Time-Triggered Fieldbus Networks – State of the Art and Future Applications

(Invited Paper)

Wilfried Elmenreich
Lakeside Labs, Mobile Systems Group,
Institute of Networked and Embedded Systems
University of Klagenfurt
wilfried.elmenreich@uni-klu.ac.at

Abstract

The time-triggered paradigm encompasses a set of concepts and principles that support the design of dependable real-time systems. By using the properties of physical time and the mechanism of clock synchronization, coordinated interaction between distributed nodes can be facilitated. This paper briefly explains the time-triggered approach, defines a taxonomy for real-time requirements and discusses seven protocols that provide time-triggered features. Finally, two application examples are given that show the potential of the time-triggered approach.

1. Introduction

Whenever a system interacts with a real environment through its sensors and actuators, the aspect of time has to be considered in order to achieve a correct and meaningful behavior. Although soft real-time systems may cope with this problem by just providing enough processing and communication power, there is a fundamental difference in designing architectures that are required to guarantee hard real-time properties.

The time-triggered approach is an established paradigm for designing hard real-time systems with predictable and guaranteed behavior. The application of the time-triggered approach is beneficial from the viewpoint of (i) establishing predictable real-time response times within the network, (ii) reducing system complexity, and, therefore, facilitating the design of real-time applications, by introducing an architecture with a global notion of time and well-defined interfaces, (iii) facilitating the processing of sensor data, since measurements can be synchronized and interpreted on a global timescale, and (iv) enable the coordination of timely correlated actions of actuators. The time-triggered approach has already been successfully applied to safety-critical applications in cars [29], railway control systems [9] and to flight-critical functions in aircraft and aircraft engines [29].

The objective of this paper is to provide an overview of the currently available fieldbus networks providing time-triggered features and to discuss future applications showing the potential of the time-triggered approach in fieldbus networks.

This paper is structured as follows: The next section shortly reviews the time-triggered paradigm. Section 3 discusses advantages and disadvantages of the time-triggered approach in the context of fieldbus applications. Section 4 presents a taxonomy of real-time requirements for fieldbus systems. Section 5 reviews existing fieldbus systems that apply the time-triggered concept, at least partially. Section 6 gives application examples that show further potential of the time-triggered approach for fieldbus systems. Section 7 concludes the paper.

2. The Time-Triggered Paradigm

There are two major design paradigms for constructing real-time systems, the *event-triggered* and the *time-triggered* approach. In principle, an event-triggered system follows the principle of reaction on demand. In such systems the environment enforces temporal control onto the system in an unpredictable manner (interrupts), with a lot of undesirable problems of jitter, missing precise temporal specification of interfaces and membership, scheduling etc. On the other hand, the event-triggered approach is well-suited for sporadic actions and data, low-power sleep modes, and best-effort soft real-time systems with high utilization of resources. Event-triggered systems do not ideally cope with the demands for predictability, determinism, and guaranteed latencies – requirements that must be met in a hard real-time system.

Time-triggered systems derive control from the global progression of time, thus the concept of time that appears in the problem statement appears also as basic mechanism for the solution:

A real-time system is time-triggered if the control signals, such as sending and receiving of messages or recognition of an external state change are derived solely from the progression of a (global) notion of time (cf. [10]).

This approach supports a precise temporal specification of interfaces and the implementation of "temporal firewalls" to protect error propagation via control signals. The Time-Triggered Architecture (TTA), a computing infrastructure for the design and implementation of dependable distributed embedded systems [12], implements the time-triggered paradigm and supports membership identification,

interoperability, and replica determinism.

A basic concept in the time-triggered paradigm is the *global time*. For most real-time applications it is sufficient to model time according to Newtonian physics without regarding relativistic effects [14]. Unlike the concept of logical clocks [17], the global time is thus bound to physical time with a given accuracy. In order to establish a meaningful global time with a given granularity, an ensemble of physical clocks must be synchronized with a precision better than the intended granularity. Since all actions are derived from the global time, the process of creating synchronization is of utmost importance. A fault during synchronization will thus propagate to many relevant parts of the system. Therefore, clock synchronization approaches for time-triggered systems typically implement concepts of fault tolerance and self-stabilization. Some faults might even cause two sets of correctly working hardware to diverge into two cliques with different synchronization. Such situations are handled by clique avoidance mechanisms, typically by keeping the larger clique and restarting and reintegrating the other one. Novel design such as the BRAIN approach [25] allow to set special policies for startup and restart in order to keep the nodes which are most important to the application or which are difficult to restart.

