TT^{ET}: Event-Triggered Channels on a Time-Triggered Base*

Vilgot Claesson and Neeraj Suri
Department of Computer Science
TU Darmstadt
64283 Darmstadt, Germany
www.deeds.informatik.tu-darmstadt.de

Abstract

This paper develops solutions for efficient transfer of sporadic (event-triggered) data over a time-triggered communication channel. We present novel and efficient techniques for composite provisioning of the event triggered (ET) and time triggered (TT) paradigms to achieve both the predictability and flexibility inherent in TT and ET systems respectively. As a tangible demonstration of the techniques, present and assess a variation of the TTP/C protocol, where minor operational changes allow for its efficient handling of sporadic message traffic.

1. Introduction

The techniques to access a shared communication media using either demand based access (Event-Triggered) or slotted controlled access (Time-Triggered) have garnered their share of strong opinions in the community.

The implications of choosing either the time-triggered or event-triggered approaches are not particularly easy to assess. Especially within a complex system where the various design considerations span contending tradeoffs across flexibility, efficiency, predictability, and dependability. The event-triggered (ET) approach is widely perceived to offer a highly flexible communication solution, for example in the ease of adding nodes and adapting to changing communication loads in nodes. The caveat being the potential variability of message delivery times in ET operations. In contrast, the time-triggered (TT) approach offers considerably less flexibility compared to ET. Instead the TT approach offers highly predictable communication behavior on account of its static slotted access approach; predictability being a significant factor in its usage for design of safe and reliable systems.

* This work was funded in part by EU Next TTA #IST-2001-32111

In this paper, we consider reliability and safety to be the primary design considerations with correspondingly desired attributes of predictable communication. However, a capability of handling sporadic messages such as high-priority interrupts is also essentially desired. We contend that the preferred way of maintaining the predictability attribute is to start with a time-triggered base and build the necessary event-triggered channels on top. Consequently, we investigate approaches for combining the benefits of both time-triggered and event-triggered communication within a common framework.

We explicitly emphasize that a pre-dominant consideration over our proposed approaches is to ensure that the safety related essence of the TT paradigm, specifically predictability, is in no way compromised over the process of extending its handling of sporadic ET traffic.

This paper has two main parts. The first part suggests approaches for improved handling of sporadic messages in TT systems. In the second part, we restrict ourselves to a specific and well-established safety-critical TT communication protocol, namely the TTP/C protocol [2, 3]. Our goal is to derive an adaptation to the protocol that improves the efficiency of sporadic message transfer. It is important to achieve this without any disruption of the time-triggered traffic or the underlying periodic behavior. As this protocol is directed toward safety-critical systems, much effort and work has been expended in ensuring its correct behavior. Thus, our solution must work with very limited changes to the protocol specification so as to minimize any subsequent re-verifications. We especially try to modify only those aspects within TTP/C that do not change any behavioral characteristics of TTP/C. For example, if an existing TTP/C frame is utilized for handling sporadic messages in place of periodic traffic, from the TTP/C protocol viewpoint, there is no change done to the frame specification of the protocol.

For a background and short history on multiple-access communication we direct the reader to [4] and [5] that comprehensively survey the area.

2. System Model

We target systems that have stipulated requirements of reliability, safety, and also of real-time specifications. In these systems, we consider *n* autonomous nodes that communicate via a broadcast media. When using such a media, all nodes can simultaneously listen and receive information transferred on the media. However, only one node can send at a time; if more than one node transmits concurrently, the information will be garbled.

2.1. The Time-Triggered Base

In the case of a *time-triggered system*, nodes have local counters that are used to control the sending and receiving of messages. Furthermore, the nodes operate in a cyclic manner where each node can send one or more times in each TDMA communication round, as shown in Figure 1.

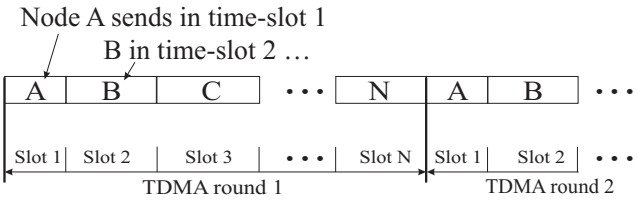

Figure 1. A TDMA communication round.

