

Embedding Complex Embedded Systems in Ethernet-based Networks

Armin Wasicek, Christian El-Salloum, Oliver Höftberger
Vienna University of Technology, Austria
Institute for Computer Engineering
Email: {armin, salloum, oliver}@vmars.tuwien.ac.at

Abstract—The rollout of the Ethernet protocol is the next logical step in the evolution of embedded systems. Ethernet strives to replace prevalent domain-specific communication systems and unify fieldbus level communication enabling more efficient and flexible systems. This development is accompanied by the integration of embedded applications in large networks like the Internet. At the same time, embedded systems themselves face a paradigm shift towards Multi-Processor System-on-a-Chip (MPSoC) architectures. A dedicated on-chip network will connect several IP-blocks in such a system. Both trends lead to the anticipation the future embedded systems will deploy networks on different abstraction levels. Each network has different characteristics and goals. Particularly, embedded real-time systems require that temporal guarantees are handed over between these abstraction levels. This work focuses on the interconnection of multiple embedded systems, which can be as complex as a modern MPSoC in a larger Ethernetbased network. Our solution centers around a dedicated gateway component that implements the required interfaces towards the external Ethernet and the internal Network-on-Chip (NoC) networks. We address several challenges when designing a gateway for such an interconnection, e.g., temporal issues when interfacing timetriggered and eventtriggered networks, allocating addresses, security, and the synchronization of the time internal to an MPSoC to an external reference. Finally, we evaluate the feasibility and applicability of our solutions to the above listed challenges in a prototype of the TTSoC Architecture which is a novel research architecture for dependable real-time systems.

I. INTRODUCTION

The success story of Ethernet-based networks has started in the 90s by networking home and office computers and currently reaches the domain of embedded systems. Up to now, distributed embedded systems have been connected by deploying special purpose protocols like for example Controller Area Network (CAN). With the observable decrease of cost of Ethernet equipment and the requirement to exercise remote control over all kinds of embedded systems, standard Ethernet has the potential to prevail as a standard communication protocol for all kinds of embedded applications. Moreover, the specification of the IPv6 protocol by the Internet Engineering Task Force (IETF) and the resulting extension of the Internet Protocol (IP) addresses space laid foundation to the envisioned *Internet of Things* [1], where not only human beings interact over a Wide Area Network (WAN) but also machines.

The advantages of Ethernet that contributed to its dominance as the mainstream networking technology are its simplicity of installation, its connectivity to backbone networks, and – nowadays – its vast rollout. A major drawback of standard Ethernet is that it can give only best-effort guaranties whereas

many embedded systems have hard real-time requirements and need a predictable communication network. This has been acknowledged by the extension of the Ethernet protocol family through variants that satisfy hard real-time requirements at the highest Safety Integrity Levels. An interesting variant is Time-Triggered Ethernet (TTE) which was designed to support best-effort and predictable time-triggered traffic [2] over the same communication link. A commercial version of this protocol has been implemented for NASA's Orion spacecraft¹.

In order to deliver a common service within the scope of a larger system, single systems have to exchange information like process variables, system states, or control signals. A *gateway* is an intermediate node participating in two or more networks that redirects information between them. Kopetz states in [3] that *"the purpose of a gateway is to exchange relative views between two interacting clusters. In most cases, only a small subset of the information in one cluster is relevant to the other cluster."* Thus, a gateway is an integral element in every system architecture to connect distinct subsystems.

In this paper we focus on the design of such a gateway, which can be used to connect complex real-time systems like (e.g., an MPSoC) in a larger Ethernet-based network *maintaining certain network properties*. Due to the conflicting properties of standard Ethernet communication (e.g., best-effort, possible packet loss) and common requirements for real-time communication (e.g., timeliness, dependability), established solutions from other domains are not portable. Our solution to connect single systems is based around the TTE technology. The combination of event-triggered and time-triggered traffic in a single communication system allows maximum flexibility when integrating single systems.

Addressing the presented challenges, the contribution of this paper includes

- the design of a gateway connecting two real-time networks, and
- the temporal alignment of real-time networks.
- redirection messages between event-triggered and time-triggered networks

The remainder of this paper is organized as follows: Section II gives a problem statement when connecting networks of different paradigms. Section III elaborates on the system model and Section IV lists the requirements for the gateway. Next, Section V concerns with the design and implementation

¹<http://www.tttech.com/fileadmin/content/pdf/TTTech-NASA-Casestudy-Orion.pdf>

of a gateway and Section VI discusses its accordance to the requirements. We draw a conclusion in Section VII.

