Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Masterprojekt2-Ausarbeitung WS2012

Florian Bartols

Cluster Simulation of
Real-time Ethernet based Electronic Control Units
in Context of Automotive Applications:
Calculation of Suitable Hardware Setups

Fakultät Technik und Informatik
Department Informatik

Florian Bartols

Thema der Masterprojekt2-Ausarbeitung
WS2012
Restbussimulation von Echtzeit Ethernet-basierten Steuergeräten im Automobil: Berechnung passender Aufbauten

Stichworte
Echtzeit Ethernet, TTEthernet, Restbussimulation, Bussysteme, Automotive Anwendungen, FIBEX

Kurzzusammenfassung
Die zunehmende Komplexität automobiler Netzwerke und deren hohe Bandbreitenanforderungen erfordert neue Konzepte in der Vernetzung der elektronischen Komponenten im Fahrzeug. Real-time Ethernet ist ein geeigneter Kandidat diese Problemstellung zu lösen. Um frühzeitig im Entwicklungsprozess Steuergeräte zu testen hat sich die Restbussimulation als adäquates Mittel etabliert und wird im Projekt am Beispiel des TTEthernet-Protokolls für Real-time Ethernet erarbeitet. In dieser Arbeit werden prinzipielle Hardwareaufbauten und Topologien für eine Real-time Ethernet Restbussimulation erläutert. Des weiteren wird ein formales Modell auf Basis der Mengentheorie vorgestellt, dass die Berechnung passender Topologien in Abhängigkeit ausgewählter Testlinge ermöglicht.

Title of the paper
Cluster Simulation of Real-time Ethernet based Electronic Control Units in Context of Automotive Applications: Calculation of Suitable Hardware Setups

Keywords
Real-time Ethernet, TTEthernet, Cluster Simulation, Bussystems, Automotive Applications, FIBEX

Abstract
The increasing complexity of automotive networks and their high bandwidth requirements will demand new concepts in networking of electronic automotive components. Real-time Ethernet is a suitable candidate to solve this problem. In order to allow testing of control units in an early point in the development process, cluster simulation has been established as an adequate manner and therefore in this project it is developed for Real-time Ethernet using the example of TTEthernet. In this work, hardware and topology setup approaches for a Real-time Ethernet based cluster simulation are explained. Furthermore, a formal model basing on set theory is presented, that allows for calculating suitable topologies in dependency of selected devices under test.

Contents

1 Introduction

Today's cars are complex, mixed critical [1], distributed systems with more than 70 single components, called electronic control units (ECUs) [2]. These units are responsible for either information and entertainment systems e.g. radio navigation or safety relevant functions such as x-by-wire or adaptive cruise control systems. Thus, the requirements for these systems differ in bandwidth, reliable data transmission and real-time behavior. Safety critical functions have high demands in real-time characteristics, whereas entertainment systems require high bandwidth capabilities. Due to these different requirements, the in-car network infrastructure became heterogeneous in the past. For each kind of application class a dedicated system for on-board networks has been utilized. As a consequence, the required on-board network topology became more and more complex and nowadays, it is hardly manageable [3]. Also next-generation driver assistance functions like camera or radar based systems will tighten the problematic of future in-car networks. These applications both have high real-time requirements and high demands in bandwidth. State-of-the-art network systems like CAN [4] and FlexRay already [5] operate on their bandwidth limits or do not meet the real-time requirements (MOST [6]), which makes them unsuitable in the field of these new assistant systems. A possible solution for the addressed problems is the application of Ethernet-based networks in vehicles [7, 8, 9]. Real-time Ethernet (RTEthernet) is a suitable candidate for future on-board networks to overcome the issues imposed by next-generation driver assistance. It provides reliable real-time data transmission while supply high bandwidth as well. Additionally, the complexity of the network infrastructure can also be reduced, due to the ability of dedicated message transmission in different classes, which would result in one 'flat' network system only.

Further on, the utilization of software controlled automotive applications is rising rapidly, too. It is the key driver for new innovations in the automotive industry and its utilization has increased in a few years from 20 percent to 80 percent. Forecasts claim that 90 percent of the automotive functions will be realized in software in this decade [10]. This fact has a direct influence on the costs during the development process of a new automobile. To minimize these costs, the software has to be tested in an early development stage. The functionality spread over several units makes this goal even more difficult, since the development of new cars is also more and more distributed [11]. The Original-Equipment-Manufacturer (OEM) designs the car and its features in specifications, while external suppliers produce only parts. Therefore, the suppliers have to test their developed ECUs entirely on their own, prior to the first assembly of all components during the system integration. Currently missing nodes that are responsible for

an essential part of the application can make this testing complex. For this purpose, a testing method called cluster simulation has to be applied by the suppliers. A cluster simulator imitates the missing nodes and their behavior inside the network and therefore enables a solution for unit-testing purposes. Cluster simulation has already proven to be utilized inside the development processes for state-of-the-art technologies, since many commercial products have been developed [12].

The to be tested devices are connected to the cluster simulator via the real network interface, in order to get triggered with regular messages. Since switched networks, like RTEthernet, provide a separate collision domain for each participant, the network topology differs to the state-of-the-art technologies, which are basing on a bus structure. Therefor, the hardware setup and the corresponding topology for RTEthernet based cluster simulation is also different compared to bus based network systems. In contrast to a bus based cluster simulation, the number of parallel connected DUTs has a significant influence on the RTEthernet simulation setup as well. Since the setup effects also the cost for the cluster simulation, it should be possible to calculate the required attributes of a cluster simulator in dependency of the number of DUTs in order to find the best trade-off between cost and performance.

In this paper, topology alternatives for RTEthernet based cluster simulation are presented and discussed. Furthermore, the influence of the number of connected DUTs is investigated. A formal model, basing on set theory, allows calculation, whether a topology alternative in dependency of the number of DUTs can be utilized to perform a cluster simulation properly and to find the best trade off.

The remainder of this paper is organized as followed: Section 2 on the following page presents related work on cluster simulation hardware setups for the TTP/C protocol and introduces setups and corresponding topologies for RTEthernet based cluster simulation. In Section 3 on page 18 a formal model to calculate required attributes of a RTEthernet based cluster simulation is outlined and results of an example network are presented in 4 on page 32. Section 5 on page 38 concludes and gives an outlook on upcoming work.

2 Hardware Setups of Real-Time Ethernet based Cluster Simulation

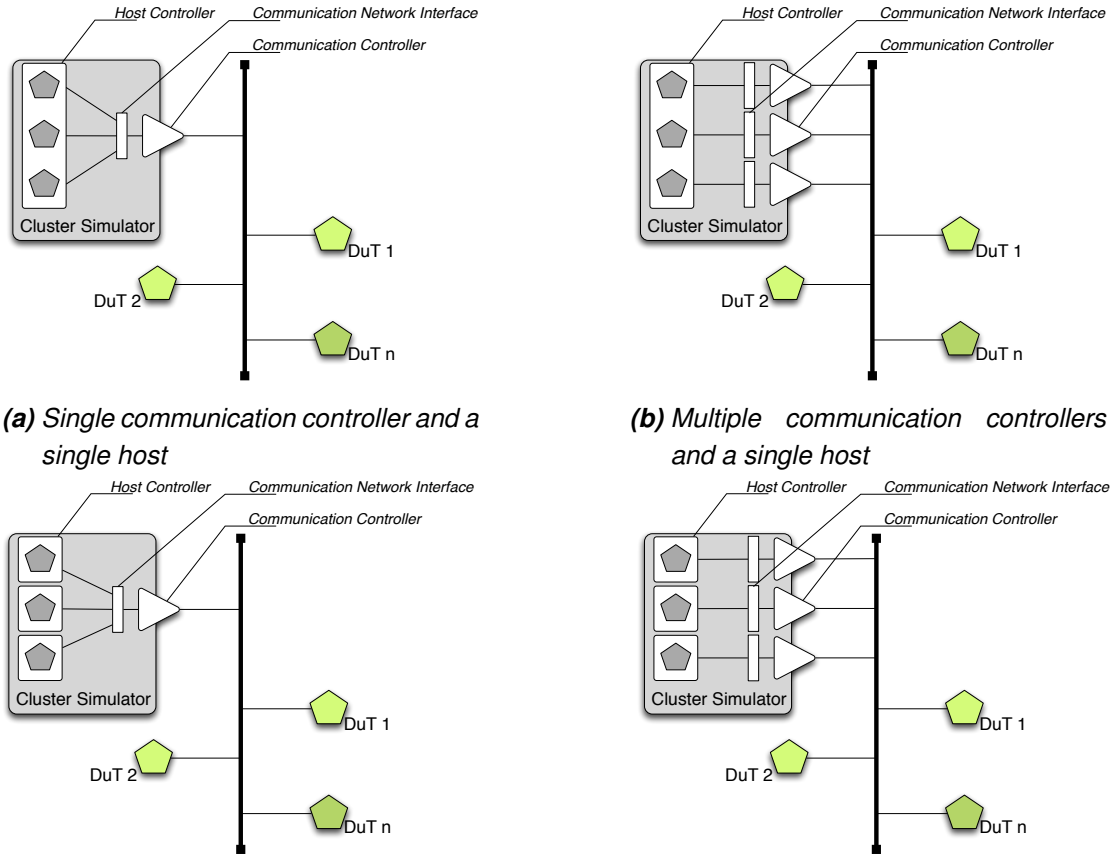
Cluster simulation is a suitable method to test electronic control (ECU) units in an early phase during the development process. Former solutions mostly confirm to real-time aware bus systems like CAN or FlexRay, basing on a shared media access scheme that only allows transferring one message at the same time. Switched networks however, provide a separate collision domain for each network participant, which allows parallel transmission of messages. Due to different media access approaches, bus and switched networks utilize different network topologies. Since cluster simulation uses the real network to trigger and stimulate the devices under test (DUTs), it might be that the simulation topology for switched networks has to be adopted to test the devices properly and cost efficient. This section will present hardware setups of a former used bus based cluster simulation and afterwards, setups for RTEthernet based cluster simulation. Finally, their characteristics will be compared to each other and a short conclusion outlines irrelevant topologies.

