Simulation based Timing Analysis of FlexRay Communication at System Level

Stefan Buschmann, Till Steinbach, Franz Korf, Thomas C. Schmidt
HAW-Hamburg, Department Informatik
Berliner Tor 7, D-20099 Hamburg, Germany
{stefan.buschmann, till.steinbach, korf, schmidt}@informatik.haw-hamburg.de

ABSTRACT

In modern cars the communication infrastructure consists of different application specific bus systems that are interconnected with each other. Due to the growing complexity of the communication infrastructure, the corresponding timing analysis at system level is currently a hot topic in the automotive industry.

FlexRay is a state-of-the-art fieldbus for cars. While FlexRay simulations below system level are already established in automotive tool chains, FlexRay system level simulation is not yet common. This paper focuses on simulation-based timing analyses for FlexRay communication. Based on different scenarios the practical relevance is shown. The performance results promise that our simulation approach is a good building block for the simulation of heterogeneous communication consisting of several bus systems and technologies. An evaluation of the simulation results in comparison with the CANoe network simulator proves the conformance of the implementation with the FlexRay specification.

Categories and Subject Descriptors

I.6.5 [**Simulation and Modeling**]: Model Development; C.2.2 [**Computer-Communication Networks**]: Network Protocols—FlexRay; C.4 [**Performance of Systems**]: Performance attributes

General Terms

Design, Performance, Reliability

Keywords

FlexRay, System Level Simulation, Timing Analysis, OMNeT++

1. INTRODUCTION

The simulation of diverse fieldbus designs and configurations is a key challenge in automotive development. Currently most commercial and open source simulation solutions for fieldbuses such as FlexRay [2] or CAN [7] focus on the accurate simulation of the hardware and physical communication layer to allow for a precise analysis of various network metrics. These simulations are suitable when the focus of interest is only on one single communication technology. When a large system with several buses is analysed, a more abstract view is required to visualise the systems behaviour and timing with reasonable simulation effort. Especially when the system uses multiple communication technologies and domains that are interconnected, a careful analysis of the influences of gateways, that interchange messages beyond the borders of a technology or domain, is indispensable. New automotive architectures, such as Ethernet-based in-car backbones [10] consolidate state-of-the-art technologies and cannot be simulated in conventional simulation environments. Thus, conventional toolchains are inappropriate for the evaluation of next generation in-car network architectures.

The FlexRay protocol [2] is a time-triggered protocol. During a communication cycle the static segment provides deterministic bus access according to a *coordinated time division multiple access (TDMA)* scheme. Moreover the dynamic segment supports event-triggered strictly prioritised communication. This work contributes a simulation model for FlexRay based automotive fieldbuses that abstracts from implementation and hardware details. To the best of our knowledge it is the first open source FlexRay simulation model that focuses on data link layer (layer 2) according to the OSI model. Many layer 2 related configuration parameters defined in the FlexRay standard are used in the simulation model. This tight connection between simulation and hardware parameters realises a smooth transfer from the model to the implementation.

Our simulation model is designed to be compatible with other simulation models such as Ethernet-AVB [6] or TT-Ethernet [9] to simulate gateway and migration concepts. We prove the validity of the simulation results obtained from our model by comparing results of reference networks in OMNeT++ and CANoe.

The evaluation of the simulation model shows two typical applications: The simulation model can be used to detect protocol violations in the static segment of FlexRay during early design phases. These errors might be caused by protocol violations of applications or insufficient hardware, such as inaccurate oscillators. Furthermore, our simulation can be used to detect end-to-end delay problems in the dynamic segment, which might be caused by insufficient bandwidth or prioritisation problems.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
OMNeT Workshop 2013, March 05-07
Copyright © 2013 ICST 978-1-936968-76-3
DOI 10.4108/icst.simutools.2013.251724

