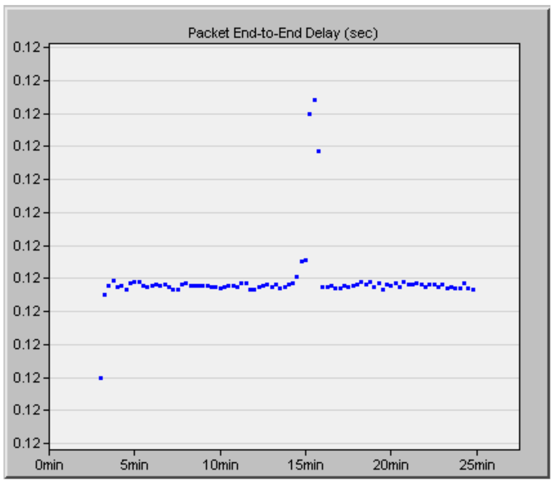


(b) Packet delay

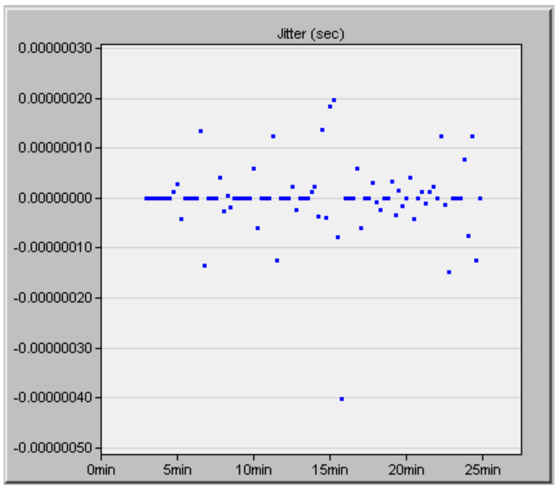
Figure 60. Statistics for signalling of stationary trains (Loaded WiFi).



(a) Sent vs. received data



(b) Packet delay



(c) Jitter

Figure 61. Statistics for voice of stationary trains (Loaded WiFi).

The results obtained for the mobile train are equivalent to the previous scenario. The presence of the stationary trains and their traffic exchange do not affect the performances perceived by the applications of the moving train.

Overall, these results show that the WiFi network implemented in the station area can largely support the traffic generated by stationary and mobile trains. It remains that the EDCA QoS management mechanism implemented by WiFi is known to be non-performing in overload conditions.

3.4.3 Results for the Co-simulation using LTE technology (IFS)

The goal of the co-simulation consists in validating the GCG communication with an LTE terminal, through an end to end communication path formed by an emulation leg (LTE eNodeB + LTE EPC) and a simulation leg (Backhaul network)

3.4.3.1 Test Equipment

The Lab is shown in Figure 62. The GCG Simulator is running on a laptop as a host (E8). UE (E4) can interact with the GCG host (E8) via the path of the LTE emulator and the OPNET simulation. The GCG Simulator is installed on a Linux based laptop as a GCG host (E8). Details of the GCG Simulator are described in Section 3.7 in D3.5, *Proof-of-concept implementation of T2G Test Environment*. The test tools library for the test execution is used as BASH script files. The list of GCGSim meta-test commands is listed in Section 3.5.1 in D3.5.

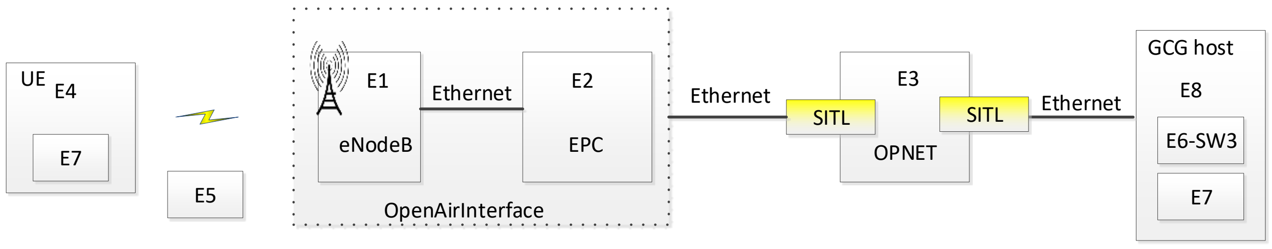


Figure 62. System layout in test setup.

The system layout in the test setup consists of the following subsystems:

- E1: LTE eNodeB.

  A laptop that runs an OpenAirInterface program with Software Defined Radio (SDR) which acts as eNodeB. The SDR is used for emulating signal loss and consequently a communication failure. Two Ethernet cards are required (USB-Ethernet converter can work).

- E2: LTE EPC.

  A laptop that runs an OpenAirInterface program which acts as EPC [23]. Two Ethernet cards are required (USB-Ethernet converter can work).

- E3: A laptop as Riverbed's OPNET simulator.

  A laptop that runs an OPNET program which acts as backbone network. Two licenses for SITL interfaces are required. [24]

  SITL (System-In-The-Loop): a tool from Riverbed that provides an interface to connect real network to simulated network. It converts data packets from real to simulated environment and vice versa.

- E4: LTE UE

  A PC with LTE Communication dongle

- E5: Signal Sniffer.

  Device that allows visualizing the radio frequency signals. An oscilloscope-like UI is used to check the presence of the LTE signal.

- E6: Generator of TCP connection and UDP stream.

  A laptop that runs an iperf traffic generator (SW3) [25] which generates network traffic.

- E7: Packet Sniffer.

  A laptop that runs the Wireshark program in order to monitor and log the packets exchanged in the network.

- E8: GCG host.

  A laptop which represents a simple custom GCG device. This laptop can include Ethernet Packet Generator (E6-3) and Packet Sniffer (E7) functionalities.

The GCGSim test cases in Section 5.4 in D3.5 are re-examined in this hybrid LTE emulator/OPNET simulator test environment. The GCG simulator is running on the GCG Host (E8) and the user connects to the server with a pre-assigned URL (in our case, http://10.10.10.6:8085) from the UE device (E4).

The GCGSim and test tools are configured to interact with each other directly on the same PC in the tests described in D3.5. In the test herein described, the end-to-end communication is performed over the LTE wireless network, the Open Air Interface emulator and the simulated backhaul network in Riverbed's OPNET simulator.

3.4.3.2 Test 1. Validation of GCGSim Environment

Description: This test verifies the communication between UE and GCG host under GCGSim Environment

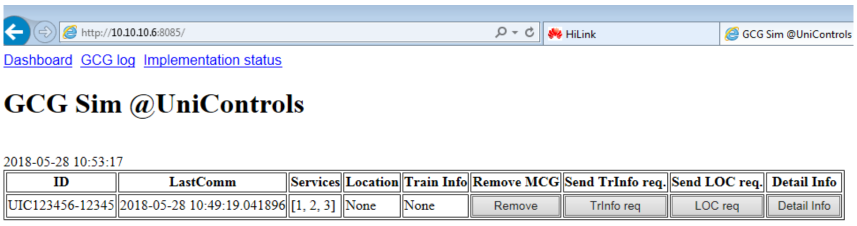
Precondition:

- The LTE emulator and OPNET simulator (E1-E4) are activated.
- The GCG host (E8) has been configured and interacted with UE (E4).
- Packet Sniffers (E7) is activated to allow the logging of the packets.

Step	Step Description	Expected Result
1	Connectivity test	UE connects and interacts with GCG host (results shown in Test 1.1)
2	Announcement Service	UE shows announcement information when the command is set in GCG host (results shown in Test 1.2)
3	Remove MCG Service	User can remove MCG from UE or MCG can be removed from GCG hosts (results shown in Test 1.3)
4	Location Notification Service	UE shows location information when the command is set in GCG host (results shown in Test 1.4)
5	Train Info Notification Service	UE shows train information when the command is set in GCG host (results shown in Test 1.5)
6	File Upload Service	File can be downloaded from UE when the command is set in GCG host (results shown in Test 1.6)
7	Reset MCG Service	UE shows MCG information when the command is set in GCG host (results shown in Test 1.7)
8	Train Info Request Service	User can request train info from both UE and GCG hosts (results shown in Test 1.8)
9	Train Location Request Service	User can request location info from both UE and GCG hosts (results shown in Test 1.9)

Test 1.1: Connectivity test at UE (E4)

The test is used to validate the successful connection between UE (E4) and GCG host (E8). Figure 63 shows the connection between UE and GCG using the announce command. Figure 64 further illustrates location and train information.



http://10.10.10.6:8085/

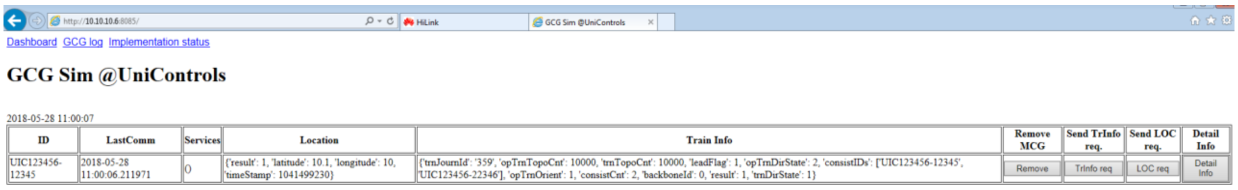
Dashboard GCG log Implementation status

GCG Sim @UniControls

2018-05-28 10:53:17

ID	LastComm	Services	Location	Train Info	Remove MCG	Send TrInfo req.	Send LOC req.	Detail Info
UIC123456-12345	2018-05-28 10:49:19.041896	[1, 2, 3]	None	None	Remove	TrInfo req	LOC req	Detail Info

Figure 63. UE connection with GCG host, with the announce command.



http://10.10.10.6:8085/

Dashboard GCG log Implementation status

GCG Sim @UniControls

2018-05-28 11:00:07

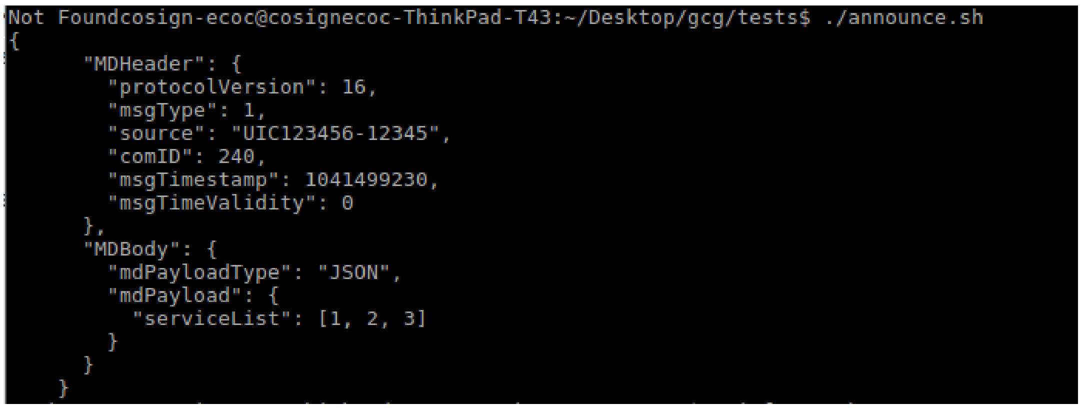
ID	LastComm	Services	Location	Train Info	Remove MCG	Send TrInfo req.	Send LOC req.	Detail Info
UIC123456-12345	2018-05-28 11:00:06.211971	0	{'result': 1, 'latitude': 10.1, 'longitude': 10, 'timeStamp': 1041499230}	{'trnJournId': '359', 'opTrnTopoCnt': 10000, 'trnTopoCnt': 10000, 'leadFlag': 1, 'opTrnDirState': 2, 'consistIDs': ['UIC123456-12345', 'UIC123456-22346'], 'opTrnOrient': 1, 'consistCnt': 2, 'backboneId': 0, 'result': 1, 'trnDirState': 1}	Remove	TrInfo req	LOC req	Detail Info

Figure 64. UE connection with GCG host with location info and train info command.

Test 1.2: Announcement Service

This test checks the announcement service provided by GCGSim. The results depicted in Figure 65 return a service list with three elements. It uses the following script:

- announce.sh - sendServiceList()



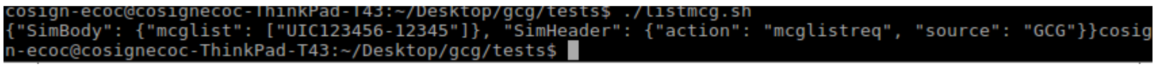
```
Not Foundcosign-ecoc@cosignecoc-ThinkPad-T43:~/Desktop/gcg/tests$ ./announce.sh
{
      "MDHeader": {
        "protocolVersion": 16,
        "msgType": 1,
        "source": "UIC123456-12345",
        "comID": 240,
        "msgTimestamp": 1041499230,
        "msgTimeValidity": 0
      },
      "MDBody": {
        "mdPayloadType": "JSON",
        "mdPayload": {
          "serviceList": [1, 2, 3]
        }
      }
    }
```

Figure 65. Test tool result of announce command.

Test 1.3: Remove_MCG Service

This test checks the TE API remove_mcg service provided by GCGSim and lists the available MCGs as shown in Figure 66. Two scripts are used:

- remmcg.sh - simRemMcgReq()
- listmcg.sh - simMcgListReq()



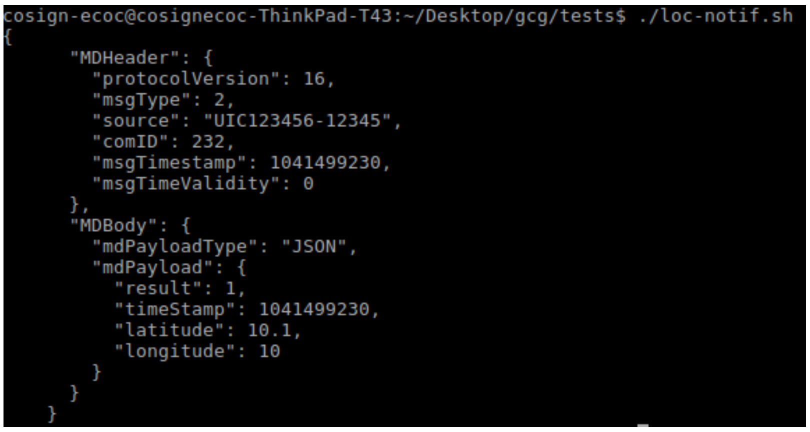
```
cosign-ecoc@cosignecoc-ThinkPad-T43:~/Desktop/gcg/tests$ ./listmcg.sh
{"SimBody": {"mcglist": ["UIC123456-12345"]}, "SimHeader": {"action": "mcglistreq", "source": "GCG"}}cosig
n-ecoc@cosignecoc-ThinkPad-T43:~/Desktop/gcg/tests$ 
```

Figure 66. Test tool result of list MCG command.