2.1 Related Work: Former Utilized Setups for Time-Triggered Bus Systems

In general, the hardware setup for bus based systems is limited, due to the shared media access, which means the network setup for cluster simulation and the real network is identical. A topology for bus based cluster simulation consists of DUTs and a simulation platform interconnected via the bus medium. However, there can be differences in the system architecture of the cluster simulator. In his dissertation [13], Thomas M. Galla has presented four different hardware setup approaches for TTP/C cluster simulation. TTP/C is a time-triggered bus system in the context of the Time-Triggered Architecture [14] utilized to interconnect components in order to form a highly dependable real-time system [15]. A TTP/C node is composed of a host microcontroller, that runs the actual application, and a communication controller, that establishes the connection to the network by running the protocol stack. Both controllers communicate via the ‘Communication Network Interface’ (CNI) which can be implemented as dual port memory.

Since the developed cluster simulator had to be conformally implemented to the general system architecture of TTP/C nodes, its system architecture is divided into two subsystems (host and communication system), too.

The cluster simulator host subsystem is composed of either one or multiple micro controllers, that are responsible for the simulation of nodes. The communication sub-system, again composed of one or multiple controllers, to allow protocol compliant communication. Four different hardware setups are depicted in Figure 2.1 and are discussed afterwards.



(a) *Single communication controller and a single host*

(b) *Multiple communication controllers and a single host*

(c) *Single communication controller and multiple hosts*

(d) *Multiple communication controllers and multiple hosts*

Figure 2.1: *Bus based cluster simulation hardware setups in comparison*

Approach 1 This approach consists of one communication controller and one host controller to perform the cluster simulation and is depicted in Figure 2.1a. It is the most simple and cost efficient opportunity to achieve cluster simulation for time-triggered bus systems. All simulated nodes are executed on one host controller. Therefor, a coordination is unnecessary to prevent the CNI from concurrent access. However, this hardware setup can produce two conflict scenarios which may have to be resolved:

1. The mapping of several tasks, executed on real nodes to one physical processor. If the overall CPU simulation utilization exceeds the CPU performance of the simulation platform, simplifications to the simulation model of the simulated nodes have to be applied.
2. Since the specification of TTP/C allows communication controllers to send only one time during a time-division-multiple-access (TDMA) round, the communication protocol for the cluster simulator has to be modified to be able to send in each simulated slot.

Approach 2 This approach is depicted in Figure 2.1b on the preceding page and is composed of one host controller and a dedicated communication controller for each simulated node. Due to the fact that only one host controller is utilized, a coordination to prevent the CNI from concurrent access is unnecessary, too. While solving the protocol conflict, the mapping of several tasks on one physical processor must still be taken into account.

Approach 3 Figure 2.1c on the previous page depicts the approach with a single communication and a dedicated host controller for each simulated node. On the one hand, this alternative provides sufficient CPU performance, which makes simplifications to the simulation model rather unnecessary. But on the other hand, additional effort has to be taken into account to coordinate the access to the CNI from the host controllers to prevent concurrent access. Still, the protocol has to be modified to be able to transmit in each simulated TDMA slot, since only one communication controller is used.

Approach 4 An approach with multiple communication and host controllers is shown in Figure 2.1d on the preceding page. To be precise, this approach does not really conform to a cluster simulation approach, since for each simulated node a dedicated communication and host controller is used. Thus, this approach alternative solves all conflicts imposed by alternative 1. It provides sufficient CPU performance and multiple communication interfaces to make simplifications and modifications to the protocol unnecessary. From the economic perspective, this approach has a major drawback in the amount of utilized hardware.

Under the assumption that the component costs for host controllers were rather high, approach 1 and 2 fit best to perform an economically feasible cluster simulation for time-triggered bus systems. Under simulation capability perspective, approaches 3 and 4 outperform the previous alternatives. Nevertheless, Thomas M. Galla used the single communication and single host configuration (approach 1) to implement a hardware cost efficient cluster simulator for time-triggered bus systems. The modification of the protocol was accepted for economic reasons and the simulator was successfully used in the development process of a steer-by-wire prototype system.

In contrast to the four presented setup approaches for time-triggered bus systems, setups for RTEthernet based cluster simulation differ in the connection between the DUTs and the simulator platform. Nevertheless, concepts for the simulation platform composition can be also used in RTEthernet. In general, the simulation of nodes has to be simplified as well, if the computation performance is not sufficient enough. Also the principle of using more computation units to get more simulation performance can be adopted. On the other hand, if the DUTs are communicating with each other, a switch has to be simulated or must be physically available in hardware.

2.2 Real-time Ethernet Based Hardware Setups

The last section has depicted hardware setups for time-triggered bus systems, which have a shared collision domain among all participants. This section deals with hardware setups for RTEthernet based network systems. RTEthernet networks are based on switched Ethernet, where each network participant has its own collision domain, which allows parallel message transmission. Due to the access scheme of RTEthernet, the hardware setup does not only affect the cluster simulation platform composition, but also the used network topology. This results in different possible setup configurations, that are discussed in detail in this section. The discussion focus is set towards setup dependent attributes, performance and economic characteristics. In general, setups can be classified in approaches with a direct connection between the cluster simulator and the DUTs (refer Figure 2.2, described in Section 2.2.1 on the next page and setups with additional physical RTEthernet switches, outlined in Section 2.2.2 on page 12. Both approaches allow increasing the simulation performance by using multicore platforms, like symmetric multiprocessing (SMP) architectures. In both setup approaches it is desirable to record the traffic inside the network to allow additional test analysis.

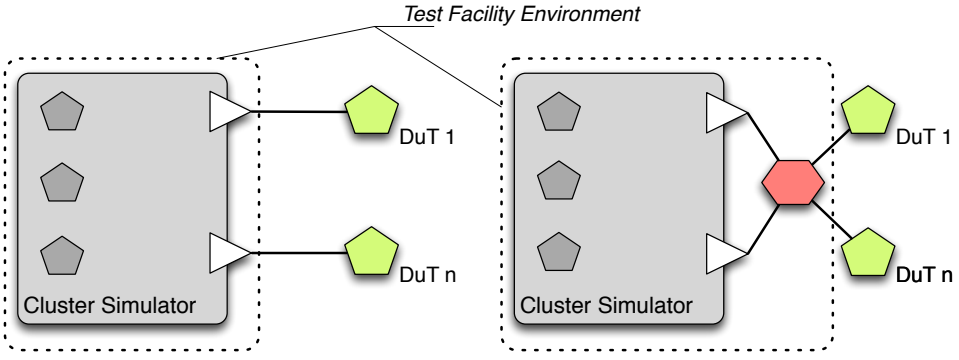


Figure 2.2: *Comparison of a direct cluster simulation and an approach with additional RTEthernet switch*

2.2.1 Directly Connected Devices Under Test

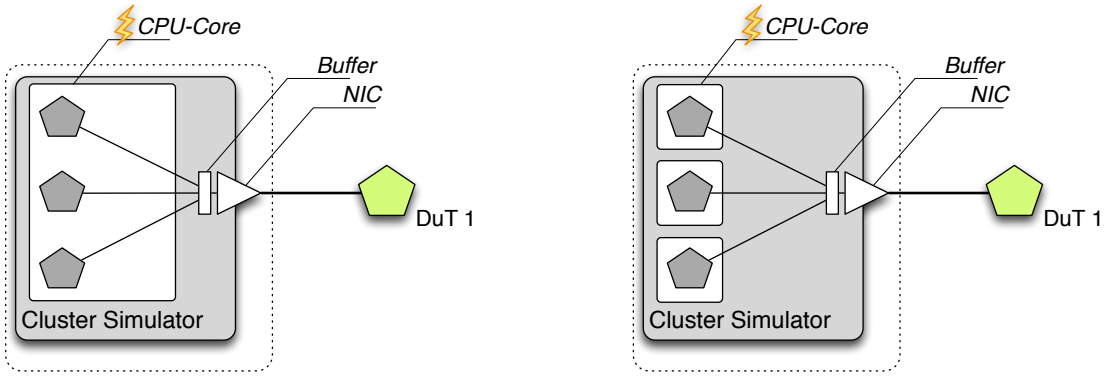
Hardware setups with a direct connection do not explicitly need further RTEthernet hardware like switches and the network topology corresponds to a peer-to-peer approach. The DUTs are directly coupled to the simulator platform which results in the following setup scenarios:

- One network interface card for one connected device under test
- Multiple network interface cards for multiple connected devices under test
- Distributed simulator for multiple connetected devices under test

The used hardware platform must be real-time capable, since messages in RTEthernet are scheduled in a time-triggered (TT) or rate-constrained (RC) manner. TT-messages expect a high timely precision, whereas RC-messages have to be transmitted according a configured 'bandwidth allocation gap' (BAG). Additionally, the cluster simulator has to completely support the time synchronization mechanism of RTEthernet. Since no further hardware is required, only the simulation platform has to be configured.

One NIC for One DUT

The first alternative is composed of a cluster simulator with one network interface card (NIC), that is directly connected to the device under test and consists of a single or SMP CPU architecture (Fig. 2.3a respectively Fig. 2.3b). In general, this is the simplest hardware setup alternative to perform a RTEthernet cluster simulation, but it is limited to one single device under test.



(a) *One network interface and one CPU core*

(b) *One network interface and multiple CPU cores*

Figure 2.3: *Hardware setup for a direct connected device under test*

Since only one DUT is connected to the cluster simulator, the message scheduling can be taken from the DUT's configuration with little adaptation. As already presented in the setup alternatives for bus systems (refer to Section 2.1 on page 3), the mapping of simulated nodes on one or multiple cores has to be achieved, too. Again, the CPU utilization of a core must not exceed the CPU performance. An adequate approach to simplify the simulation of nodes is to execute only tasks that are responsible for the generation and reaction of messages, which are consumed and produced by the DUT, instead of imitating the complete behavior of a simulated node.

These tasks can be either implemented in a single-process, (only reasonable on a single core system), or a multi-process environment. In a single-process, the simulation tasks are scheduled sequentially as functions, which excludes concurrent NIC buffer access. If the simulation tasks are implemented as a multi-process system, the tasks can run independently and the NIC access has to be synchronized in order to avoid concurrent access. Since the platform is composed of only one network interface, the simulation tasks do not have to be assigned to dedicated interfaces.