The global time is used to define the instances when communication and computation of tasks take place in a time-triggered system. Communication comprises send and receive operations by particular nodes. Typically, sending messages takes place in a broadcast manner. The message length and message sender are known *a priori* according to the predefined message schedule.

Computation is realized by the execution of *Simple Tasks*, that is tasks that consume their input at task start and provide their output with task completion. Simple Tasks do not have synchronization points within the task, and, therefore, cannot be blocked. For each task an *a priori* known upper bound for their Worst Case Execution Time (WCET) [27] is assumed.

Using a static scheduling algorithm, the tasks and messages are scheduled to form a collision-free communication pattern where it is guaranteed that all tasks can finish in time before their results are used.

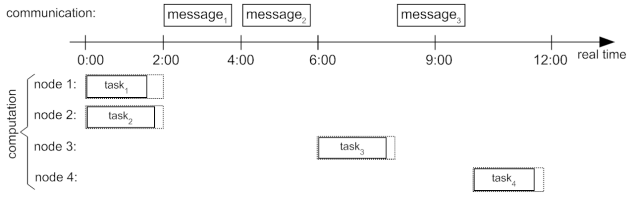

Figure 1. Time-triggered scheme for communication and computation

Such a communication and computation pattern forms so-called *rounds* which are periodically repeated. Figure 1 depicts an example for such a time-triggered schedule. For demonstrative reasons, the timeline has been denoted according to the 12 hours as on the face of an analog clock. The boxes above the timeline represent the planned messages on a shared communication medium. Collisions are avoided by design using Time Division Multiple Access (TDMA) scheduling. The boxes below the timeline correspond to the execution of local tasks. The upper bound of a task's execution time is denoted by the dotted lines.

3. Applicability of the Time-Triggered Approach

Depending on the application type, the time-triggered approach has its strengths and weaknesses; both are briefly discussed in the following. Note that there exist also hybrid approaches that combine the time-triggered approach with a more flexible approach (e. g., [22], [24]).

3.1 Advantages of the Time-Triggered Approach

With respect to embedded real-time systems, the time-triggered approach has proven to show the following advantages:

- The low jitter for message transmission and task execution is especially advantageous for distributed control loops.
- The predictable communication scheme simplifies diagnosis of timing failures. Furthermore, a timing failure of a node can be barred from the bus using the concept of a bus guardian.
- The periodically transmitted messages enable a short and bounded error detection latency for timing and omission errors.
- The principle of resource adequacy guarantees the nominative message throughput independent of the network load. Problems like increasing delays at message floods or thrashing [3] are avoided by design.
- Due to the predefined schedule, it is possible to derive the message ID and the message sender from the instant when a message was received. Using this information enables high protocol efficiency.
- The time-triggered paradigm avoids bus conflicts using a TDMA scheme, making an explicit bus arbitration obsolete.
- By using a *sparse time base*, replica determinism between time-triggered components can be achieved without the need of complex agreement protocols.
- The TTA supports temporal composability, i. e., the constructive design of dependable distributed real-time systems out of previously validated components while retaining the previously validated properties [13].