II. PROBLEM STATEMENT

During the past decades, the special requirements of different hard and soft real-time applications lead to a fragmentation of real-time networks into many protocols and standards. Each respective application area has its own highly specialized solutions. For instance, the automotive industry uses serial protocols like CAN, LIN, MOST, in automation buses like LON and ASi are deployed, and in airplanes AFDX and ARINC 429 are implemented.

The term *Island of Automation (IoA)* emphasizes this isolation of systems and protocols, because every application has its unique characteristics, constants, and common practices and represents therefore a virtual island. Formally, an IoA is a *disconnected group of systems with no other obvious integration point than the end user* [4]. This can be the case in an industrial plant, where many different machines work to achieve a common goal but don't exchange explicit control information other than by a human operator.

Modern automation systems are often linked to some backbone network for remote access or to coordinate actions among single systems. Connectivity is the trend, even the connection between enterprise and fieldbus systems is an integration goal [5]. One more recent term, which reflects this circumstance is *System-of-Systems (SoS)* which is *a system constructed from autonomous component systems* [6]. The keyword in this definition of an SoS is *autonomous* which means *independence with respect to existence, operation and/or evolution*. In this sense, when we connect several systems to an SoS, we want the overall system to achieve a certain goal, but the single system to stay autonomous. In order to prevent unwanted interaction between autonomous systems, we have to focus on the 'glue' between them, which is the gateways. By strictly determining what is *relevant information* – as in Kopetz's definition earlier – we can control and restrict the systems' interaction and avoid side-effects from the integration.

Relevant information does not only encompass a set of data values that are useful in another IoA, but also implicit properties like temporal accuracy [3], reliability, consistency, security, etc., that depend on the way the data values are handled and processed. For example, a time-triggered message transport service can be used to periodically broadcast state information. Each message transported by this service carries implicit timing information defined by the period of the message and its phase (the offset within a period). Every receiver of such a message will know within some bounds (the jitter) the message send instant. In the Time-Triggered Protocol (TTP) this information is even used for implicit clock synchronization [7]. During the transition of such a time-triggered message to an event-based network, it would lose this implicit information, because it might be queued (delayed) or even dropped at an intermediate node. The other way around – forwarding a message from a network which is not time-aware to a time-triggered network – is also not trivial. In order to send a time-triggered message in a timely fashion, it has to be present in the sender's Communication Network Interface (CNI) before its message send instant. The question

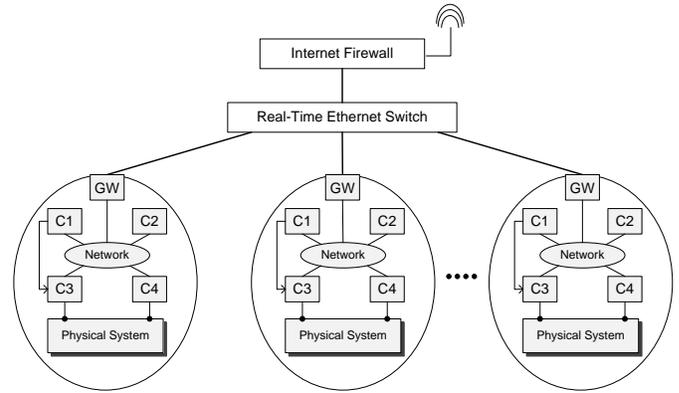


Fig. 1. Connection of several IoAs in a network

is, how can be guaranteed that a message from an event-based network is available at the genuine message send instant in the gateway's CNI.

III. SYSTEM MODEL

In our system model, we consider several IoAs that are connected over a backbone network. Each IoA is a complex embedded system containing several interconnected nodes, e.g., sensors, actuators, controllers. Figure 1 depicts such a system arranged in a star fashion, which is also the typical topology for switched Ethernet. A couple of IoAs are wired through a real-time Ethernet network. The intermediary between an IoA and the backbone is a gateway. The whole SoS is connected to a WAN via an *Internet Firewall* that can be a standard off-the-shelf component [8].

An IoA itself can be realized either as stand-alone fieldbus network or integrated within an SoC. In this paper, we assume that a single IoA is implemented as a GENESYS²-compliant System-on-a-Chip [9]. Following the GENESYS terminology, this paper elaborates on the integration of chips on the device level. The openness property of GENESYS requires this interface to be open to integrate legacy systems, therefore real-time Ethernet is an appropriate choice for this connection due to the reasons presented in the introduction of this paper.