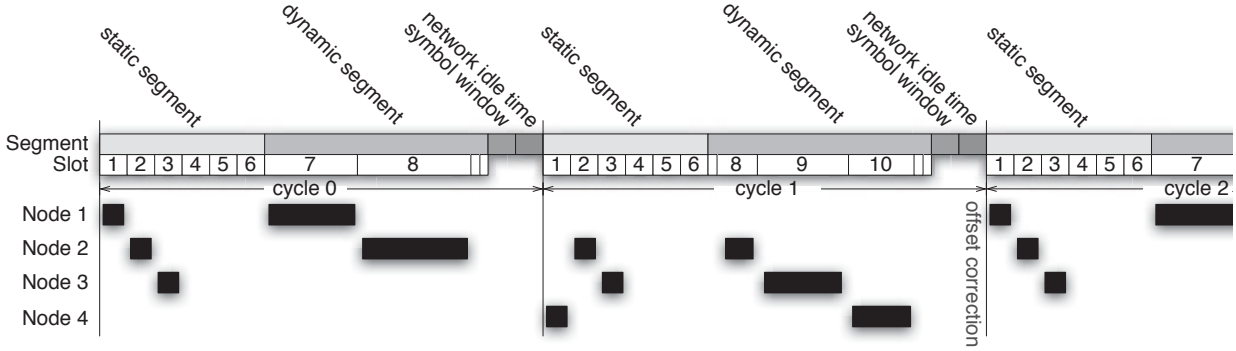

Figure 1: The FlexRay communication cycle

The remainder of the paper is organised as follows: In Section 2 the background on the FlexRay protocol and related work is given. Section 3 shows the concept behind the simulation model and its architecture. The evaluation of the simulation model and the discussion of the results of typical use-cases is shown in Section 4. Finally Section 5 concludes and gives an outlook.

2. BACKGROUND & RELATED WORK

2.1 FlexRay

FlexRay is a communication protocol designed as a dependable automotive bus. It offers higher bandwidth than preceding protocols such as CAN or LIN. It is suitable for the communication of safety critical applications with high timing requirements concerning latency and jitter. Furthermore, it offers two different methods to transmit frames: A *coordinated time division multiple access* strategy is used to send statically scheduled messages. Prioritised event-triggered communication is supported by a so-called mini-slot protocol. Due to the TDMA approach each node has its own clock that must be precisely synchronised. The local clocks are synchronised using offset- and rate-correction in specific time slots of the static segment.

2.1.1 Communication Cycle

The communication in a FlexRay system is divided into continuously repeated cycles. A cycle may consist of up to four different elements: the *static segment*, the *dynamic segment*, the *symbol window* and the *network idle time* (see Fig. 1). Static segment and network idle time are required in every cycle while dynamic segment and symbol window are optional. The duration of each element is configured before startup and cannot be changed during operation.

According to TDMA the static segment is divided into multiple slots. Each slot has the same fixed duration. A slot will be assigned to exactly one node. When the beginning of a slot is reached, the node transmits the corresponding frame.

In the dynamic segment a dynamic mini-slot-based scheme comes to use. Similar to the static segment, it is divided in multiple mini-slots, where each mini-slot can be assigned to exactly one node.

As soon as a mini-slot is reached, the node can transmit its dynamic frame. If the node transmit a frame, the mini-slot is extended according to the length of the frame. Otherwise the size of the mini-slot will is not changed. Only the end of the dynamic segment restricts the end of the transmission. If a node sends a frame and extends its mini-slot, the beginning of consecutive slots is delayed.

The symbol window offers the possibility to send signals, which are used within the startup of the network, for example to avoid collisions.

Within the network idle time there is no communication on the medium. During this time, nodes perform different tasks and calculate clock correction. The offset-correction is applied at the end of this segment.

2.1.2 Synchronisation

For synchronisation up to 15 nodes can be defined as redundant *sync nodes*. Such a node transmits a sync frame in each cycle, which is used for the clock synchronisation process. The local time of a FlexRay node is represented by cycles, macroticks and microticks. A macrotick consists of a number of microticks and a cycle of a number of macroticks. A microtick is standardised as 12.5 ns, 25 ns or 50 ns.

The rate correction of the local clock changes the number of microticks per macroticks. The offset correction of the local clock adjusts the duration of the network idle time.

For the calculation of both correction values a *fault-tolerant midpoint algorithm (FTM)* is used [2]. During a cycle, data concerning the current – and the expected – arrival time of each sync frame will be stored. Based on the number of entries up to two of the smallest and largest values will be discarded. Afterwards the largest and the smallest values are averaged and the floor function is applied.