Test 1.4: Location Notification Service

This test checks the train location notification service provided by GCGSim. In this case, the latitude is 10.1 and the longitude 10.0 degree (shown in Figure 67). It uses the script:

- loc-notif.sh - sendLocNotif()



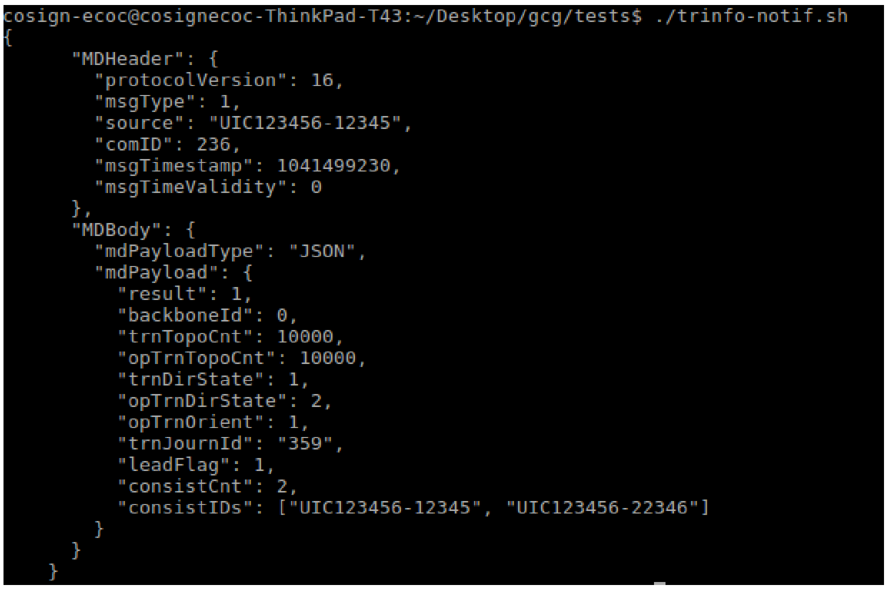
```
cosign-ecoc@cosignecoc-ThinkPad-T43:~/Desktop/gcg/tests$ ./loc-notif.sh
{
      "MDHeader": {
        "protocolVersion": 16,
        "msgType": 2,
        "source": "UIC123456-12345",
        "comID": 232,
        "msgTimestamp": 1041499230,
        "msgTimeValidity": 0
      },
      "MDBody": {
        "mdPayloadType": "JSON",
        "mdPayload": {
          "result": 1,
          "timeStamp": 1041499230,
          "latitude": 10.1,
          "longitude": 10
        }
      }
    }
```

Figure 67. Test tool result of sent location notification command.

Test 1.5: Train Info Notification Service

This test checks the train information notification service provided by GCGSim. It returns information such as the backbone and consist IDs, the orientation or a lead flag (Figure 68). The related script is:

- trinfo-notif.sh - sendTrainInfoNotif()



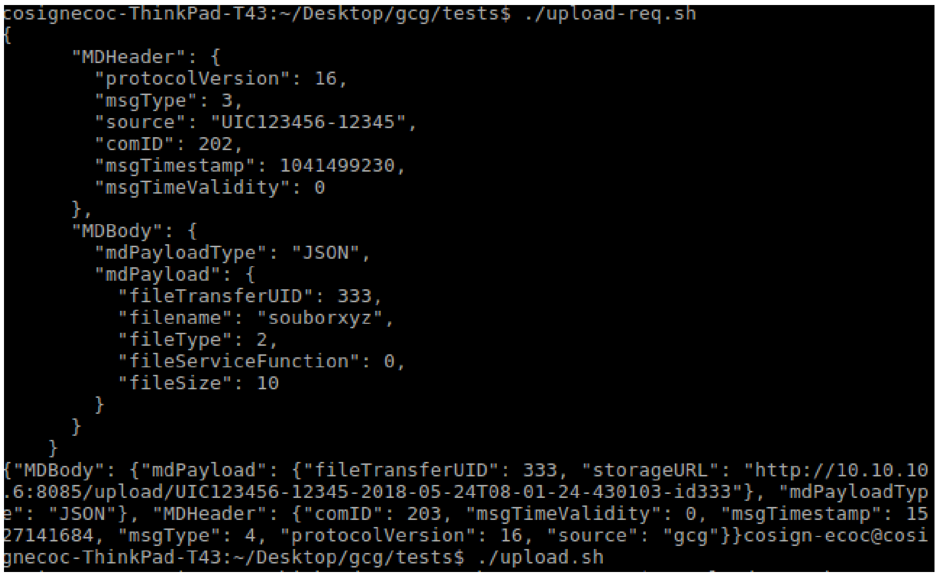
```
cosign-ecoc@cosignecoc-ThinkPad-T43:~/Desktop/gcg/tests$ ./trinfo-notif.sh
{
      "MDHeader": {
        "protocolVersion": 16,
        "msgType": 1,
        "source": "UIC123456-12345",
        "comID": 236,
        "msgTimestamp": 1041499230,
        "msgTimeValidity": 0
      },
      "MDBody": {
        "mdPayloadType": "JSON",
        "mdPayload": {
          "result": 1,
          "backboneId": 0,
          "trnTopoCnt": 10000,
          "opTrnTopoCnt": 10000,
          "trnDirState": 1,
          "opTrnDirState": 2,
          "opTrnOrient": 1,
          "trnJournId": "359",
          "leadFlag": 1,
          "consistCnt": 2,
          "consistIDs": ["UIC123456-12345", "UIC123456-22346"]
        }
      }
    }
```

Figure 68. Test tool result of sent train info notification command.

Test 1.6 File Upload Service

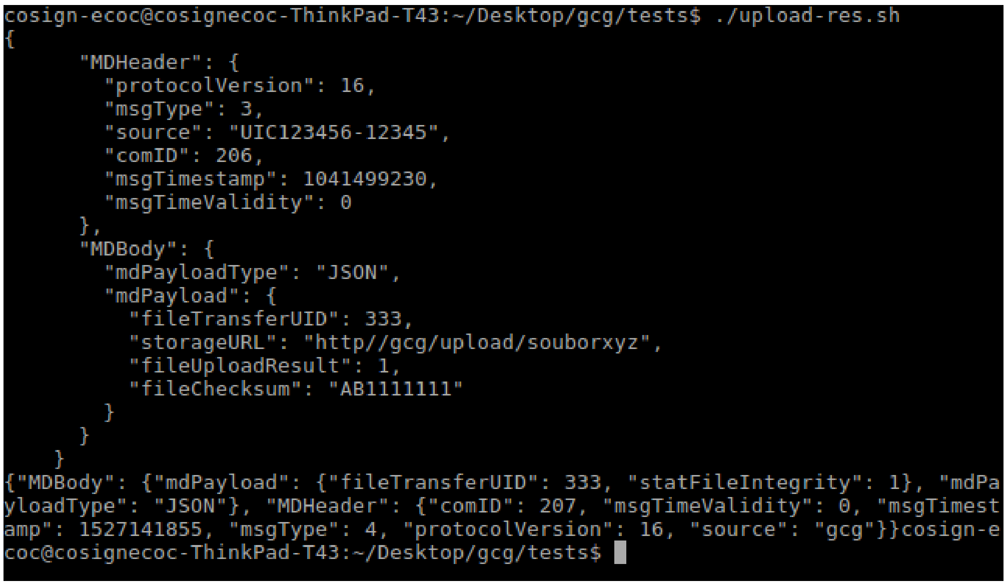
This test checks the file upload service provided by GCGSim. The exchanged data contains information about the file to upload (Figure 69) and if the upload was successful (Figure 70). It uses three scripts:

- upload-req.sh - sendUploadReq()
- upload.sh - uploadFile()
- upload-res.sh - sendUploadResult()



```
cosignecoc-ThinkPad-T43:~/Desktop/gcg/tests$ ./upload-req.sh
{
      "MDHeader": {
        "protocolVersion": 16,
        "msgType": 3,
        "source": "UIC123456-12345",
        "comID": 202,
        "msgTimestamp": 1041499230,
        "msgTimeValidity": 0
      },
      "MDBody": {
        "mdPayloadType": "JSON",
        "mdPayload": {
          "fileTransferUID": 333,
          "filename": "souborxyz",
          "fileType": 2,
          "fileServiceFunction": 0,
          "fileSize": 10
        }
      }
    }
{"MDBody": {"mdPayload": {"fileTransferUID": 333, "storageURL": "http://10.10.10
.6:8085/upload/UIC123456-12345-2018-05-24T08-01-24-430103-id333"}, "mdPayloadTyp
e": "JSON"}, "MDHeader": {"comID": 203, "msgTimeValidity": 0, "msgTimestamp": 15
27141684, "msgType": 4, "protocolVersion": 16, "source": "gcg"}}cosign-ecoc@cosi
gnecoc-ThinkPad-T43:~/Desktop/gcg/tests$ ./upload.sh
```

Figure 69. Test tool result of upload request and upload command.



```
cosign-ecoc@cosignecoc-ThinkPad-T43:~/Desktop/gcg/tests$ ./upload-res.sh
{
      "MDHeader": {
        "protocolVersion": 16,
        "msgType": 3,
        "source": "UIC123456-12345",
        "comID": 206,
        "msgTimestamp": 1041499230,
        "msgTimeValidity": 0
      },
      "MDBody": {
        "mdPayloadType": "JSON",
        "mdPayload": {
          "fileTransferUID": 333,
          "storageURL": "http//gcg/upload/souborxyz",
          "fileUploadResult": 1,
          "fileChecksum": "AB1111111"
        }
      }
    }
{"MDBody": {"mdPayload": {"fileTransferUID": 333, "statFileIntegrity": 1}, "mdPa
yloadType": "JSON"}, "MDHeader": {"comID": 207, "msgTimeValidity": 0, "msgTimest
amp": 1527141855, "msgType": 4, "protocolVersion": 16, "source": "gcg"}}cosign-e
coc@cosignecoc-ThinkPad-T43:~/Desktop/gcg/tests$
```

Figure 70. Test tool result of sent upload result command.

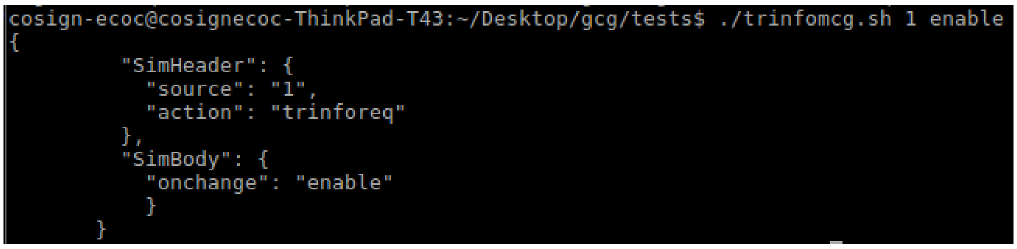
Test 1.7 Reset MCG Service

This test checks the TE API reset_mcg service provided by GCGSim. The result of this test shows the same as the announcement in Test 1.2.

Test 1.8 Train_Inf Request Service

This test checks the TE API send_train_info_req service provided by GCGSim. The results in Figure 71 show the activation of the info service. The following script is used:

- trinfomcg.sh - simTrInfoReq()



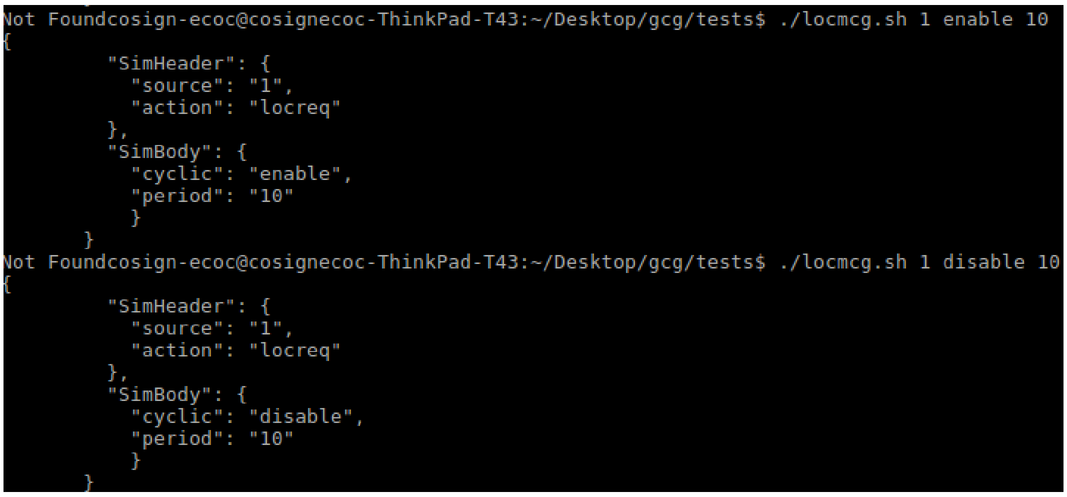
```
cosign-ecoc@cosignecoc-ThinkPad-T43:~/Desktop/gcg/tests$ ./trinfomcg.sh 1 enable
{
         "SimHeader": {
           "source": "1",
           "action": "trinforeq"
         },
         "SimBody": {
           "onchange": "enable"
           }
       }
```

Figure 71. Test tool result of train info MCG command.

Test 1.9 Train Location Request Service

This test checks the TE API send_train_loc_req service provided by GCGSim. Figure 72 shows an example where the service enabled and disabled. It uses the script:

- locmcg.sh - simLocReq()



```
Not Foundcosign-ecoc@cosignecoc-ThinkPad-T43:~/Desktop/gcg/tests$ ./locmcg.sh 1 enable 10
{
         "SimHeader": {
           "source": "1",
           "action": "locreq"
         },
         "SimBody": {
           "cyclic": "enable",
           "period": "10"
           }
       }
Not Foundcosign-ecoc@cosignecoc-ThinkPad-T43:~/Desktop/gcg/tests$ ./locmcg.sh 1 disable 10
{
         "SimHeader": {
           "source": "1",
           "action": "locreq"
         },
         "SimBody": {
           "cyclic": "disable",
           "period": "10"
           }
       }
```

Figure 72. Test tool result of location MCG command.

3.4.3.3 Test 2. Performance testing

These tests try to introduce variability in network KPIs (Key Performance Indicators) to verify the functionality of the testing setup/ tested equipment. In this work, we have tested behaviour under variations of Packet Loss, Latency and Jitter.

Description: This test verifies the loss, latency and jitter of packets when the system is commanded to do so.

Precondition:

- The Ethernet Packet Generator (E6) is loaded with SW1.
- The LTE emulator and OPNET simulator (E1-E4) are activated.
- The GCG host (E8) has been configured and interacted with UE (E4).
- Packet Sniffers (E7) is activated to allow the logging of the packets.