IV. REQUIREMENTS

This section analyzes the properties of real-time networks that have to be addressed when transferring information between different real-time networks.

a) *Integrated Model of Time:* Single clocks in a distributed system will significantly differ in granularity, reliability, drift rate, error correction, and error probability [3]. An integrated model of time throughout an entire SoS encompasses methods and algorithms to globally synchronize all clocks in the system. Dealing with the low-level properties and requirements on the SoS level is not feasible, therefore, an appropriate abstraction is required [10]. The gateway must support this integrated model of time in order to maintain the temporal order of any event.

²<http://www.genesys-platform.eu>

b) *Determinism*: The system property determinism and in particular *replica determinism* [11] is not a singular property but rather spans the entire system. It is sort of a Boolean property, which is either present or absent. It is impossible to build a deterministic distributed real-time system, if simultaneous events cannot be resolved *consistently*. The execution of agreement protocols, which maintain a common view on the events in a computer system, must be supported by the gateway.

c) *Dependability*: Redirecting messages between networks with different dependability properties requires a defined strategy. For instance, if a receiving node is disconnected or an entire network is temporarily unavailable, the gateway has to implement some procedure how to handle data at hand. Common solutions encompass storing the data for a later retry, thrashing messages if a receiving node is offline, or sending an error message if a sending node has not delivered the expected data. It is the responsibility of the gateway to implement an appropriate strategy, which deals with the different reliability properties. Furthermore, the gateway itself must fulfill certain dependability requirements, because a system state in which the connected networks function properly, but the gateway device cannot deliver its service, is highly unfavorable.

d) *Fault and Error containment*: Fault containment safeguards the correct operation of subsystems regardless of any arbitrary logical or electrical fault outside the subsystem, a so-called Fault Containment Region (FCR). Error containment is concerned with the prevention of error propagation between different partitions of a system in such a way that the error is corrected or masked before it can corrupt the rest of the system [3]. It is the function of a gateway to define an interface for fault and error containment between different networks belonging to separate fault and error containment regions. Obviously, an autonomous IoA represents such an FCR.

e) *Performance*: Common parameters for network performance throughput, transfer rate, latency, and access arbitration. If the networks connected to the gateway differ significantly in terms of performance, the gateway must introduce reasonable limits in order to maintain performance guarantees even in the worst case like peak load scenarios.

f) *Energy efficiency*: Communication cost in terms of energy can vary significantly between network types. Consider for example an NoC and an Ethernet network. Most NoCs are built for high throughput whilst consuming little energy. On the contrary, wirebound and especially wireless communication has a much higher power consumption on the physical layer. Therefore, applications must use off-chip communication carefully.

g) *Security*: Security policies from one system might not be compatible with those from another system. Furthermore, the emerging services available through the connection between different IoAs will have new security requirements. Users from one system will need access rights for the other system and vice versa.

V. DESIGN AND IMPLEMENTATION

This section presents the Time-Triggered System-on-a-Chip (TTSoC) architecture [12] that is a precursor of the

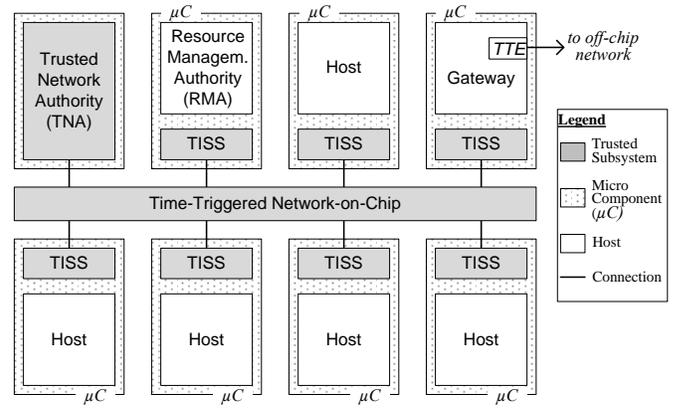


Fig. 2. The TTSoC Architecture

ACROSS MPSoC Architecture. ACROSS³ is a research project that aims to develop and implement an ARTEMIS cross-domain reference architecture for embedded systems based on the architecture blueprint developed in the European FP7 project GENESYS.