The offset-correction value is calculated in every cycle but the clock offset-correction is only executed in odd cycles. For the calculation, the deviation values of the current cycle are processed and forwarded to the FTM algorithm. The result of the algorithm reflects the offset-correction value.

The rate-correction value is calculated in the odd cycles but it uses the values of two consecutive cycles. The differences between the values of both cycles is stored and forwarded to the FTM algorithm. The result is added to the previous rate-correction value. The value is valid for the next two cycles until the next rate-correction starts.

2.2 Related Work

Lauer et. al. [5] proposed discrete event based simulation, especially for design decisions at early stages of the development process. The authors argue that discrete simulation is

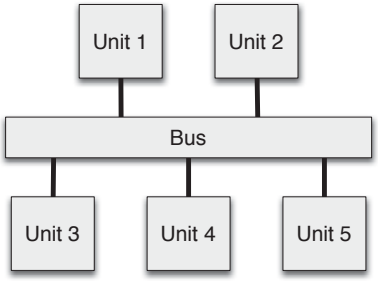

Figure 2: OMNeT++ Components of the FlexRay Simulation Model

in particular useful for sensitivity analyses from an average-case perspective. The average-case covers the aspects during normal system operation. The results of the timing analysis show the significant impact of clock drift on communication latency and jitter.

In general, discrete FlexRay models are rare: CANoe [12] is a well known commercial automotive network simulation environment provided by Vector Informatik GmbH. Besides the software simulation CANoe can also be applied for cluster simulation to test physically attached nodes.

There are examples for the simulation of fieldbus protocols in the OMNeT++ event-based simulation framework. Frauenhofer FOKUS introduced a FIELDBUS Framework for OMNeT++, but the model that was intended as basis for the simulation of ControlNet and DeviceNet never became stable [3].

Approaches like the ones given by Cheikhwafa et. al. [1] or Rolfs et. al. [8] use extensive abstraction to realise a performant simulation-based system analysis. The simulation model provided in this paper is based on layer 2 to allow for a more detailed timing analysis of the FlexRay protocol.

Graf et. al. introduced an abstract model for the analysis of automotive networks using virtual components [4]. Their concept bases on an actor concept with abstract simulation models at system-level. In an experimental setup it was shown how the simulation can be used to optimise parameters of the communication infrastructure to reduce round-trip delays.

3. CONCEPT, ARCHITECTURE & IMPLEMENTATION DETAILS

The proposed simulation model includes the communication within the static and the dynamic segment. The network idle time is used to calculate the correction values of the synchronisation process and to adjust the duration of the cycle in the simulation. At the start of a simulation it is assumed that all nodes of the network are already connected and synchronised with each other. Since the startup is not simulated the implementation of the symbol window is not required but is subject of future work.

3.1 Concept & Architecture

FlexRay Model

Each FlexRay node and the bus will be modeled as OMNeT++ module (see Figure 2). The bus module represents the physical structure of the FlexRay bus, such as two FlexRay channels. Bus access will be modelled by OMNeT++ messages. Static and dynamic frames will be sent from the transmitter node to the bus module. The bus module then sends the message to all other connected nodes.

The simulation implements time-triggered scheduling during the static segment of FlexRay. Each node is aware of the slot and corresponding frameID of all FlexRay frames that it was assigned to based on its configuration. When the slot is reached, the node sends a message with the simulation relevant information to the bus module. The bus module forwards the message to all nodes that are connected to the corresponding communication channel. Each FlexRay node contains checkers that report collisions, missing frames, and time violations.

In the dynamic segment the allocation of the slots is handled similar to the static segment. One exception is a node will not use its mini-slots during all cycles. If it is unused the bus stays idle during this slot. Utilised mini-slots are allowed to be extended over the configured duration. This results in a later transmission time of messages that will be sent later in the cycle (see Fig. 1). The additional delay is directly related to the variable size of the dynamic frame, that is allowed changing over multiple cycles.

Each FlexRay node has a local clock. To support the time-triggered approach of the static segment, these clocks must be synchronised according to the protocol. At the beginning of the network idle time of each odd cycle, the calculation of the offset and rate-correction values is performed. Depending on the result of the offset correction, the duration of the idle time is lengthened or shortened. The rate-correction will be used at the start of the next cycle to adjust the number of microticks per cycle.

