Mode-Changes in COTS Time-Triggered Network Hardware without Online Reconfiguration

Florian Heilmann, Ali Syed and Gerhard Fohler
Chair of Real-time Systems
Technische Universität Kaiserslautern, Germany
{heilmann,syed,fohler}@eit.uni-kl.de

ABSTRACT

Time-triggered networks are widely used for safety-critical applications. Being offline scheduled, flexibility and adaptivity typically come at the price of very low resource utilization, if possible at all. In this paper, we present the Stacked Scheduling Approach (SSA) for time-triggered networks to enable mode changes and implicit adaptation in such networks by enabling reuse of network bandwidth reservations. We describe SSA in detail and conduct a case study to show that SSA can be implemented in COTS time-triggered network hardware and validate the approach by implementing an example in a COTS TTEthernet network.

Keywords

Time-Triggered Networks; Scheduling; Mode Changes

1. INTRODUCTION

Safety-critical applications require a high degree of reliability and predictability. In the network domain, time-triggered (TT) networks are widely used to satisfy these requirements. Safety-critical applications, however, may also require a high degree of flexibility and adaptivity to deal with changes in system state, environment or application requirements. This is a problem for TT networks, because they are scheduled offline. Flexibility and adaptivity in such networks can only be achieved either at the price of very low resource utilization, or cannot be achieved at all.

This limitation results in various approaches to augment TT network technologies to enable adaptive behavior.

These approaches increase the flexibility of COTS TT networks, however, they have drawbacks, which either manifest themselves in expensive online reconfigurations or the aforementioned inefficient use of already scarce network bandwidth. The integration of mode changes or adaptation mechanisms, which use network bandwidth efficiently and do not require online reconfiguration, into the specifications of existing TT networking technologies is infeasible. Such an integration would incur significant cost and design overheads since protocols and hardware would require redesign and recertification. To the best of our knowledge, no existing work enables adaptation and mode changes in existing TT network technologies without requiring online reconfiguration or sacrificing bandwidth.

In this paper, we present the *Stacked Scheduling Approach (SSA)* which makes use of the system specification to identify stackable messages (i.e. messages that are never ready to transmit at the same time) and stack them into the same scheduled TT transmission slot. This approach enhances existing TT networks with increased flexibility. We outline how SSA can be used to enable system wide mode changes as well as node-local adaptation. We conduct a case study showing that COTS TT networks can either execute schedules created with an SSA enabled scheduler directly, or require only minor changes. In addition, by reusing network bandwidth reservations with the help of SSA, more functionality can now be added without compromising network operation.

The remainder of this paper is structured as follows: Section 2 focuses on related work while Section 3 presents the system model and terminology used in this paper. Section 4 compares different mode change implementations. We present the Stacked Scheduling Approach in Section 5 and perform a case study on SSA in TTEthernet in Section 6 before concluding the paper in Section 7.

2. RELATED WORK

The state of the art explores several options to provide adaptation in TT networks. One possible approach is the inclusion of adaptation support into the network specification itself. Time-Triggered Networks like TTCAN [5] or FlexRay [3] provide limited flexibility to safety-critical systems by partitioning the network bandwidth into static and dynamic segments. While the dynamic segments do not assign network bandwidth to specific message and thus can provide a platform for adaptation, the response time of a message in this segment depends on whether it wins or (repeatedly) loses arbitration, which makes it difficult to provide response time guarantees [12]. Another example is the Time-Triggered Protocol (TTP) proposed by Kopetz et al. [9], which includes support for adaptation through mode changes in its specification [10]. The number of mode changes is, however, limited. Moreover, TTP is not able to service event-triggered (ET) traffic alongside TT traffic which may be desired by the system designer.