A. The TTSoC Architecture

1) *Overview*: The TTSoC architecture assembles single Intellectual Property (IP) cores called *micro components* around a dedicated time-triggered NoC [13] and establishes a global time base among the connected cores. It enables a temporal firewall interface [14] for read and write accesses to the network. In addition to micro components for the application, the architecture employs dedicated infrastructural IP cores for reconfiguration (TNA), resource management (RMA), diagnosis (DU), and for the interconnection of the NoC to off-chip networks (gateways) as depicted in Figure 2. Host applications access the communication services via the Trusted Interface Subsystem (TISS).

2) *Encapsulation*: A key objective of the TTSoC architecture is to facilitate independent development of application subsystems and the integration of subsystems with mixed criticality levels. This is accomplished by the use of encapsulation mechanisms that prevent any unintended interference between subsystems. The encapsulation [15] mechanisms of the TTSoC architecture prevent temporal interference, e.g., stalling messages or delaying computations in another micro component, and spatial interference, e.g., overwriting a message produced by another micro component. Note that this error containment strategy also establishes security properties.

The *implicit properties of the encapsulation mechanisms should be maintained when redirecting messages between two SoCs* [16]. Therefore, an intermediate network supporting the temporal and spatial partitioning of the communication like TTE is required.

3) *Prototype*: The TTSoC prototype [17] developed at our institution implements the concepts of the TTSoC architecture in a FPGA based development environment. It is assembled by a main board (Altera Cyclone II 2C70) hosting the NoC and nine connected extension boards (Altera Cyclone II 2C35)

³<http://www.across-project.eu/>

plus IO synthesizing eight micro component and one off-chip gateway.

B. TTE prototype

TTE [2], [18], which is a real-time Ethernet system, unifies real-time and non-real-time traffic into a single coherent communication architecture. A TTE system consists of a central TTE switch and several connected TTE controllers which communicate using the Time-Triggered Ethernet Protocol. One controller acts as rate master and establishes a global time base [3], [14] by a master-slave clock rate correction algorithm. TTE supports two message classes, event-triggered and time-triggered messages. At each node, a data structure called a Message Descriptor List (MEDL) configures the TTE controller's time-triggered send and receive operations. A MEDL entry consists of message transmission period, phase, and a couple of configuration bits like, e.g., interrupt enable. The identifier of a message is uniquely defined by its transmission instant (period and phase), because no other node may access the communication medium at this instant.

The time-triggered message class supports *periodic messages* that are typically used to transmit state information. The controller triggers periodically their send operation according to the transmission instant as specified in the MEDL. The event-triggered message class of TTE supports standard Ethernet (IEEE 802.3) messages. Sporadic messages belong to a hybrid message class that is triggered at a specific instant, but not periodically.

C. Harmonic Time Model

Both architectures TTE and TTSoC implement a cyclic time format [15] representing a periodic control system based on the *harmonic time model*. The harmonic time model defines *harmonic periods* (based on the negative power of 2), i.e., a period can be $1s$, $\frac{1}{2}s$, $\frac{1}{4}s$, and so forth. Note that a lower period α repeats a higher period β $2^{\beta-\alpha}$ times. In order to avoid collisions and to allow a sequence of activities within one period, each activity encompasses a 2-tuple (π, ϕ) of a period π and a phase ϕ which is the offset within the period. Both values are coded in a 64 bit register constituting a *harmonic time format*. The period π is represented by the set bit with the highest index concerning the bit position (i.e., leftmost). We call this the *period bit*. Obviously, because this is a binary value with just a single one, it can be represented as power of 2 as postulated for π . The phase ϕ is coded by the remaining bits which are to the right side of the period bit. These bits are interpreted as a binary number. We chose a binary base for the harmonic time format, because it is easy to compute by digital logic. The harmonic time model is designed to *simplify the interleaving of cycles, the generation of cyclic schedules and the synchronization of the activity of a system with an external time reference* [9]. A detailed description of the representation of the harmonic time model can be found in [15], [18].