The synchronization of the nodes in the communication system can be divided into a number of steps, namely: (a) collection of nodes clock values, (b) calculating adjustment value and (c) clock adjustment. Thus, the clock synchronization process requires that the receiver of a message knows the sender's identification (ID) and local time reference (e.g., when the message was sent). For the TDMA communication this information can be extracted from the arrival time of messages, as they are known statically. The differences between the local and senders clocks are calculated by the differences in the expected arrival time and the real arrival time.

We assume that the TDMA synchronization is handled by a standard clock synchronization algorithm controlling the progress of the local clocks, for example, using the daisy-chain clock synchronization algorithm [6]. Other examples of existing suitable synchronization algorithms can be found in [7, 8, 9].

Synchronization ensures that every node has the same view of the current position in the communication schedule, which suffices for nodes to know when to send their messages. See [2, 10] for advantages of synchronized nodes.

Each node must store the communication schedule, which will contain the information of when and what a node is expected to send. The storage requirement for this information is normally relatively small, typically a few kilobits, since we deal with embedded systems with a limited number of nodes.

2.2. Event-Triggered Transfer

In the case of an *event-triggered information*, normally a node will immediately send its message if the media is free. If the media is occupied, the transmission is delayed until the media becomes free. However, if more than one node is waiting to send when the media becomes free, a collision is likely to occur. In this paper we will discuss different approaches to transfer the event-triggered information on top of a TT-system instead.

Synchronization: In an event-triggered system synchronization is not really necessary, as nodes do not synchronize their sending with other nodes. However, there is nothing that prevents the introduction of synchronization in the event-triggered system. In for example CAN, additional mechanisms has been added to achieve such solutions, e.g., see [11].

In our approach the nodes are already synchronized due to the underlying TT-base.

2.3. Fault Model

When considering faults, we assume that the nodes follow *fail-silent* semantics thus preventing faulty nodes failing by continuously transmitting on the communication channel. Such a failure would overflow the media and prevent any normal communication, including synchronization traffic. The fail-silent property [12, 13, 14] relies on high coverage of the error detection mechanisms of the nodes. It can be argued that sufficient coverage may be difficult to achieve and therefore, such failure semantics is unsuitable for hard real-time system. However, recent work indicates that using rigorous design and error detection methods, a very high coverage can be achieved [15].

The fail-silent semantics also restrict the number of diverse failure scenarios viable in the system.

The Media For the broadcast media, we assume well-accepted omission failure semantics [16, 17] where messages are either received correctly and on time or not at all. By designing nodes to be fail-silent and using a broadcast media, we effectively exclude timing failure on the media. Similarly, Byzantine failures are not regarded as they are considered to be avoided by design, using a broadcast media combined with message checksums.

3. Lower Priority Non-Periodic Messages

In this section, we study scenarios where we have low-priority sporadic traffic that must be scheduled "over" the time-triggered traffic. The intention is to make as many sporadic messages meet their (soft) deadlines as possible.

3.1. Pre-Scheduled Slack

In this method, each node schedules some slack at pre-runtime, which is pre-destined for sporadic/event triggered traffic from that node, see Figure 2. Thus, a node fills a pre-defined part of the message-frame with periodic data, and when a node has sporadic messages to send, it will pack as many as possible in the remaining part of the frame. The size of the periodic part is determined pre-runtime, as the frame sizes are fixed. Thus, the size available for sporadic messages is also fixed. The properties that follow this method include:

- The sporadic data normally needs addressing information, such that the nodes can distinguish between different sporadic messages from a node, i.e., if there they receive several from the same node. Thus, each data entity must include an address or ID. Please note that for periodic data, address information is implicit as this data is statically scheduled.
- The length of a message-frame is constant and the same as the slot-length. A node cannot adaptively change the amount of data it sends. Furthermore, a node cannot increase or decrease the amount of "event data" to send, as the slack is statically scheduled for each node.
- A node cannot utilize unused slack of other nodes. This implies that there is no global priority on sporadic data. The priorities among the sporadic messages are handled internally in each node, i.e., each node handles its own slack.