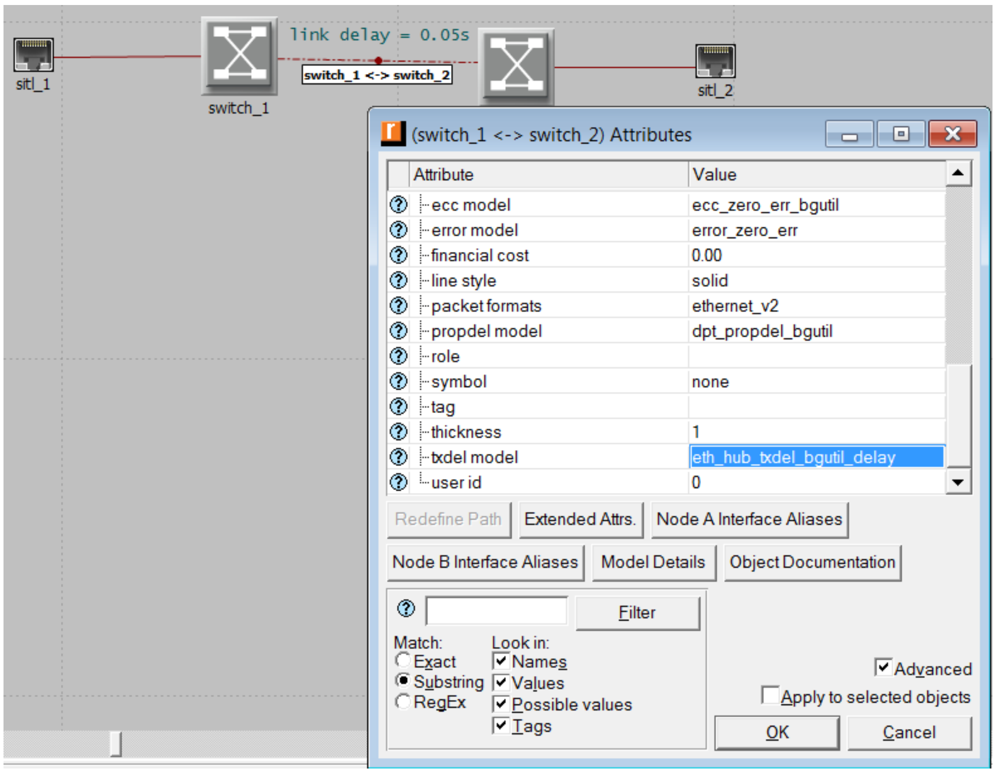
Step	Step Description	Expected Result
1	Inserting delay and loss at UE (by changing wireless condition)	Results have been reported in previous deliverable & paper.
2	Inserting delay at OPNET	See Test 2.1
3	Inserting delay at OpenAirInterface	See Test 2.2

Test 2.1 Inserting delay in OPNET

Configuration:

The backhaul network in the OPNET simulation environment (shown in Figure 73) contains two SITL gateways for conversion between the real traffic packet format and simulated packet format. Furthermore there are two Ethernet switches for routing the traffic to the destination. The transmission delay is inserted into the link between two switches in order to mimic the real world data transmission.

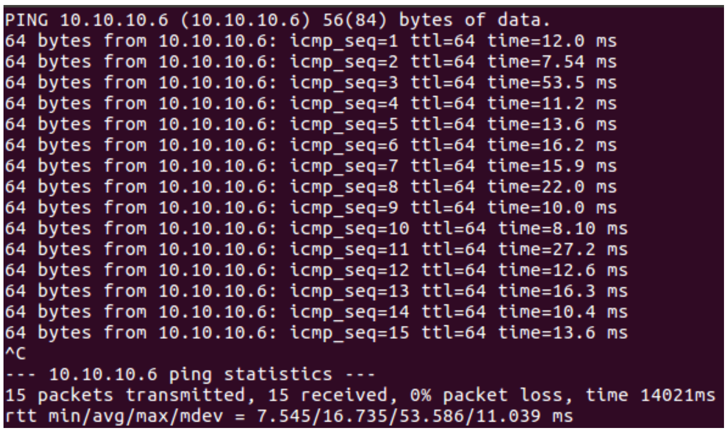
In OPNET, the transmission delay model, which is programmed in the pipeline stage file (C code), is modified with a value of the transmission delay.



Attribute	Value
ecc model	ecc_zero_err_bgutil
error model	error_zero_err
financial cost	0.00
line style	solid
packet formats	ethernet_v2
propdel model	dpt_propdel_bgutil
role	
symbol	none
tag	
thickness	1
txdel model	eth_hub_txdel_bgutil_delay
user id	0

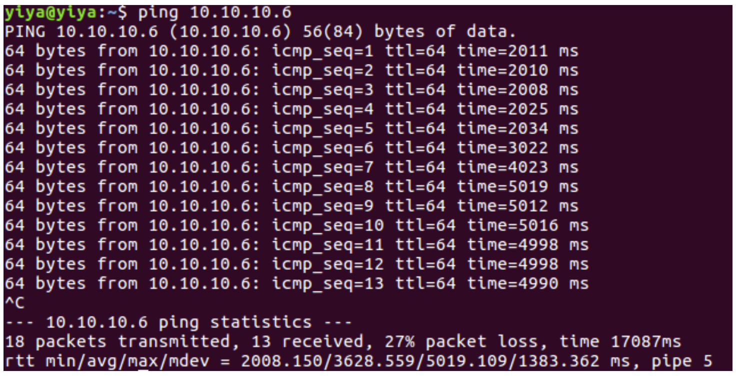
Figure 73. Backhaul network in OPNET and link delay settings.

Test results:



```
PING 10.10.10.6 (10.10.10.6) 56(84) bytes of data.
64 bytes from 10.10.10.6: icmp_seq=1 ttl=64 time=12.0 ms
64 bytes from 10.10.10.6: icmp_seq=2 ttl=64 time=7.54 ms
64 bytes from 10.10.10.6: icmp_seq=3 ttl=64 time=53.5 ms
64 bytes from 10.10.10.6: icmp_seq=4 ttl=64 time=11.2 ms
64 bytes from 10.10.10.6: icmp_seq=5 ttl=64 time=13.6 ms
64 bytes from 10.10.10.6: icmp_seq=6 ttl=64 time=16.2 ms
64 bytes from 10.10.10.6: icmp_seq=7 ttl=64 time=15.9 ms
64 bytes from 10.10.10.6: icmp_seq=8 ttl=64 time=22.0 ms
64 bytes from 10.10.10.6: icmp_seq=9 ttl=64 time=10.0 ms
64 bytes from 10.10.10.6: icmp_seq=10 ttl=64 time=8.10 ms
64 bytes from 10.10.10.6: icmp_seq=11 ttl=64 time=27.2 ms
64 bytes from 10.10.10.6: icmp_seq=12 ttl=64 time=12.6 ms
64 bytes from 10.10.10.6: icmp_seq=13 ttl=64 time=16.3 ms
64 bytes from 10.10.10.6: icmp_seq=14 ttl=64 time=10.4 ms
64 bytes from 10.10.10.6: icmp_seq=15 ttl=64 time=13.6 ms
^C
--- 10.10.10.6 ping statistics ---
15 packets transmitted, 15 received, 0% packet loss, time 14021ms
rtt min/avg/max/mdev = 7.545/16.735/53.586/11.039 ms
```

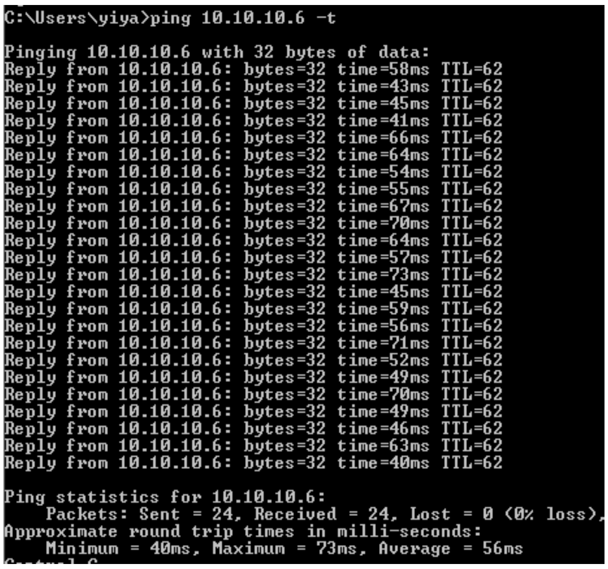
Figure 74. Ping from EPC without OPNET delay (average delay = 16.735ms).



```
yiya@yiya:~$ ping 10.10.10.6
PING 10.10.10.6 (10.10.10.6) 56(84) bytes of data.
64 bytes from 10.10.10.6: icmp_seq=1 ttl=64 time=2011 ms
64 bytes from 10.10.10.6: icmp_seq=2 ttl=64 time=2010 ms
64 bytes from 10.10.10.6: icmp_seq=3 ttl=64 time=2008 ms
64 bytes from 10.10.10.6: icmp_seq=4 ttl=64 time=2025 ms
64 bytes from 10.10.10.6: icmp_seq=5 ttl=64 time=2034 ms
64 bytes from 10.10.10.6: icmp_seq=6 ttl=64 time=3022 ms
64 bytes from 10.10.10.6: icmp_seq=7 ttl=64 time=4023 ms
64 bytes from 10.10.10.6: icmp_seq=8 ttl=64 time=5019 ms
64 bytes from 10.10.10.6: icmp_seq=9 ttl=64 time=5012 ms
64 bytes from 10.10.10.6: icmp_seq=10 ttl=64 time=5016 ms
64 bytes from 10.10.10.6: icmp_seq=11 ttl=64 time=4998 ms
64 bytes from 10.10.10.6: icmp_seq=12 ttl=64 time=4998 ms
64 bytes from 10.10.10.6: icmp_seq=13 ttl=64 time=4990 ms
^C
--- 10.10.10.6 ping statistics ---
18 packets transmitted, 13 received, 27% packet loss, time 17087ms
rtt min/avg/max/mdev = 2008.150/3628.559/5019.109/1383.362 ms, pipe 5
```

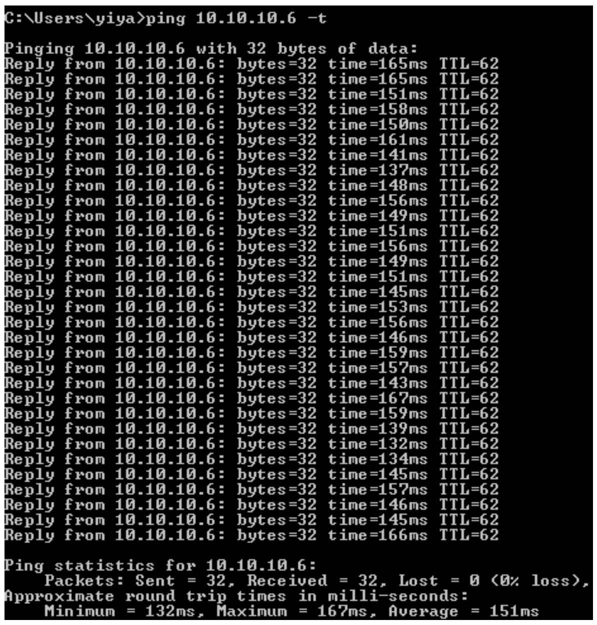
Figure 75. Ping from EPC with tx_delay=1s for all packets (both directions, average delay = 3628ms).

The tx_delay equals to 1s is chosen only for showing an obvious result. The communication between the LTE UE and the server does not work under 1s add-on delay. Therefore, under the test of an end-to-end communication, the tx_delay in OPNET is selected as 0.05s. Without any OPNET delay, the average delay is 16.375ms (Figure 74). Using the OPNET delay, the average delay increases to 3628ms.



```
C:\Users\yiya>ping 10.10.10.6 -t

Pinging 10.10.10.6 with 32 bytes of data:
Reply from 10.10.10.6: bytes=32 time=58ms TTL=62
Reply from 10.10.10.6: bytes=32 time=43ms TTL=62
Reply from 10.10.10.6: bytes=32 time=45ms TTL=62
Reply from 10.10.10.6: bytes=32 time=41ms TTL=62
Reply from 10.10.10.6: bytes=32 time=66ms TTL=62
Reply from 10.10.10.6: bytes=32 time=64ms TTL=62
Reply from 10.10.10.6: bytes=32 time=54ms TTL=62
Reply from 10.10.10.6: bytes=32 time=55ms TTL=62
Reply from 10.10.10.6: bytes=32 time=67ms TTL=62
Reply from 10.10.10.6: bytes=32 time=70ms TTL=62
Reply from 10.10.10.6: bytes=32 time=64ms TTL=62
Reply from 10.10.10.6: bytes=32 time=57ms TTL=62
Reply from 10.10.10.6: bytes=32 time=73ms TTL=62
Reply from 10.10.10.6: bytes=32 time=45ms TTL=62
Reply from 10.10.10.6: bytes=32 time=59ms TTL=62
Reply from 10.10.10.6: bytes=32 time=56ms TTL=62
Reply from 10.10.10.6: bytes=32 time=71ms TTL=62
Reply from 10.10.10.6: bytes=32 time=52ms TTL=62
Reply from 10.10.10.6: bytes=32 time=49ms TTL=62
Reply from 10.10.10.6: bytes=32 time=70ms TTL=62
Reply from 10.10.10.6: bytes=32 time=49ms TTL=62
Reply from 10.10.10.6: bytes=32 time=46ms TTL=62
Reply from 10.10.10.6: bytes=32 time=63ms TTL=62
Reply from 10.10.10.6: bytes=32 time=40ms TTL=62

Ping statistics for 10.10.10.6:
    Packets: Sent = 24, Received = 24, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 40ms, Maximum = 73ms, Average = 56ms
```