Figure 3 depicts a representation of both time formats aligned as described in the next paragraph. For each representation, one period bit was chosen to adjust it to the International System of Units. Therefore, TTE assigns bit 24 and TTSoC assigns bit 32 to the $1s$ period. This choice spans a different horizon and granularity for each communication system. The

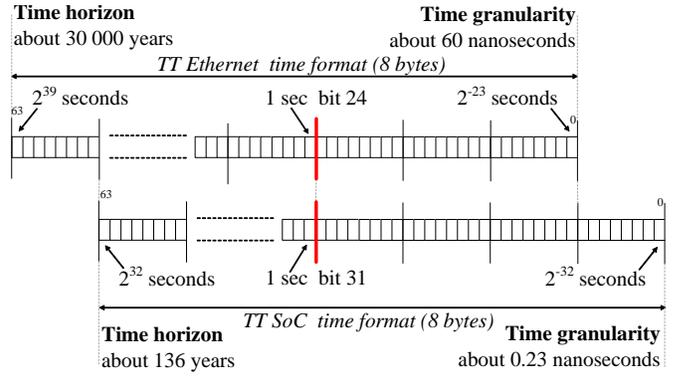


Fig. 3. Aligning TTSoC and TTE time format

assignment reflects the circumstance that in a NoC a much finer granularity is required than in an Ethernet system.

D. Temporal Alignment of Networks

Since the envisioned gateway micro component is required to perform timely network activities in both networks, it has to align the different time formats in a consistent manner. This is done in two steps:

- 1) Set up a consistent representation of time for both networks facilitating an *intermediate time format*
- 2) Implement *transfer functions* to convert between the different time formats

Figure 3 depicts the alignment of the time format of TTE and TTSoC. Since both time formats origin from a harmonic time format, they have a compatible representation of time. The temporal alignment is simply aligning both $1s$ periods (marked by the bold vertical line). All other periods are then equally aligned, because both time formats implement a binary base. The intermediate time format is defined by the subset of possible tuples where both registers overlap. Thus, the intermediate time format inherits the smaller horizon and the bigger granularity of each time format. The transfer functions to convert a period representation (π, ϕ) between networks are implemented by simply shifting the 64 bit register.

E. Gateway

The gateway micro component is able to redirect messages between the internal time-triggered NoC as well as the off-chip Ethernet. There are several functions the gateway micro component provides:

- message conversion between the two network protocol formats
- synchronization of the global time between the TTE and the TTNoC
- a debugging and diagnostic interface

Figure 4 depicts a detailed view on the gateway micro component from Figure 2. The gateway micro component contains three elements:

- A *processor* running a software, which initially reads configuration data like a MEDL and parameters for the TISS from a persistent storage (e.g., a flash memory) and writes it to the hardware.

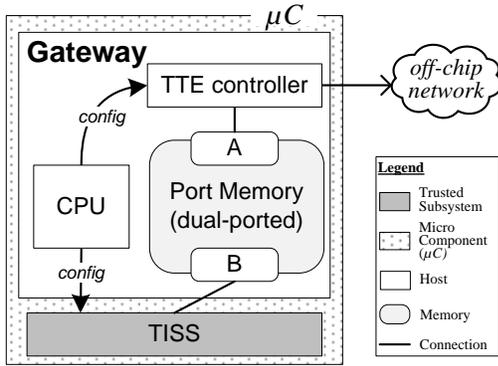


Fig. 4. Gateway design

- The *Port Memory* which is used to intermediately store messages crossing the gateway. It is implemented as a dual-ported memory and can be concurrently accessed by the TISS and the TTE controller.
- The TISS that is the intermediary to the NoC.
- The *TTE controller*, which provides access to a TTE network.

The Port Memory buffers messages in transit between the two network interfaces, the TTE controller and the TISS. A forward operation at the gateway is triggered by pushing a message into the Port Memory by the source network interface. The trigger is determined by the receive instant of the message. After copying the message data to the Port Memory, the CPU receives an interrupt, which then notifies the target network interface that a message is pending. At the message send instant, the respective controller fetches and transmits the according message data. This procedure applies to either the NoC and the TTE controller. Schedules for the respective send and receive instants for each message are configured the TTE's MEDL and in the TISS' configuration.

In order to be efficient in terms of energy and bandwidth, the gateway restricts redirected messages to the sporadic time-triggered message class. Therefore, both networks reserve bandwidth for the respective send instants, but avoid sending redundant messages (if not the user explicitly transmits a periodic event). For each message redirection, the processor has to notify the sender that a new message is present in the Port Memory. In addition, the gateway supports the aggregation of messages from the more performant NoC to the less performant Ethernet, which is to subsume several related messages (e.g., with the same receiver) in a single Ethernet message.

