A Real-time Ethernet Prototype Platform for Automotive Applications

Kai Müller, Till Steinbach, Franz Korf, Thomas C. Schmidt
Department of Computer Science
Hamburg University of Applied Sciences, Germany
kaim@online.de, {till.steinbach, korf, schmidt}@informatik.haw-hamburg.de

***Abstract*—The increasing number of driver assistance, infotainment and entertainment systems in automobiles results in higher requirements for bandwidth, fault tolerance and timing behaviour concerning the in-vehicle communication structure. In future, in-vehicle networks based on current technologies will reach their limits due to insufficient scalability and complexity. Real-time (RT) Ethernet is a new, scalable approach to reduce the complexity of these networks significantly.**

This paper demonstrates the architecture of a RT Ethernet prototype platform based on an ARM centred system-on-chip, which achieves timing and bandwidth characteristics of a typical future automotive application. It is based on an advanced interrupt driven architecture.

I. Introduction

The number of electronic control units (ECUs) in automobiles increases continuously. From computer science point of view, cars become more and more complex distributed RT systems. Applications like chassis control, driver assistance or infotainment and entertainment require novel concepts for an integrated in-car communication backbone. RT Ethernet is a new approach to satisfy the challenging requirements of automotive backbone networks.

Ethernet has already been proven to be a highly scalable, widely deployed and flexible protocol, but does not offer reliable temporal bounds in message transmission that are necessary for safety critical tasks.

TTEthernet [1] is a time-triggered RT Ethernet protocol specified by TTTech [2] in collaboration with Honeywell [3]. It satisfies the special requirements of automotive and avionic networks. To support the diversity of miscellaneous application specific timing requirements, it joins three traffic classes – *time-triggered*, *rate-constrained* and *best-effort* – with different RT attributes on the same standard Ethernet based physical infrastructure, allowing a homogeneous network all through the vehicle.

For our research on the TTEthernet protocol and RT Ethernet based in-car networks, we built a prototype platform based on an ARM9 system-on-chip. To the best of our knowledge this is the first microcontroller based software implementation of the TTEthernet protocol stack. Using this architecture, first design concepts for in-car networks can be evaluated on a hardware platform that is based on standard components and offers realistic performance and temporal attributes.

This paper presents an architecture for a TTEthernet end system that allows full-duplex transmission and reception of time-triggered, rate-constrained and best-effort traffic with variable priority. A preemptive scheduler that is realised by an interrupt driven approach keeps track of correct task execution. An advanced buffer structure that shares its memory with the attached network cores reduces expensive copy operations. The presented architecture fully complies with the TTEthernet specification [1] and the TTEthernet-API [4].

An extensive evaluation of the temporal attributes of the implementation shows the achievable precision in synchronisation and message processing as well as deterministic task execution even under heavy load. Even though the architecture is designed to satisfy the specifications of TTEthernet, it is easily transferable to other time-triggered Ethernet protocols, which are based on multiple prioritised message types like Profinet [5].

The remainder of this paper is organised as follows: In section II, the background of RT Ethernet and the TTEthernet protocol is given and related work is presented. The concept and architecture of the prototype platform is described in section III. Selected details of the implementation are outlined in section IV. Section V describes the test-bed used for evaluation and the results for the analysed metrics. Finally, section VI concludes and gives an outlook.

II. Background & Related Work

For in-car communication RT Ethernet approaches are still under investigation. However, a variety of application domains like process automation have already successfully adopted this Ethernet technology.

A. Real-time Ethernet Variants

RT Ethernet variants can be classified in three major groups:

Token-based systems use a shared resource – the token – to control network access. Only the owner of this token is allowed to send messages to the network, which ensures that no congestion emerges. In token-based networks the whole protocol logic is transferable to host side, which allows the use of standard Ethernet switches. The biggest challenge is the detection and recovering of a lost token. Token recovery directly affects the worst-case delay. A well distributed token-based protocol is EtherCat [6] utilised in process automation. EtherCat supports different communication media, topologies and a redundancy concept.