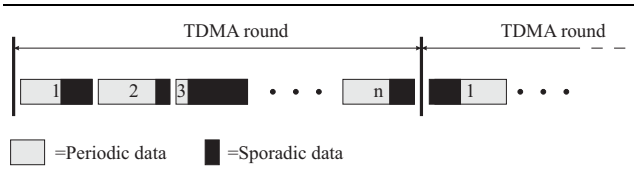

Figure 2. The TDMA approach, with slack, for event-triggered/sporadic traffic.

The drawback of this approach is that message-frames are fixed, both in position and length. This prevents a node from adapting to differences in the need for transferring sporadic data, a node cannot share the unutilized portion of the media-access time that is pre-assigned to it.

3.1.1. Improvement suggestion As the frame sizes are fixed, only node-internal measures can be taken. Thus, one approach is to permit a flexible assignment within the frame for periodic and sporadic information, i.e., a frame from a node can be filled with only sporadic data, only periodic data, or anything in-between. This would naturally depend on the current communication requirements of the node and the priority of data to send. In some cases it could necessitate that a periodic message at some occasions would be delayed/skipped to make way for a sporadic message. Naturally, this only applies to situations where the system is expected to perform such prioritizing among the messages, and must be decided by the system designer.

3.2. A Mixed Access Method

In this method periodic data is scheduled first and no slack is used in individual nodes. The TDMA-round is divided in two parts, one where nodes access the bus by TDMA and a second part for sporadic data, using an alternate media access method. Thus, in each TDMA-round there will be a certain free time not occupied by statically scheduled data, which is free for sporadic traffic. The drawback with this method is that it needs a special bus access procedure for the sporadic messages. Examples of possible bus access methods include:

- Minislotting. The combination of TDMA and minislotting is, for example, used in Arinc 629 [18].
- The use of bit arbitration.
- The use of tokens.
- A Media Master who sends the schedule in the periodic message. For example, each node asks/requests a message slot in the "sporadic area" in the periodic message, the last node who sends in the periodic area is a "master" and decides when and how much bandwidth each node is assigned. This would require that all nodes send a message in each TDMA-round such that they can demand a piece of the "sporadic message area".

This method can have variants where a TDMA-round is divided into a number of periods, with periodic and sporadic areas that alternate. Each periodic area would end with a node functioning as a "master" in the following sporadic area, which will be partitioned among the nodes, by the master node.

4. TT^{ET}: An New Approach based on TTP/C

In this section we present a new approach for sending event-triggered data on a time-triggered channel. We develop a modified version of the existing TTP/C protocol [3, 2] as the base protocol. Our modification is labeled as TT^{ET} and we say ET on TT. Our intention is to introduce as few changes as possible to the TTP/C protocol to maintain its predictability feature, and not to impose any additional complexity. As stated in the previous section, there will be no "preemption" of the normal time-triggered communication in order to send higher priority messages. Instead we have restricted this case to low-priority event-triggered channels, where the focus is on robustness and short access times.

To facilitate sporadic data transfer with TTP/C, our approach introduces two new concepts, (1) the Sporadic Information Transfer (SIT) bits, and (2) *free-frames*. The SIT bits consist of one or more bits, included in the normal data frames. The purpose with the SIT bits is to either directly send sporadic data or requesting some additional slots for sporadic data. After such a request a node will normally be assigned a slot, i.e., a *free-frame* in which it can send its sporadic data. In Figure 3 a normal communication frame is shown with an additional SIT bit. This normal data frame makes it possible to implement the required functionality without any major changes.

Conceptually, the *free-frames* are empty slots reserved for sporadic/event-triggered data. These newly introduced *free-frames* are part of the communication schedule, as they are statically scheduled. However, any node is allowed to send sporadic data in these *free-frames*, see Figure 4. This will require a method for avoiding collisions in these slots, and we will come back to how this is handled in Section 4.1.

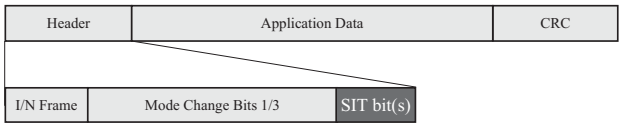

Figure 3. A TTP/C normal frame with the addition of a SIT bit.