The gateway micro component implements a simple design and follows the end-to-end principle. It stores no state information about the connection other than the defined message send and receive instants. A crash of the gateway micro component will result in a disruption of communication between the endpoints, but the message schedules can be restored from a persistent memory. If one of the networks fails silently, no error propagates to the other network because no notification is issued. Another common failure mode is the babbling idiot failure, which can only manifest in the TTE, because the TTSoc architecture avoids this failure mode by encapsulation.

In this scenario, the TTE controller protects the other network from erroneous messages by thrashing all untimely messages. If one of the two networks is unreachable the present message is trashed. Real-time guarantees can be given, because a message's end-to-end delay can be precisely calculated in the time-triggered case.

F. Clock synchronization

A common notion of time is established by external clock synchronization. The TTE and the TTNoC are synchronized by a mechanism implemented in the hardware (FPGA design) of the TISS. The NoC's macro tick is adjusted to the synchronization message issued by the rate master node in the TTE network. The clock synchronization in TTE achieves a precision of $2^{-21}s = 476.8ns$ [18].

VI. EVALUATION AND DISCUSSION

This section evaluates how the design described in Section V conforms to the requirements from Section IV.

a) *Integrated Model of Time*: The temporal relations between the different abstraction levels (system, device) are established by deploying the same harmonic time format on all levels. The design of the harmonic time format allows to align networks with different granularities and makes temporal measures in the different networks comparable. Hence, the definition of a time format that is applicable to all different abstraction levels can suffice this requirement. The ACROSS MPSoC goes one step beyond the harmonic time format and facilitates *free running periods* which do not have any restrictions. This gives a system designer more flexibility to define periodic activities, but also puts a higher demand on the implementation of the logic that mediates the time format between abstraction levels. In the TTSoc Architecture this logic is trivial.

b) *Determinism*: All channels inherit the weakest *determinism* property of the underlying networks. The prototype implementation facilitates the implementation of *phase alignment*. A system supports phase alignment, if the temporal relationship of different periodic actions can be precisely defined. In our implementation, two actions having the same period can be aligned via their phase offsets. The starting instant of both periods in the different SoCs are synchronized via the synchronization mechanism of the TTE controller. Hence, our prototype implementation supports the notion of replica determinism [11]. In ACROSS, we follow a similar goal and require replica determinism between different connected MPSoCs.

c) *Dependability*: Table I presents some possible strategies to achieve different levels of *dependability*. The strategy offering the highest dependability strategy is using time-triggered to time-triggered strategy with phase alignment according to the harmonic time format as described in Section V-D.

d) *Error containment*: Both TTE and the TISS facilitate *error containment* by temporal and spatial partitioning of the communication. The gateway relies on the encapsulation mechanisms of both networks. Whenever a subsystem behaves erroneously, these mechanisms ensure that no error propagates.

TABLE I
DEPENDABILITY STRATEGIES

from/to	event-triggered	time-triggered
event-triggered	store and forward	aggregated cut-through
time-triggered	aggregated cut-through	phase aligned

e) Performance and Energy efficiency: These requirements are met by providing the possibility to aggregate several small messages in longer one. This saves energy and bandwidth, because less low level bus utilization is required. Moreover, by using the sporadic message, redundant transmission of messages is avoided if not explicitly configured by the user.

f) Security: Concerning security, the encapsulation mechanisms provide on-chip protection against eavesdropping (channels cannot be redirected or listened without approval of the TNA). Off-chip protection can be attained by deploying a middleware for encryption [19]. ACROSS offers an extensive range of security mechanisms covering secure communication and secure clock synchronization services.

VII. CONCLUSION

In this paper we worked out the function of gateways as intermediaries between systems in order to build large SoSes. We investigated on the redirection of messages between different network paradigms in general and between time-triggered networks having a different granularity in particular. Moreover, we presented the design of a gateway connecting a NoC and an Ethernet network under real-time constraints. We outlined the harmonic time format that facilitates the easy temporal alignment of two time-triggered networks and evaluated these concepts within the TTSoc Architecture. Moreover, we pointed out connections to the ACROSS MPSoc which is an advanced implementation of a MPSoc using a time-triggered NoC. In conclusion, we found out that an model of time that integrates over all abstraction levels is a major cornerstone to build a dependable and deterministic system.