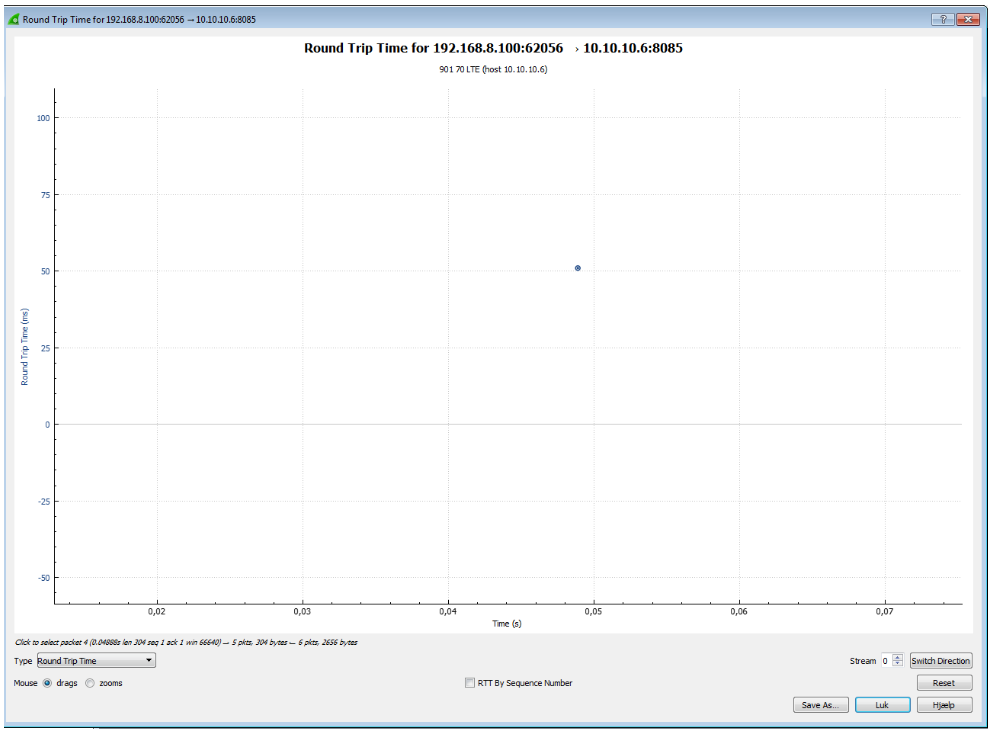
Figure 76. Ping from UE to server without OPNET delay (average delay = 56ms).



```
C:\Users\yiya>ping 10.10.10.6 -t

Pinging 10.10.10.6 with 32 bytes of data:
Reply from 10.10.10.6: bytes=32 time=165ms TTL=62
Reply from 10.10.10.6: bytes=32 time=165ms TTL=62
Reply from 10.10.10.6: bytes=32 time=151ms TTL=62
Reply from 10.10.10.6: bytes=32 time=158ms TTL=62
Reply from 10.10.10.6: bytes=32 time=150ms TTL=62
Reply from 10.10.10.6: bytes=32 time=161ms TTL=62
Reply from 10.10.10.6: bytes=32 time=141ms TTL=62
Reply from 10.10.10.6: bytes=32 time=137ms TTL=62
Reply from 10.10.10.6: bytes=32 time=148ms TTL=62
Reply from 10.10.10.6: bytes=32 time=156ms TTL=62
Reply from 10.10.10.6: bytes=32 time=149ms TTL=62
Reply from 10.10.10.6: bytes=32 time=151ms TTL=62
Reply from 10.10.10.6: bytes=32 time=156ms TTL=62
Reply from 10.10.10.6: bytes=32 time=149ms TTL=62
Reply from 10.10.10.6: bytes=32 time=151ms TTL=62
Reply from 10.10.10.6: bytes=32 time=145ms TTL=62
Reply from 10.10.10.6: bytes=32 time=153ms TTL=62
Reply from 10.10.10.6: bytes=32 time=156ms TTL=62
Reply from 10.10.10.6: bytes=32 time=146ms TTL=62
Reply from 10.10.10.6: bytes=32 time=159ms TTL=62
Reply from 10.10.10.6: bytes=32 time=157ms TTL=62
Reply from 10.10.10.6: bytes=32 time=143ms TTL=62
Reply from 10.10.10.6: bytes=32 time=167ms TTL=62
Reply from 10.10.10.6: bytes=32 time=159ms TTL=62
Reply from 10.10.10.6: bytes=32 time=139ms TTL=62
Reply from 10.10.10.6: bytes=32 time=132ms TTL=62
Reply from 10.10.10.6: bytes=32 time=134ms TTL=62
Reply from 10.10.10.6: bytes=32 time=145ms TTL=62
Reply from 10.10.10.6: bytes=32 time=157ms TTL=62
Reply from 10.10.10.6: bytes=32 time=146ms TTL=62
Reply from 10.10.10.6: bytes=32 time=145ms TTL=62
Reply from 10.10.10.6: bytes=32 time=166ms TTL=62

Ping statistics for 10.10.10.6:
    Packets: Sent = 32, Received = 32, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 132ms, Maximum = 167ms, Average = 151ms
```

Figure 77. Ping from UE to server with OPNET delay of 0.05s (average delay = 151ms).

When the GCG Simulator is running, the Round-Trip-Times (RTT) are collected with and without the OPNET delay of 0.05s using the Wireshark program (installed at LTE UE). Without the delay the average communication delay accounts to 56ms (Figure 76). If the OPNET delay is introduced, the average delay increases to 151ms (Figure 77). Figure 78 (no delay) and Figure 79 (with delay) show related statistics from Wireshark. The calculation of the RTT in the *ping*-command can be different from the packet capture in Wireshark.


Round Trip Time for 192.168.8.100:62056 → 10.10.10.6:8085

Round Trip Time for 192.168.8.100:62056 → 10.10.10.6:8085

901 70 LTE (host 10.10.10.6)

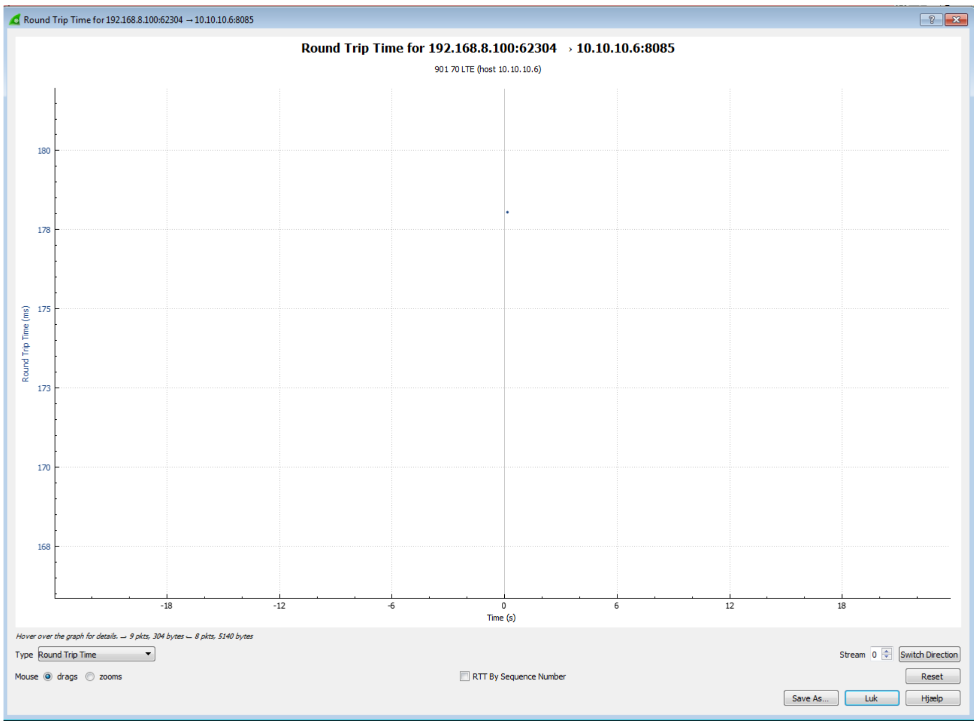
Click to select packet 4 (0.04888s len 304 seq 1 ack 1 win 66640) → 5 pkts, 304 bytes ← 6 pkts, 2656 bytes

Type: Round Trip Time

Mouse [✓ drags] zooms

Stream 0

Figure 78. RTT Result with RTT = 58 ms when no delay is introduced.


Round Trip Time for 192.168.8.100:62304 → 10.10.10.6:8085

Round Trip Time for 192.168.8.100:62304 → 10.10.10.6:8085

901 70 LTE (host 10.10.10.6)

Hover over the graph for details. → 9 pkts, 304 bytes ← 8 pkts, 5140 bytes

Type: Round Trip Time

Mouse [✓ drags] zooms

Stream 0

Figure 79. RTT = 178 ms when the OPNET delay is enabled.

Test 2.2 Inserting a delay in the LTE OpenAirInterface emulator

Configuration:

The LTE emulator includes an EPC and an eNodeB. Both of them are implemented on a Linux PC based on the OpenAirInterface open source program.

A delay can be added in on two communication links: (I) the link between the LTE emulator (EPC) and the backhaul network (OPNET) and (II) the link between the LTE eNodeB and the LTE EPC.

Traffic control (tc) is a very useful Linux utility that gives the ability to configure the kernel packet scheduler. The utility is able to simulate packet delay and loss for TCP or UDP applications.

General commands:

- tc qdisc add dev eth0 root netem delay 200ms

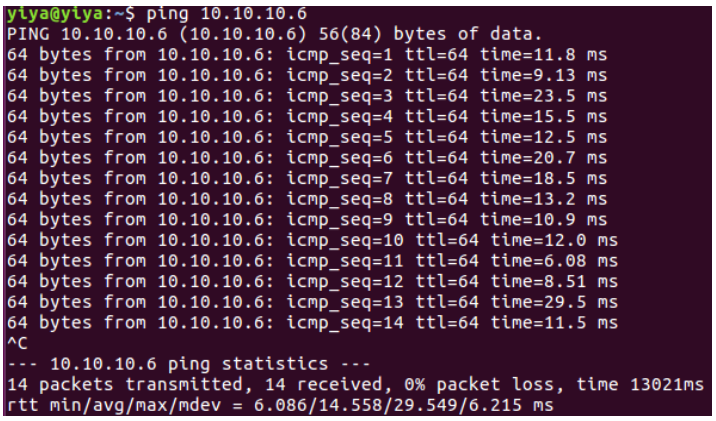
This command shows how to add constant delay (200ms) to an interface (eth0).

- tc qdisc add dev eth0 root netem loss 10%

This command introduces a packet loss of 10% to an interface (eth0).

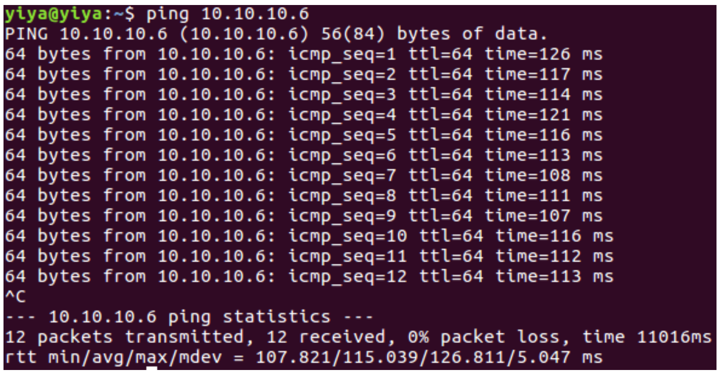
Test results

Sub-test1 adds a delay of 100ms in the link between the EPC and OPNET. Without the additional delay, the communication takes 14.558ms in average as shown in Figure 80. However, the ping's RTT is increases to 115.039ms if a delay of 100ms is introduced. This effect can be seen in Figure 81.



```
yiya@yiya:~$ ping 10.10.10.6
PING 10.10.10.6 (10.10.10.6) 56(84) bytes of data.
64 bytes from 10.10.10.6: icmp_seq=1 ttl=64 time=11.8 ms
64 bytes from 10.10.10.6: icmp_seq=2 ttl=64 time=9.13 ms
64 bytes from 10.10.10.6: icmp_seq=3 ttl=64 time=23.5 ms
64 bytes from 10.10.10.6: icmp_seq=4 ttl=64 time=15.5 ms
64 bytes from 10.10.10.6: icmp_seq=5 ttl=64 time=12.5 ms
64 bytes from 10.10.10.6: icmp_seq=6 ttl=64 time=20.7 ms
64 bytes from 10.10.10.6: icmp_seq=7 ttl=64 time=18.5 ms
64 bytes from 10.10.10.6: icmp_seq=8 ttl=64 time=13.2 ms
64 bytes from 10.10.10.6: icmp_seq=9 ttl=64 time=10.9 ms
64 bytes from 10.10.10.6: icmp_seq=10 ttl=64 time=12.0 ms
64 bytes from 10.10.10.6: icmp_seq=11 ttl=64 time=6.08 ms
64 bytes from 10.10.10.6: icmp_seq=12 ttl=64 time=8.51 ms
64 bytes from 10.10.10.6: icmp_seq=13 ttl=64 time=29.5 ms
64 bytes from 10.10.10.6: icmp_seq=14 ttl=64 time=11.5 ms
^C
--- 10.10.10.6 ping statistics ---
14 packets transmitted, 14 received, 0% packet loss, time 13021ms
rtt min/avg/max/mdev = 6.086/14.558/29.549/6.215 ms
```

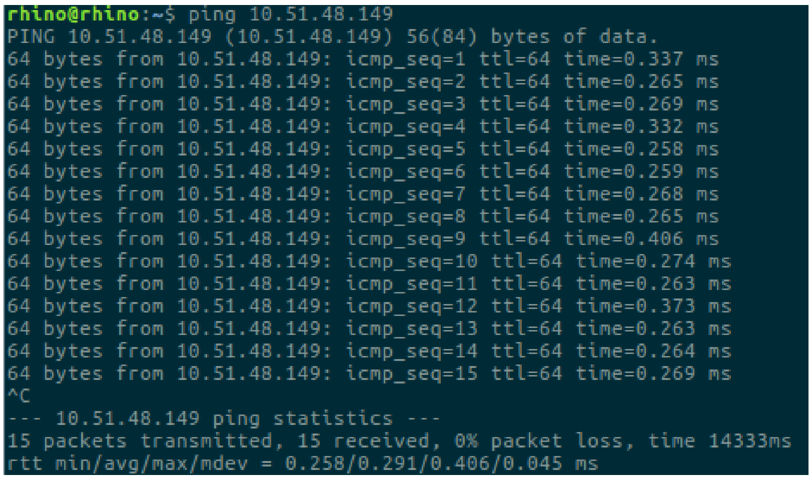
Figure 80. Ping from EPC without delay (average delay = 14.558ms).



```
yiya@yiya:~$ ping 10.10.10.6
PING 10.10.10.6 (10.10.10.6) 56(84) bytes of data.
64 bytes from 10.10.10.6: icmp_seq=1 ttl=64 time=126 ms
64 bytes from 10.10.10.6: icmp_seq=2 ttl=64 time=117 ms
64 bytes from 10.10.10.6: icmp_seq=3 ttl=64 time=114 ms
64 bytes from 10.10.10.6: icmp_seq=4 ttl=64 time=121 ms
64 bytes from 10.10.10.6: icmp_seq=5 ttl=64 time=116 ms
64 bytes from 10.10.10.6: icmp_seq=6 ttl=64 time=113 ms
64 bytes from 10.10.10.6: icmp_seq=7 ttl=64 time=108 ms
64 bytes from 10.10.10.6: icmp_seq=8 ttl=64 time=111 ms
64 bytes from 10.10.10.6: icmp_seq=9 ttl=64 time=107 ms
64 bytes from 10.10.10.6: icmp_seq=10 ttl=64 time=116 ms
64 bytes from 10.10.10.6: icmp_seq=11 ttl=64 time=112 ms
64 bytes from 10.10.10.6: icmp_seq=12 ttl=64 time=113 ms
^C
--- 10.10.10.6 ping statistics ---
12 packets transmitted, 12 received, 0% packet loss, time 11016ms
rtt min/avg/max/mdev = 107.821/115.039/126.811/5.047 ms
```

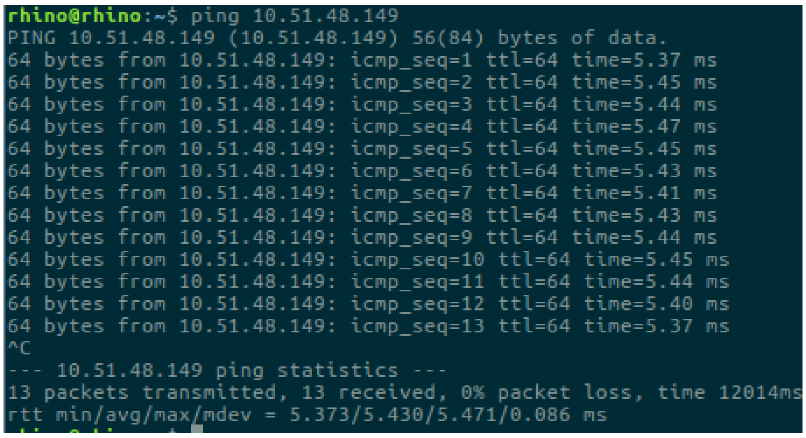
Figure 81. Ping from EPC with 100ms delay (average delay = 115.039ms).