The *free-frames* are used when large amounts of sporadic data transfer is needed. In such cases, the introduced SIT bits will be used as indicators that a node wants to use the "*free-frames*". If data is sent directly using the SIT bits we get a low bandwidth solution and when transferring using the *free-frames* we get a high bandwidth solution. Thus, depending on the amount of sporadic data a node wants to transfer, we use the SIT bits in two different ways: (1) in case of small amount of sporadic data, the SITs bits are used to transfer data, or (2) in case of larger amounts, the SIT bit can work as requests for *free-frames*.

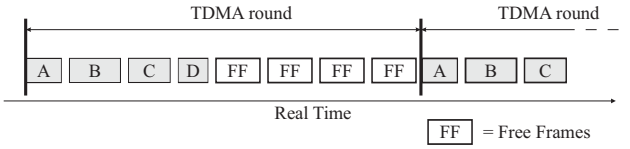

Figure 4. A TDMA-round with event-triggered channels in form of *free-frames* located in the end of the TDMA-round.

The decision to use only a single or several (yet very few) SIT bits is based on limiting the overhead for introducing a sporadic channel. However, using only a single bit has some implications, e.g., we have to decide how to use this SIT bit pre-runtime. When transferring data via the SIT bits, we call it a Low Bandwidth (LB) solution. The available bandwidth for sporadic messages at run time is static in such a case. If a node uses LB, it needs a local priority queue for sporadic messages. In case of more than one receiver we have to handle addressing, start and stop of messages, etc.

By using the SIT bits for message transfer we have created a low bandwidth channel for event-triggered communication over a time triggered system. The advantage of this method is the low complexity which facilitates the design of a robust system. However, it offers limited bandwidth and that bandwidth can not be used by any other node when nothing is sent.

In the second option, we used the SIT bits as requests for further sporadic bandwidth, i.e., requests for *free-frames*. This makes it possible to share the *free-frames* among all nodes, i.e., all nodes can use this bandwidth. This approach implements a High Bandwidth (HB) event-triggered channel for nodes that have a lot of sporadic data to send. This also has implications on the working of the protocol. Specifically, we have to decide how the sporadic bandwidth is divided among requesting nodes. There is a number of options which we will discuss in Section 4.1. Finally, a node can use a combination of the above such that the node has a LB channel and a HB channel.

In the following sections we will describe and investigate how a number of variants on assigning *free-frames* affect the event-triggered channel. Another parameter that is investigated is when the *free-frames* are scheduled, e.g., com-

posed at the end of a TDMA-round or scattered over the whole TDMA-round.

4.1. Prioritization of Sporadic Messages

In this section we discuss alternate solutions for controlling the *free-frame* access among nodes; and also across messages within a node. The goal is to obtain short access times and high throughput for the sporadic data.

We only consider how nodes will share the *free-frames*, as communication via the SIT bits is strictly handled node internally. Locally, a node must handle sporadic messages such that they get queued in a node internal queue according to their priority. This is independent of whether the data will be sent via the SIT bits or *free-frames*. To transfer sporadic messages we need to handle extra information, compared to the periodic data, as the sporadic traffic is not static:

- Start and stop information of sporadic data.
  - Start and stop bits, indicating the start and stop of a message.
  - Messages can be sent starting with a specified offset from the start of, for example, the cluster cycle, i.e., a number of repeated TDMA-rounds, or TDMA-round. When using sporadic messages with predefined length, they can be synchronized to the clusters cycles.
- Destination address or message ID serving as address. This is necessary to transfer in order for receivers to know two whom the message is directed. However, sender address is not necessary as the node from which the frame arrived is known.

As discussed earlier, the *free-frames* approach requires a media access method to avoid collisions when sending (using) the free frames. We have investigated two approaches for avoiding collisions, one central and one distributed approach, described in the next two sections.

4.1.1. Central Prioritization In this approach, one node will make a central/global decision about which node is allowed to send in a specific *free-frame*. The nodes sending in the static area can request bandwidth for sporadic data in the form of *free-frames*. The last node in the static part of the TDMA-round, i.e., node i_e, will prioritize and decide which of the requesting nodes are allowed to send, and at what time. Node i_e will then include the schedule of the sporadic event-triggered data in its message, see Figure 5.