3.2 Disadvantages of the Time-Triggered Approach

A time-triggered system is a specialization of an event-triggered system where only the time is used as a trigger. Therefore, there are problems, for which an event-triggered approach is better suited than a strict time-triggered scheme:

- When a system is required to achieve a low energy consumption over time, as it is the case for wireless sensor networks. In event-triggered systems, messages are created on demand, i. e., on the occurrence of respective events. In the time-triggered scheme messages and computations are triggered periodically which causes a permanent and constant energy consumption even during minimum load situations. However, for wireless systems with a low duty cycle the time-triggered approach can be also a way to enhance the system lifetime – see Section 6 for an example.
- When the average response time of the system is of concern. Event-triggered systems may outperform time-triggered systems in that aspect since the latter are designed to have a constant response time independent of the system load, thus the average response time equals the worst-case response time.
- Time-triggered systems have to plan for an upper bound for the execution time of each task, in contrast an event-triggered approach can work with weaker assumptions such as a global time budget for a set of tasks.
- When it is difficult to fit messages with differing periods into a static schedule. The length of the static schedule is defined by the least common multiple of the message periods, which can lead to a very extensive static schedule causing memory problems in embedded systems.
- In wireless scenarios where a considerable rate of link failures cannot be handled by the standard time-triggered approach. If the node connectivity is also dynamic, a dynamic re-routing algorithm does not allow for a static time-triggered approach.
- When it is not possible to establish the required precision of the global time, e.g., in state-of-the-art chip design with clock frequencies of several Gigahertz, it is very difficult to distribute the clock signal across the chip [28, 20].

Most of the problems listed above arise in applications, where non-real-time or soft real-time requirements are prevalent over dependability issues such as reliability and safety.

4. Taxonomy of Real-Time Applications

A real-time computer system must react to stimuli from the controlled object (or the operator) within time intervals dictated by its environment. The instant at which a result must be produced is called a deadline. [10, p.2]

This definition does not include systems like a real-time clock that does not react on stimuli but has to generate output strongly aligned to the progression of real-time. Since an embedded system typically can have various tasks to be performed with respect to the progression of real-time, we present the following classification of real-time applications (typically, an embedded system will apply several of the following tasks):

- *Performing some action locally with respect to real time*, such as generating a particular Pulse-Width Modulation (PWM) signal or making a measurement every 100 ms. If the quality (drift) of the (local) clock source is sufficient to provide a useful time base, this application can be achieved without the need for clock synchronization.
- *Timestamping events* can be used to temporally relate measurements to each other and, in consequence, make global interpretations based on a set of distributed measurements. In order to create timestamps with a global validity, a synchronized global time is required among the participating nodes. In most cases, clock synchronization has to be done periodically in order to compensate for the drift of the local clocks. Once a global time is established, timestamping does not impose real-time requirements on the communication system, since timestamped events can be locally stored.
- *Bounded maximum reaction time* requires the communication system to deliver messages within a specified time interval. Standard feedback control algorithms also require low message jitter in order to work correctly.
- *Globally synchronized actions* require the synchronized generation of action triggers in different nodes. This can be achieved by a multi-cast message or assigning actions to an instant on the globally synchronized time scale.

This classification can be combined with the notion of hard/soft real-time requirements, that is a deadline can be either *hard*, i. e., deadlines must be met under all circumstances or *soft*, i. e., the system is still of use if deadlines are violated infrequently. Some architectures implement a subset of the described features or provide different features with hard or soft real-time behavior. For example, the LAAS architecture [1] for component-based mobile robots specifies local hard real-time behavior for tasks such as a locally closed control loop or the instrumentation of an ultrasonic sensor, while at higher levels, e. g., for globally synchronized actions it provides only soft real-time behavior.

5. State of the Art of Time-Triggered Fieldbus Systems

5.1. Time-Triggered Protocol (TTP)

Time-Triggered Protocol (TTP), Time-Triggered Protocol for SAE class A applications (TTP/A) and Time-Triggered Ethernet (TTE) are protocols out of a family of protocols for the TTA. TTP (or TTP/C) focuses on the interconnection of components in order to form a highly dependable real-time system that is sufficient for critical applications such as X-by-wire in the automotive and avionics domains. TTP implements a replicated bus system and a guardian that prevents babbling idiot failures. Currently, it supports transmission rates of up to 25 MBit/s.

The protocol does not require a central node as time master or bus manager, instead the nodes interact at startup to agree on a common synchronized timebase that is used to define instants of action and communication in the system. The protocol is fault-tolerant to a single arbitrary faulty node in the start-up phase as well as during the synchronous operation.