Furthermore, the transmission of event-based messages, like RC- and BE-messages, can be delayed, since TT-messages have always precedence. They are queued in an according buffer and are sent, regarding their scheduled transmission.

A suitable use-case for this alternative is the unit testing approach, where the logical behavior of an ECU application can be tested regarding conformance, by analysing the received data. Further, this alternative allows rapid prototyping of new applications if the final ECU hardware is not yet completely specified or physically available in hardware. The application is executed on the cluster simulation platform and can be investigated in another testing or prototype environments.

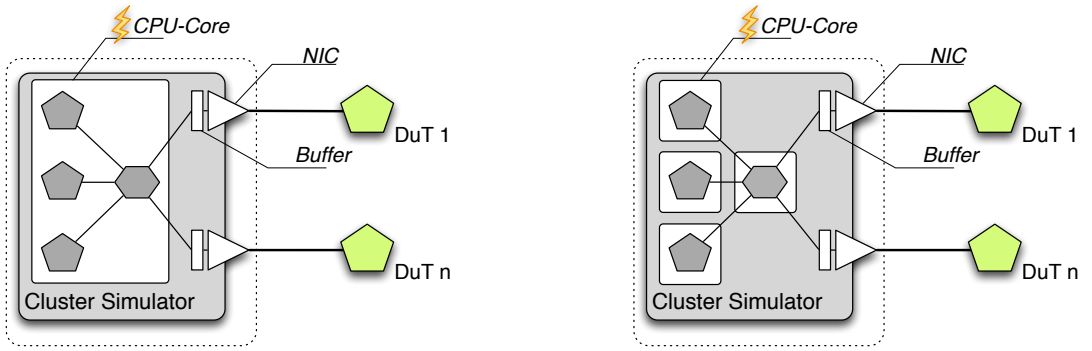
Advantages Easy implementation of the cluster simulator itself, since the message scheduling can be taken from the DUT with little effort. Also the simulation task scheduling is simple, if the simulator is implemented for single process architecture. Under the assumption that the cost for implementation is rather high, this approach is economic attractive. The CPU performance for cluster simulation should be sufficient, since only one device is tested. Thus, the risk of performance issues is quite small.

Disadvantages This approach is limited to one DUT. More on, the cluster simulator must be real-time capable which results in higher hardware costs.

Multiple NICs for Multiple DUTs

The second alternative for cluster simulation is depicted in Figure 2.4 on the following page. Again, the simulator is directly coupled with the to be tested devices. In contrast to the first

alternative, the platform is composed of one network interface per DUT. This allows communication between the DUTs via a simulated software switch, if one DUT depends on data from another DUT. With this alternative, it is possible to test multiple devices at the same time without using further RTEthernet hardware in terms of additional switches, but the to be transmitted messages have to be assigned to dedicated NICs, in order to trigger the correct DUT.



(a) *Multiple network interfaces and one CPU core*

(b) *Multiple network interfaces and multiple CPU cores*

Figure 2.4: *Hardware setup for a direct connection with multiple NICs*

Multiple DUTs are connected to the cluster simulator which results in a more complicate implementation than the previous alternative, since the number of simulated messages are likely higher in dependence of the amount of DUTs. The implementation becomes even more complex, if a parallel transmission and reception of messages has to be applied. This boosts the risk of performance issues significantly, if the simulator is build upon a single-process simulation architecture.

If multiple DUTs expect the reception of independent TT-messages at the same point in time in a single process, the according messages have to be sequentially calculated and must be buffered in the NIC before their actual transmission start. The message release can then be performed nearly in parallel by activating the start of transmission in all NICs. If independent TT-messages are to be received at the same point in time, they have to be sequentially read from the NICs. In a multi-process, multi-core simulator architecture (Fig. 2.4b), the messages can be calculated in parallel, but access from processes to one NIC has to be synchronized in order to avoid concurrent access.

This setup allows exchanging messages through the devices under test. Therefor, the shared traffic has be passed on at the simulation platform from one interface to the other regarding the simulated software switch. Since handing messages over at the cluster simulator induces a larger and a non-deterministic delay than a typical RTEthernet switch, the timing behavior of TT-messages can be negatively affected, because the message routing has to be done in software. At worst, TT-messages miss their transmission slot and will be dropped at receiver

side. As a solution, a special NIC with several separate ports can be used, that hands the traffic over network card internal without using the hosts data bus.

On the other side, it is simply possible to analyze the traffic that is shared among the DUTs without using further hardware which results in network sniffer capabilities by default. The traffic can be collected at the simulated switch and passed to a further process, that stores the traffic.

While the first presented alternative is used for unit testing, this setup can also be used during the integration phase, where single devices are connected to build a system for the first time.

Advantages Under the assumption that the costs for RTEthernet switches are rather high, this setup provides the capability to test multiple devices at once, without using further expensive hardware. Therefor, only the cluster simulator itself have to be configured regarding the selected devices under test.

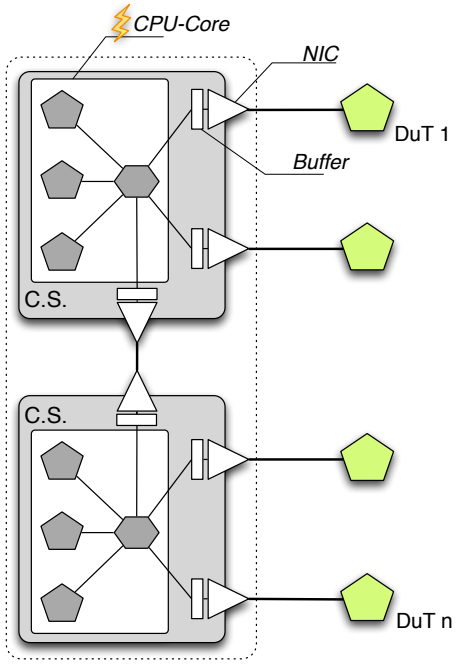
Disadvantages Although further hardware costs for external devices can be saved, the implementation cost for the simulator rises significantly with the amount of DUTs and the to be simulated messages. Also the actual hardware requirements are higher, because the simulator itself must be real-time capable. The utilization of a special NIC with multiple ports raises also the cost significantly.

Distributed Simulator for Multiple DUTs

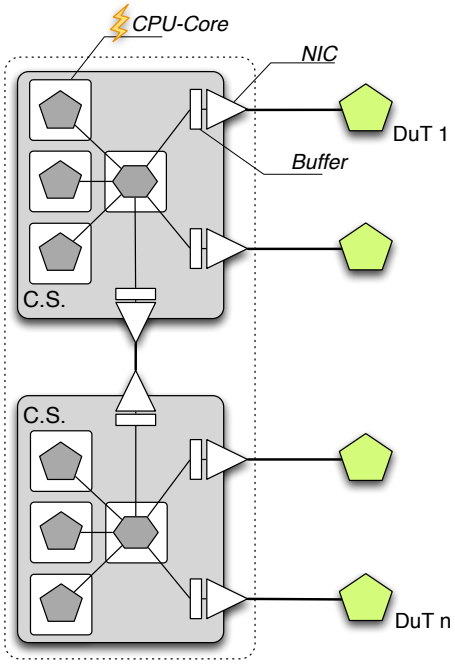
As a third and last alternative, a distributed approach (Fig. 2.5 on the following page) for a direct connected cluster simulation is possible. The cluster simulator is arranged on several separate hardware platforms, each composed of one NIC per DUT. The distributed simulators are also directly connected to each other, to allow exchanging messages between the subsystems under test.

The distributed setup can be seen as a special case from the previously presented approaches ‘One NIC for One DUT’ on page 7 and ‘Multiple NICs for Multiple DUTs’ on page 8, but causes further problems. Since the simulation is distributed in several parts, all cluster simulator subsystems have to be synchronized, in order to correctly schedule messages and to provide a system wide consistent time base. Further, if a DUT depends on data from other DUTs, the traffic may have to be traversed between the cluster simulator subsystems, which can induce further non deterministic delays. As corresponding use case for this setup, the integration phase can be seen.

Advantages In the case of only one connected DUT per cluster simulator subsystem, the scheduling can be directly taken from the DUT’s scheduling table with little adaptation.



(a) *Distributed approach with one CPU core*



(b) *Distributed approach with multiple CPU cores*

Figure 2.5: *Hardware setup for a distributed appraoch with a direct connection*

Disadvantages From the economic perspective, the major disadvantage is rather obvious. Since more than one cluster simulation platform has to be utilized, the hardware costs are significantly higher. If more than one DUT is connected to the cluster simulator, the implementation costs are expensive as well. Furthermore, the induced delays can be greater, since the traffic may have to be handed over between several cluster simulation subsystems.

2.2.2 Indirectly Connected Devices Under Test

A direct connection between the cluster simulator and the DUTs is not the only option, to perform a cluster simulation. Due to the fact, that RTEthernet is based upon switched Ethernet, a dedicated RTEthernet switch can be used to avoid the non deterministic traversal behavior if multiple DUTs are connected. A utilization of such hardware, results in the following setup alternatives:

- One network interface card for one or multiple devices under test
- Multiple network interface cards for multiple devices under test
- Distributed simulator for multiple devices under test

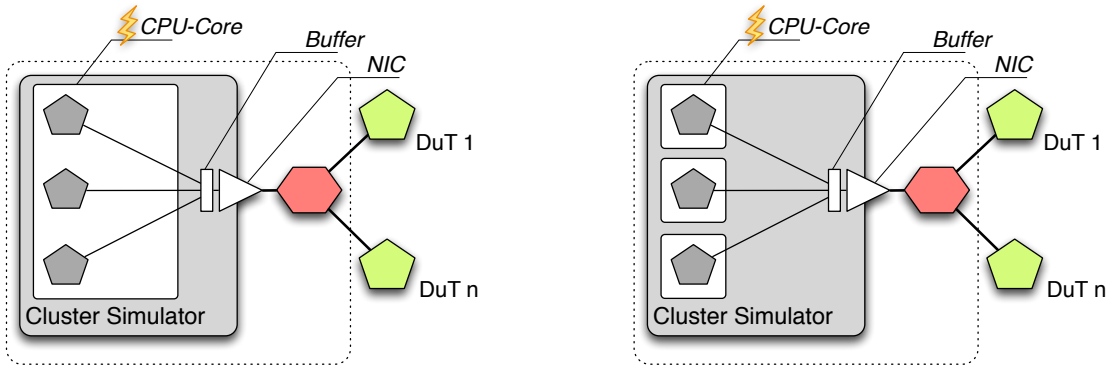
In this hardware setup classification, the utilization of one or more dedicated RTEthernet switches is mandatory, since standard Ethernet switches are not capable of time-triggered or rate-constrained message transmission, which results in additional costs for further RTEtherner capable hardware. The message routing and the proper scheduling has to be statically configured to the switches.