Sub-test2 adds a delay of 5ms in the link between eNodeB and the EPC. Figure 82 shows a ping without any additional delay. The average delay accounts for 0.291ms. If an additional delay of 5ms is introduced as presented in Figure 83, the average delay increases to 5.430ms.



```
rhino@rhino:~$ ping 10.51.48.149
PING 10.51.48.149 (10.51.48.149) 56(84) bytes of data.
64 bytes from 10.51.48.149: icmp_seq=1 ttl=64 time=0.337 ms
64 bytes from 10.51.48.149: icmp_seq=2 ttl=64 time=0.265 ms
64 bytes from 10.51.48.149: icmp_seq=3 ttl=64 time=0.269 ms
64 bytes from 10.51.48.149: icmp_seq=4 ttl=64 time=0.332 ms
64 bytes from 10.51.48.149: icmp_seq=5 ttl=64 time=0.258 ms
64 bytes from 10.51.48.149: icmp_seq=6 ttl=64 time=0.259 ms
64 bytes from 10.51.48.149: icmp_seq=7 ttl=64 time=0.268 ms
64 bytes from 10.51.48.149: icmp_seq=8 ttl=64 time=0.265 ms
64 bytes from 10.51.48.149: icmp_seq=9 ttl=64 time=0.406 ms
64 bytes from 10.51.48.149: icmp_seq=10 ttl=64 time=0.274 ms
64 bytes from 10.51.48.149: icmp_seq=11 ttl=64 time=0.263 ms
64 bytes from 10.51.48.149: icmp_seq=12 ttl=64 time=0.373 ms
64 bytes from 10.51.48.149: icmp_seq=13 ttl=64 time=0.263 ms
64 bytes from 10.51.48.149: icmp_seq=14 ttl=64 time=0.264 ms
64 bytes from 10.51.48.149: icmp_seq=15 ttl=64 time=0.269 ms
^C
--- 10.51.48.149 ping statistics ---
15 packets transmitted, 15 received, 0% packet loss, time 14333ms
rtt min/avg/max/mdev = 0.258/0.291/0.406/0.045 ms
```

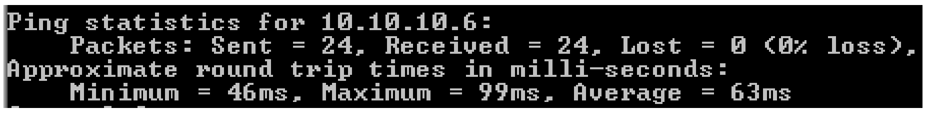
Figure 82. Ping from eNodeB without delay (average delay = 0.291ms).



```
rhino@rhino:~$ ping 10.51.48.149
PING 10.51.48.149 (10.51.48.149) 56(84) bytes of data.
64 bytes from 10.51.48.149: icmp_seq=1 ttl=64 time=5.37 ms
64 bytes from 10.51.48.149: icmp_seq=2 ttl=64 time=5.45 ms
64 bytes from 10.51.48.149: icmp_seq=3 ttl=64 time=5.44 ms
64 bytes from 10.51.48.149: icmp_seq=4 ttl=64 time=5.47 ms
64 bytes from 10.51.48.149: icmp_seq=5 ttl=64 time=5.45 ms
64 bytes from 10.51.48.149: icmp_seq=6 ttl=64 time=5.43 ms
64 bytes from 10.51.48.149: icmp_seq=7 ttl=64 time=5.41 ms
64 bytes from 10.51.48.149: icmp_seq=8 ttl=64 time=5.43 ms
64 bytes from 10.51.48.149: icmp_seq=9 ttl=64 time=5.44 ms
64 bytes from 10.51.48.149: icmp_seq=10 ttl=64 time=5.45 ms
64 bytes from 10.51.48.149: icmp_seq=11 ttl=64 time=5.44 ms
64 bytes from 10.51.48.149: icmp_seq=12 ttl=64 time=5.40 ms
64 bytes from 10.51.48.149: icmp_seq=13 ttl=64 time=5.37 ms
^C
--- 10.51.48.149 ping statistics ---
13 packets transmitted, 13 received, 0% packet loss, time 12014ms
rtt min/avg/max/mdev = 5.373/5.430/5.471/0.086 ms
```

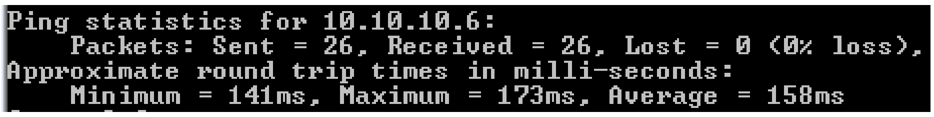
Figure 83. Ping from eNodeB with 5ms delay (average delay = 5.430ms).

Sub-test3 adds a delay of 10ms in the link between eNodeB and the EPC and a delay of 100ms between the EPC and OPNET. Without additional delay, the average RTT for a ping between UE and Server results in 63ms while the TCP communication between UE and GCG host requires 158ms. The RTTs increase to 158ms and 163ms if the delay is added. The results are presented in Figure 84 to Figure 87.



```
Ping statistics for 10.10.10.6:
    Packets: Sent = 24, Received = 24, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 46ms, Maximum = 99ms, Average = 63ms
```

Figure 84. Ping from UE to server without delay (average delay = 63ms).



```
Ping statistics for 10.10.10.6:
    Packets: Sent = 26, Received = 26, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 141ms, Maximum = 173ms, Average = 158ms
```

Figure 85. Ping from UE to server with in total 110ms delay (average delay = 158ms).

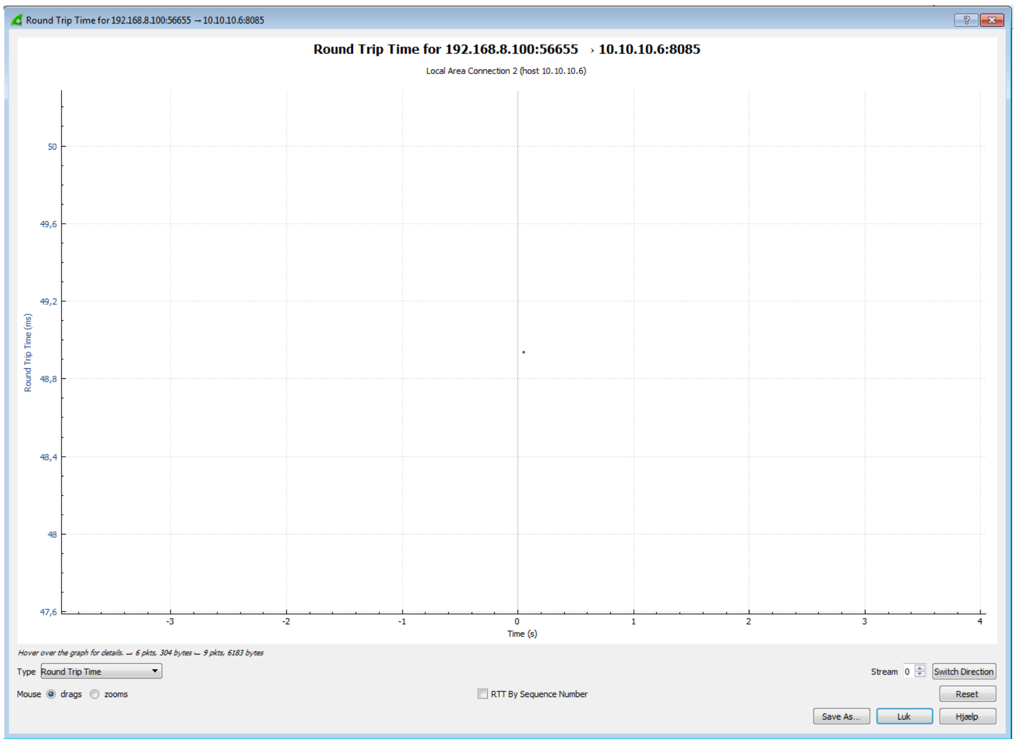


Figure 86. TCP RTT between UE and GCG host (48ms).

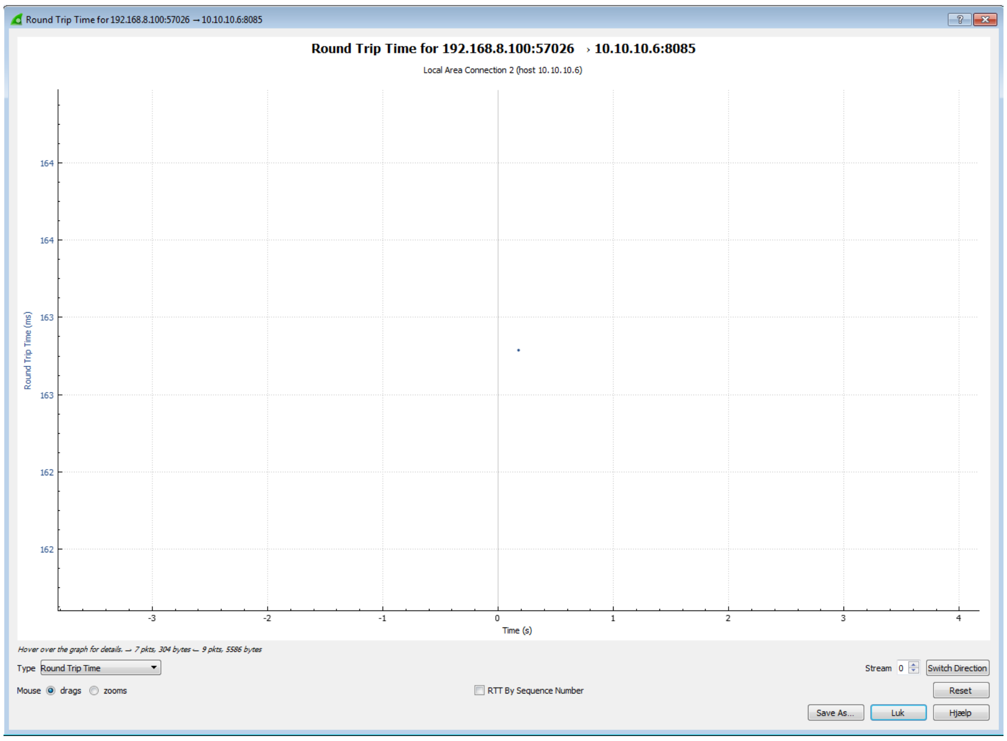
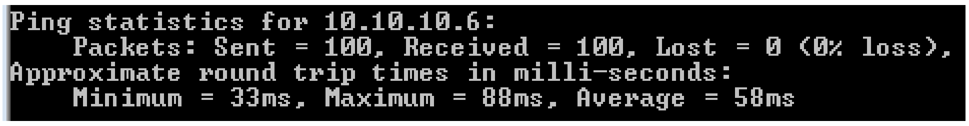


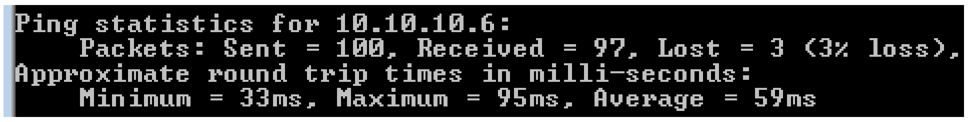
Figure 87. TCP RTT between UE and GCG host with 110ms delay (163ms).

Sub-test4 adds a loss of 5% in the link between the EPC and OPNET. While there is no packet loss configured in Figure 88, Figure 89 shows a configured loss of 5%. In fact, without loss all packets are received whereas only 3% of the packets are lost if a loss is configured. Packet loss is a probability. Due to the small number of pings, the number of lost packets differs from the defined value.



```
Ping statistics for 10.10.10.6:
    Packets: Sent = 100, Received = 100, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 33ms, Maximum = 88ms, Average = 58ms
```

Figure 88. Ping from UE to server without loss.



```
Ping statistics for 10.10.10.6:
    Packets: Sent = 100, Received = 97, Lost = 3 (3% loss),
Approximate round trip times in milli-seconds:
    Minimum = 33ms, Maximum = 95ms, Average = 59ms
```

Figure 89. Ping from UE to server with 5% loss configured. 3% of packets are lost in fact.

3.4.3.4 Test 3: Redundancy test in backhaul network

The purpose of this test is to examine the network simulation ability in the Riverbed Modeler. The performance measurement focuses on network availability and redundancy.

Configuration:

A network consisting of 8 switches is simulated in the Riverbed Modeler (Figure 91). Two SITL gateways are connected as the real-sim/real-sim interfaces to the LTE OAI EPC and the GCG service host. The overall test scenario is shown in Figure 90.

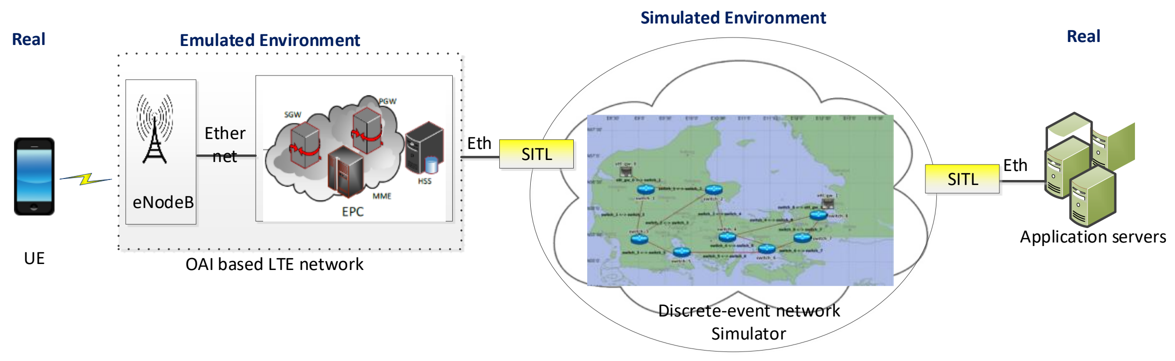


Figure 90. Overall test scenario with simulated backbone network.

General operations:

The traffic is routed between switch_1 and switch 8 via the link switch_1 <-> switch_2 <-> switch_4 <-> switch_8. The redundancy property is tested by setting the link between switch_2 and swtich_4. If a failure occurs, the traffic is routed via a new path as switch_1 <-> switch_2 <-> switch_3 <-> switch_5 <-> switch_4 <-> switch_8.