The time-base that has been established by the start-up is used for distributed message scheduling as well as for local process scheduling. Furthermore the timebase is available to the application as global synchronized time. Thus, TTP inherently supports all four kinds of real-time requirements as defined in Section 4.

TTP assumes to have *a priori* defined action and communication patterns. In order to support also event-triggered legacy systems such as CAN applications, the time-triggered layer of TTP has been enhanced to emulate event messages.

5.2. TTP/A

TTP/A [15] is a time-triggered master-slave fieldbus system. The master establishes a global time and announces the beginning of a communication round by issuing a so-called *fireworks message*. Following the fireworks, all nodes follow a common collision-free communication pattern as depicted in Figure 2.

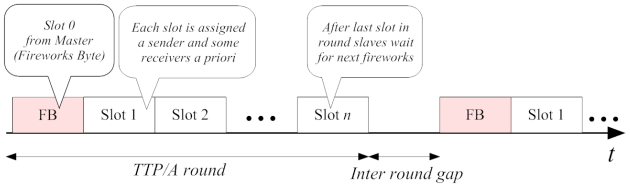

Figure 2. TTP/A communication round

As in TTP, the schedule and global time is known to all nodes, however TTP/A supports an on-line update function of the schedule during operation. TTP/A inherently supports all four kinds of real-time requirements as defined in Section 4.

TTP/A was designed to support an easy and economically feasible integration of sensors and actuators into a real-time network. TTP/A can be implemented in software on low-cost microcontrollers. The interface concept of TTP/A supports a modular design and an easy integration and management of transducers.

In contrast to TTP, TTP/A has no fault-tolerant capabilities that can handle arbitrary node faults, but is more flexible by providing means of online configuration. The interface implemented by the TTP/A protocol has been standardized with the OMG Smart Transducer Interface [23]. An implementation of TTP/A for Atmel AVR is available under an Open Source License[1]. At present, TTP/A supports transmission rates of up to 100 KBit/s.

5.3. Time-Triggered Ethernet

TTE [11] is a time-triggered protocol that uses Ethernet as physical layer. It establishes a global synchronized time that is then used to execute a distributed time-triggered communication scheme. TTE allows also the use of standard Ethernet frames to support event-triggered data, whereas a dedicated TTE-Switch takes care that time-triggered frames are not delayed by other frames.

TT Ethernet builds on a well-known physical layer and is intended to support all types of applications, from simple data acquisition systems, to multimedia systems up to safety-critical real-time control systems.

5.4. Flexray

Flexray [5] is a fieldbus system for automotive applications such as X-by-wire. Flexray has been developed and is supported by a consortium of automotive manufacturers and suppliers including BMW, DaimlerChrysler, Volkswagen, Bosch, General Motors, Freescale and NXP Semiconductors.

The FlexRay protocol consists of a time-triggered part, where messages are scheduled according to an *a priori* defined TDMA schedule and a flexible part supporting sporadic traffic. The flexible part is based on the Byteflight protocol [26] that uses minislotting in order to provide a collision-free communication that does not interfere with the time-triggered part. Flexray implements a global synchronized timebase that supports synchronized actions.

Flexray supports a communication speed of up to 10 MBit/s.

5.5. TTCAN

Time-Triggered Controller Area Network (TTCAN) [6] is a time-triggered protocol that builds on the event-triggered CAN protocol [7]. In its extension level 2, TTCAN establishes a global synchronized time derived from periodically broadcasted synchronization frames by a time master node. This synchronized time can be used to program an event-trigger in the application code, thus enables synchronized actions.

The TTCAN protocol is implemented in hardware using a dedicated TTCAN controller. TTCAN integrates time-triggered frames with standard event-triggered frames. The event-triggered part uses the standard CAN arbitration to avoid collisions.

5.6. WorldFIP and Foundation Fieldbus

WorldFIP [21] and Foundation Fieldbus [30] are industrial fieldbusses providing a hybrid approach for transmitting time-triggered and event-triggered data. Both approaches use a similar scheme, since in fact the Foundation Fieldbus is a functional superset of WorldFIP. The Foundation Fieldbus variant FF H1 is indented for automation applications at field level.