Bandwidth-limiting protocols use predefined bandwidth contingents to ensure a reliable message transmission. The switch monitors the traffic and ensures that no client exceeds its contingent. A further advance is the definition of bandwidth allocation gaps (BAG). This periodical breaks between every attempt forces each sender to maintain the artificially reduced bandwidth. The senders do not operate synchronised, therefore latency is not predictable but a computable upper bound exists [7]. In aviation an established bandwidth-limiting protocol is the avionics full-duplex switched Ethernet (AFDX) [8].

Time-Triggered RT Ethernet variants use a coordinated time division multiple access (TDMA) media access strategy to ensure predictable message transmission. Through synchronisation, a global time base is shared among all participants. An offline defined schedule determines the dedicated transmission times for each node in the network. This ensures that no congestion on outgoing linecards delays the transmission of RT frames and thus allows transport with low latency and jitter. The more precise the system can be synchronised, the more accurate timings can be achieved. A popular time-triggered real-time Ethernet protocol for industrial applications is Profinet [5].

B. TTEthernet

TTEthernet is a time-triggered RT Ethernet extension that focuses on the specific demands of automotive and avionic applications. The TTEthernet specification [1] was developed by TTTech and is currently proposed for standardisation by the Society of Automotive Engineers [9].

TTEthernet relies on switched Ethernet. Any topology is formed of switches that relay the messages. Redundancy is achieved by multiple redundant channels. As usual in time-triggered protocols it is centered around periodic cycles. For time-triggered communication, each node is assigned to offline configured timeslots. This coordinated TDMA based access policy ensures predictable transmission delays without queuing, and therefore low latency and jitter. To allow each node to access its dedicated transmission slot, each component has its own local clock and transmission schedule. Since a global synchronised time across all participants is needed, the TTEthernet specification defines a fail-safe synchronisation protocol.

The TTEthernet synchronisation protocol defines three roles: The *synchronisation master* starts the synchronisation, the *compression master* calculates the global time and the *synchronisation client* receives the global time. Synchronisation in TTEthernet is based on Ethernet messages called protocol control frames (PCF). Each synchronisation master sends its PCF at a dedicated time in the cycle. The compression master collects the PCFs of all synchronisation masters and calculates a global time of all messages. Then, the new time is broadcasted in a new PCF to all participants, which adjust their local clock to the global time.

Besides the *time-triggered* (TT) traffic, TTEthernet defines two other traffic classes. *Rate-constrained* (RC) traffic is based on the AFDX-Protocol [8] and intended for communication with less rigid temporal requirements. *Best-effort* (BE) traffic has the lowest priority and is based on standard Ethernet. TTEther-Networks are capable of working with hosts that are unaware of the time-triggered protocol and thus remain unsynchronised. Those hosts only communicate by best-effort traffic.

RT traffic uses a content oriented addressing format, similar to Ethernet multicast. Instead of addressing a node in the network, the content of a frame is determined by the destination address. The 48 bit destination address is divided into two parts. The first part is the critical traffic marker. It is used to detect RT traffic. The critical traffic ID (CT-ID) is located in the second part. It is unique for each message in the cycle and used to address the message in the TTEthernet schedules. Routing decisions are based on the CT-ID.

For rate-constrained traffic, it is ensured that sufficient bandwidth is available for each message (identified by its CT-ID). To achieve this, *Bandwidth Allocation Gap accounts* (BAG-accounts) are defined. BAG-accounts determine the minimum period of time between two subsequent messages with the same CT-ID. The application sending the RC message has to respect the constraints of the configured BAG-account, otherwise the message will be considered as invalid and dropped by the switch.

The best-effort traffic in TTEthernet equals standard Ethernet traffic. It has the lowest priority of all three traffic classes and is relayed in idle times within the cycle. Best-effort traffic provides no guarantee for the upper bound of latency or even the delivery of packages.

TTEthernet is a typical time-triggered RT Ethernet variant. Thus the concepts presented in this paper can be easily transferred to other protocols such as PROFINET that use a similar partitioning of traffic classes.