These switches in general have a higher timing precision than software based RTEthernet endsystems, because the protocol is implemented in hardware. Therefor, the cluster simulator does not have to precisely send critical messages according their configured time slot, but at least before their actual scheduled receiving point in time at the DUT. The cluster simulator can use the timing precision of the switch to transmit the messages to the DUT. In order to schedule TT-messages in time, the cluster simulation platform has to be synchronized to the time of the system, too. Nevertheless, these timing requirements are less than a setup with a direct connection to the DUTs, since the TT-messages can be buffered and precisely sent by the switch and not by a software stack in the cluster simulator's host. Therefor, TT-messages are permitted to jitter for a certain interval while transmitted from the actual cluster simulator platform. An in-time scheduling could be realized by receiving TT-messages with a configured bigger receive-window at the switch. The RTEthernet switch is then responsible for the precise time-triggered transmission of the message to the DUT.

Since this setup allows testing of multiple DUTs, these devices can interact with each other by sending messages, which makes a traffic analysis desirable. It is not as comfortable as in the alternatives 'Multiple NICs for Multiple DUTs' on page 8, since the cluster simulator is not located between the DUTs. Therefor, additional monitoring routes to the simulation platform have to be configured to the switches. The next subsections will discuss the attributes in detail for the presented setups.

One NIC for One or Multiple DUTs

The first setup alternative with an additional switch is presented in Figure 2.6. Similar to the alternative 'One NIC for One DUT' on page 7, the cluster simulator platform is composed of one network interface only, that establishes the connection to the RTEthernet switch, while the DUTs are coupled to the switch.



(a) *One network interface and one CPU core*

(b) *One network interface and multiple CPU core*

Figure 2.6: *Hardware Setup with Additional Switch and one Network Interface*

The scheduling of simulated messages at the cluster simulator has to be configured to achieve an in-schedule message reception at the DUTs. If only one device is tested, the messages scheduling can be taken from the DUT's scheduling configuration. Additionally to the physically imposed transmission delay, the scheduling delays imposed by the switch have to be considered as well. As addressed in the introduction of this setup classification, the cluster simulator does not need to exactly transmit TT-messages, since the RTEthernet switch have a higher timing precision. This reduces the timing requirements for the simulator significantly, because the messages only have to be causally scheduled.

If more than one DUT is connected to the switch, the cluster simulation becomes more complex and can cause conflicts that have to be resolved before configuration. In RTEthernet networks

it is possible that parallel message transmission occurs. Therefor, the DUTs connected to the switch can send and receive messages independently which leads to the following conflict scenarios because multiple Ethernet connections are integrated to a single connection:

Message scheduling conflict Different TT-messages are configured to be received or sent at the DUTs at the same point in time. Since in this setup only a single connection to the cluster simulation is available, a concurrent transmission is excluded by default. The TT-messages have to be sent sequentially to the RTEthernet switch, which can buffer the messages for a certain amount of time. It is then responsible for the in-schedule transmission to the DUTs. If multiple DUTs are transmitting independent TT-messages at the same point in time to the cluster simulator, the RTEthernet switch have to sequence and integrate the concurrent messages, which leads to the following conflict:

Bandwidth conflict Since multiple connections are integrated to a single connection, a further conflict scenario may be the available bandwidth at the connection from the RTEthernet switch to the cluster simulator. If the DUTs are using a lot of bandwidth, for example to send a raw video stream, the overall used bandwidth of all connected DUTs must be less than the available bandwidth on the connection from the cluster simulator to the RTEthernet switch. Otherwise a cluster simulation can not be accomplished properly and TT-messages cannot be sequenced by the RTEthernet switch.

If the switch has to sequence TT-messages it adds an additional delay to the TT-messages. The timing constraints of TT-messages cannot be observed by the cluster simulator anymore.

To allow an analysis of the traffic that is shared between the DUTs, a further route to the simulation platform has to be configured to the switch. This can only be applied if enough bandwidth is available within the regular messages.

Moreover, performance issues are likely to happen more often, if more then one DUT is connected. Like in the previously presented cluster simulation classification, simplifications have to be applied. Furthermore, a multi-core, multi-process architecture (Fig. 2.6b on the preceding page) can be used to improve the simulation capabilities.

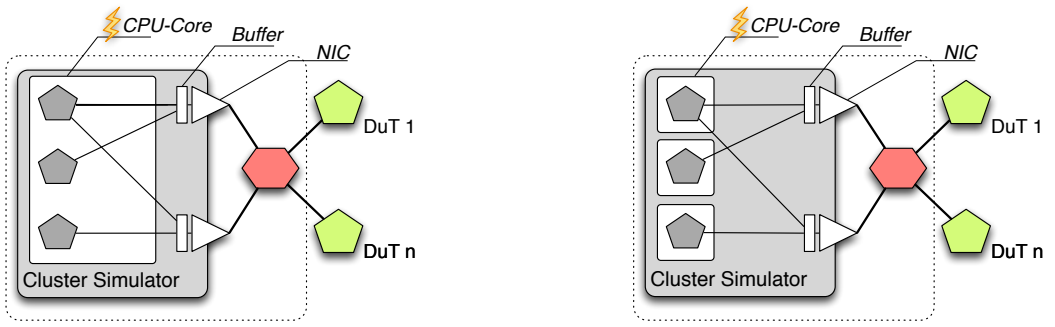
This setup can be used during the unit testing or integration phase, depending on the amount of connected DUTs.

Advantages This setup provides the option to test multiple devices by using one NIC. If only one DUT is connected to the switch, the configuration and hardware cost for the actual cluster simulator platform are rather small. The scheduling of the to be simulated messages can be taken directly from the DUTs. Since the precision of the additional switch is used, the cluster simulators hardware can be built upon off-the-shelf components, that run a RTEthernet stack.

Disadvantages This setup requires an additional RTEthernet switch, which results in further hardware costs. Moreover, the disadvantage of this setup outcrops when multiple devices are tested. The bottleneck is the available bandwidth at connection between the cluster simulator and the additional RTEthernet switch. Further message scheduling and bandwidth conflicts have to be resolved or at least analyzed. Additionally, an analysis of the shared traffic can not be easily applied in this approach.

Multiple NICs for Multiple DUTs

The next possible setup alternative is depicted in Figure 2.7. In this setup, the cluster simulator has multiple interfaces (one interface per DUT), to establish a connection to the RTEthernet switch and can be seen as a combination of the alternatives 'Multiple NICs for Multiple DUTs' on page 8 and 'One NIC for One or Multiple DUTs' on page 13.



(a) *Multiple network interfaces and one CPU core*

(b) *Multiple network interfaces and multiple CPU core*

Figure 2.7: *Hardware Setup with additional RTEthernet switch and multiple network interfaces*

This setup solves the bandwidth and scheduling conflict scenarios from the previously presented approach, imposed by the bottleneck behavior. Since in this approach one NIC per DUT is used, the switch does not need to integrate several connections to a single connection. Like in the approach 'Multiple NICs for Multiple DUTs' on page 8, the messages have to be assigned to dedicated NICs in order to trigger the correct DUT.

Traffic analysis that is shared between the DUTs is still difficult, since additional messages routes have to be configured. In contrast to the previous approach, these routes does not impose bandwidth conflicts, since the simulator platform is composed of one NIC per DUT. Nevertheless, the collected traffic from different NICs has to be merged, in order to get an overall consistent view.

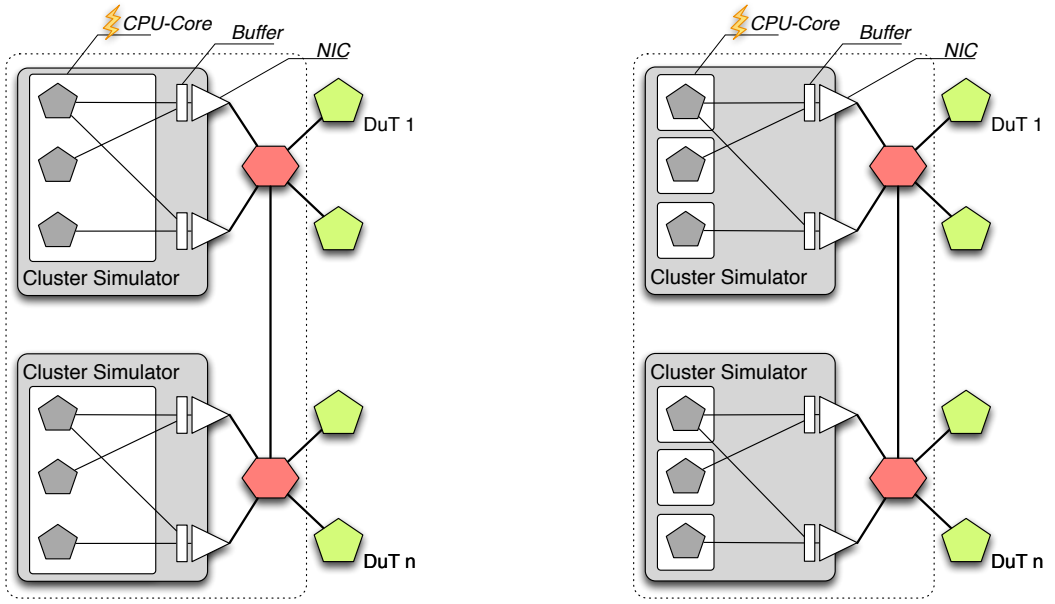
The use-case domain is the integration phase, since more than one DUT can be connected.