The concept is implemented by *periodic* and *aperiodic* processes. Periodic processes are time-triggered processes initiated at predetermined points in time. Aperiodic processes handle event-triggered traffic that is delivered as soon as possible but with considerable jitter in the message delivery time. Both message types are scheduled on a single bus with a MAC protocol based on centralized arbitration by a bus manager.

[1] http://www.vmars.tuwien.ac.at/ttpa/

Name	Bandwidth	Clock synchronization	Dependability	TT concept	Level of Automation
TTP	25 MBit/s	fault-tolerant distributed	high	fully TT	cell level
TTP/A	100 kBit/s	master-slave	low	fully TT	field level
TTE	100 MBit/s	fault-tolerant distributed	high	coexistent TT and ET traffic	field or cell level
Flexray	10 MBit/s	fault-tolerant distributed	high	coexistent TT and ET traffic	field level
TTCAN	1 Mbit/s	master-slave	medium	coexistent TT and ET traffic	field or cell level
FF H1	31.25 kBit/s	master-slave	medium	TT only at application level	field level
LIN	20 kBit/s	master-slave	low	TT only at application level	field level

Table 1. Feature comparison of time-triggered buses.

From the application's viewpoint, the Foundation Fieldbus could be considered a time-triggered protocol. Note however the following properties that are untypical for time-triggered systems: (i) The scheduling table for the periodic data is not provided to the single nodes (ii) The nodes are not aware of a global time which could be used in the application (iii) The scheduling decisions are not directly based on a global time, but are done by the bus manager (which bases its decision on timing)

5.7. Local Interconnect Network (LIN)

LIN is tailored to serve as sub-bus for body electronics in automobiles. Design guidelines for LIN had been low hardware costs, robust operation, real-time support, and easy implementation (e.g. by a software protocol layer in an 8-Bit Microcontroller Unit (MCU)).

LIN is basically a polling protocol, where a central master issues request messages to the slave nodes. The master node acts also as a gateway to a higher network. The slave nodes are smart transducers which are waiting for specific request messages in order to set a control value or to send a measured value as reply. The master issues request messages on a predefined schedule, while the slave nodes are not aware of a global time or the current state of the schedule. This simplifies the implementation of the slave nodes, but does not support coordinated actions like synchronized measurements. Due to the polling principle (requesting a value involves the request message, a "thinking time" for the slave node and sending the reply message), the effective bandwidth of a LIN network supports only applications with low bandwidth requirements, such as less critical body electronic functions in cars.

In order to save bandwidth, the LIN 1.3 specification was enhanced to LIN 2.0 including unconditional frames and event-triggered frames. These frames must be also triggered by the master, but the slaves' response may depend on local information, e. g., if a measurement value has changed since the last time or not.

The polling principle in LIN makes a node's implementation very simple, but causes an overhead on the network due to the frequent message requests from the master. Moreover, since the LIN slaves do not know the time of a request *a priori*, it becomes difficult to time a measurement adequately or to synchronize measurements.

5.8. Comparison

Table 1 depicts the main features of the presented time-triggered protocols. The level of automation refers to the concept of field, cell, and management level in industrial automation.

TTP is considered mainly feasible for the cell level when considering cost constraints. TTE might be a promising candidate for the cell level (due to its high speed and dependability) as well as for the field level (because of its expected low cost). A similar prospect exists for TTCAN.

At the low-cost end, TTP/A, LIN, and Foundation Fieldbus provide time-triggered solutions at the fieldbus level. However, only TTP/A can be considered a true time-triggered protocol that fully utilizes the concept of global time and thus supports all four types of applications as discussed in Section 4. While LIN is the solution with the lowest cost, Foundation Fieldbus is the only bus with considerable market share in the automation domain among those three.

6. Auspicious Applications of Time-triggered Fieldbus Systems

The previous sections gave an overview of the state of the art of applying the time-triggered paradigm in fieldbus systems. In the following, we will discuss two application examples from the domain of wireless systems and high-dependable control systems.