C. Previous & Related Work

Since RT Ethernet is a new candidate for in-car communication networks [10] compares TTEthernet and FlexRay as state-of-the-art technology. This mathematical analysis shows the potential of RT Ethernet based in-car backbones. Further the simulation based analysis of network metrics [11] in real-time Ether-networks shows the potential of real-time Ethernet based in-car backbones.

The realisation of RT Ethernet on any given hardware is currently a very attractive field for both research and industry. In general it can be distinguished between software and hardware based approaches, whereas both have advantages in their environment. The development of software stacks is much more cost efficient. In contrast hardware based approaches offer a higher temporal precision. Most prototype implementations rely on standard PC hardware with a RT enabled operating system. Due to the overhead of such systems realistic scenarios for automotive applications cannot be achieved. [12] gives an overview of software based implementations of the time-triggered Ethernet variant [13] developed at TU-Wien. The commercial TTEthernet applied in this prototype builds on these concepts.

Furthermore, in [14] the time-triggered Ethernet protocol was also implemented in hardware by using a system on a programmable chip and separating the protocol controller from the host controller, which is responsible for executing the application itself. The communication between them is realised by using dual-port memory. It was shown by experiments, that this implementation meets the strict timing requirements for RT communication.

In contrast to the related concepts, the architecture presented in this paper uses a hybrid approach based on a multicore hardware design and a software implementation of the RT protocol stack.

III. Concept & Architecture

For realising a prototype with reliable timings in scheduling of message transmission and task execution, the choice of hardware is an essential part of the concept, which has direct influence on protocol metrics. The synchronisation process is important for time-triggered protocols, and therefore must be supported by the architecture. Furthermore, the bandwidth of 100 Mbit/s per port demands a memory architecture that is able to manage high data throughput. Due to fault tolerance through redundancy, the system has to handle transmission and reception on multiple ports concurrently, which avoids bottleneck behavior by design.

A. Platform

The concept is based on a highly integrated system-on-chip design developed by Hilscher [15]. The modular design of this platform provides four configurable communication channels for various communication technologies like Ethernet or CAN [16]. Each channel provides its own Arithmetic Logic Unit (ALU), which acts as MAC (Medium Access Controller), PEC (Protocol Execution Controller) and, especially for Ethernet, PHY (Physical interface) and can be programmed via microcode. The application itself is executed on an integrated 200 MHz ARM9 CPU. The ALUs of all channels are able to operate independently, even to the main ARM core and allow processing Ethernet frames and applications in parallel. Communication and data transfer between channels and CPU is provided by a data switch, which replaces the AMBA (Advanced Microcontroller Bus Architecture). The switch is able to open up to five communication channels at the same time. With this architecture we are able to send and receive multiple frames on different ports concurrently, which allows to support the TTEthernet conform fault tolerance approach.

A dedicated system time module takes timestamps of every incoming frame and can be configured to handle time-triggered code execution. It is clocked with 100 MHz but with an adjustable increment in a resolution of 2^{-28} ns to compensate deviation and drift relating to a system wide clock. This precisely adjustable clock is used by the synchronisation module to manage and maintain a stable time base.

B. Architecture

TTEthernet is an extension of the standard Ethernet protocol. Therefore, the prototype is built on top of the Ethernet hardware abstraction layer (HAL) and provides portability to any hardware, which is capable of standard Ethernet communication. To apply a RT Ethernet protocol, a couple of modules are essential. Figure 1 depicts the concept of the TTEthernet stack, which consists of a synchronisation module, a scheduler and a buffer structure. The TTEthernet capable API provides functionality and setup through a configuration module to all applications running on the prototype.