Using this method, we get extra overhead for explicitly sending the schedule for the event-triggered traffic. If we assume that we indicate whether a node should send or not, we use n bits followed by n times the maximum number of *free-frames* that are sent per node, assuming x bits we get a total overhead (H) of:

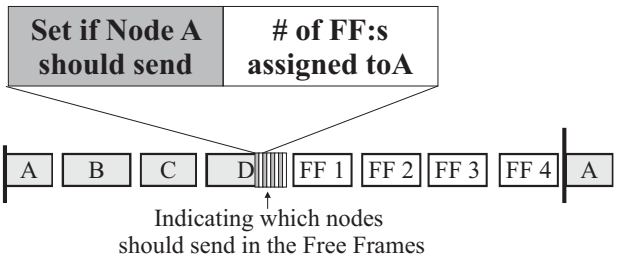

Figure 5. A central node decides which nodes are allowed to send in the event-triggered channel, formed by the *free-frames*.

$$H = n + n \cdot x$$

One consequence of using this approach is that the schedule is explicitly sent on the bus. It is a low complexity solution, but only one node controls the schedule, which can potentially introduce a single point of failure. However, normally the safety-critical information is transferred via the more predictable time-triggered channels. Only when using one central unit for prioritization of the event-triggered messages can the implementation be easily changed without having to make changes at all nodes. Nodes that do not receive this message are considered to observe silence semantics.

4.1.2. Distributed Prioritization The previous method has the disadvantage of introducing a single point of failure (if the central node fails, the event-triggered channel will collapse). A more attractive solution, with fault tolerance aspects, is to use a distributed approach when deciding the allocation of *free-frames* to nodes. In this solution, each participating node makes a decision based on received information, which makes it very important that all nodes receive the same information. Thus, the nodes make a distributed decision about which nodes can access the *free-frames*. Our method has the purpose of achieving low overhead, small delay, and uniform bandwidth among requesting nodes. However, this basic method can easily be modified to give one node higher priority. Although, in the following we only describe the basic method where all nodes will be able to send and where we focus on short media access times.

The sporadic transfer efficiency is dependent on the number and positions of the *free-frames* in the TDMA-rounds. This approach basically gives priority to the node with the earliest request. For example, assume nodes a, b, and c send (in that order) their time-triggered frames, just

before a *free-frame* is scheduled. If all of them request a *free-frame*, then node a will get the highest priority followed by b and then c. If there were other nodes in the queue before a, b, and c that could make there requests, they will be put in the queue after the new nodes, i.e., node a, b, and c. This means that nodes might not get access to the event-triggered channel if there are more nodes than *free-frames* in the system.

We also note that in order to minimize the access time to the event-triggered channel, there should be a *free-frame* in every second frame, see Figure 6.

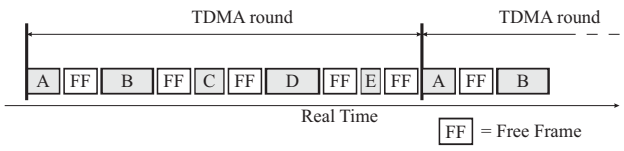

Figure 6. Distributed method where the *free-frames* are scattered throughout the TDMA-round in order to minimize the buss access time for sporadic data.

If a node fails to receive a message with a request for a *free-frame* it will get an inconsistent view of parameters can be set before runtime, there is a possibility to minimize the number of priorities in the system. This can minimize the number of bits necessary to transfer the requests and corresponding priority.

When half the frames are *free-frames*, each node will have the possibility to send in a *free-frame* each TDMA-round. If there are less *free-frames* and all nodes want to send, the lowest prioritized nodes will not have the chance to send. Thus, the will be subjects to starvation if they never can access any *free-frames*. However, this could be handled by circulating the priorities among nodes during different TDMA-rounds.