Advantages This setup solves the message scheduling and bandwidth conflicts. Furthermore, shared traffic between the DUTs does not have to be traversed at the cluster simulator. This results in an overall better simulation performance, since no integration of data paths have to be applied and the simulation platform does not have to deal with the connection between the DUTs.

Disadvantages In contrast to Multiple NICs for Multiple DUTs, an additional RTEthernet switch is required and the implementation costs rise with the number of DUTs.

Distributed Simulator for Multiple DUTs

A distributed simulator approach is also possible for alternatives with additional RTEthernet switches and is depicted in 2.8. The simulator is distributed on several sub simulation platforms, that are interconnected via additional RTEthernet switches and multiple connected DUTs.



(a) *Distributed approach with one CPU core*

(b) *Distributed approach with multiple CPU cores*

Figure 2.8: *Hardware setup for a distributed approach with a additional RTEthernet switch*

Like alternative ‘Distributed Simulator for Multiple DUTs’ on page 10, this setup is a special case from the previously presented approaches, where the simulation sub systems have to be synchronized. Also an overall traffic analysis is hard to achieve, due to the fact that each subsystem is likely to collect messages in its own subsystem. The captured traffic has to be

merged from all subsystems to get an system wide consistent view. Again, the corresponding utilization domain is the integration phase.

Advantages If a subsystems consists of multiple network interfaces (one per DUT), this setup has no bandwidth or timing restrictions.

Disadvantages On the other hand, such an alternative setup has the highest hardware and implementation costs, which makes this approach under economical sight rather infeasible.

2.2.3 Discussion of RTEthernet Cluster Simulation Hardware Setups

The last section has presented possible hardware setups for RTEthernet based networks in detail. In dependency of the number of to be tested devices and the characteristics of the alternatives, some setups become irrelevant. This section will conclude and determine the relevant setup topologies. If only one device shall be tested, ‘One NIC for One DUT’ on page 7 is economically the best opportunity to perform unit testing, since no further hardware is needed. Performance issues are likely not to happen, since only serial data transmission occurs. An abstraction from the real behavior of a simulated node has to be applied to simplify the cluster simulation, if the simulation model is too complex to be executed properly.

On the other side, if a system is to be integrated, setups that provides a direct connection to the DUTs are not suitable, because of their resulting timing conflicts, when communication between the DUTs occurs. The hand over of messages from one interface to another can influence the simulation performance and makes a real-time behavior impossible. Therefore, alternative ‘One NIC for One or Multiple DUTs’ can be utilized, if no bandwidth and scheduling conflicts appear. The conflicts can be resolved, by utilizing one network interface per DUT (presented in alternative ‘Multiple NICs for Multiple DUTs’). If simulation performance is a problem, the utilization of a multi-process, multi-core architecture is possible. Distributed alternatives (‘Distributed Simulator for Multiple DUTs’ and ‘Distributed Simulator for Multiple DUTs’) provides additional problems and should not be applied. It is better to use a RTEthernet switch with several ports.

Hence, a cluster simulation should be implemented to achieve the best trade-off between simulation performance in hardware and implementation cost. Therefore an abstract model is needed to calculated, whether a selected topology is suitable in dependency of the DUTs. The next chapter presents the a formal model of a message flow system and according functions to enable such a calculation.

3 Formal Description of Messages in Cluster Simulation

This chapter deals with a formal representation of messages inside a RTEthernet network in order to calculate required attributes of a cluster simulation in dependency of the number of DUTs. The base elements are taken from the FIBEX4TTEthernet [16] extension, which is used to describe a complete RTEthernet network as a XML document. In the beginning, first fundamental definitions are clarified. Afterwards, set definitions to model a message-flow-system of RTEthernet networks and according functions to calculate the required attributes are presented. Finally, limitations to the model are described.

3.1 Fundamental Definitions

Definition 3.1.1 (*Definition of Natural Numbers*)

In this work, $\mathbb{N}$ *is defined as followed:*

$$\mathbb{N} = \mathbb{N}^{+} = \mathbb{N}_0 \backslash \{0\} = \{1, 2, 3, \ldots\}$$

Definition 3.1.2 (*Tuple Selector*)

Let $tuple$ be a n-tuple:

$$tuple = (a_1, a_2, \ldots a_n) \in TUPLE$$
$$TUPLE \in (M_1 \times M_2 \times \ldots M_n),\ n \in \mathbb{N}$$

The access of an element of the n-tuple is defined via selector functions

$$tuple.a_1 : TUPLE \rightarrow M_1$$
$$tuple.a_1(a_1, a_2, \ldots a_n) = a_1$$
$$tuple.a_2 : TUPLE \rightarrow M_2$$
$$tuple.a_2(a_1, a_2, \ldots a_n) = a_2$$
$$\vdots$$
$$tuple.a_n : TUPLE \rightarrow M_n$$
$$tuple.a_n(a_1, a_2, \ldots a_n) = a_n$$

3.2 Set Definitions of a Message Flow System

3.2.1 Base Elements

Definition 3.2.1 (*ECU*)

*Electronic control units **ECU** is a finite not empty set.*

$$ECU = \{ecu_1, ecu_2, \ldots ecu_n\} \neq \emptyset, n \in \mathbb{N}$$

Electronic control units can be sensors, actuators or controllers in a distributed real-time system.

Definition 3.2.2 (*CE*)

*Coupling elements **CE** is a finite not empty set.*

$$CE = \{ce_1, ce_2, \ldots ce_n\} \neq \emptyset, n \in \mathbb{N}$$

In the case of RTEthernet, coupling elements are switches that interconnect (couple) ECUs or other coupling elements (switches).

Definition 3.2.3 (*DEV*)

*Devices **DEV** is a finite not empty set.*

$$DEV = ECU \cup CE \text{ , where}$$
$$ECU \cap CE = \emptyset$$

Devices in RTEthernet networks are either ECUs or switches.

Definition 3.2.4 (*PORT*)

***PORT** is a finite not empty set.*

$$PORT = \{port_1, port_2, \dots port_n\} \neq \emptyset, n \in \mathbb{N}$$

A port defines a physical element (network interface) that establishes a connection to the RTEthernet network.

Definition 3.2.5 (*FRAME*)

Let id be the identifier of a frame, then $Id_{be} \subseteq \mathbb{N}$ is defined as best effort, $Id_{rc} \subseteq \mathbb{N}$ as rate constrained, $Id_{tt} \subseteq \mathbb{N}$ as time triggered id, where:

- $Id_{be} \cap Id_{rc} = \emptyset$
- $Id_{be} \cap Id_{tt} = \emptyset$
- $Id_{tt} \cap Id_{rc} = \emptyset$

*A **frame** $frame$ is a triple.*

$$frame = (id, len, payload) \in ((Id_{be} \cup Id_{rc} \cup Id_{tt}) \times \mathbb{N} \times PAYLOAD)$$
$$PAYLOAD = \{(x_1, x_2, \dots x_n) \mid x_i \in \{0,1\} \text{ , } n \in \mathbb{N}\} \text{ , where:}$$

- id *is the identifier,*
- len *the length in bytes and*
- $payload$ *the payload of the frame.*

*A frame is the container in which information is transmitted through the network. **FRAME** is the corresponding finite not empty set.*

$$FRAME \in \mathcal{P}((Id_{be} \cup Id_{rc} \cup Id_{tt}) \times \mathbb{N} \times PAYLOAD) \backslash \emptyset$$

3.2.2 Composed Elements

Definition 3.2.6 (*DEVICE PORT*)

*A **device port** dp is a tuple that assigns a port to a device.*

$$dp = (port, dev) \in (PORT \times DEV) \text{ where :}$$

- $port$ *is the physical port*
- dev *is the dedicated device*

***DEVPORT** is the corresponding finite not empty set.*

$$DEVPORT \in \mathcal{P}(PORT \times DEV) \backslash \emptyset$$

A device port is the logical network port of a device that establishes a connection to the network.

Definition 3.2.7 (*BEMESSAGE*)

*A **best effort message** m_{be} is a triple*

$$\begin{aligned} m_{be} =& (beSenderPort, BeReceiverPorts, frame) \\ & \in (DEVPORT \times \mathcal{P}(DEVPORT) \backslash \emptyset \times FRAME) \text{ where:} \end{aligned}$$

- $beSenderPort$ *is the sending device port of the,*
- $BeReceiverPorts$ *the set of receiving device ports,*
- $frame$ *is the frame of a message*

and where:

- $BeSenderPort \notin ReceiverPorts$
- $frame.id \in Id_{be}$

A best effort message is used to transmit non-critical messages in a real time Ethernet network. ***MESSAGE$_{BE}$** is the corresponding finite not empty set.*

$$MESSAGE_{BE} = \mathcal{P}(DEVPORT \times \mathcal{P}(DEVPORT) \backslash \emptyset \times FRAME) \backslash \emptyset$$

Definition 3.2.8 (*RCPORT*)

*A **rate-constrained device port** $rcdp$ is a triple.*

$$rcdp = (port, dev, bag) \in (PORT \times DEV \times \mathbb{N}) \text{ where:}$$

- $port$ *is the physical port*
- dev *is the dedicated device*
- bag *is the corresponding bandwidth allocation gap (BAG) according to the AFDX [17] protocol in nanoseconds. A BAG defined as zero is not allowed according the AFDX protocol, since this would result in a constant message transmission.*

*A rate-constrained device port extends the device port to allow defining timing constraints at ports. Therefor, a $rcdp$ assigns a time value to the port. **RCDEVPORT** is the corresponding finite not empty set.*

$$RCDEVPORT \in \mathcal{P}(DEVPORT \times \mathbb{N}) \backslash \emptyset$$

Definition 3.2.9 (*RCMESSAGE*)

*A **rate-constrained message** m_{rc} is a triple.*

$$m_{rc} = (rcSenderPort, RcReceiverPorts, frame) \in (RCDEVPORT \times \mathcal{P}(RCDEVPORT)\backslash\emptyset \times FRAME)$$ *where:*