Oscillator Model

A very important and at the same time very critical element of the simulation is the model of the oscillator. It is essential for the synchronisation process and has a huge influence on the simulation performance. With a precise model, it is possible to achieve a high accuracy in the simulation results even though the physical layer itself is not simulated.

The accuracy of oscillators is the most important attribute. Each of them has a certain inaccuracy called clock drift [11]. The drift of the clock may occur due to different reasons. For example, due to the physical architecture of the oscillator or environmental conditions like temperature changes. These drifts have a significant influence on the synchronisation process of the FlexRay nodes and must therefore be carefully included in the oscillator model.

To create a very accurate model of an oscillator it would be necessary to simulate each tick as a separate event. This approach would generate a huge amount of events and the simulation time would be slowed down significantly. Thus, it is not suitable to effectively simulate a network.

To provide a fast oscillator model, several ticks between two events are combined. To simulate the clock drift a constant drift value is used for a certain interval. The duration between the current time and the following events can be calculated as shown in equation 1:

$$t' = t + \delta * (\Delta t_{Tick} + \Delta t_{Drift}) \quad (1)$$

where t' represents the simulation time of an upcoming event. The time for the remaining ticks is added to the current simulation time t. It is calculated using the number of ticks (δ),

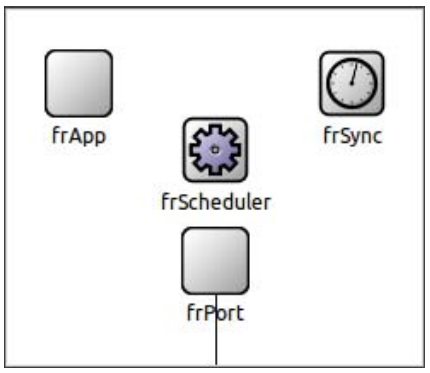

Figure 3: Structure of the FlexRay node

the fixed time for one tick (Δt_{Tick}) and the current drift value (Δt_{Drift}).

The same oscillator module can be used as well for simulations of other time-triggered protocols such as time-triggered Ethernet [9]. Based on a shared oscillator module we will then be able to simulate the complete communication matrix of one car or provide technology independent synchronisation services.

3.2 Implementation Details

The simulation consists of two main modules - one for the node and one for the bus topology. The node is divided in four different submodules according to their tasks: FRApp, FRScheduler, FRSync and FRPort (see Figure 3).

For traffic generators the *FRApp* submodule creates all frames that should be sent in the static or the dynamic segment and initialises the transmission. If simulation will be used for early application software tests, the application software will be plugged into FRApp.

The *FRScheduler* submodule implements the organisation of the communication cycle and the oscillator model. At the beginning of each cycle, the oscillator model provides the new clock drift for each node. Afterwards all already scheduled events, and the current value of the rate-correction that was calculated during the idle time of the previous cycle, have to be adjusted to the new drift.

$$gdMacrotick = microPerMacro * currentTick \qquad (2)$$

To apply the clock drift to future events the duration of the macrotick will be adjusted as given in equation 2. The variable $microPerMacro$ defines the number of microticks per macrotick. It depends on the configured values for the network and the current value of the rate-correction, which is added to the complete cycle. $currentTick$ represents the duration of one microtick including the actual drift.

The number of microticks per cycle that is adjusted with the rate-correction is distributed over all macroticks. This approach is different from the corresponding concept of the FlexRay-specification, where the microticks are added to or removed from several macroticks. Figure 4 shows an example where only one additional microtick is distributed over the cycle. This difference does not influence the timing behaviour on the data link layer, which is the focus of our simulation model. Our rate-correction model significantly reduces the complexity of the simulation.

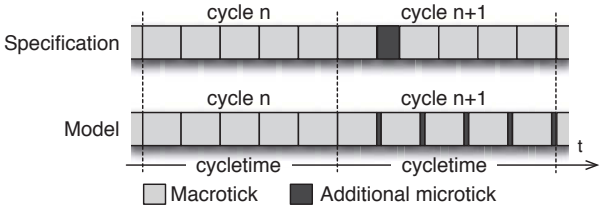

Figure 4: Distribution of an additional microtick

The *FRSync* submodule is responsible for clock synchronisation according to the FTM-algorithm described in Section 2.1.2. Additionally, it tracks the deviation of incoming sync frames and selects the values that are passed to the FTM. *FRFrame* is the message object for FlexRay frames. It stores FlexRay-specific parameters such as the *frameID* or the *syncFrameIndicator*. If the simulation is used for early application software tests, the payload is stored in this message object as well. Otherwise, the payload can be omitted to speed up the simulation.