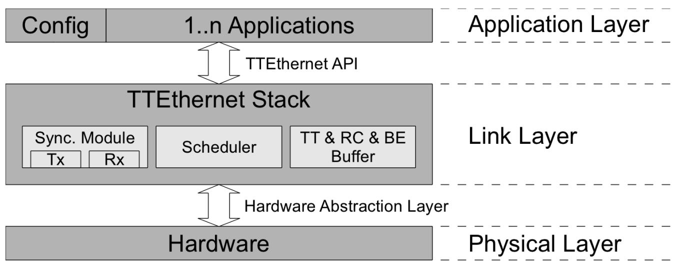

Fig. 1. Conceptual overview of the TTEthernet capable prototype

Synchronisation module: To maintain a synchronised system wide clock, a dedicated time synchronisation module is necessary to constantly adjust the local time.

Contrary to the discrete master and client state-machines specified by TTTech [2], we decided to provide a sub-module for reception and transmission of synchronisation frames respectively. By generalising the reception and computation of protocol control frames, we have the ability to use the same receive sub-module in all synchronisation roles. Complete synchronisation client functionality is provided by the sub-module. If the prototype is configured as synchronisation master, the transmission sub-module is activated, too. This enables each participant to down or upgrade dynamically and allows master nodes to preserve synchronisation even in case of an error without influencing scheduling.

Scheduler: The scheduling is needed to maintain periodic events like task execution or sending TT messages. Further, it has to handle external triggered events like callback tasks. This demands a hybrid design. There are two approaches which are known as preemptive and non-preemptive.

For non-preemptive scheduling the starting point in time as well as the duration of an event has to be known. This multiplies the number of entries to be scheduled, unnecessarily. Furthermore, due to the fact that scheduled tasks are implemented in the application, and the unpredictable occurrence of external events, the determination of these timings is not reliable.

In contrast for preemptive scheduling this is not needed. Priorities for each potential occurring event have to be defined, building a hierarchical order, which avoids interruption of critical tasks like TT transmission by BE events. This assumes that every possibly occurring event has to be specified as part of a clearly defined priority classification.

Since TTEthernet requires prioritisation and offline configured events, a preemtive scheduling approach is chosen.

Buffer: The TTEthernet protocol binds every TT and RC

TABLE I
SYSTEM LOAD WITH DIFFERENT RECEIVING MODES AND FRAME SIZES

reception strategy	**execution time per frame**	**system load at max. frame size**	**system load at min. frame size**
event-based:	5.43 µs	4.4 %	89.3 %
periodical:	4.67 µs	3.8 %	76.8 %

message to a dedicated buffer. Each is arranged either as queued or double buffer. Through a dedicated interface the application is able to define its own buffer structures, too.

In general queued buffers are intend to handle BE and RC traffic. By using this buffer type the message order is preserved for transmission and reception. Contrary, double buffers are designed to always supply the newest message to the network interface or the applications and thus are mainly used to handle TT traffic.

In exception, synchronisation messages which are principally marked as RC are double buffered by default to assure that the current transparent clock is provided at all times. All buffer types support concurrent access and provide a handler of the current message to every application.

IV. IMPLEMENTATION

The design was implemented on the previously introduced microcontroller.

A. Burst Behaviour & Dropping of Frames

The controller provides 32 Kbyte of memory per communication channel. In this configuration, the standard Ethernet HAL is capable of storing 20 frames, due to overhead through management structures and the predefined slots for full sized frames. Combining the infrastructure of two channels doubles the amount of frames. To ensure that critical frames are never dropped due to full buffers through BE bursts, a dropping algorithm has to prune the minor relevant frames.

Since the number of slots is limited, bursts of minimum sized frames are the worst case for the system. Resulting from propagation time and interframe gap it has to process each frame in 6.08 µs. To reduce processing time, all received frames are periodically fetched when no TT communication is expected, resulting in a periodical buffer clean, too. This period has to be configured as long as possible without the risk of losing frames. For TT reception, the receiver acts event-triggered to maintain a low and predictable delay for message processing and callback execution at the cost of speed. Through delegating send and receive attempts to the respective communication cores, the delay is constant even for varying frame sizes.

To give an overview of the system load with different receiving strategies, table I shows a comparison between best and worst case relating on frame size. Though, the load is in linear dependency to the amount of received data, the execution time remains constant for every strategy used by the system.