- $rcSenderPort$ *is the sending device port,*
- $RcReceiverPorts$ *the set of receiving device ports,*
- $frame$ *is the frame of a message*

and where:

- $rcSenderPort \notin RcReceiverPorts$
- $frame.id \in Id_{rc}$
- $\forall\, rcRecPort \in RcReceiversPorts : rcRecPort.bag = rcSenderPort.bag$

Rate constraint messages are used to transmit event based time critical messages in a real-time Ethernet network. ***MESSAGE$_{RC}$*** *is the corresponding finite not empty set.*

$$MESSAGE_{RC} \in \mathcal{P}(RCDEVPORT \times \mathcal{P}(RCDEVPORT)\backslash\emptyset \times FRAME)\backslash\emptyset$$

Definition 3.2.10 (*TTPORT*)

*A **time-triggered device port** $ttdp$ is a triple.*

$$ttdp = (port, dev, actionPit) \in (PORT \times DEV \times \mathbb{N}_0)$$ *where:*

- $port$ *is the physical port*
- dev *is the dedicated device*
- $actionPit$ *is the absolute point in time in which the action (send / receive) will be triggered.*

A time-triggered device port extends the device port to allow defining an absolute timing constraint for an action at the according port. Therefor $ttdp$ assigns a time value to the port. ***TTDEVPORT*** *is the corresponding finite not empty set.*

$$TTDEVPORT \in \mathcal{P}(DEVPORT \times \mathbb{N}_0)\backslash\emptyset$$

Definition 3.2.11 (*TTMESSAGE*)

*A **time-triggered message** m_{tt} is a quadruple*

$$m_{tt} = (ttSenderPort, TtReceiverPorts, frame, period) \\ \in (TTDEVPORT \times \mathcal{P}(TTDEVPORT)\backslash\emptyset \times FRAME \times \mathbb{N})$$ *where:*

- $ttSenderPort$ *is the sending device port*
- $TtReceiverPorts$ *the set of receiving device ports*
- $frame$ *is the frame of a message*
- $period$ *is the period in which the message transmission is repeated*

and where:

- $ttSenderPort \notin TtReceiverPorts$
- $frame.id \in Id_{tt}$
- $\forall\ ttRecPort \in TtReceiverPorts : ttRecPort.actionPit \gg ttSenderPort.actionPit$

Time triggered messages are used to transmit periodically sent time critical messages in a real-time Ethernet network. ***MESSAGE$_{TT}$*** *is the corresponding finite not empty set.*

$$MESSAGE_{TT} \in \mathcal{P}(TTDEVPORT \times \mathcal{P}(TTDEVPORT)\backslash\emptyset \times FRAME)\backslash\emptyset$$

Definition 3.2.12 (*MESSAGES*)

*$\mathcal{M}$ is the set of all **messages***

$$\mathcal{M} = (MESSAGE_{BE} \cup MESSAGE_{RC} \cup MESSAGE_{TT})$$

where:

- $(MESSAGE_{BE} \cap MESSAGE_{RC}) = \emptyset$
- $(MESSAGE_{BE} \cap MESSAGE_{TT}) = \emptyset$
- $(MESSAGE_{RC} \cap MESSAGE_{TT}) = \emptyset$

These different messages allows a RTEthernetwork defining a mixed critical system.

Definition 3.2.13 (*Message Flow System*)

*A **message flow system** $messageFlow$ is a tuple that represents the message flow of a mixed critical RTEthernet network system.*

$$mfsys = (Dev, Messages, clustercycle) \in (\mathcal{P}(DEV) \times \mathcal{P}(\mathcal{M}) \times \mathbb{N}) \text{ where:}$$

- Dev *is the set of devices,*
- $Messages$ *the set of Messages in the network and*
- $clustercylce$ *describes the cycle of the cluster for the time-triggered traffic*

***MFSYS** is the corresponding finite not empty set.*

$$MFSYS \in \mathcal{P}(\mathcal{P}(DEV) \times \mathcal{P}(\mathcal{M}) \times \mathbb{N}) \backslash \emptyset$$

A message flow system is dependent on the overall participating devices in the network, the messages that are transmitted and the cycle in which the time-triggered message scheduling will repeat. The cluster cycle must be the least common multiple of all periods of the messages. Otherwise no suitable schedule can be calculated.

3.3 Definitions of Functions

This section will present functions that allow calculating the cluster simulation attributes, in order to find the best matching RTEthernet cluster simulation topology in dependency of the number of devices under test.

Definition 3.3.1 (*Transmitted Messages from Device under Test*)

Let $mfsys \in MFSYS$ a valid message flow system
and $dut \in mfsys.Dev$ then
$messages_{transmitted} : DEV \rightarrow \mathcal{P}(\mathcal{M})$

$$\begin{aligned} messages_{transmitted}(dut) = \{m \mid m \in \mathcal{M} : \ & m.beSenderPort.dev = dut \\ & \vee m.rcSenderPort.dev = dut \\ & \vee m.ttSenderPort.dev = dut\} \end{aligned} \tag{3.1}$$

This function is used to calculate the messages that are transmitted by a device under test.

Definition 3.3.2 (*Received Messages by Device under Test*)

Let $mfsys \in MFSYS$ a valid message flow system
and $dut \in mfsys.Dev$ then
$messages_{received} : DEV \rightarrow \mathcal{P}(\mathcal{M})$

$$messages_{received}(dut) = \{m \mid m \in \mathcal{M} \wedge \exists\, recv \in (\ m.BeReceiverPorts \cup m.RcReceiverPorts \cup m.TtReceiverPorts) : recv.dev = dut\} \tag{3.2}$$

This function is used to calculate the messages that are received by a device under test.

Definition 3.3.3 (*All to be simulated messages by the cluster simulator*)

Let $mfsys \in MFSYS$ be a valid message flow system
and $DUT \subseteq mfsys.Dev$ then
$allSimMessagesByCS : \mathcal{P}(DEV)\backslash\emptyset \rightarrow \mathcal{P}(\mathcal{M})$

$$allSimMessages(DUT) = \bigcup_{dut \in DUT} messages_{received}(dut)\backslash ((\bigcup_{dut \in DUT} messages_{received}(dut)) \bigcap (\bigcup_{dut \in DUT} messages_{transmitted}(dut)) \tag{3.3}$$

This function calculates all messages that have to be imitated by the cluster simulator in dependence of the selected devices under test. Messages that are exchanged between the selected devices under test are not to be simulated.

Definition 3.3.4 (*All to be received messages by the cluster simulator*)

Let $mfsys \in MFSYS$ be a valid message flow system
and $DUT \subseteq mfsys.Dev$ then
$allRecMessagesByCS : \mathcal{P}(DEV)\backslash\emptyset \rightarrow \mathcal{P}(\mathcal{M})$

$$allRevMessages(DUT) = \bigcup_{dut \in DUT} messages_{transmitted}(dut)\backslash ((\bigcup_{dut \in DUT} messages_{received}(dut)) \bigcap (\bigcup_{dut \in DUT} messages_{transmitted}(dut)) \tag{3.4}$$

This function calculates all messages that have to be received by the cluster simulator in dependence of the selected devices under test. Again, messages that are exchanged between the selected devices under test are not to be received.

Definition 3.3.5 (*All messages located at the cluster simulator*)

Let $mfsys \in MFSYS$ *be a valid message flow system*
and $DUT \subseteq mfsys.Dev$ *then*
$allMessagesAtCS : \mathcal{P}(DEV)\backslash\emptyset \rightarrow \mathcal{P}(\mathcal{M})$

$$allMessagesAtCS(DUT) = allRevMessages(DUT) \cup allSimMessages(DUT) \tag{3.5}$$

This function calculated all messages that are either received or transmitted by the cluster simulator in dependence of the selected devices under test.

Definition 3.3.6 (*Message Bandwidth Consumption*)

Let $mfsys \in MFSYS$ *be a valid message flow system*
and $m \in mfsys.Messages \wedge m \in MESSAGE_{TT}$ *then*
$bandwidth_{TT} : MESSAGE_{TT} \rightarrow \mathbb{Q}$

$$bandwidth_{TT}(m) = \frac{m.frame.len + 20}{m.period} \tag{3.6}$$

or $m \in mfsys.Messages \wedge m \in MESSAGE_{RC}$ *then*
$bandwidth_{RC} : MESSAGE_{RC} \rightarrow \mathbb{Q}$

$$bandwidth_{RC}(m) = \frac{m.frame.len + 20}{m.rcSenderPort.bag} \tag{3.7}$$

These functions calculate the message bandwidth consumption on a direction. Hence, $bandwidth_{RC}$ *(equation 3.7) can only estimate the worst case bandwidth consumption, since it is not clear, whether message transmission occurs. Since the Ethernet's specific inter packet gap (IPG) and preamble, which is 12 respectively 8 Bytes [18], also affects the bandwidth consumption, these values have to be added to the fame size.*

Definition 3.3.7 (*Accumulated Critical Message Bandwidth Consumption of a DuT*)

Let $mfsys \in MFSYS$ *be a valid message flow system*
and $dut \in mfsys.Dev$ *then*
$bandwidth_{DuT} : DEV \rightarrow \mathbb{Q}$

$$bandwidth(dut) = \sum_{i \in I} \begin{cases} bandwidth_{TT}(m_i), & \text{if } m_i \in MESSAGE_{TT} \\ bandwidth_{RC}(m_i), & \text{if } m_i \in MESSAGE_{RC} \\ 0, & \text{else} \end{cases} \tag{3.8}$$

with $I = \{m \in messages_{transmitted}(dut)\}$

This function calculates the accumulated consumed worst case bandwidth of all critical messages send by the DUT. For best effort messages, no transmission period is defined which allows an estimation of the worst case bandwidth. Thus, the worst case bandwidth of best effort messages cannot be estimated.