4. EVALUATION & RESULTS

To validate the correctness of the simulation results, the OMNeT++ model is compared with the software tool CANoe and two typical use-cases are shown.

4.1 Comparison of Latency Results: OMNeT++ Simulation Model and CANoe

To evaluate the FlexRay simulation model we compared the results of a reference network in OMNeT++ and CANoe [12]. CANoe is a well-established commercial network simulator that is precisely configurable. We evaluated both, the static and the dynamic segment of the FlexRay communication cycle by comparing the CANoe traces with the OMNeT++ event log.

Our reference network consists of three nodes. All FlexRay parameters – such as the number of slots, the size of the static and dynamic segment, duration of micro- and macroticks – are equally configured in both setups. The cycle length is set to 5 ms, all nodes send in the static and the dynamic segment. Our FlexRay model offers less configuration options than CANoe, because of the higher abstraction. These additional parameters focus on layer 1 and 2 and influence the frame size or the transmission time.

To each node a slot in the static segment is assigned, in which it sends a static frame. Within the dynamic segment, node 1 has the highest priority, but sends a message only in every second cycle. Node 2 sends in the second slot in every cycle. The third node, which has the lowest priority, tries to send in the end of the cycle. To ensure that a comparison between CANoe and OMNeT++ is possible, all messages have the same size in of the dynamic segment.

For the comparison we turned off the clock drift in our FlexRay model as well as in the CANoe simulation. This ensures that we validate only the model of the FlexRay communication and not the clock model. As the model behind the clock in CANoe is not accessible for us, this is the only possibility to ensure comparable results.

The results of our simulation over 200 cycles show the same behaviour in CANoe and OMNeT++. All frames in the static segment are scheduled, while in the dynamic seg-

# nodes	# sync nodes	channel	sec_{Sim}/sec_{CPU}
10	10	single	∼0.96
20	15	single	∼0.58
30	15	single	∼0.45
10	10	dual	∼0.62
20	15	dual	∼0.32

Table 1: CPU-time for different network structures

ment the last frame is repressed when both previous nodes access the bus. When comparing the latency between CANoe and the FlexRay model we have observed a difference of approximatly 100 ns that is below the precision of the trace in CANoe. We attribute the time difference to the more detailed configuration options of CANoe.

The validation allows assuming correct simulation results of the model according to the abstraction level.

4.2 Performance Evaluation

To get an overview of the performance of the FlexRay simulation, the used CPU-time for different networks was analysed to compare them with each other. For the test we simulate three networks, which operate only on a single channel, and two networks, which send frames on both channels. To show the scalability of our simulation the dynamic segment is omitted. This is acceptable since the simulation time to transmit a message is in the same magnitude for both segments. All networks have a similar topology. In more detail: The amount of static slots is equal to the amount of nodes. Within the static segment every node sends one message. All other parameters such as the duration of a slot, a macrotick or the network idle time are equal.

Table 1 lists the results for all networks on a standard PC system. Each network ran for 60 seconds. The memory usage of each network was between 31 and 32 MB and is therefore not further considered.

The parameters *number of macroticks per slot*, *duration of a macrotick* and *number of unused slots* have a significant influence on the required CPU-time. In the performance evaluations given above we chose these values in such a way, that the required CPU-time is high. For example, in the simulation setup shown in Table 1 a slot takes 40 µs. If this value is changed to 200 µs for the last simulation, the proportion of sec_{Sim} to sec_{CPU} changes from ∼0.32 to ∼1.67 simulation seconds per CPU second.

The performance evaluation shows that the model offers sufficient performance for the simulation of typical automotive networks.