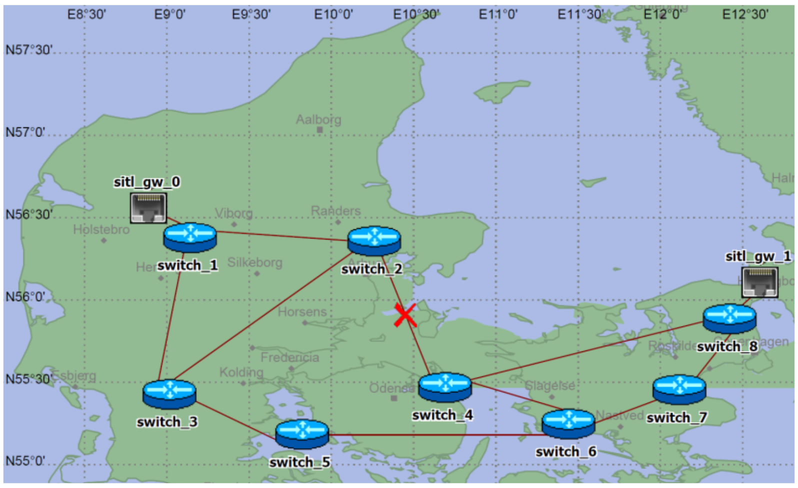
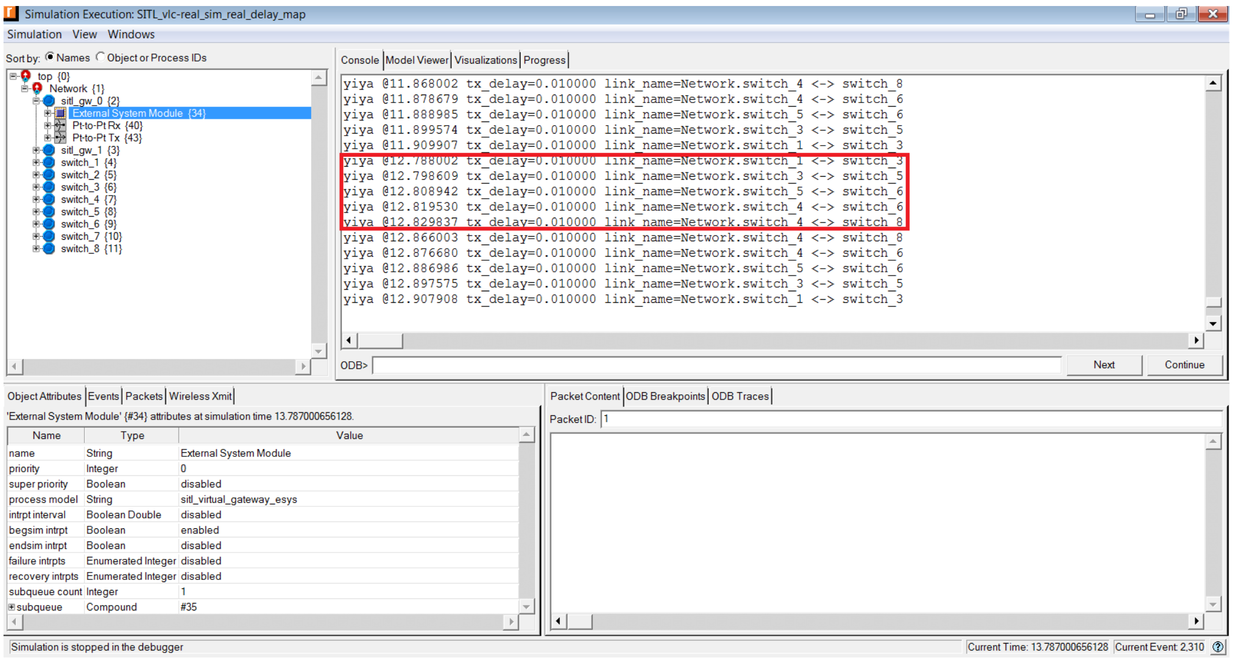


Figure 91. Simulated Backhaul Network – with failure at link switch_2 <-> switch_4.

Test results:

In order to test the network redundancy and traffic re-routing, the transmission delay is examined as a performance metric. In order to highlight the differences in results, an extra link transmission delay is added on each link as *tx_link_delay = 0.01s.* The packet forwarding is traced using the debugging interface in Riverbed Modeler and the redundancy is shown in the traffic routing trace (Figure 92).


Simulation Execution: SITL_vlc-real_sim_real_delay_map

Sort by: (●) Names ( ) Object or Process IDs

```
top {0}
  Network {1}
    sitl_gw_0 {2}
      External System Module {34}
      Pt-to-Pt Rx {40}
      Pt-to-Pt Tx {43}
    sitl_gw_1 {3}
    switch_1 {4}
    switch_2 {5}
    switch_3 {6}
    switch_4 {7}
    switch_5 {8}
    switch_6 {9}
    switch_7 {10}
    switch_8 {11}
```

```
yiya @11.868002 tx_delay=0.010000 link_name=Network.switch_4 <-> switch_8
yiya @11.878679 tx_delay=0.010000 link_name=Network.switch_4 <-> switch_6
yiya @11.888985 tx_delay=0.010000 link_name=Network.switch_5 <-> switch_6
yiya @11.899574 tx_delay=0.010000 link_name=Network.switch_3 <-> switch_5
yiya @11.909907 tx_delay=0.010000 link_name=Network.switch_1 <-> switch_3
yiya @12.788002 tx_delay=0.010000 link_name=Network.switch_1 <-> switch_3
yiya @12.798609 tx_delay=0.010000 link_name=Network.switch_3 <-> switch_5
yiya @12.808942 tx_delay=0.010000 link_name=Network.switch_5 <-> switch_6
yiya @12.819530 tx_delay=0.010000 link_name=Network.switch_4 <-> switch_6
yiya @12.829837 tx_delay=0.010000 link_name=Network.switch_4 <-> switch_8
yiya @12.866003 tx_delay=0.010000 link_name=Network.switch_4 <-> switch_8
yiya @12.876680 tx_delay=0.010000 link_name=Network.switch_4 <-> switch_6
yiya @12.886986 tx_delay=0.010000 link_name=Network.switch_5 <-> switch_6
yiya @12.897575 tx_delay=0.010000 link_name=Network.switch_3 <-> switch_5
yiya @12.907908 tx_delay=0.010000 link_name=Network.switch_1 <-> switch_3
```

'External System Module' {#34} attributes at simulation time 13.787000656128.

Name	Type	Value
name	String	External System Module
priority	Integer	0
super priority	Boolean	disabled
process model	String	sitl_virtual_gateway_esys
intrpt interval	Boolean Double	disabled
begsim intrpt	Boolean	enabled
endsim intrpt	Boolean	disabled
failure intrpts	Enumerated Integer	disabled
recovery intrpts	Enumerated Integer	disabled
subqueue count	Integer	1
⊞ subqueue	Compound	#35

Packet ID: 1

Simulation is stopped in the debugger

Current Time: 13.787000656128 Current Event: 2,310

Figure 92. Simulated Backhaul Network – with failure at link switch_2 <-> switch_4.

Figure 93 and Figure 94 present the resulting RTTs for a ping without link failure (Figure 93) and if a failure is introduced (Figure 94). The resulting delay is increased from 115ms to 149ms as a proof that the packet is re-routed via a longer path.

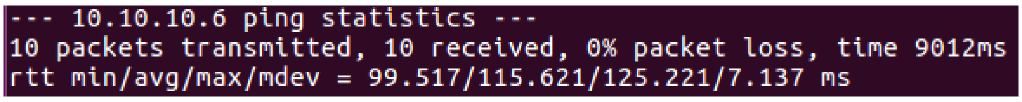

```
--- 10.10.10.6 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9012ms
rtt min/avg/max/mdev = 99.517/115.621/125.221/7.137 ms
```

Figure 93. Ping from EPC without failure (average delay = 115.6ms).

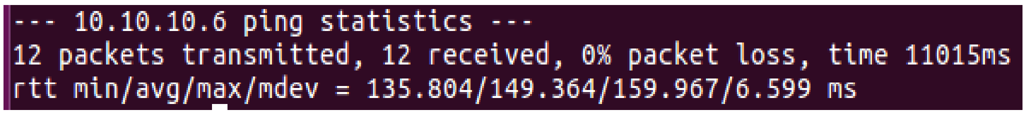

```
--- 10.10.10.6 ping statistics ---
12 packets transmitted, 12 received, 0% packet loss, time 11015ms
rtt min/avg/max/mdev = 135.804/149.364/159.967/6.599 ms
```

Figure 94. Ping from EPC with failure (average delay = 135.8ms).

Chapter 4 Conclusions and further recommendations

4.1 Conclusions and recommendations for the distributed simulation framework (IKL/SIE/IAV)

The evaluation results presented in Chapter 2 show the proper functionality of the distributed simulation framework (communication emulator) developed during the Safe4RAIL project. Besides the presentation of the results, also various conclusions and recommendations are further explained in detail. In the following, Section 4.1.1 summarizes the conclusions and recommendations already given. They are refined to requirement recommendations for drive-by-data and the embedded platform (Section 4.1.2).

4.1.1 Conclusions and recommendations

This section summarizes the conclusions and recommendations already given in Chapter 2. For each recommendation, references are given to the related section which led to the definition.

- The simulation durations if the Internet is used are quite large due to the delays introduced by the network and the VPN used to secure the communication (Section 2.2.2). A hierarchical RTI should be used if multiple simulation bridges are located in one LAN and connected to other simulation bridges via the Internet.
- To reduce the communication delays introduced by the Internet, a future project should evaluate the usage of Quality of Service (QoS) settings between the simulation hosts. A proper QoS service class in the communication might reduce the delays.
- Section 2.2.3 shows large delays introduced by the packet capturing using PCAP. The parameters used to open a packet capture shall be improved. Furthermore, different packet capturing mechanisms shall be evaluated.
- To enable the parallel execution of multiple functionalities in the simulation bridges the application uses threads in which the modules are executed. The evaluation of PCAP (Section 2.2.3) and the state-estimation (Section 2.2.5) both show delays and missed injection events moreover in case of the estimation. Hence, the host used to run a simulation bridge should include a CPU with at least six cores or hardware threads to limit context changes due to the Operating System. Furthermore, each simulation bridge should run on a distinct system.
- The state-estimation (Section 2.2.5) enables the injection of packets with real-time requirements. To ensure a timely correct injection, the simulation bridge should be executed on a host providing a real-time operating system.
- If the state-estimation is used to ensure the reception of messages in time on a real device, the device and the PC hosting the simulation bridge should be synchronized in time to ensure a timely correct injection (Section 2.2.5).
- Tests with real train hardware and enabled time management (Section 2.3.2) have shown that the mechanism used to synchronize the simulation bridge and the con-

nected device is inappropriate. In future projects a mechanism should be implemented which provides sufficient information for the simulation bridge to advance in time.

- For the use-case in which several federates (EDs) are simulated in the same Simulation host (SIL), the current design of the CE forces the use of one SB per federate. As it can be seen in Figure 95, all the traffic sent by each node must be captured by its SB and send to the Central PC (RTI) which sends these messages to the SB of the Network Simulator afterwards. The Network Simulator simulates the network elements of the TCMS.

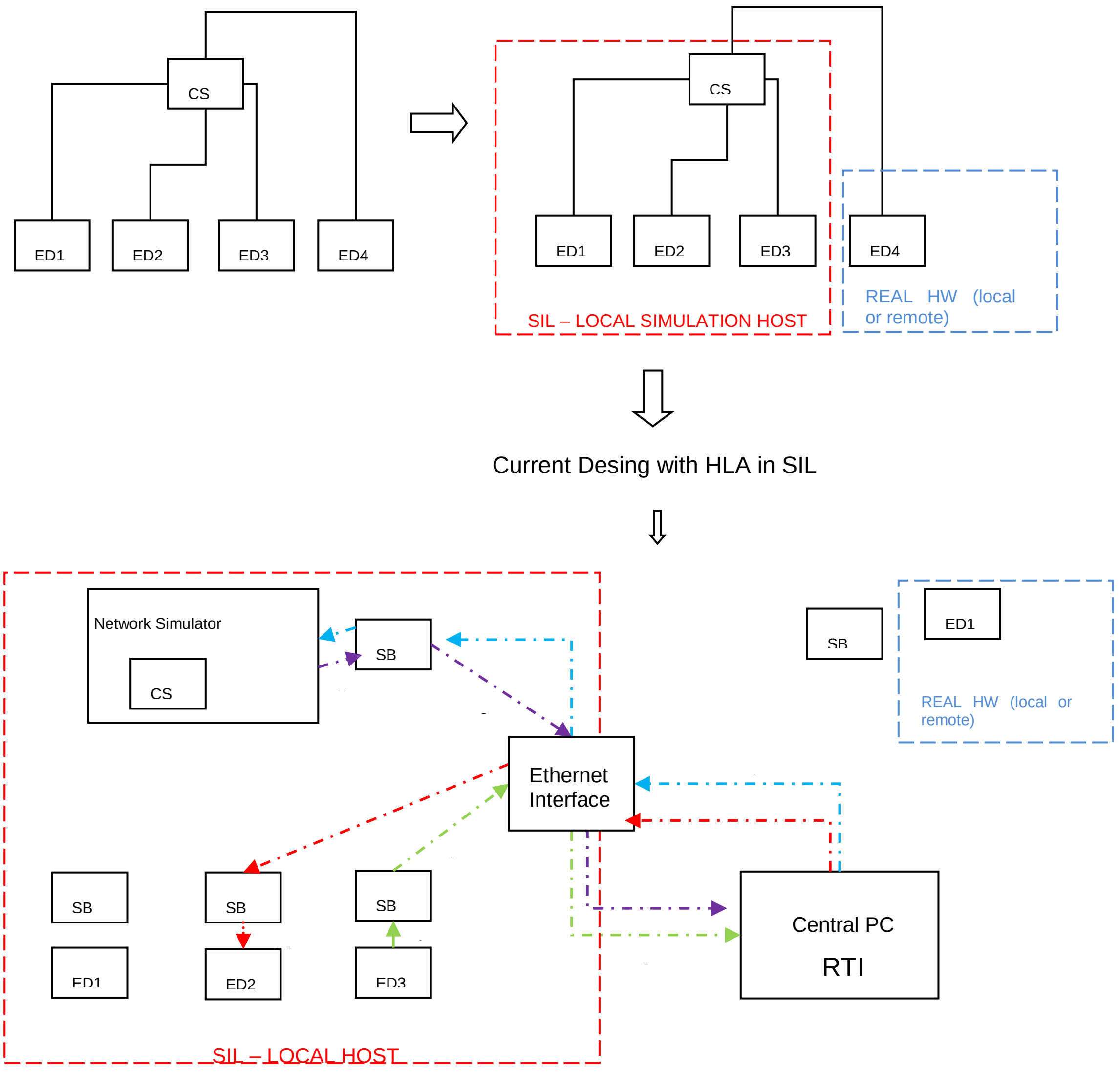


Figure 95. Current proposal SIL with SB with VPN and HLA.

All these messages introduce a high level of traffic inside the local host, and between the local host and the Central PC where RTI runs. Moreover, the message must travel across the RTI although the RTI could run in the PC of the local Simulation Host.

Taking into account that CONNECTA is not currently interested in the synchronization mechanisms provided by the HLA, the design inside the Local Host could be simplified. The use of HLA mechanism for the communication of those simulated devices that are inside the same Local Simulation Host can be avoided. In this case, the proposal is to use a reduced version of the SBs that send the messages produced by its device directly to the Network Simulator. They just introduce a small header to indicate to the Network Simulator to which CS (and to which port of the CS) the message should be delivered. Similar, the SB of the Network Simulator should know where to forward the packet. If the destination device is connected to the same host, the SB of the Network simulator should know the local destination SB. In case the destination device is not part of the SIL environment on the local host, it must forward the packet to the RTI running on the Central PC. The proposed reduced version of the SB should not include functionalities that introduce extra processing times such as the HLA and a VPN. Both should only be used for the communication between different PCs as proposed in Figure 96.