Definition 3.3.8 (*Replication Rate of a TTMessage*)

Let $mfsys \in MFSYS$ *be a valid message flow system*
and $m \in mfsys.Messages \wedge m \in MESSAGE_{TT}$ *then*
$replRate : MESSAGE_{TT} \rightarrow \mathbb{N}$

$$replRate(m) = \frac{mfsys.clustercycle}{m.period} \tag{3.9}$$

The replication rate calculates how often a time-triggered message is transmitted during the cluster cycle.

Definition 3.3.9 (*Absolute Action Points in Time*)

Let $mfsys \in MFSYS$ *be a valid message flow system*
and $m \in mfsys.Messages \wedge m \in MESSAGE_{TT}$
and $ttport \in m.TtReceiverPorts$ *then*
$absActionPITs : TTDEVPORT \times MESSAGE_{TT} \rightarrow \mathcal{P}(\mathbb{N}_0 \times MESSAGES_{TT})$

$$absActionPITS(ttport, m) = \bigcup_{i=1}^{N} (ttport.actionPit * i, m) \ \in (\mathbb{N}_0 \times MESSAGES_{TT}) \tag{3.10}$$

with $N = replRate(m)$

This function calculates a set of action points to a corresponding message.

Definition 3.3.10 (*Action Points in Time of Transmitted TTMessages at a Device*)

Let $mfsys \in MFSYS$ *be a valid message flow system*
and $dut \in mfsys.Dev$ *then*
$actionPITofTransmittedTTMessages : DEV \rightarrow \mathcal{P}(\mathbb{N}_0 \times MESSAGE_{TT})$

$$actionPITsofTransmittedTTMessages(dut) = \bigcup_{m \in M} absActionPITS(m.ttSenderPort, m) \tag{3.11}$$

with $M = \{m \mid m \in mfsys.Messages \wedge m \in MESSAGE_{TT} : m.ttSenderPort.device = dut\}$

$actionPITsofTransmittedTTMessages$ *calculates the all action points of transmitted TTMessages at the given device under test.*

Definition 3.3.11 (*Action Points in Time of Received TTMessages at a Device*)

Let $mfsys \in MFSYS$ *be a valid message flow system*
and $dut \in mfsys.Dev$ *then*
$actionPITsofReceivedTTMessages : DEV \rightarrow \mathcal{P}(\mathbb{N}_0 \times MESSAGE_{TT})$

$$actionPITsofReceivedTTMessages(dut) = \bigcup_{m \in M} absActionPITS(ttRecvPort, m) \tag{3.12}$$

with $M = \{m \mid m \in mfsys.Messages \wedge m \in MESSAGE_{TT} \wedge \exists\, recvPort \in m.TtReceiverPorts : recvPort.device = dut\}$
and $ttRecvPort = \exists!\, ttRecvPort \in m.TtReceiverPorts : ttRecvPort = dut$

$actionPITofReceivedTTMessages$ *calculates the all action points of received TTMessages at the given device under test.*

Definition 3.3.12 (*Transmitted TTMessages with the same Action Points by selected DUTs*)

Let $mfsys \in MFSYS$ *be a valid message flow system*
and $DUT \subseteq mfsys.Dev$ *then*
$messagesWithSameTransmissionActionPIT : \mathcal{P}(DEV)\backslash\emptyset \rightarrow \mathcal{P}(MESSSAGES_{TT})$

$$messagesWithSameTransmissionActionPIT(DUT) = \{m \mid \forall ap \in AP \wedge m \in ap.ttMessage : ap.actionPIT = ap.actionPIT'\} \tag{3.13}$$

with $AP = \bigcup_{dut \in DUT} \{(actionPIT, ttMessage) \in actionPITofTransmittedTTMessages(dut)\}$

$messagesWithSameActionPIT$ *calculates the messages, that have the same transmission action point in time at selected DUTs. Only if this this set is empty an, a cluster simulation can be performed without an adaptation of the schedule in the an RTEthernet switch, that relays the messages from the DUTs to the simulator.*

Definition 3.3.13 (*Received TTMessages with the same Action Points by selected DUTs*)

Let $mfsys \in MFSYS$ *be a valid message flow system*
and $DUT \subseteq mfsys.Dev$ *then*
$messagesWithSameReceiveActionPIT : \mathcal{P}(DEV)\backslash\emptyset \rightarrow \mathcal{P}(MESSSAGES_{TT})$

$$messagesWithSameReceiveActionPIT(DUT) = \{m \mid \forall ap \in AP \wedge m \in ap.ttMessage : ap.actionPIT = ap.actionPIT'\} \tag{3.14}$$

with $AP = \bigcup_{dut \in DUT} \{(actionPIT, ttMessage) \in actionPITofReceivedTTMessages(dut)\}$

$messagesWithSameReceiveActionPIT$ *calculates the messages, that have the same receive action point in time at selected DUTs. Only if this this set is empty, a cluster simulation can be performed without an adaptation of the schedule at the cluster simulator.*

3.4 Limitations of the Formal Model

The Fibex4TTEthernet model allows assigning dedicated offsets to periods of TTMessages, in order to refine the schedule of TTMessages. This is currently not realized in the formal model of this message-flow-system, since the focus was set towards cluster simulation attributes calculation capabilities and not to re-implement the RTEthernet protocol in another model.

4 Using the Framework for an Example Network Configuration

This Chapter will present results of the formal framework, implemented as a Java application, utilized on an example RTEthernet network cluster. This network is used to demonstrate RTEthernet structures, where critical and uncritical applications communicate over the same physical infrastructure. First, the network structure is presented and afterwards results from the framework.

4.1 Decription of the used example Network

The example network is taken from an existing RTEthernet application that is used to demonstrate an automotive application with different message-classes. The structure is depicted in Figure 4.1 on the following page and consists of two RTEthernet switches connecting an RTEthernet backbone and six different ECUs. Whereas five ECUs are located at the left switch and two ECUs on the right.

The ‘Controller’ is responsible to control a critical drive-by-wire application with according sensors and actuators represented by the ECUs ‘Wheel Bridge’ and ‘Steering Wheel Bridge’. Both bridges transmit status information and command confirmation back to the ‘Controller’ to enable a control-loop. The ‘Infotainment’ ECU controls the ‘Headlights’ and can modify the steering wheel attributes like force and torque. The ‘Headlights’ also communicate their current light status back to the ‘Infotainment’, which further streams uncritical video data to the ‘Rearseat Display’ via a second network interface. As devices-under-test the ‘Controller’, ‘Bridge Wheel’ and both ‘Headlights’ are selected, in order to allow closely realistic results.

These applications represents the typically existing message-classes inside a modern car infrastructure. Critical control-loop tasks, like a drive-by-wire application uses TT-messages which have the highest timing accuracy. Event-based critical tasks, like headlight are modeled within RC-messages. Multimedia streaming applications are modelled with BE-messages, since they are not allowed to influence critical tasks.

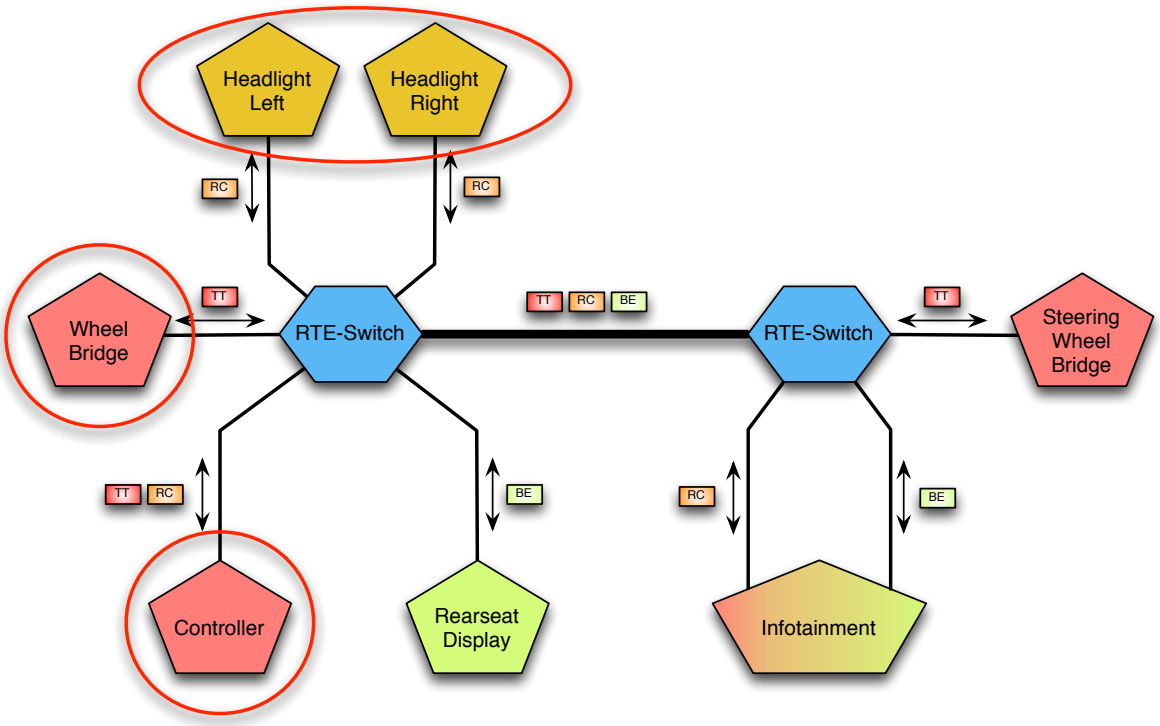


Figure 4.1: *The used example Network*

4.2 Framework Results

In this section, results from the message-flow framework are presented. First, the ‘Controller’ is selected as a single DUT. Afterwards, to check cluster simulation configurations with multiple devices, the ‘Controller’ and the ‘Bridge Wheel’ are chosen. As last use-case, both ‘Headlights’ represent the DUTs.

4.2.1 Using the Framework for the Controller as DUT

Messages at the Cluster Simulator

Table 4.1 depicts all received and transmitted messages by the cluster simulator. Received messages are generated by the DUT Controller and to be simulated messages are transmitted by the cluster simulator. In this example, the controller does not receive any BE-messages.