B. Improved Scheduling

The scheduling uses an advanced interrupt controlled approach to maintain different priorities triggered by the local system time. The system time is bound to a single delegated FIQ (Fast Interrupt Request) to trigger execution with the highest priority, minimum delay and maximum precision in time. Every upcoming event is registered to the system time module to be triggered at a specified point in time. This avoids unnecessary CPU load in idle times and makes a classification in micro- and macro-ticks, commonly used by scheduling approaches, dispensable. Thus, the achieved precision in event registration is related to the modules frequency and based on our hardware 10 ns. Once the FIQ has been triggered, the execution is delegated to a specified IRQ representing the priority of the current event. Since this is a preemptable approach, lower prioritised events are interruptible by those with higher priority. Multiple requests with the same or lower priority are chained and the sequence of execution is ordered in descending priority. This also applies to the external triggered events issued by the hybrid capability of the scheduling algorithm.

Due to the introduced approach, the execution time of non RT tasks has not to be estimated, which reduces the complexity of the scheduling algorithm significantly.

C. Memory Management & Buffering

The HAL only supports standard Ethernet communication and therefore is not able to distinguish between TT, RC and BE frames. This demands an extension of the provided memory organisation. Every TTEthernet buffer, even those that are defined and created by the application through the appropriate interface, is part of a pool, which is directly mapped to all memory banks of the Ethernet ports. Due to the fact that allocation and release of every frame is related to basic memory operations performed on the data pool, this approach avoids the usage of expensive copy operations and therefore is able to handle the high throughput.

In matter of complexity, arrangement of buffers for each message ID allows to transfer CPU extensive operations to initialisation phase, which reduces the complexity of any buffer operation executed in scheduling to $\mathcal{O}(1)$.

V. EVALUATION RESULTS

The test setup shown in figure 2 was built with three prototypes and a standard PC as traffic generator connected via a TTEthernet switch. One controller (1) was configured as synchronisation master, the other two as clients. One client (2) sends TT messages and the other (3) RC messages to the master. The cycle length of the schedule for TT message transmission was 3 ms, whereas the RC messages were defined with a bandwidth allocation gap (BAG) of 2 ms. A standard PC generated best-effort traffic with minimum frame size, which was routed to every prototype to generate load. An oscilloscope monitored the schedule of all prototypes to verify the synchronisation status and its precision. Furthermore, every

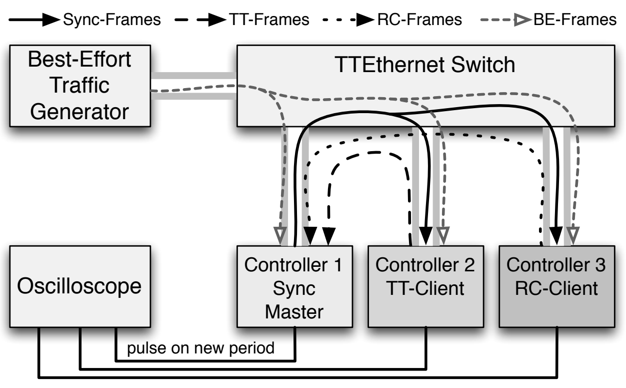

Fig. 2. Overview of the test-bed used for evaluation

received frame was timestamped by each controller to trace jitter and latency.

The analysis showed that incoming TT frames were always received in schedule with a maximum jitter below 1 µs. The scheduled receive window was 10 µs. During this time all other messages are not allowed to be send, because Ethernet frame fragmentation is not specified by TTEthernet. This proves that even under heavy system load, scheduling of critical tasks operates predictably and the dropping of best-effort frames does not influence the RT behaviour. The prototypes run more than 10,000 periods under test, where no event was executed out of schedule nor appeared any indication that this condition would be violated in further testing.

For rate-constrained traffic the test shows the reliable compliance with the bandwidth allocation gaps. Since the media is not reserved for RC messages, those frames can be delayed for a maximum of a single BE frame due to a busy interface.