The adaptivity through mode changes allows systems to adapt to major changes in the environment [4] or system state, e.g. fault recovery. Kopetz et al. identified two different types of mode changes, which are triggered by a host node by firing a mode change request [10]. In deferred mode changes, all system nodes change mode after the request at the end of the hyperperiod. In immediate mode changes, only one system node changes operation mode soon after the request, i.e. before the end of the hyperperiod. The principle is consistency for the former and speed for the lat-

ter. Kopetz et al. [10] pointed out that an immediate mode change may lead to consistency problems for the applications and therefore a number of tasks or messages must be aborted but should not go in an undefined state. This notion of an (immediate) mode change requires careful application design and tight coupling between application and scheduler. However, Fohler asserted that consistency is required in both mode change types to keep schedulability during and after the mode changes [4]. His notion of a mode change decouples the application from the scheduler and hence is more suited for complex systems.

Another possible approach to provide adaptation enables mode change support using online reconfiguration by recomputing a suitable TT schedule on the fly. This approach is investigated by Klobedanz et al. for FlexRay [6][7][8], and Ashjaei et al. for FTT-Switched Ethernet [1]. The common issue with these approaches comes with the overhead incurred by reconfigurations. Online recomputation of TT schedules may take a significant amount of time, during which the network may or may not be available. Moreover these works focus on fault-tolerance or admission control specifically.

Craciunas and Serna Oliver investigated the synthesis of TTEthernet schedules [2]. Their implementation, however, leads to the assumption that mode changes can only be implemented by aggregating all messages from all modes into a single "super schedule", an approach similar to the one investigated by Klobedanz et al. where frame-packing is used to dynamically alter the contents of scheduled messages [7]. These super schedules allow the network to exhibit behavior akin to seamless online mode changes. Due to all messages of all modes being assigned their own slots in such super schedules, collisions cannot happen and reliable operation is ensured. However, this method wastes network bandwidth, as the super schedule reserves network bandwidth for all messages of all modes at any given time, which leaves the network bandwidth reserved for inactive modes idle. This approach limits the functionality that can be integrated into the network, since the available network bandwidth is quickly exhausted if all modes are aggregated into one schedule.

3. SYSTEM MODEL AND TERMINOLOGY

In this paper, a generic TT network Θ is modeled by the tuple

$$\langle \mu_\Theta, \mathcal{N}_\Theta, L_\Theta \rangle \tag{1}$$

where μ_Θ represents the time granularity of the TT network. For a given network type, the lower limit for the time granularity $\mu_{\Theta_{min}}$ is defined by the hardware. The application designer may select $\mu_\Theta > \mu_{\Theta_{min}}$, based on the application requirements. $\mathcal{N}_\Theta$ and L_Θ collectively represent the network topology. $\mathcal{N}_\Theta$ is a set of communication nodes, i.e. switches and processing nodes, and L_Θ is a set of communication links, i.e. the links between switch/processing nodes or a bus. A link $l \in L_\Theta$ is further defined by the tuple $\langle N_l, \upsilon_l \rangle$, where N_l defines the set of nodes connected by the link l while υ_l defines the link speed.

For switched networks, N_l is represented by an ordered pair $\langle n_a, n_b \rangle$, which defines a directed logical communication link connecting two nodes $n_a, n_b \in \mathcal{N}_\Theta$. The network is a full duplex network when $\forall \langle n_a, n_b \rangle \in l_i$, there exists $\langle n_b, n_a \rangle \in l_j$. Moreover, υ_l can vary for different links, i.e. multi-speed links. For a bus based network, $N_l = \mathcal{N}_\Theta$.