Received messages	To be simulated messages
RCMessage: FRAME-VL_0	RCMessage: FRAME-VL_281
RCMessage: FRAME-VL_602	RCMessage: FRAME-VL_582
RCMessage: FRAME-VL_603	RCMessage: FRAME-VL_583
RCMessage: FRAME-VL_604	RCMessage: FRAME-VL_584
RCMessage: FRAME-VL_PCF	TTMessage: FRAME-VL_182
TTMessage: FRAME-VL_200	TTMessage: FRAME-VL_183
TTMessage: FRAME-VL_201	TTMessage: FRAME-VL_210
TTMessage: FRAME-VL_20A	TTMessage: FRAME-VL_211
TTMessage: FRAME-VL_252	TTMessage: FRAME-VL_21A
TTMessage: FRAME-VL_80	

Table 4.1: *Received and to be simulated messages by the cluster simulator*

Worst case Bandwidth Accumulation

In this example the bandwidth allocation gap (BAG) is defined as 500 µs. All messages have a fixed size of 64 Byte, which results in the following worst case bandwidth of all accumulated to be received messages 6.182MBit/s.

Message Scheduling of Controller

Since only one device will be tested, the TT-message scheduling is sequential. No conflicts can be observed, since all action points in time appear only once.

Action PIT [µs]	Received Message	Transmitted Message
225	-	TTMessage: FRAME-VL_201
475	-	TTMessage: FRAME-VL_80
595	-	TTMessage: FRAME-VL_200
675	TTMessage: FRAME-VL_211	-
805	-	TTMessage: FRAME-VL_252
1060	TTMessage: FRAME-VL_210	-
1200	-	TTMessage: FRAME-VL_20A
1425	TTMessage: FRAME-VL_182	-
1465	TTMessage: FRAME-VL_183	-
1605	TTMessage: FRAME-VL_21A	-

Table 4.2: *Messages scheduling at the DUT*

4.2.2 Using the Framework for the Controller and Bridge Wheel as DUTs

Messages at the Cluster Simulator

In contrast to the first example with one DUT, the messages located at the cluster simulator are less, since most of the messages are exchanged only between the controller and the wheel.

Received messages	To be simulated messages
RCMessage: FRAME-VL_604	RCMessage: FRAME-VL_584
TTMessage: FRAME-VL_200	TTMessage: FRAME-VL_210
TTMessage: FRAME-VL_201	TTMessage: FRAME-VL_211
TTMessage: FRAME-VL_20A	TTMessage: FRAME-VL_21A

Table 4.3: *Received and to be simulated messages by the cluster simulator*

Worst case Bandwidth Accumulation

Since the cluster simulator receives less messages than in the previous example, also the worst case bandwidth consumption is less and is accumulated with 1.747MBit/s.

Message Scheduling of Controller and Bridge Wheel

In the example with two connected DUTs, the results show no scheduling conflicts.

Action PIT [µs]	Received Message	Transmitted Message
225	-	TTMessage: FRAME-VL_201
595	-	TTMessage: FRAME-VL_200
675	TTMessage: FRAME-VL_211	-
1060	TTMessage: FRAME-VL_210	-
1200	-	TTMessage: FRAME-VL_20A
1605	TTMessage: FRAME-VL_21A	-

Table 4.4: *Message scheduling of the DUTs*

4.2.3 Using the Framework for the Headlight Controllers

Messages at the Cluster Simulator

Received messages	To be simulated messages
RCMessage: FRAME-VL_410	RCMessage: FRAME-VL_400
RCMessage: FRAME-VL_411	RCMessage: FRAME-VL_401
	RCMessage: FRAME-VL_PCF

Table 4.5: *Received and to be simulated messages by the cluster simulator*

Worst case Bandwidth Accumulation

The worst case bandwidth for these two DUTs is 2.688MBit/s.

Message Scheduling of the Headlightcontrollers

The headlight controllers are triggered with RC-messages and on the other side, they transmit RC-messages only. As a result, no scheduling conflicts can occur, as long as the messages stick to the BAG.

4.2.4 Discussion of the Applied Model

TT-messages in this network are scheduled sequentially. As a consequence, scheduling conflicts (TT-messages have to be transmitted, or received at the cluster simulator) are impossible. Furthermore, all critical frames have a fixed frame size of 64 Byte and the cluster cycle repetition is 5ms. TT-messages have the same period as the cluster cycle (5 ms) or a configured BAG with 500 µs which results in a relatively low bandwidth consumption. The repetition rate of BE-messages is not modeled within the presented formal framework, since it is not modeled within FIBEX, too.

As conclusion a cluster simulation can be properly performed for all described cases with alternative ‘One NIC for One or Multiple DUTs’ on page 13. For one device (the controller in this example) also alternative ‘One NIC for One DUT’ on page 7 is possible.

5 Conclusion & Outlook

This chapter concludes and gives and outlook about upcoming work.

RTEthernet is a switched network technology and provides a dedicated collision domain for each network participant, which allows a parallel transmission of independent messages at the same time. Therefor hardware setups and topologies for RTEthernet cluster simulation are different compared to setups and the topology for bus based networks. Nevertheless, the idea to use a multicore / multi-process approach can be adapted to gain more simulation performance. If the simulation model is too complex to run properly on the cluster simulator, it has to be simplified.

In general, RTEthernet cluster simulations can be distinguished in hardware setups with a direct connection between the simulator and the DUTs and in setups with a an additional RTEthernet switch. If more than one device is to be tested, setups with an additional switch are preferred. In setups with a direct connection, the shared traffic has to handed over at the cluster simulator, which results in a not deterministic timing behavior. However, setups with an additional switch can produce scheduling and bandwidth conflicts if only one network interface is used by the cluster simulator. If these conflicts cannot be resolved, the cluster simulator must be composed of multiple network interfaces.

To calculate whether there exist any conflicts, a formal model basing on set definitions was introduced. This model is independent of the used network configuration file (i. e. FIBEX or TTEthernet Network Description) as input. The formal model provides functions to calculate the consumed bandwidth or scheduling conflicts in dependency of selected DUTs. The formal model was implemented as a Java application and applied on an example RTEthernet network, which is used to demonstrate future in-car communication. The results show that a cluster simulation with one network interface and an additional switch can be properly performed.

Since this network was rather simple and did not provide any concurrent transmission of TT-messages the formal model has to be applied to a more complex network scenario. Furthermore, the formal model can be used to generate configuration files for a cluster simulation.

Bibliography

[1] W. Steiner and G. Bauer, “Mixed-criticality networks for adaptive systems,” in *2010 IEEE/AIAA 29th Digital Avionics Systems Conference (DASC)*, Oct. 2010, pp. 5.A.3–1–5.A.3–10.

[2] B. Müller-Rathgeber, M. Eichhorn, and H.-U. Michel, “A unified Car-IT Communication-Architecture: Network switch design guidelines,” in *IEEE International Conference on Vehicular Electronics and Safety*, Sep. 2008, pp. 16–21.

[3] J. Badstübner, “Kollaps im Bordnetz: Schluss mit Can, Lin und Flexray,” *KFZ-Betrieb*, no. 17, pp. 68–70, 2008.

[4] Robert Bosch GmbH, “Controller area network.” [Online]. Available: http://www.semiconductors.bosch.de/

[5] FlexRay Consortium, “Flexray,” Stuttgart. [Online]. Available: http://flexray.com/

[6] MOST Cooperation, “Media Oriented Systems Transport.” [Online]. Available: http://www.mostcooperation.com/

[7] T. Steinbach, F. Korf, and T. C. Schmidt, “Real-time Ethernet for Automotive Applications: A Solution for Future In-Car Networks,” in *2011 IEEE International Conference on Consumer Electronics - Berlin (ICCE-Berlin)*. Piscataway, New Jersey: IEEE Press, Sep. 2011, pp. 216–220.

[8] B. Müller-Rathgeber, M. Eichhorn, and H.-U. Michel, “A unified car-it communication-architecture: Design guidelines and prototypical implementation,” in *IEEE Intelligent Vehicles Symposium, 2008*, Jun. 2008, pp. 709–714.

[9] F. Reimann, A. Kern, C. Haubelt, T. Streichert, and J. Teich, “Echtzeitanalyse Ethernet-basierter E/E-Architekturen im Automobil,” in *GMM-Fachbericht - Automotive meets Electronics, (Automotive meets Electronics (AmE’10), Dortmund, Germany, Apr. 15-16, 2010)*. Berlin: VDE Verlag, 2010, pp. 9–14.

[10] E. Bringmann and A. Krämer, “Model-based testing of automotive systems,” in *2008 1st International Conference on Software Testing, Verification and Validation*, Apr. 2008, pp. 485–493.

[11] J. Schäuffele and T. Zurawka, *Automotive Software Engineering*. Wiesbaden: Vieweg und Teubner, 2010.

[12] F. Bartols, “Restbussimulation von Time-Triggered Ethernet in Automotive Anwendungen: Verwandte Arbeiten,” Aug. 2011, bericht.

[13] T. M. Galla, “Cluster Simulation in Time-Triggered Real-Time Systems,” Ph.D. dissertation, TU Wien, Wien, Dec. 1999.

[14] Hermann Kopetz AND Günther Bauer, “The Time-Triggered Architecture,” *Proceedings of the IEEE*, vol. 91, no. 1, pp. 112–126, Jan. 2003.

[15] W. Elmenreich and R. Ipp, “Introduction to TTP/C and TTP/A,” in *Proceedings of the Workshop on Time-Triggered and Real-Time Communication Systems*, Dec. 2003, pp. 1–9, eingeladen; Vortrag: Workshop on Time-Triggered and Real-Time Communication Systems, Manno, Switzerland; 2003-12-02.

[16] F. Bartols, “Cluster Simulation of Real-time Ethernet based Electronic Control Units in Context of Automotive Applications: Extending FIBEX for Real-time Ethernet,” Sep. 2012, bericht.

[17] Aeronautical Radio Incorporated, “Aircraft Data Network, Part 7, Avionics Full-Duplex Switched Ethernet Network,” ARINC, Standard ARINC Report 664P7-1, 2009.

[18] Institute of Electrical and Electronics Engineers, “IEEE 802.3: Carrier Sense Multiple Access with Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications,” IEEE, Standard IEEE 802.3-2008, Sep. 2008.