During times when no nodes request event-triggered data transfer, the *free-frames* can be statically assigned to specific nodes. This serves two purposes:

- A nodes' response times can be reduced by assigning default designations to *free-frames*. Thus, all *free-frames* will be preassigned to a node. If a node notices that its designated *free-frame* is not reserved, i.e., requested, it is free to use that *free-frame*. If this node has data to transmit, and did not have the opportunity to request a *free-frame*, it is free to send in that *free-frame*. This can, during low load situations, lower the access time for nodes.
- Nodes that normally do not send every TDMA-round, can be assigned a *free-frame* which gives such nodes access to those TDMA rounds under low load situations, i.e., those nodes have the possibility to send in rounds when they are normally not sending any time-triggered frames.

With the described method, a node may be assigned more than one *free-frame*. However, if one *free-frame* suffices for the node, it would occupy more *free-frames* than necessary. As we use the same message-frames both for time-triggered and event-triggered messages, we can use the SIT bits to indicate when a node has no more sporadic data to send. Thus, if a node is assigned two or more *free-frames* but only needs one, it can indicate "no more sporadic data" using the SIT bits in the *free-frame*. This will improve the utilization of the media, as no *free-frame* is assigned to a node with no more data to send.

5. Properties

In this section we briefly describe some of the properties of our adaptation of the existing TTP/C protocol and some representative simulation results. The primary intent is to confirm that the TTP/C behavior has not been perturbed. We defer detailed simulations as a future elaboration to the approach. We have focused our approach on using distributed prioritization, where every second slot is a *free-frame*. We also assume that all time-triggered message-frames have the same length, and all *free-frames* have the same length. However, time-triggered message-frames and *free-frames* do not necessarily have the same length.

We emphasize that our main focus has been to achieve flexible handling of sporadic messages without disturbing the time-triggered base. In Table 1 we compare a few important properties of our modified TTP-protocol, termed as TT^{ET}, with the classic TTP and the CAN [19] protocols. For the TTP protocol we assume that sporadic messages are handled using preassigned slack as described in Section 3.1. In Table 1 we show the (1) worst-case (WC) delay of a sporadic-message, (2) the overhead corresponding to sporadic message handling, and (3) how much sporadic data can be assigned to a single node, under the condition that they all have the same amount of data and the same period T of periodic messages.

In the first column we have the WC delay where our TT^{ET} and TTP have the same WC delay, i.e., one period T. The CAN protocol has a very short WC delay, which is when the longest message must finish sending before the next may access the bus.

In the Overhead-column of Table 1, the overhead related to the sporadic message transfer is shown. In our TT^{ET} the overhead is related to the SIT bits, one for each node, assuming n nodes. For TTP using pre-assigned slack there is

no extra overhead. For the CAN protocol, the ID-field is also used to resolve priorities accessing the media. Thus, it should be noted that comparing these may be a bit favorable for the TTP protocols.

Finally, the last column indicates how much media access can be assigned a single node, in the best case. For our TT^{ET} we can basically assign all *free-frames* to one single node, i.e., $T/2$. For the standard TTP we cannot change the pre-scheduled slack, and assuming each node is assigned the same amount of slack for sporadic messages, one node gets the size of approximately one *free-frame* (*FF*). In this column we can see the major improvement of our TT^{ET} approach which allows a node to utilize more than one *free-frame*. This significantly improves the transfer rate, when a single node has a lot of data to transfer.

Approach	WC delay	Overhead	Max trans
TT^{ET}	T	n bits	T/2
TTP	T	0 bits	approx. 1 *FF*
CAN	Max ML	n · ID-field	T/2

Table 1. Properties of different communication approaches

One question naturally arises as to how the average delays are affected. In the following simulation our goal has been to present some preliminary validations of the expected behavior of the TT^{ET}. In future work, we plan to conduct more detailed simulations using varying parameters of TT^{ET} and simulation parameters. In Figure 7, we observe that although we have introduced a more flexible handling of sporadic messages in the TTP protocol with TT^{ET}, we still maintain, at least, the same average delay of sporadic messages. At low load we even manage to improve the average delay, although the only optimization, described in Section 4, implemented is simply that the *free-frames* have preassigned nodes. We believe that refined optimizations will help improve the average delay characteristics even further. In Figure 8, we have verified that this behavior is still valid if we increase or decrease the size of the *free-frames* compared to the normal time-triggered message-frames.