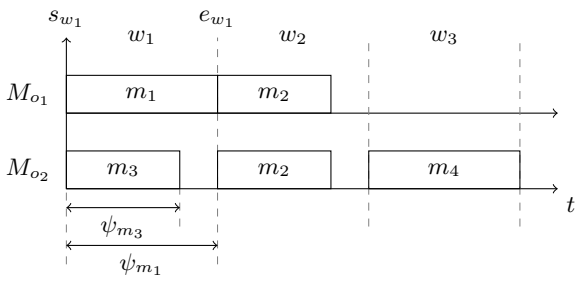


Figure 1: Example schedule for a link l

An application model is represented by a set of operation modes O. An operation mode o is defined by $\langle M_o, P_o \rangle$, where M_o is the set of messages of the mode and P_o is a set of routes for these messages through the links L_Θ in a switched network. For a bus based network P_o is an empty set $\varnothing$. A message $m \in M_o$ is represented by the tuple

$$\langle \psi_m, \gamma_m \rangle \tag{2}$$

where ψ_m is the message transmission time. Moreover, γ_m is defined by $\langle \gamma_s, \gamma_r \rangle$, where γ_s is the sender node and γ_r is the set of receiver nodes. For multi-speed networks, ψ_m is a function of $p_m \in P_o$, while for single speed networks, ψ_m is a constant.

We assume that the message deadlines are implicit (i.e. equal to the message period) and the message phase is constrained by the message period. Furthermore, we assume that $\forall m \in M_o$, the sender/receiver nodes γ_m and the routes $p_m \in P_o$ are known and fixed.

As we are focusing on TT messages, all activities are assumed to be triggered by the passage of time. The schedule of a link l is uniformly partitioned into a set of non-overlapping TT transmission slots W_l. A TT transmission slot $w \in W_l$ is defined by the tuple

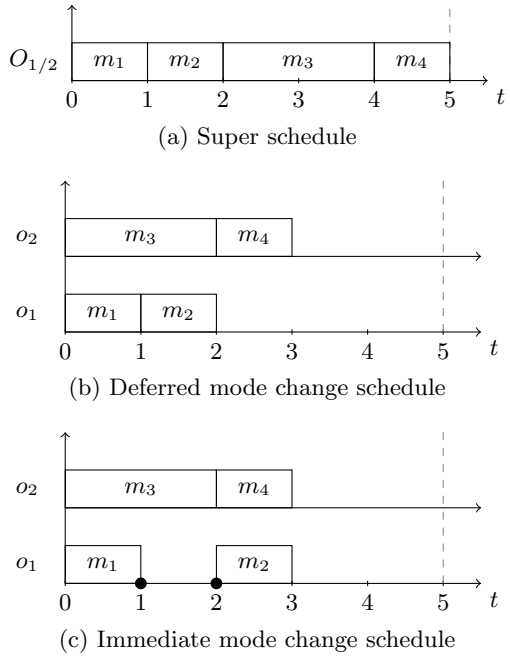
$$\langle s_w, e_w, M_w \rangle \tag{3}$$

where s_w is the start time of the slot, e_w is the end time and M_w is the set of messages scheduled in slot w. The parameters s_w and e_w are integral multiples of the granularity μ_Θ. Note that the parameters s_w, e_w and υ_l collectively define the reserved bandwidth for the messages M_w. The set of scheduling tables T_l is generated offline by the message scheduler and is defined by a set of W_l and the hyperperiod HP, i.e. the LCM of all message periods.

An example schedule for link l is shown in Figure 1. In the figure, w_i denotes the TT transmission slots, m_j denotes the messages and M_{o_k} denotes the sets of messages in mode o_k. The figure assumes that $M_{w_1} = \{m_1, m_3\}$.

4. COMPARISON OF MODE CHANGE IMPLEMENTATIONS

To compare the shortcomings and bottlenecks of different mode change implementations, scheduling tables for a link l are presented in Figure 2. The figure assumes mode o_1 with messages m_1 and m_2 and mode o_2 with messages m_3 and m_4. For all messages, the phase is 0 and the period is 5 units (i.e. the HP is 5 units). All messages have unit transmission time except m_3 which requires 2 units for transmission.