4.3 Timing Analyses in the Static Segment

One objective of the FlexRay simulation is to detect timing errors and incorrect behaviour caused by the network configuration or insufficient hardware. Typical detectable errors are:

- Violations of TDMA timing requirements (e.g. the transmission of a frame starts before the corresponding slot begins)
- Simultaneous transmissions in the same slot and channel
- Too many sync nodes
- Frames, received in the wrong slot

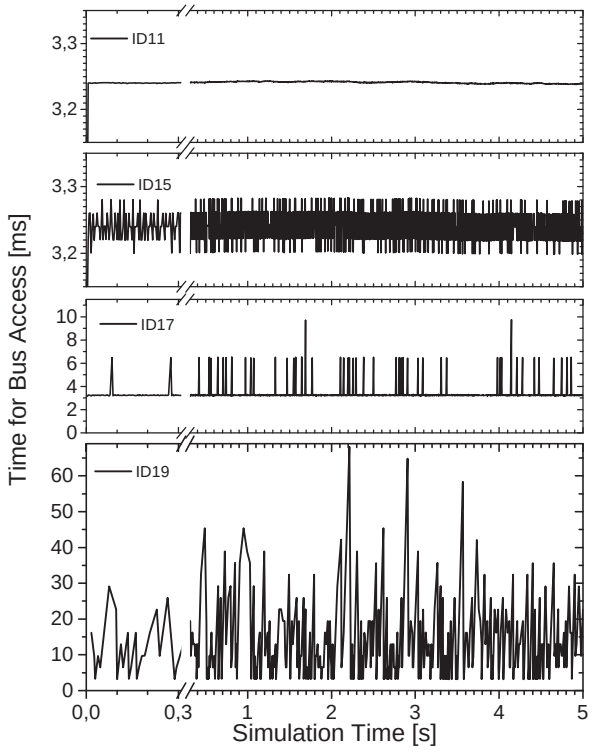

Figure 5: Latency of Frames with different IDs in the dynamic Segment over time

Network parameters such as the assignment of slots to messages and nodes have to be done in early design states. Erroneous configurations that are detected during bring up phase often require significant modifications. Thus, the simulation-based analysis of these error sources is of practical relevance.

Within the configuration of a FlexRay bus the *action point offset*, which is related to the beginning of a slot, defines the time window, in which a node may start the transmission of a frame. If the accuracy of the oscillator of a node is insufficient, the simulation detects these violations of TDMA requirements. Hence, our simulation approach can be used to adjust FlexRay configuration parameters such as the *action point offset*. Vice versa the accuracy of all oscillators can be validated against FlexRay configuration parameters in the early design phases, before the first prototype is available.

4.4 Latency Analysis of dynamic transmission

The latency analysis of the dynamic segment is demonstrated based on a small network. Four nodes send in a dynamic segment of 10 minislots. The corresponding IDs of the dynamic segment are 11 to 20, due to the 10 slots of the static segment. The four active nodes send their messages in the slots 11, 15, 17 and 19. Each node sends its messages with a randomised size such that the transmission will require 1 to 3 minislots. The cycle time is configured as 3.24 ms. The arrival time of each message type is recorded during the simulation of 3100 FlexRay cycles.

Figure 5 shows for each node the delay to get access to the bus. The optimal time between two accesses is 3.24 ms,

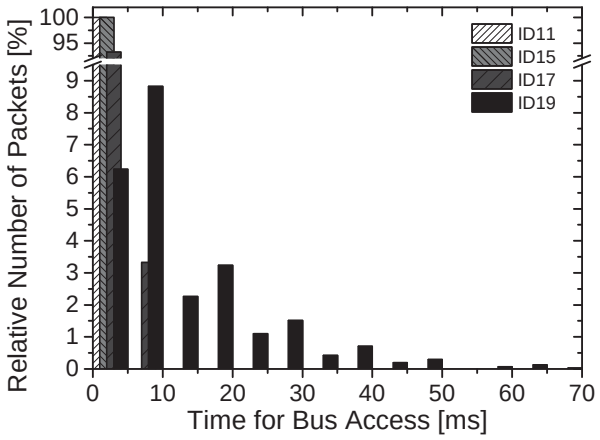

Figure 6: Latency Distribution of Frames with different IDs in the dynamic Segment

which is the cycle time. This can be guaranteed for the first two nodes. If node 3 and 4 – which are assigned to minislot 15 and 17 – try to transmit a message that requires 3 minislots during the same cycle, node 4 cannot transmit its message. The majority of messages of node 4 – which is assigned to ID 19 – have a high latency which implies that various messages were not sent. The node can never transfer a message that requires 3 minislots and it may take up to over 20 cycles to get bus access.