6.1. Time-triggered Wireless Networks

Up to now, real-time wireless fieldbus systems were mostly reduced to cable replacement [16] using IEEE 802.15.1/Bluetooth (e. g., replacing traditional serial interfaces such as RS232, RS422 and RS485 by a wireless point-to point or master/slave multi-point connection) and applications with low or soft real-time requirements. Other initiatives like the R-Fieldbus [8] have not found much acceptance among industrial applications.

The time-triggered approach has found even less application in the wireless domain, due to the strict limitations on bandwidth and battery lifetime. In wireless systems the event-triggered paradigm is typically preferred over the time-triggered one, since in event-triggered systems a mes-

sage is only sent if there is the need to send. In that way less energy is consumed for sending data.

However, typical radio nodes such as the AVR®Z-Link™802.15.4/ZigBee nodes have a similar high power consumption for sending and being in receive mode. Using a time-triggered communication scheme, the energy consumption can be reduced by entering a sleep mode and turning off the radio in periods, when, due to *a priori* knowledge no transmission is to be expected.

Thus, for wireless systems with a low duty cycle the time-triggered approach can be less energy consuming since the knowledge of message transmission instants allows to suspend the receiving units for the duration between two transmissions. A case study using this approach has been implemented by establishing global synchronization using a self-organizing algorithm and then applying a time-triggered communication scheme.[18]

Results (see Figure 3) indicated the possibility to extend the lifetime of the battery powered nodes up to a factor of about four (depending on the duty cycle).

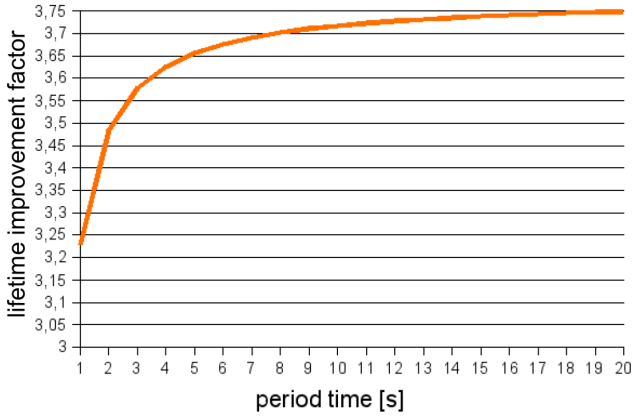

Figure 3. The lifetime improvement as a function of the period time.

6.2. Fault-tolerant Actuating

In the time-triggered protocols that claim to be fault-tolerant (TTP, TTE, Flexray, TTCAN) the fault hypothesis mainly focused on network faults like faulty messages. However, when it comes to a fault-tolerant application, also the points of data acquisition, i. e., measurement and actuation is of interest. Solutions for fault-tolerant and robust measurement can be found in [19, 2, 4]. In the following we briefly discuss fault-tolerant actuation in the context of time-triggered systems.

Time-triggered systems are of special interest to fault-tolerant actuating, since in most cases, (i) a faulty action cannot be undone at a later instant, and, (ii) the non-synchronous execution of correct actions from independent actuators may lead to unwanted behavior. Think, for example, of a car having wheels that can be steered independently. First, an incorrect action, like an unwanted turn might lead to an uncontrollable skidding car and further consequences. Second, when deciding to make a turn having one wheel turning before the other one due to synchronization problems might lead to the same behavior, even if the two actions separately may be within tolerance.

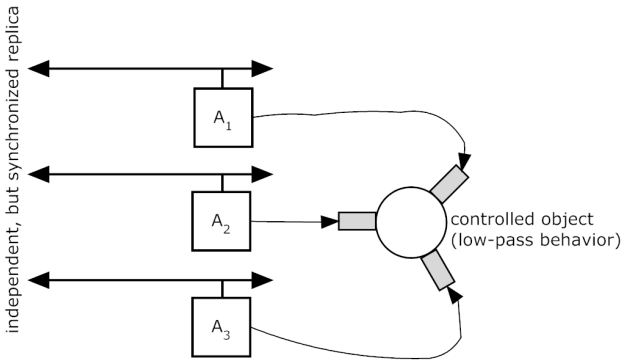

Figure 4. Fault-tolerant actuating using triple modular redundancy.