(a) Super schedule

(b) Deferred mode change schedule

(c) Immediate mode change schedule

Figure 2: Comparison of mode change implementations

The super schedule approach is presented in Figure 2a, where all messages are scheduled sequentially, irrespective of their operation mode. The advantage of this approach is its ease of implementation. However, using this approach, scheduling is hard for more than trivial loads. Moreover, this approach has very bad bandwidth utilization and does not support adaptation.

The scheduling tables for networks capable of handling mode changes can be seen in the Figure 2b and 2c. In this paper, this approach is referred to as individual schedule approach. This approach is easy to implement and to schedule. However, the scope of adaptivity in this approach is limited to operation modes, and implicit adaptation (see Section 5.4.2) is not possible without overprovisioning.

The dark circles on the time-line in Figure 2c represent mode change blackouts [4]. A mode change blackout defines that changing modes at this point in time will lead to either a message not being transmitted completely or a queued message being discarded. In COTS network controllers, mode change blackout information cannot be utilized, which further limits the scope of the individual schedule approach. Note that the mode change blackout is only required for immediate mode changes. In deferred mode changes, the mode change can only occur at the end of HP, which is always a safe point to switch for systems with implicit deadlines[10].

5. STACKED SCHEDULING APPROACH

5.1 Definitions

The description of SSA requires the definition of two terms. SSA makes use of the stackability of two or more messages to stack messages into TT transmission slots in the TT schedules of the network.

The definition for the stackability relation between two messages in a system is provided in Definition 1.

Definition 1. For a given TT transmission slot $w \in W_l$, two or more messages are termed stackable, if, during runtime, only one of the messages is ready to be transmitted during this transmission slot based on system or environmental state.

Examples for stackable messages include messages sent from a sender node to different receiver nodes based on different environmental/system state (XOR constraints as per Fohler [4]) or messages sent during different modes of operations. The definition of stacked TT transmission slot is provided in Definition 2.

Definition 2. A TT transmission slot $w \in W_l$ is termed a stacked TT transmission slot, if $|M_w| > 1$ and all messages $m_i \in M_w$ are stackable.

An example for a stacked TT transmission slot is provided in Figure 1 where w_1 is a stacked transmission slot with two messages, m_1 and m_3 assigned to it.

5.2 Methodology

SSA can be implemented as an extension to existing TT network schedulers.

During scheduling, the scheduler can assign the message it is currently scheduling to a transmission slot which already contains one or more messages if the resulting message set only contains stackable messages. A message cannot be assigned to a transmission slot that occurs before the message is ready to transmit. Without SSA, these messages would be assigned to separate transmission slots. SSA results in a single schedule, which utilizes network bandwidth more efficiently than a super schedule approach (see Figure 2a) by occupying less TT transmission slots and does not require online network reconfiguration since only a single schedule is used.

In order to use SSA with existing TT networks, only the following two conditions need to be satisfied during runtime:

Condition 1. Stackability constraints hold during system operation.

Condition 2. The network controller accepts schedules containing stacked transmission slots and services requests for all messages contained in stacked TT transmission slots during runtime.

Condition 1 can be satisfied by the system designer by, for example, using mode changes and making sure that all nodes switch modes simultaneously. Condition 2 relates to the capabilities of the COTS network controller. Due to the stacking of messages, different modes of operation and (implicit) adaptations are now possible.

5.3 Advantages

Using SSA in a TT network provides numerous advantages to the system designer. Compared to scheduling techniques without SSA, assuming stackable messages exist, less network bandwidth is now required, which allows to implement additional functionality or provide more bandwidth to existing functionality. SSA can provide the ability to add functionality to legacy systems. In such systems, the existing functionality is already scheduled. SSA can be used to create stacked TT transmission slots containing legacy messages and new messages if they are stackable. Finally, compared to other approaches such as super schedules, SSA can, in some cases, improve the response time of a message by moving the it to an earlier TT transmission slot containing only messages which are stackable.