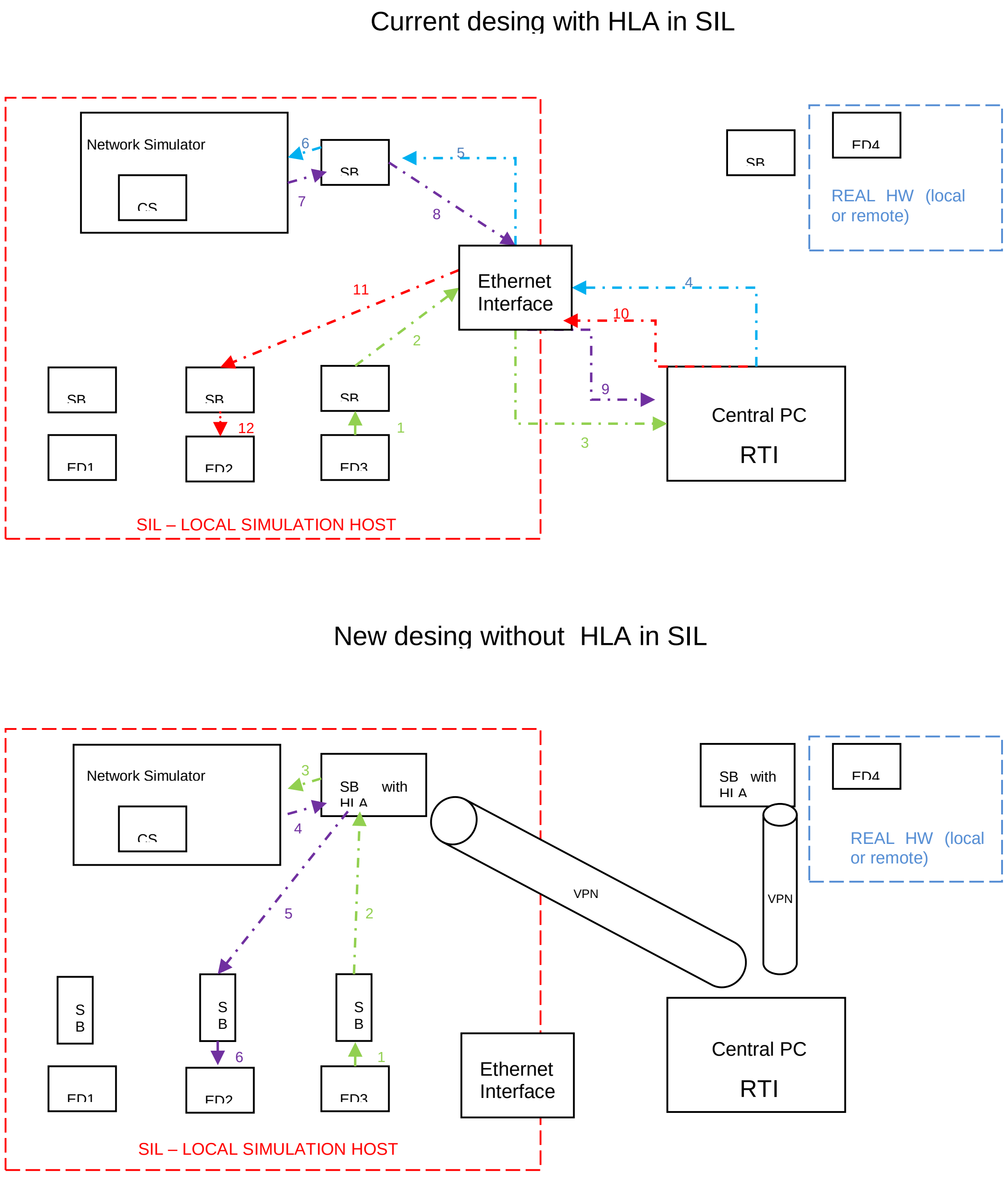


Figure 96. New proposal SIL with SB without VPN and HLA.

4.1.2 Derived requirement recommendations for drive-by-data and the embedded platform

This section presents requirement recommendations for the drive-by-data-concept. They are derived from the evaluation results gained in this deliverable, especially on the recommendations given for future projects. These recommendations are written on a high level and have to be refined according to the requirements and capabilities of the drive-by-data-concept and the embedded platform.

- The communication between the drive-by-data-concept and the embedded platform shall be based on Ethernet data transfer.
- The drive-by-data-concept shall provide a time synchronization mechanism between a real network node device and the PC hosting the simulation bridge. This mechanism is required to provide information to the simulation bridge when it can advance in time if the time management shall be used.
- The time synchronization mechanism used shall be compatible with Linux and Windows to enable the usage of the time management.
- The data exchange between the drive-by-data-concept and the embedded platform shall be based on a known communication schedule to enable the usage of the state-estimation mechanism.
- The communication schedule shall provide the information about the instant when a packet has to be injected into the network by the state-estimation mechanism.
- The communication protocol of the drive-by-data concept shall provide a mechanism which enables the simulation bridge to identify the destination of the message sent, e.g. using the destination MAC address of the Ethernet protocol.
- If the data exchange in the drive-by-data-concept has to be encrypted, the concept shall provide sufficient information in the packet so that the simulation bridge can identify the destination of the packet.
- The network simulation used to simulate network nodes in the drive-by-data concept shall provide a suitable interface to enable the connection with the CE. This interface has to be based on FMI or the tool has to send Ethernet frames via a dedicated network interface on the simulation host.

4.2 Conclusions and recommendations for the Train-to-Ground test environment

The work on the Train-to-Ground Test Environment (T2G TE) in WP3 provides means for validating the communication protocol between moving trains and a ground equipment implemented according to the IEC standard 61375-2-6 [11]. The work has been done in a tight cooperation with the CONNECTA project partners (Bombardier, Siemens, CAF, Alstom).

The CONNECTA project partners implemented a Mobile Communication Gateway (MCG) according to the standard, UniControls implemented a related Ground Communication Gateway (GCG). After the implementation phase, validation tests have been performed between the three available MCGs and the GCG using the T2G Test Environment. During all phases of the project (analysis, design, implementation, validation testing) many issues have been identified the IEC 61375-2-6 standard, such as:

- ambiguities in definitions and requirements
- inconsistencies throughout the text
- undefined behaviour of the components involved in the communication in certain situations (particularly handling of error situations)
- errors in some telegram content definition

These technical issues have been passed as comments to the national IEC standardization authorities during the final voting phase of the standardization process. But because the technical changes in the standard are not allowed during this phase, the issues have not been resolved. Therefore, we suggest that the standard will be revised in the near future taking into account the comments, otherwise individual implementations of the T2G communication will not be interoperable.

The possible changes in the T2G will of course also influence the T2G Test Environment, so even the TE shall be revised afterwards. We suggest that the T2G Test Environment, a GCG and an MCG shall be designed and implemented for early detection of problematic points in the revised standard before the standardization process reaches its final phase.

4.2.1 Comments on T2G standard

In Table 36 we provide a list of the comments on the T2G standard [11].

Table 37. Comments on the T2G standard.

Clause/ Subclause	Paragraph Figure/ Table	Comment	Proposed change
		Small letters in the term “end device” are used in its definition. But “End Device” and “End device” can be found in the document.	Use the correct form of the term throughout the document.
3	3.1.46	The definition of the term “job” is out of the alphabetical order.	Move the definition to the proper place.
4.2	Table 1	The requirement 13 “A job can result in a number of transactions” is unnecessary because a job is defined as sequence of transactions in clause 3.	Delete the requirement 13.
4.5	Table 5	Bulk Data class (item 8) is not defined in IEC 61375.	Use Best Effort instead of Bulk Data.
4.9.1	List of zones	Rules applied for the determination of the zones should be stated (at least by reference).	Beyond the sentence “…architecture is shown in” add the sentence “The zones in this model were determined by the application of the rules defined in IEC 62443-3-2”.
4.9.2		Some overview of security measures is provided but these measures should be related to the threats (high-level threats) identified before.	Add the section containing the list of high-level threats to the T2G system. Give the threat(s) addressed by that measure for each measure listed in Table 9.
4.9.2	1st paragraph	It should be mentioned that the list of the security measures in Table 9 is not exhaustive.	Modify the sentence to “…. are listed in Table 9 (the list is not exhaustive)”.

Clause/ Subclause	Paragraph Figure/ Table	Comment	Proposed change
4.9.3		Missing conclusions from the section describing the AAA model.	Express the conclusions in the form of policy.
4.9.4	Headline	The headline of the section does not express properly the content of the section. The section is about MSG classification by means of network categories.	Change the headline to "MSG classification"
4.9.4.1	Last but one paragraph	"The following clauses define security measures for a category 1 MCG". Those clauses define security requirements.	Replace "security measures" by "security requirements" in the sentence.
4.9.4.2 and 4.9.4.3	Headlines	The headlines should be formulated in similar way. From their formulation it is not clear at the first glance what the relation of the two subsections is.	Change the headlines to: "MSG connecting closed system to ground network" and "MSG connecting both closed and open system to ground network".
5.3.1	2nd bullet	"Access to a specific train journey, regardless …." refers to the access to a train, i.e. a journey train.	Change the sentence to "Access to a specific journey train, regardless …".
5.3.5.2	Headline and 1st paragraph	"Train journey addressing" – but it is about train addressing, i.e. journey train addressing.	Change to "Journey train addressing". Change "train journey" in the 1st paragraph to "journey train".
5.4.1	List of data classes	Each word of data class name starts with a capital letter (IEC 61375-1)	Change to Process Data, Message Data, Stream Data, Best Effort Data.
5.4.3 and 5.4.4		The sections describing the T2G transfer of data of particular data class in more detail should follow the order the data classes given in the list in 5.4.1., i.e. process data first.	Exchange the order of the sections 5.4.3 and 5.4.4.
5.6.2.3.2.1	Table 18	The ComID identifier is defined in Table 18 as "Identifier for the message body", which corresponds with the definition in IEC 61375-2-3 ("The ComId is a unique identifier for the user data structure of the PDU"). This means that there are different ComIDs for request and response messages. However, in the protocols in Section 6 the same ComID is used for both messages of the request-response pair.	Use the ComID according to the definition or change the definition.
5.6.1	Item c) in the list	"File transfer" is not a data type.	Change "File transfer" to "File".
5.6.2.2	1st paragraph	"The message data exchange service" - wrong name of the service	Change to "The message data communication service"

Clause/ Subclause	Paragraph Figure/ Table	Comment	Proposed change
5.6.2.2	Figure 18	The Figure 18 and its description (bullets at the end of the section) do not match.	Correct the description.
5.3.3.2.1	List a) to h)	The description is too verbose lacking comprehensibility (e.g. f)).	Simplify the wording.
5.6.3.2.1	Figure 20	The description in bullet f) does not correspond with Figure 20.	Correct the Figure 20 showing the interaction with the storage.
5.6.3.2.2.2	Table 29	The GCG should be able to reject the service in the MD reply.	Extend the ComID 203: MD body or add a message with status information.
5.6.3.2.2.4	Table 35	The GCG should be able to inform the MCG about the result of integrity check of the file transferred.	Extend the ComID 207: MD body.
5.6.3.2.2.5		Generally, the error handling concerning file exchange should not rely only on the time-outs. The involving of status messages could improve the protocol efficiency.	Consider the changes in the protocol.
5.6.3.3:1	Item e) in the list	Step e) is not shown in Figure 25	Correct the Figure 25.
	Item g) in the list	Unclear formulation	Reword the text.
5.6.3.3.1	Item h) in the list	“… inform one (or more) End Device(s) about…” – end device should be only in singular.	Change to “…inform one (or more) end device about…”.
5.6.3.3.1	Figure 25	Wrong orientation of “Download file” arrow.	Correct the Figure 25.
5.6.3.3.2.2	Table 41	It is not clear what the meaning of the “fileCheckResult” field in the ComID 209 message is. At the time this message is sent the file has not yet been downloaded, as can be seen in Figure 25.	Check the design of the protocol.
5.6.3.3.3	Table 47	The states indicating the failure of the download should be added. Consider the replacement of three state items by a single item that will present the state of the download state machine. The processing would be simpler. With three items there can be invalid combinations which should be checked.	Add the states “download to MCG failed” and “download to ED failed”.
	Table 48 to Table 55	Wrong alignment of the text in the columns.	Align text left (left alignment is generally used in the document’s tables).
6.3.1.5	Headline	“Capabilty detection”.	Correct to “Capability detection”.

4.2.2 Conclusions and recommendations for the T2G TE regarding the Access Network Simulator

This deliverable has presented two complementary strategies to evaluate the TCMS application traffic exchange between MCG and GCG through an LTE or WiFi network: a pure simulation platform and a software and hardware in the loop based platform. The pure simulation platform is based on the use of the discrete-event simulator Riverbed Modeler. All of these works are published in the following conferences [15], [16] and [17].

Simulation results for a high-speed scenario show that LTE can be the alternative communication technology, as it allows the integrity of transferring the interesting application traffic, when passengers use LTE simultaneously.

Regarding the scenario with adding a jammer node around eNodeB, simulation results show that this attack affect the transmission of TCMS traffic. Thus, it is interesting to integrate an anti-jamming technique for the network security and the safety transferring interesting data.

Regarding the scenario related to the establishment of WiFi-LTE heterogeneous connectivity, the results obtained show that this heterogeneous connectivity allows a relative continuity in the communications of a mobile train. However, the results of voice applications show that there are several disconnections that cause packet losses during the handovers between WiFi Access Points (AP) and during the handover from WiFi to LTE. Regarding the critical signalling application, the observed Packet losses and delay increase remain in compliance with the imposed limits. This is not the case for the voice and video applications for which the losses are likely to cause disturbances perceived by users. These disconnections can be minimized with more optimization of the mobility management. For the intra-AP handovers, improvements have to be made in terms of handover triggers and also in the downlink traffic redirection at the switch that connects the APs. For WiFi to LTE handover, proactive redirection mechanisms must be implemented to transfer downlink traffic from the WiFi core network to the LTE core network.

When other stationary trains are introduced in the scenario, results show that the WiFi network implemented in the station area can largely support the traffic generated by stationary and mobile trains. It remains that the *EDCA* QoS management mechanism implemented by WiFi is known to be non-performing in overload conditions. It would be interesting to study the performances perceived by the TCMS applications under overload conditions. It will be interesting to consider the scenarios where of the WiFi network is used for other applications such as station video surveillance and VoIP communications of station staff.

Regarding the co-simulation scenarios, tests and evaluation results lead to the following conclusions: First, the GCGSim simulator environment works successfully with the co-simulation test environment. All functions are tested and results are verified. Second, the network performance, such as delay and packet loss, are examined with the GCG host services. The artificial delay and packet loss are inserted in both hardware and software environment in order to mimic the real network transmission environment. Then, a few recommendations are drawn for using a co-simulation platform to conduct research and test scenarios for railway communication systems:

1. The LTE emulator provides a physical transmission environment for testing LTE signalling protocols and wireless signal conditions. The chosen LTE emulator is generic in nature and does not include natively artefacts specific for railway scenarios, such as channel effects in tunnels or mechanisms to emulate high speed movement and the related hand overs. Nevertheless, these could be built as add-ons to the lab set-up, as additional elements from the feeding lines from the antennas to the emulator elements.