ACKNOWLEDGMENT

This document is based on the ACROSS project in the framework of the ARTEMIS programme. The work has been funded in part by the ARTEMIS Joint Undertaking and National Funding Agencies of Austria, Germany, Italy and France under the funding ID ARTEMIS-2009-1-100208. The responsibility for the content rests with the authors. The authors would like to thank Christian Paukovits for the fruitful teamwork.

REFERENCES

- [1] International Telecommunications Union (ITU), "Internet reports 2005: The internet of things," Executive Summary, 2005.
- [2] H. Kopetz, A. Ademaj, P. Grillinger, and K. Steinhammer, "The Time-Triggered Ethernet (TTE) Design," in *8th IEEE International Symposium on Object-oriented Real-time distributed Computing (ISORC)*, May 2005, pp. 22–33.
- [3] H. Kopetz, *Design Principles for Distributed Embedded Systems*, 4th ed. Kluwer Academic Publishers, 1997.
- [4] R. Fichera, "Islands Of Automation Are Dead – Long Live Islands Of Automation," Forrester Research Report, Quick Take., August 2004.
- [5] A. Nagashima, "Technologies for Achieving Field-Ubiquitous Computing," in *Proceedings of the 5th Conference on Industrial Informatics*, 2007.
- [6] M.-C. Gaudel, V. Issarny, C. Jones, H. Kopetz, E. Marsden, N. Moffat, M. Paulitsch, D. Powell, B. Randell, A. Romanovsky, R. Stroud, and F. Taiani, "Final version of the DSoS conceptual model," *DSoS Project (IST-1999-11585) Deliverable CSDA1*, 2002.
- [7] TTTech, "Time-Triggered Protocol TTP/C High-Level Specification Document Protocol Version 1.1," TTTech Computertechnik AG, Specification V 1.5.2, January 2004.
- [8] S. M. Bellovin and W. R. Cheswick, "Network firewalls," *IEEE Communications Magazine*, vol. 32, no. 9, pp. 50–57, 1994.
- [9] R. Obermaisser and H. Kopetz, Eds., *GENESYS: A Candidate for an ARTEMIS Cross-Domain Reference Architecture for Embedded Systems*. Süd dwestdeutscher Verlag für Hochschulschriften (SVH), 2009.
- [10] H. Kopetz, "The Complexity Challenge in Embedded System Design," in *The 11th IEEE International Symposium on Object/component/service-oriented Real-time distributed Computing (ISORC)*, May 2008, Research Report, pp. 3–12.
- [11] S. Poledna, *Fault-Tolerant Real-Time Systems: The Problem of Replica Determinism*. Kluwer Academic Publishers, 1995.
- [12] C. El-Salloum, R. Obermaisser, B. Huber, and H. Kopetz, "The Time-Triggered System-on-a-Chip Architecture," in *Proceedings of the IEEE International Symposium on Industrial Electronics*, 2008.
- [13] C. Paukovits and H. Kopetz, "Concepts of switching in the time-triggered network-on-chip," in *Proceedings of the 14th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA)*, 2008.
- [14] H. Kopetz and G. Bauer, "The time-triggered architecture," in *Proceedings of the IEEE Special Issue on Modeling and Design of Embedded Software*, October 2001.
- [15] C. Paukovits and H. Kopetz, "Building Encapsulated Communication Channels in the Time-Triggered System-on-Chip Architecture," Institut für Technische Informatik, Treitlstr. 1-3/182-1, 1040 Vienna, Austria, Research Report 8/2009, 2009.
- [16] B. Huber, C. El-Salloum, and R. Obermaisser, "A Resource Management Framework for Mixed-Criticality Embedded Systems," in *Proceedings of the 34th Annual Conference of the IEEE Industrial Electronics Society (IECON'08)*, November 2008.
- [17] C. Paukovits, "The Time-Triggered System-on-Chip Architecture," Ph.D. dissertation, Technische Universität Wien, Institut für Technische Informatik, Treitlstr. 3/3/182-1, 1040 Vienna, Austria, Dec. 2008.
- [18] K. Steinhammer, "Design of an FPGA-Based Time-Triggered Ethernet System," Ph.D. dissertation, Technische Universität Wien, Institut für Technische Informatik, Treitlstr. 3/3/182-1, 1040 Vienna, Austria, 2006.
- [19] A. Wasicek and C. El-Salloum, "End-to-End Encryption in the TTSoc Architecture," in *Proceedings of the 3rd Workshop on Embedded Systems Security, ESWEK'08, Atlanta, USA*, 2008.