6. Summary

In this paper we have studied and presented different approaches for transferring sporadic messages using a communication system based on the time-triggered paradigm.

We have developed novel approaches for transferring varied priority sporadic messages using a time-triggered base. More importantly, we have demonstrated a new method based on the TTP/C protocol for transferring sporadic messages in a flexible manner. Our main focus has been to provide for enhanced flexibility in communication without disrupting the fundamental predictability of the TTP/C protocol. This has been achieved with a minor change in normal message-frames which does not change any functional properties of TTP/C. Using our new approach, nodes share *free-frames* on a request basis. This is a significant improvement, compared to using preassigned slack, as one node can utilize many *free-frames* when another node has no sporadic messages to send. Our preliminary simulation results show that, without disturbing the time-triggered traffic, we achieved average delay times comparable to a time-triggered approach using pre-assigned slack. At low loads we have even improved the average access times. We have also shown via preliminary simulations that there is no indication that these results are affected by the size of the *free-frames* compared to the normal time-triggered frames.

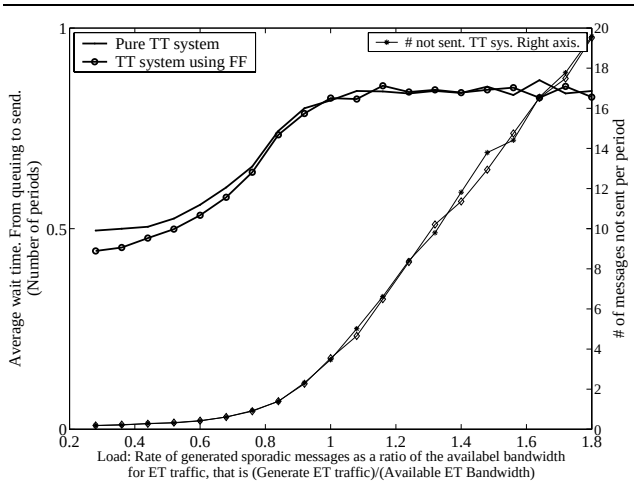

Figure 7. A comparison of our TT^{ET} and a basic TTP using pre-assigned slack.

References

[1] Claesson, V., Ekelin, C. and Suri, N., "The event-triggered and time-triggered medium-access methods," in *The 6th IEEE International Symposium on Object-oriented Real-time distributed Computing (ISORC2003), Hakodate, Hokkaido, Japan*, May 14-16, 2003.

[2] Kopetz, H. and Grunsteidl, G., "TTP- a protocol for fault-tolerant real-time systems," *IEEE Computer*, vol. 27, no. 1, pp. 14–23, 1994.

[3] Time-Triggered Technology, TTTech Computertechnik GmbH, www.tttech.com, *TTP/C protocol, Specification of the Basic TTP/C protocol*, 1.0 edition, Jul 1999.