Figure 4 gives an example of a fault-tolerant actuation. The controlled object is assumed to show low-pass behavior due to physical issues. In order to ensure correct behavior even in the case of failure of one of the replicated actuators, it must be ensured that (i) the control signals are synchronized with a precision better than the cut-off frequency of the low pass element, and, (ii) the control decisions of the correctly operating components are replica-deterministic. In order to achieve these requirements, a time-triggered architecture using a precisely synchronized fieldbus is adequate. Furthermore, a closely synchronized action setting is of advantage in order to minimize mechanical stress that is created if actuators work against each other. Thus, time-triggered systems support a timely closely correlated action setting while the decision process can be derived independently redundant via different fault containment regions.

7. Conclusion and Outlook

The time-triggered paradigm is more than just applying a TDMA message scheduling to a bus. The discussed protocols differ, besides parameters such as cost and speed also in the way, how much the concept of time is used as the solution to a problem than just being the problem itself. TTP, TTE, Flexray show similarities in the way that they implemented a distributed clock synchronization algorithm and provide a global synchronized time that can also be used by the application to achieve real-time tasks. TTP/A and TTCAN use a central master to provide some coordination, but nevertheless establish a global synchronized time in all the nodes.

On the other hand, LIN, WorldFIP and Foundation Fieldbus have chosen a concept where the slave nodes are triggered by a central master instead of being triggered by the

progression of time. Although at application level this approach provides similar features like a true time-triggered one, there are shortcomings when it comes to diagnosis and support for coordinated actions. These features are still possible with these systems, however it requires additional effort to implement them, while in fully time-triggered systems they come for free.

Acknowledgments

I would like to thank my colleagues Michael Gyarmati and Michael Paulitsch for proofreading and comments on an earlier version of this paper. This work was supported by the Austrian FWF project TTCAR under contract No. P18060-N04.

References

[1] R. Alami, R. Chatila, S. Fleury, M. Ghallab, and F. Ingrand. An architecture for autonomy. *International Journal of Robotics Research*, 17(4):315–337, Apr. 1998.

[2] P. Chew and K. Marzullo. Masking failures of multidimensional sensors. In *Proceedings of the 10th Symposium on Reliable Distributed Systems*, pages 32–41, Pisa, Italy, Oct. 1991.

[3] P. J. Denning. Thrashing: Its causes and prevention. In *Proceedings AFIPS Fall Joint Computer Conference*, volume 33, pages 915–922, 1968.

[4] W. Elmenreich. Fusion of continuous-valued sensor measurements using confidence-weighted averaging. *Journal of Vibration and Control*, 13(9-10):1303–1312, 2007.

[5] Flexray Consortium. *FlexRay Communications System Protocol Specification Version 2.1*, 2005. Available at http://www.flexray.com.

[6] T. Führer, B. Müller, W. Dieterle, F. Hartwich, R. Hugel, and M. Walther. Time triggered communication on CAN (Time Triggered CAN–TTCAN). In *7th international CAN Conference*, 2000.

[7] R. B. GmbH. CAN specification version 2.0, Sept. 1991.

[8] J. Haehniche and L. L. Rauchhaupt. Radio communication in automation systems: the r-fieldbus approach. In *Proceedings of the IEEE International Workshop on Factory Communication Systems*, pages 319–326, Porto, Portugal, Sept. 2000.

[9] G. Heiner and T. Thurner. Time-triggered architecture for safety-related distributed real-time systems in transportation systems. In *Proceedings of the The Twenty-Eighth Annual International Symposium on Fault-Tolerant Computing*, pages 402–407, 1998.

[10] H. Kopetz. *Real-Time Systems, Design Principles for Distributed Embedded Applications*. Kluwer Academic Publishers, Boston, Dordrecht, London, 1997.