5.4 Applications

5.4.1 Mode Changes

By definition, messages that belong to two modes of operations o_1 and o_2 are stackable. We exploit this property of the operation modes and employ SSA to support multiple operation modes in COTS TT network controllers that do not implement mode change functionality.

Deferred mode changes can be implemented using SSA without major modifications to the scheduler for systems with constrained deadlines. In addition to stacking the messages of the different modes, a message for mode change requests has to be added to the network. This message is used to exchange mode change requests among the nodes. If such a message is received, the processing nodes change modes at the end of the HP.

In order to implement immediate mode changes using SSA, the following conditions, corresponding to the argumentation in the Section 2, have to be satisfied in addition to Conditions 1 and 2 in Section 5.1.

Condition 3. Mode change blackout [4] information is provided by the scheduler.

Condition 4. Changing mode in a node and not in the others also leads to a valid operation mode.

Note that, in the case of an immediate mode change using SSA, a TT message for a mode change request is not required.

5.4.2 Implicit Adaptation

Mode changes allow the system to switch between states that influence a large portion of the system. However, some adaptations may be too minor to warrant a full system mode change because they are limited to a single node or a single application. These implicit adaptations focus on changes in the message parameters (see tuple 2 in Section 3).

With SSA system designers can implement small, implicit adaptations by defining multiple redundant messages with different parameter sets and stacking them into the same TT transmission slot. During run-time, the node will select the message with the parameter set that is best suited for the applications' current requirements. Hence, only one message from the stacked transmission slot is selected at any given time and the stackability condition holds (Condition 1 in Section 5.4.1).

Example: Electronic Stability Program (ESP).

We illustrate such an implicit adaptation by defining the following example:

A car manufacturer wants to develop a new electric car, with one motor used for acceleration as well as braking of each tire. The car is also equipped with an Electronic Stability Program (ESP), which helps the driver avoid over- or under-steering by selectively braking a single tire at a time. The network topology is depicted in Figure 3. The ESP application is running on a single ECU (n_{ESP}) built into the dash of the car. A switched Ethernet network, consisting of a single switch (n_{SW}) and five physical links (l_1 through l_5), is used to transmit messages from n_{ESP} to the nodes controlling the tire motors ($n_{FrontRight}$, $n_{FrontLeft}$, $n_{RearRight}$ and $n_{RearLeft}$). The transmission of messages is modeled through flows (virtual end-to-end communication channels), for which a schedule is generated offline and stored in all involved devices. We compare two approaches to schedule these flows without the use of SSA to an approach which utilizes SSA.

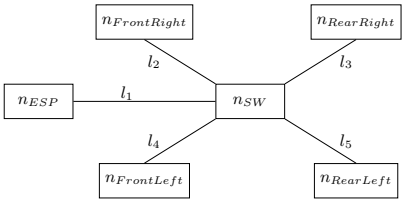


Figure 3: ESP Example Network Topology

Single Flow without SSA(SF-noSSA) A single multicast flow is used to carry the information for all 4 tire nodes. The advantage of this approach lies in the efficient usage of transmission slots, since only one transmission slot has to be used on all network links. However, this approach also introduces unnecessary network activity in all links from n_{SW} to nodes belonging to tires that do not need to brake. On these links, messages that carry no active information are transmitted. This "inactive load" consumes bandwidth that could be used, for example, by non-time-triggered traffic.

Individual Flows without SSA (IF-noSSA) Four individual unicast flows are used to carry the information for each respective tire node. n_{ESP} can now, depending on which tire has to brake, release a message which will only be routed to that specific tire node. As a result, no inactive load will be introduced to the links from n_{SW} to the tires that do not need braking. However the overhead in the schedule, especially on l_1 is significant for this approach. Four separate transmission slots have to be allocated to the messages of the ESP functionality, even though only one slot will have a message that is ready at any given time. Moreover, assuming that n_{ESP}, independent on which tire needs to brake, releases a message at the same time, some messages may have a larger response time than others because they have to wait longer for the transmission slot they are assigned to.