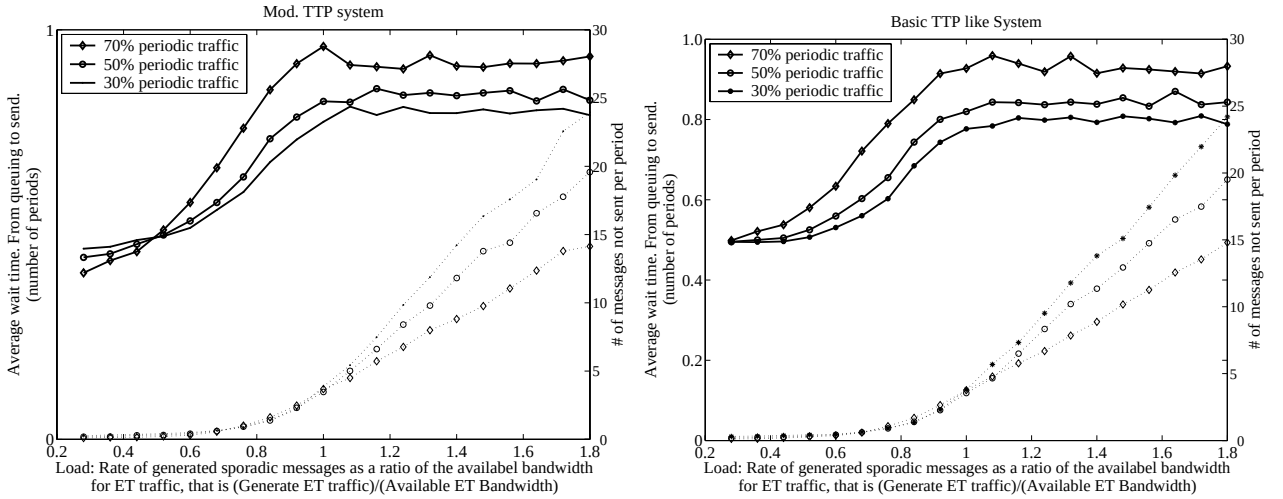

Figure 8. Illustrating the similarities in behavior of the basic TTP and the TT^{ET} when changing the amount of the time-triggered part.

[4] Kurose, J., Schwartz, M. and Yemini, Y., "Multiple-access protocols and time-constrained communication," *Cumputing Surveys*, vol. 16, no. 1, pp. 43–70, 1984.

[5] Malcolm, N. and Zhao, W., "Hard real-time communication in multiple-access networks," *Real Time Systems*, vol. 9, no. 1, pp. 75–107, 1995.

[6] Lönn, H. and Snedsbøl, R., "Synchronisation in safety-critical distributed control systems," in *IEEE International Conference on Algorithms and Architectures for Parallel Processing, ICA3PP*, Brisbane, Australia, vol., pp., 891–899, 1995.

[7] Suri, N., Hugue, M. and Walter, C., "Synchronization issues in real-time systems," *Proceedings of the IEEE*, vol. 82, no. 1, pp. 41–54, 1994.

[8] Ramanathan, P., Shin, K. and Butler, R., "Fault-tolerant clock synchronization in distributed systems," *Computer*, vol. 23, no. 10, pp. 33–42, 1990.

[9] Kopetz, H. and Ochsenreiter, W., "Clock synchronization in distributed real time svstems," *IEEE Trans. Computers.*, vol. 36, no. 8, pp. 933–940, 1987.

[10] Kopetz, H., Krüger, A., Hexel, R., Millinger, D., Nossal, R., Pallierer, R. and Temple, C., "Redundancy management in the time-triggered protocol," Technical Report4/1996, Technical University of Vienna, 1996.

[11] Leen, G. and Heffernan, D., "Time-triggered controller area network," *IEEE Computing & Control Engineering Journal*, vol. 12, no. 6, pp. 245–256, December 2001.

[12] Chrque, M., Powell, D., Reynier, P., Richier, J.-L. and Voiron, J., "Active replication in Delta-4," in *Twenty-Second International Symposium on Fault-Tolerant Computing, FTCS-22.*, pp., 28–37, 1992.

[13] Kopetz, H. and Damm, A. and Koza, C. and Mulazzani, M. and Schwabl, W. and Senft, C. and Zainlinger, R., "Distributed fault-tolerant real-time systems: The MARS approach," *IEEE Micro*, vol. 9, no. 1, pp. 25–40, 1989.

[14] Temple, C., "Avoiding the babbling-idiot failure in a time-triggered communication system," in *Twenty-Eighth Annual International Symposium on Fault-Tolerant Computing*, pp., 218 – 227, 1998.

[15] Folkesson, P., *Assessment and Comparison of Physical Fault Injection Techniques*, Ph.D. thesis, Chalmers University of Technology, 1999.

[16] Perry, K.J. and Toueg, S., "Distributed agreement in the presence of processor and communication faults.," *IEEE Transactions on Software Engineering*, vol. 12, no. 3, pp. 477–482, 1986.

[17] Powell, D., "Failure mode assumptions and assumption coverage," in *Twenty-Second International Symposium on Fault-Tolerant Computing, FTCS-22.*, pp., 386–395, 1992.

[18] Aeronautical Radio, Inc., *Multi-Transmitter Data Bus, Part 1, Technical Description*, Dec. 1995.

[19] Robert Bosch GmbH, *CAN Specification Version 2.0*, 1991.