To evaluate the quality of the synchronisation algorithm, the distance of the cycles of master and both clients were measured and results in a deviation below 0.5 µs. During startup phase, the system initialises and the synchronisation process needs to stabilise due to its two stage control algorithm. This time can be specified as less than 2 s.

VI. Conclusion & Outlook

RT Ethernet is a realistic candidate for next generation in-car backbones. Therefore, realistic prototypes of RT Ethernet based ECUs gain importance for both research and industry.

This paper presented a RT Ethernet prototype platform for automotive applications that is able to serve traffic streams of applications with different requirements concerning bandwidth and timing in parallel. Due to an advanced interrupt driven scheduling, the system is able to execute tasks, like driver assistance, with strict respect to the configured priorities.

The evaluation has shown that micro controller based clients can fulfill the rigid temporal requirements of RT Ethernet. Our analysis proves, that the proposed architecture maintains its functionality even under heavy load and is therefore suitable for applications in safety critical environments.

The prototype is utilised in our test setups to evaluate RT Ethernet based implementations of steer-by-wire and camera based driver assistance.

Further work will also contain real-time Ethernet gateway designs for established automotive technologies like CAN or FlexRay on the same prototype hardware. In particular the modular communication channel design supports the development of such gateways. For applications with high demand on computing power, for example image processing, it will be possible to separate the communication controller from the host by using the dual port memory as communication node interface.

References

[1] W. Steiner, "TTEthernet Specification," TTTech Computertechnik AG, Nov. 2008. [Online]. Available: http://www.tttech.com

[2] TTTech Computertechnik AG, Wien. [Online]. Available: http://www.tttech.com

[3] Honeywell International. [Online]. Available: http://www.honeywell.com

[4] TTTech Computertechnik AG, "TTEthernet Application Programming Interface," TTTech Computertechnik AG, Dec. 2008. [Online]. Available: http://www.tttech.com

[5] PROFIBUS & PROFINET International, "Profinet," Karlsruhe. [Online]. Available: http://www.profibus.com/technology/profinet

[6] EtherCAT Technology Group, "EtherCAT." [Online]. Available: http://www.ethercat.org

[7] H. Charara, J.-L. Scharbarg, J. Ermont, and C. Fraboul, "Methods for bounding end-to-end delays on an AFDX network," in *18th Euromicro Conference on Real-Time Systems*, 2006.

[8] Aeronautical Radio Incorporated, "Aircraft Data Network," ARINC, Annapolis, Maryland, Standard 664, 2002.

[9] SAE - AS-2D Time Triggered Systems and Architecture Committee, "Time-Triggered Ethernet (AS 6802)," 2009. [Online]. Available: http://www.sae.org

[10] T. Steinbach, F. Korf, and T. C. Schmidt, "Comparing Time-Triggered Ethernet with FlexRay: An Evaluation of Competing Approaches to Real-time for In-Vehicle Networks," in *8th IEEE Intern. Workshop on Factory Communication Systems*. Piscataway, New Jersey: IEEE Press, May 2010, pp. 199–202.

[11] 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*, 2011, to appear.

[12] P. Grillinger, A. Ademaj, K. Steinhammer, and H. Kopetz, "Software Implementation of Time-Triggered Ethernet Controller," in *Workshop on Factory Communication Systems*, 2006, pp. 145–150.

[13] H. Kopetz, A. Ademaj, P. Grillinger, and K. Steinhammer, "The time-triggered Ethernet (TTE) design," in *Eighth IEEE International Symposium on Object-Oriented Real-Time Distributed Computing, 2005. ISORC 2005.*, May 2005, pp. 22–33.

[14] K. Steinhammer and A. Ademaj, "Hardware Implementation of the Time-Triggered Ethernet Controller," in *IFIP Advances in Information and Communication Technology*, 2007, pp. 325–338.

[15] J. Lipfert, "Technical Data Reference Guide - netX500/100," Hilscher GmbH, Dec. 2008. [Online]. Available: http://www.hilscher.com

[16] Robert Bosch GmbH, "Controller area network." [Online]. Available: http://www.semiconductors.bosch.de/