2. The chosen LTE emulator is generic in nature and does not include natively artefacts specific for railway scenarios, such as channel effects in tunnels or mechanisms to emulate high speed movement and the related handovers. Nevertheless, these could be built as add-ons to the lab setup, as additional elements from the feeding lines from the antennas to the emulator elements.

The Riverbed Modeler brings the power to analyse the end-to-end packet transmission in a lab scale. Scenarios with scalable sizes are able to be simulated and traffic forwarding and routing parameters are configurable.

4.3 Conclusions and recommendations regarding certification aspects related to the test environment

Within this chapter, conclusions and recommendations related to the certification aspects of the test environments are provided. Certification in this context is related to the implemented applications or the developed hardware. The information is established based on the Communication Emulator, but also valid for tools in general such as the Train-to-Ground Test Environment. In addition, some brief considerations concerning the surrounding processes the tool is embedded in are given. Additional details concerning this topic can be found in Chapter 8 of D3.6, *Report on final requirements* [6].

4.3.1 General points about the needs caused by certification

Within Task 3.1, *State-of-the-art analysis and requirements for distributed simulation frameworks and solutions*, of Safe4RAL's Work Package 3, the needs from different norms for a tool which is used to test a safety application have been evaluated. The results concerning safety are documented in D3.1, *Report on state-of-the-art analysis and initial requirements for the distributed simulation framework* [1] and main elements can be summarized as follows:

- The tool must be able to detect all types of errors in a safety application.
- Not only the support of the safety application validation needs to be considered, but also the process landscape the tool is embedded including the interfaces.
- The tool must be categorized (keyword “tool failure impact”) following a standard the certification is based on.
- Depending on the classification, a tool qualification with all its components like risk assessment, qualification plan, documentation (see Section 5.1.2 in D3.1 [1]) is mandatory.
- The usage of the tool (e.g. skills of the tool users, user documentation, etc.) needs to be considered as well.

In addition to the points above, a definition of the tool and its individual components as well as an assignment of the responsibilities for the necessary work must be done.

Generally, it must be decided if the tools are planned to be used to provide the needed evidences for the certification or if the tools are “only” used during development as items to support the establishment of the applications or hardware. The evidences that the software or hardware is working properly needs to be provided then by other means which need to be treated as mentioned within this chapter.

4.3.2 Conclusions and recommendations

Following the brief information provided in Section 4.3 and Section 4.3.1 in combination with the content of Section 5.1.2 of D3.1 [1], the following conclusions can be drawn if the goal is to get a tool qualified:

- It needs to be decided first if the test environment will be used for the verification of the developed hard- and software.
- The different standards (e.g. EN50128, ED-12(DO-178), etc.) sufficiently describe the needs related to tool qualification.
- Tool qualification can be considered as important as the development of the applications (e.g. Functional Distributed Framework, Brake-By-Wire Application,…) or hardware (e.g. Integrated Modular Platform).
- The complete life cycle of the tool needs to be considered.
- The tool qualification process should be organized and performed in a way that the qualification of the tool is valid for all parties using the tool (e.g. it should be avoided that every developer of an application needs to perform the tool qualification on its own).

Thus, the following recommendations may be issued in the context of certification also because the tool qualification itself is not be part of the Safe4RAIL project:

- The decision if the Communication Emulator and the Train-to-Ground Test Environment will be used for the certification or if other means will be involved needs to be made soon.
- The decision about the standard (or standards and the relevant amendments) the tool qualification shall be based on should be felt as early as possible.
- The identification of the single components which are integrated within the Communication Emulator should to be performed. Furthermore, the responsibilities for the qualification activities of the single elements should to be defined. For example, who is responsible for the provision of evidences that the simulation of an end device which is integrated in the framework works properly?
- The main responsible for the tool qualification should be nominated as soon as possible.
- The activities (planning of the tool qualification, collection of evidences, etc.) for the tool qualification should be started as soon as possible and continued along the development process of Safe4RAIL and future projects.
- Certification authorities should be involved continuously during the tool qualification activities.
- The amount of effort for the tool qualification should not be underestimated and the activities related to that task should not be started too late.

In case the tools will be used to support the development, but not to provide the specified evidenced for the certification process, the activities for the tool qualification are not applicable for these tools. However, they must be applied for the tools which are used at the end.

Chapter 5 Summary and Conclusion

Work Package 3 (Virtual Placement in the Market) of Safe4RAIL has two goals with the aim of simplifying the verification and validation of train subsystems: a distributed simulation and validation framework and a Train-to-Ground (T2G) Test Environment. Safe4RAIL's work on the simulation framework focuses on the communication between train devices, hence it is called Communication Emulator. This framework can be used to connect geographically distributed real devices, software applications and simulation models to validate their interactions on the network-centric abstraction level. On the other hand, the T2G Test Environment focuses on the verification of the communication between an on-board Mobile Communication Gateway and a way-side Ground Communication Gateway. For this, LTE and WiFi are considered as communication media. This deliverable presents evaluation results for both environments.

The tests performed in Chapter 2 focus on the Communication Emulator. Tests where Beagle Bone Boards are connected show a communication delay of 6ms with a standard deviation of 2ms and a jitter of 13.55ms. If a simulation bridge has to handle sent and received traffic, the delay increases by 1-2ms. Furthermore, it increases if a simulation bridge has to receive traffic from two nodes. The maximum throughput accounts for 150 to 200 packets/µs. If the Central PC is located remotely (Equipment in Spain, Central PC in Germany), the delay is increased by a factor of 10 to about 60ms. Only in few cases, packets are lost or duplicated (less than 0.1% of the messages transmitted). These results are also valid for the usage of real train equipment. A second use-case evaluates temporal characteristic of the simulation bridges including time synchronization. If all subsystems are located in a LAN and FMI is used to connect the simulations and software applications, real-time simulations are possible. In this case, the main delays are introduced by the communication between the simulation bridges. However, as soon as we use PCAP for the communication with the connected devices, the delays are too large for real-time simulations. The reason is that PCAP in combination with the time management introduces large delays.

In Chapter 3, the outputs of the design, implementation and usage of the T2G Test Environment are presented. The work is done in cooperation with the European research project CONNECTA and shows issues in the present version of the IEC 61375-2-6 [11] standard. The issues together with solution proposals were submitted to the IEC standardization committee. The results for the T2G communication show that LTE is a suitable technology. They show that signalling data is transferred correctly between MCG and GCG. This also accounts for communication data which is sent using a protocol that allows retransmission such as TCP. A vertical handover scenario between WiFi and LTE shows the correct transmission if a moving train passes a station with passengers. In the train station, the train can pass several WiFi access nodes which is shown using a channel emulator. However, in case of using UDP or using a jammer node to disturb the wireless signal, packet loss can be observed.

Based on the results shown in Chapter 2 and Chapter 3, Chapter 4 presents recommendations for future projects. The main recommendations for the Communication Emulator are the usage of a synchronization mechanism between the simulation bridge and a connected, real device to enable the time synchronization between all federates (devices) in the simulation. Furthermore, a simplified simulation bridge without VPN and HLA could reduce delays between the connected federates if all components are connected to one PC. For the T2G TE, more scenarios should be defined for the validation of an error handling implemented in a GCG and an MCG. The behaviour of the devices in error conditions have to be defined in the IEC 61375-2-6 standard. Besides that, the LTE validation models can be improved implementing channel effects in tunnels or the emulation of high speed movements. Tool qualification is one major topic during the development of hard- and software components within the railway domain. As several evidences are to be provided during the certification process based on the established results or provided by these tools, they play an important role in general. Hence, Section 4.3 contains some main aspects which have to be considered. These aspects ensure that the needed evidences for the tools themselves are available in time and their absence does not hinder certification of the developed hard- and software.

Chapter 6 List of Abbreviations

Table 38. List of Abbreviations.

AC	Access Category
ANS	Access Network Simulator
AP	Access Point
API	Application Programming Interface
ARQ	Automatic Repeat Request
BBB	Beagle Bone Board
BE	Best Effort
CCU	Car Control Unit
CE	Communication Emulator
CESB	Communication Emulator Simulation Bridge
CETS	Communication Emulator Toolset
CS	Consist Switch
DL	Downlink
EDCA	Enhanced Distributed Channel Access
eNB	eNodeB
EnU	Engineer User
EPC	Evolved Packet Core
EPS	Evolved Packet System
ETBN	Ethernet Train Backbone Node
ETCS	European Train Control System
FMI	Functional Mock-up Interface
FTP	File Transfer Protocol
GBR	Guaranteed Bit Rate

GCG	Ground Communication Gateway
GPIO	General Purpose Input/Output
GSM	Global System for Mobile Communications
GSM-R	Global System for Mobile Communications – Rail
HCU	HVAC Control Unit
HIL	Hardware-In-The-Loop
HLA	High Level Architecture
HO	Handover
HVAC	Heating, Ventilation and Air Conditioning
IMS	IP Multimedia System
IP	Internet Protocol
JSON	JavaScript Object Notation
KPI	Key Performance Indicator
LAN	Local Area Network
LTE	Long Term Evolution
MCG	Mobile Communication Gateway
MD	Message Data
MoU	Mobile User
MuU	Multimedia User
NTP	Network Time Protocol
PCAP	Packet Capture
PID controller	Proportional, Integral and Derivative controller
PTP	Precision Time Protocol
PWM	Pulse Width Modulation
QCI	Quality of Service Class Identifier
QoS	Quality of Service
RSSI	Receive Signal Strength Indication

RTI	Runtime Infrastructure
RTOS	Real-Time Operating System
RTT	Round Trip Time
SB	Simulation Bridge
SFTS	Simulation Framework Toolset
SIL	Software-In-The-Loop
SITL	System-In-The-Loop
SSID	Service Set Identifier
T2G	Train-to-Ground communication [11]
T2G TE	T2G Test Environment
TCMS	Train Control & Management System
TCP	Transmission Control Protocol
TE	Test Environment
TRDP	Train Real-Time Data Protocol
UDP	User Datagram Protocol
UE	User Equipment
UI	User Interface
UL	Uplink
UMa	Urban Macro
UMi	Urban Micro
VPN	Virtual Private Network
WiFi	Wireless Fidelity

Chapter 7 Bibliography

[1] EU, Safe4RAIL, "Report on state-of-the-art analysis and initial requirements for the distributed simulation framework", D3.1

[2] EU, Safe4RAIL, "Report on Design of TCMS distributed simulation framework concept", D3.2.

[3] EU, Safe4RAIL, "Report on design of T2G Test Environment", D3.3.

[4] EU, Safe4RAIL, "Proof-of-concept implementation of distributed simulation framework", D3.4.

[5] EU, Safe4RAIL, "Proof-of-concept implementation of T2G Test Environment", D3.5

[6] EU, Safe4RAIL, "Report on final requirements", D3.6

[7] "Functional Mock-up Interface." [Online]. Available: https://fmi-standard.org/.

[8] "Railway applications - Communication, signalling and processing systems - Safety-related communication in transmission systems", Appendix A3.4 in *EN 50159:2010*, 2010.

[9] H. Kopetz. "Real-Time Systems: Design Principles for Distributed Embedded Applications". 2nd Edition. Springer Publishing Company, Incorporated, 2011.

[10] T. Pieper, R. Obermaisser. "Distributed co-simulation for software-in-the-loop testing of networked railway systems". In Proceedings of the 7th Mediterranean Conference on Embedded Computing (MECO), Budva, Montenegro, pp. 24-28, 2018.

[11] IEC Standard. IEC 61375-2-6: Electronic Railway equipment train communication network (TCN): Part 2-6: On-board to Ground Communication. IEC, 19.8.2016.

[12] EU. CONNECTA. 2018. Technical Contribution to WP2 – WP2 T2.3 T2G System Test Plan. CONNECTA, 2.3.2018. CTA-T2.3-T-BTD-035-01.

[13] EU. CONNECTA. 2018. CONTRIBUTING TO SHIFT2RAIL'S NEXT GENERATION OF HIGH CAPABLE AND SAFE TCMS AND BRAKES: D2.3 – Validation Plan and Report for the T2G System. CONNECTA, 31.8.2018. CTA-T2.3-I-BTD-008-06.

[14] https://www.railengineer.uk/2015/08/11/what-is-tcms/

[15] https://www.railengineer.uk/2018/08/13/future-train-radio-whats-possible/

[16] IEC Standard. IEC 61375-2-6: Electronic Railway equipment train communication network (TCN): Part 2-6: On-board to Ground Communication. IEC, 19.8.2016.

[17] Riverbed modeler, 2017. www.riverbed.com.

[18] https://opnetmodeler.wordpress.com/lte-long-term-evolution/

[19] Sniady A., Soler J. Performance of LTE in High Speed Railway Scenarios. Communi-

cation Technologies for Vehicles. Nets4Cars/Nets4Trains 2013. Lecture Notes in Computer Science, vol 7865. Springer, Berlin, Heidelberg.

[20] Roll2Rail Project. Deliverable 2.6: Architecture and Interface Definition for the Train to Ground Communication, 2016.

[21] B. Dusza, C. Ide, P.-B. Bok, and C. Wietfeld, "Optimized cross-layer protocol choices for LTE in high-speed vehicular environments," in Proceedings of the 9th International Wireless Communications and Mobile Computing Conference (IWCMC 2013), pp. 1046–1051, IEEE, 2013. ISBN 978-1-4673-2480-9.

[22] GROVER, Kanika, LIM, Alvin, et YANG, Qing. Jamming and anti-jamming techniques in wireless networks: a survey. International Journal of Ad Hoc and Ubiquitous Computing, 2014, vol. 17, no 4, p. 197-215.

[23] http://www.openairinterface.org

[24] www.riverbed.com

[25] https://iperf.fr/en/

[26] Maha Bouaziz, Ying Yan, Mohamed Kassab, José Soler and Marion Berbineau, Train-to-Ground communications of a Train Control and Monitoring Systems: A simulation platform modelling approach, 7th Transport Research Arena TRA 2018, April 16-19, 2018, Vienna, Austria.

[27] Maha Bouaziz, Ying Yan, Mohamed Kassab, José Soler and Marion Berbineau, Evaluating TCMS Train-to-Ground communication performances based on the LTE technology and discreet event simulations, 13th International Workshop on Communication Technologies for Vehicles – Nets4cars / Nets4trains / Nets4aircraft, May 2018, Madrid, Spain.

[28] Ying Yan, Maha Bouaziz, Mohamed Kassab, Marion Berbineau and José Soler, Co-simulation Platform for Train-to-Ground communications, 20th Nordic Seminar on Railway Technology, June 2018, Gothenburg, SWEDEN.

[29] T. Blochwitz, M. Otter, M. Arnold, C. Bausch, H. Elmqvist, A. Junghanns, J. Mauß, M. Monteiro, T. Neidhold, D. Neumerkel, et al. "The functional mockup interface for tool independent exchange of simulation models". In Proceedings of the 8th International Modelica Conference; March 20th-22nd; Technical University; Dresden; Germany, number 063, pp. 105-114. Linköping University Electronic Press, 2011.

[30] EU. CONNECTA. 2017. CONTRIBUTING TO SHIFT2RAIL'S NEXT GENERATION OF HIGH CAPABLE AND SAFE TCMS AND BRAKES: D2.1 – Requirements and Specification for the T2G System. CONNECTA, 27.06.2017. CTA-T2.1-D-CAF-004-09.