[11] H. Kopetz, A. Ademaj, P. Grillinger, and K. Steinhammer. The Time-Triggered Ethernet (TTE) design. In *Proceedings of the 8th International Symposium on Object-Oriented Real-Time Distributed Computing (ISORC)*, pages 22–33, Seattle, WA, USA, May 2005.

[12] H. Kopetz and G. Bauer. The Time-Triggered Architecture. *Proceedings of the IEEE*, 91(1):112–126, Jan. 2003.

[13] H. Kopetz and R. Obermaisser. Temporal composability. *IEE's Computing & Control Engineering Journal*, 13(4):156–162, Aug. 2002.

[14] H. Kopetz and N. Suri. Compositional design of RT systems: A conceptual basis for specification of linking interfaces. Research Report 37/2002, Technische Universität Wien, Institut für Technische Informatik, Vienna, Austria, 2002.

[15] H. Kopetz et al. Specification of the TTP/A protocol. Research Report 61/2002, Technische Universität Wien, Institut für Technische Informatik, Vienna, Austria, Sept. 2002. Version 2.00.

[16] K. Koumpis, L. Hanna, M. Andersson, and M. Johansson. Wireless industrial control and monitoring beyond cable replacement. In *Proceedings of the PROFIBUS International Conference*, pages C1:1–7, UK, June 2005.

[17] L. Lamport. Time, clocks, and the ordering of events in a distributed system. *Communications of the ACM*, 21(7):558–565, July 1978.

[18] R. Leidenfrost and W. Elmenreich. Establishing wireless time-triggered communication using a firefly clock synchronization approach. In *Proceedings of the Sixth International Workshop on Intelligent Solutions in Embedded Systems*, Regensburg, Germany, July 2008.

[19] K. Marzullo. Tolerating failures of continuous-valued sensors. *ACM Transactions on Computer Systems*, 8(4):284–304, Nov. 1990.

[20] J. Muttersbach, T. Villiger, H. Kaeslin, N. Felber, and W. Fichtner. Globally-asynchronous locally-synchronous architectures to simplify the design of on-chip systems. In *Proceedings of the twelfth Annual IEEE International ASIC/SOC Conference*, pages 317–321, Washington DC, USA, Sept. 1999.

[21] P. Noury. WorldFIP, IEC 61158 and the internet: A new look at fieldbuses, 1999. Available at http://www.worldfip.org/noury02.html.

[22] R. Obermaisser. *Event-Triggered and Time-Triggered Control Paradigms*, volume 22. Springer Real-Time Systems Series, 2005.

[23] Object Management Group (OMG). *Smart Transducers Interface V1.0*, Jan. 2003. Specification available at http://doc.omg.org/formal/2003-01-01 as document formal/2003-01-01.

[24] D. Paret. *Multiplexed Networks for Embedded Systems*. John Wiley & Sons, Ltd, 2007.

[25] M. Paulitsch and B. Hall. Starting and resolving a partioned BRAIN. In *Proceedings of the 11th International Symposium on Object-Oriented Real-Time Distributed Computing (ISORC)*, Orlando, FL, USA, May 2008.

[26] M. Peller, J. Berwanger, and R. Giessbach. Byteflight specification draft version 0.5. Technical report, BMW AG Munich, 1999.

[27] P. Puschner and A. Burns. A review of worst-case execution-time analysis. *Journal of Real-Time Systems*, 18(2/3):115–128, May 2000.

[28] P. J. Restle and A. Deutsch. Designing the best clock distribution network. In *Proceedings of the Symposium on VLSI Circuits Digest of Technical Papers*, pages 2–5, 1998.

[29] J. Rushby. A comparison of bus architectures for safety-critical embedded systems. Technical Report NASA/CR-2003-212161, National Aeronautics and Space Administration, Langley Research Center, Mar. 2003.

[30] Z. Wang, Z. Yue, K. Chen, Y. Song, and Y. Sun. Realtime characteristic of ff-like centralized control fieldbus and its state-of-art. In *Proceedings of the IEEE International Symposium On Industrial Electronics*, pages 140–145, 2002.