Figure 6 shows the interarrival time distribution. In over 92 % of the cycles the node assigned to the slot 17 was able to send its message. At the latest after approximately 10 ms the node got access. Just in a total of 6 % of the cases a frame with ID 19 was sent in two consecutive cycles.

5. CONCLUSION & OUTLOOK

This paper presents an open source FlexRay simulation model that focuses on data link layer (layer 2) according to the OSI model. It can be used for different timing analyses at system level as shown exemplarily for protocol violations and inadequate oscillators of local clocks.

Simulation gains importance due to the increasing complexity of modern car networks. The FlexRay simulation offers appropriate performance and can be used for large systems with a reasonable amount of time.

A modern car communication matrix consists of a number of different technologies, such as Ethernet, CAN and LIN, each with different properties. As shown, the FlexRay simulation is well prepared to run in parallel to CAN and real-time Ethernet simulation models. In future work we will merge these components. Based on such combined simulation environments and corresponding gateway designs, complex car matrices can be analysed. A simulation of an in-car network consists of 5 to 10 nearly independent subsystems that are coupled by gateways. This provides a good chance to reduce simulation time by parallel execution.

The implementation of the FlexRay model is currently prepared for OpenSource publication. A beta release can be found on the project webpage (`http://core.informatik.haw-hamburg.de`).

Acknowledgments

This work is funded by the Federal Ministry of Education and Research of Germany (BMBF) within the RECBAR project.

6. REFERENCES

[1] M. Cheikhwafa, S. Le Nours, O. Pasquier, and J. Calvez. Transaction level modeling of a FlexRay communication network. In *Forum on Specification Design Languages (FDL)*, pages 1–4, Piscataway, NJ, USA, Sept. 2009. IEEE Press.

[2] FlexRay Consortium. Protocol Specification. Specification 3.0.1, FlexRay Consortium, Stuttgart, Oct. 2010.

[3] Frauenhofer FOKUS. FIELDBUS Framework for OMNeT++, Sept. 2005.

[4] S. Graf, M. Streubühr, M. Glaß, and J. Teich. Analyzing Automotive Networks Using Virtual Prototypes. In *GMM-Fachbericht - Automotive meets Electronics, (Automotive meets Electronics (AmE'11), Dortmund, Germany, May. 04 .- 05., 2011)*, pages 9–14, Berlin, May 2011. VDE Verlag.

[5] C. Lauer, R. German, and J. Pollmer. Discrete Event Simulation and Analysis of Timing Problems in Automotive Embedded Systems. In *Proceedings of the 4th Annual IEEE Systems Conference, 2010*, pages 18–22, Piscataway, New Jersey, Apr. 2010. IEEE Press.

[6] H.-T. Lim, D. Herrscher, M. J. Waltl, and F. Chaari. Performance analysis of the ieee 802.1 ethernet audio/video bridging standard. In *Proceedings of the 5th International ICST Conference on Simulation Tools and Techniques*, pages 27–36, New York, Mar. 2012. ACM-DL.

[7] Robert Bosch GmbH. Controller area network.

[8] H. Rolfs, A. Liehr, and K. Buchenrieder. Communication Modeling for System-Level Performance-Simulation. In *Engineering of Computer Based Systems (ECBS), 2010*, pages 193–198, Piscataway, NJ, USA, Mar. 2010. IEEE Press.

[9] T. Steinbach, H. Dieumo Kenfack, F. Korf, and T. C. Schmidt. An Extension of the OMNeT++ INET Framework for Simulating Real-time Ethernet with High Accuracy. In *Proceedings of the 4th International ICST Conference on Simulation Tools and Techniques*, pages 375–382, New York, Mar. 2011. ACM-DL.

[10] 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)*, pages 216–220, Piscataway, New Jersey, Sept. 2011. IEEE Press.

[11] D. B. Sullivan, D. W. Allan, D. A. Howe, and F. L. Walls, editors. *Characterization of Clocks and Oscillators*. National Institute of Standards and Technology, Boulder, Colorado, 1990. technical note 1337.

[12] Vector Informatik. CANoe.FlexRay 8.0. Vektor Informatik.