Individual Flows with SSA (IF-SSA) As with the previous approach, four individual unicast flows are used for each respective tire node. However, the four messages are now stacked into one transmission slot on l_1. As a result, one singular transmission slot on l_1 is utilized regardless of which tire has to brake and the response time for all four tires is identical. Finally, as in IF-noSSA, no inactive load is introduced on the links to the three tires which are not supposed to brake.

In this example, SSA allows to combine the efficient usage of schedule space of the SF-noSSA approach and flexibility of the IF-noSSA approach. While not significant, some minor overheads may be introduced in the nodes and switches by using SSA. Nodes have to select the appropriate message parameter set and switches need to service stacked TT transmission slots. As a result, the system in this example can now adapt to application requirements implicitly without requiring a mode change of the whole system.

5.5 Limitation: Overprovisioning

TT networks can suffer from overprovisioning. Overprovisioning results in network bandwidth that is reserved but not used, meaning that during run-time bandwidth may be available, but not usable by others because of reservations.

Even though SSA, compared to other approaches, frees up network bandwidth, there might be situations in which SSA does not lead to maximum bandwidth reuse. Based on the relation between the transmission slot size ($e_w - s_w$) and the message transmission time ψ_m, the following cases can lead to overprovisioning:

Case 1: Internal OP $(e_w - s_w) = \max_{m \in M_w}(\psi_m)$

This overprovisioning is due to the fact that a message always has to fit inside one TT transmission slot. As a result, the slots have to be sized according to the largest message of the set M_w. Consequently, if a smaller message is activated in the slot, the remainder of the network bandwidth in the slot is not utilized. An example for Case 1 is shown in Figure 1. Both m_1 and m_3 are stacked into the same TT transmission slot w_1. However, m_3 does not fully occupy w_1, leading to internal OP in that slot when m_3 is activated.

Case 2: Inevitable OP $(e_w - s_w) > \max_{m \in M_w}(\psi_m)$

Similar to Case 1, due to limitations in the system software or the TT network hardware, it may be impossible to choose an arbitrarily small transmission slot size. Instead, a slot size, which is larger than all messages $m \in M_w$, has to be used. Figure 1 depicts inevitable OP in w_2. Even though m_2 is the only message assigned to this transmission slot, the slot size is larger than the message size, and thus network bandwidth remains unused.

In addition to these two cases, another source of overprovisioning, not related to message transmission time and transmission slot size, has to be considered.

Case 3: Sequential OP

While SSA can improve the network bandwidth usage efficiency through stacked TT transmission slots, in some scenarios, messages may not be perfectly stackable. Such messages have to be assigned to different TT transmission slots. If, during runtime, the system is in a mode of operation where the messages assigned to these transmission slots are not active, the network bandwidth remains unused. The resulting overprovisioning of the network is termed Sequential OP. An example for Sequential OP due to messages that are not stackable is depicted in Figure 1. m_2 and m_4 are not stacked into w_2, because m_2 is active in both modes. The two messages can thus not be assigned to the same transmission slot.

6. CASE STUDY: TTETHERNET

In a TTEthernet network, SSA can be applied for both deferred mode changes, where messages from all nodes may be stacked, if they satisfy Condition 1, as well as immediate mode changes, where only messages from the same source node are allowed to be stacked into the same transmission slot. This helps to ensure that Condition 4 from Section 5.4.1 is satisified.

6.1 Network Description

TTEthernet is a TT Ethernet based network. It combines event-triggered transmission of packets in compliance with the Avionics Full Duplex Switched Ethernet (AFDX) with scheduled transmissions of TT messages. TTEthernet specifies a global synchronization protocol to synchronize the whole network to a global time base. This synchronization protocol, along with specialized switches and network controllers facilitate the transmission of TT messages according to offline computed schedules. TTEthernet considers three traffic types:

Time-Triggered traffic (TT) for scheduled safety-critical time-triggered messages.

Rate-Constrained traffic (RC) for safety-critical event-triggered messages at a constrained rate.

Best Effort traffic (BE) for non safety-critical traffic.

TTEthernet makes use of virtual links for sending messages. A virtual link (VL) defines a unidirectional virtual message transmission channel from one sender to one or more receivers. A virtual link can either be TT (TT VL) or ET (RC VL). TT VLs utilize schedules computed offline to transmit messages through the network, while RC VLs allow for transmission at a constrained rate without the need of scheduled transmission slots. If, during a scheduled transmission slot, a TT message becomes ready, it is transmitted through the network, if not, pending RC or BE messages are transmitted. If a TT message arrives at a switch or network controller outside of its scheduled transmission slot, it is considered invalid and discarded. For this case study, we focus on TT traffic and assume no RC or BE traffic to be present in the network.

The TTEthernet schedule is divided into macroticks (network parameter μ_Θ from Section 3). The size of the macrotick can vary from several nanoseconds to a few hundred microseconds and is chosen during system design depending on hardware and software choices [2].

The schedules for a TTEthernet network are computed offline and uploaded to the devices before the system commences normal operation. If the schedule in the network has to be changed, a full reconfiguration is required. A reconfiguration consists of uploading the new schedules to all devices and performing a network reset. During this reset, synchronization is lost, and as a result, reconfigurations may result in significant service outages of the network.

6.2 SSA Offline Scheduler

We have a generic scheduler [4][13] which uses the Iterative Deepening A* (IDA*) [11] search algorithm to construct scheduling tables offline. The search-tree node in our scheduler represents a (partial/complete) schedule for all modes of operation until time t, where t is an integral multiple of μ_Θ. The search-tree nodes are generated for each possible decision at a point in time defined by t. The cost of a search-tree node depicts the earliest task/message response time. In order for earlier elimination of the wrong path in the search-tree, any search-tree node leading to a deadline miss is pruned.

The SSA Offline Scheduler is an implementation of SSA on our scheduler where search-tree nodes are generated for each possible phase as a result of the phase generation method for strictly periodic activities. Once the schedules for all the modes are generated, the messages from all modes are stacked as mentioned in Section 5 and written to the TTEthernet XML files (see Section 6.4), which can then be used to execute the generated schedules online. In this case study, we only consider deferred mode changes to simplify the scheduler implementation. Due to the fact that the SSA Offline Scheduler is not the main contribution of this paper, and due to space limitations, the complete listing of the algorithm is not presented here.

6.3 Overprovisioning Revisited

As discussed in Section 6.1, the size of the TT transmission slots can be larger than the maximum message size (see case 2 and 3 in Section 5.5), and thus TTEthernet can suffer from internal and inevitable OP. Depending on the stackability of the messages, SSA can also suffer from sequential OP. With stacked TT transmission slots, the sequential OP of SSA is, however, much lower than other approaches that work on existing hardware and don't require reconfiguration (super schedule).

6.4 Proof of Concept

In order to show that SSA can be implemented in existing COTS TT network hardware, we implement the example described in Section 5.4.2 in the AVIonics NEtwork Laboratory (AVINEL) of the Real-Time Systems Chair at TU Kaiserslautern. This laboratory utilizes the TTEthernet toolchain and development hardware provided by TTTech. We synthesize schedules both for the IF-noSSA approach as well as the IF-SSA approach. These schedules consist of four TT virtual links, each carrying network packets with a size of up to 1518 bytes and a period of 8 ms to one of four receivers. In the IF-noSSA case, each virtual link is assigned to a separate transmission slot on the link l_1 between n_{ESP} and n_{SW}, while in the IF-SSA case all virtual links are stacked into the same transmission slot. We insert the schedules of both cases into the XML files utilized by the toolchain which converts these XML files into binary device configurations and uploads them to the hardware. To compare both cases, we send 100000 packets through each virtual link for each case. We determine packet-loss by counting packets on all stacked virtual links using Wireshark, determine the validity of the transmitted messages by using a TTEthernet end system as destination and compare the distribution of packet inter-arrival times. Initial experiments show the following results:

- Applying SSA to the TTEthernet schedule does not result in packet loss.
- Transmitted packets are not transmitted outside their scheduled transmission slots (the receiving TTEthernet end system considered all incoming packets as valid)
- SSA has no measurable impact on the packet inter-arrival time

7. CONCLUSION

Time-triggered networks are widely used for safety-critical applications but lack the flexibility and adaptivity these applications may require. Existing approaches incur significant drawbacks, either by resulting in low network bandwidth utilization or requiring expensive online reconfigurations. In this paper, we proposed the stacked scheduling approach to enable mode changes and implicit adaptation without online reconfiguration for COTS TT networks. The drawbacks of existing solutions, for example low resource usage efficiency or expensive online reconfiguration are not present in SSA. We described the requirements and limitations of SSA and performed a case study using COTS TTEthernet network hardware. We presented the benefits of SSA over traditional scheduling approaches and verified, that SSA can be implemented the COTS TT network hardware without impairing normal operation.

8. REFERENCES

[1] M. Ashjaei, P. Pedreiras, M. Behnam, L. Almeida, and T. Nolte. Dynamic Reconfiguration in Multi-Hop Switched Ethernet Networks. In *6th Workshop on Adaptive and Reconfigurable Embedded Systems*, 2014.

[2] S. S. Craciunas and R. S. Oliver. SMT-based Task- and Network-level Static Schedule Generation for Time-Triggered Networked Systems. In *Proceedings of the 22nd International Conference on Real-Time Networks and Systems*, 2014.

[3] FlexRay Consortium. FlexRay communications system protocol specification version 3.0.1. *Protocol Specification*, 2010.

[4] G. Fohler. *Flexibility in Statically Scheduled Real-Time Systems*. PhD thesis, TNF, Wien, Österreich, April 1994.

[5] T. Fuehrer, B. Muller, W. Dieterle, F. Hartwich, R. Hugel, and M. Walther. Time triggered communication on CAN (Time Triggered CAN-TTCAN). *7th international CAN Conference*, 2000.

[6] K. Klobedanz, G. B. Defo, W. Mueller, and T. Kerstan. Distributed coordination of task migration for fault-tolerant FlexRay networks. In *International Symposium on Industrial Embedded System (SIES)*, 2010.

[7] K. Klobedanz, A. Koenig, and W. Mueller. A reconfiguration approach for fault-tolerant FlexRay networks. *2011 Design, Automation & Test in Europe*, 2011.

[8] K. Klobedanz, A. Koenig, W. Mueller, and A. Rettberg. Self-Reconfiguration for Fault-Tolerant FlexRay Networks. In *Object/Component/Service-Oriented Real-Time Distributed Computing Workshops (ISORCW), 2011 14th IEEE International Symposium on*, 2011.

[9] H. Kopetz and G. Grunsteidl. TTP-a protocol for fault-tolerant real-time systems. *Computer*, 1994.

[10] H. Kopetz, R. Nossal, R. Hexel, A. Krueger, D. Millinger, R. Pallierer, C. Temple, and M. Krug. Mode handling in the Time-Triggered Architecture. *Control Engineering Practice*, 1998.

[11] R. E. Korf. Depth-first iterative-deepening: An optimal admissible tree search. *Artificial Intelligence*, 1985.

[12] T. Pop, P. Pop, P. Eles, Z. Peng, and A. Andrei. Timing analysis of the flexray communication protocol. *Real-Time Systems*, 2008.

[13] A. Syed and G. Fohler. Search-tree exploration for scheduling using pida*. Technical report, August 2014.