Fault and Error Containment of Gateways in Distributed Real-Time Systems

Roman Obermaisser

Vienna University of Technology, Austria

Email: romano@vmars.tuwien.ac.at

Abstract— The composition of large embedded computer systems out of heterogenous clusters requires gateways that resolve property mismatches between networks with different communication protocols and diverse computational models. At the same time, fault and error containment mechanisms are needed in order to prevent common mode failures that could be induced by the propagation of faults between clusters. This paper presents gateways to systematically tackle these challenges. The introduced gateways contain structured collections of time-sensitive variables associated with timing information (called a real-time database). The real-time database facilitates fault and error containment by temporally decoupling clusters. For accessing the real-time database from attached networks, we introduce network adaptors that ensure compliance with temporal specifications expressed using state machines with timing constraints. The containment of faults within clusters is of particular importance for mixed-criticality systems and increases the robustness of the overall systems. The proposed solution is experimentally evaluated in an example system with a time-triggered Ethernet network and a Controller Area Network (CAN).

Index Terms— gateways, fault containment, error containment

I. INTRODUCTION

In several application domains, such as the automotive or aerospace industry, complex embedded systems have emerged that comprise multiple application subsystems realized by multi-cluster electronics systems with gateways between the individual networks. For example, in-vehicle electronic systems can comprise clusters for body, passive safety, vehicle dynamics, powertrain, infotainment, and driver assistance [1]. Each of these clusters is implemented with a set of node computers interconnected by networks such as CAN [2], MOST [3], LIN [4], and FlexRay [5].

In general, the decomposition of the overall electronic system does not lead to completely independent clusters. Many application services inherently involve an interplay of clusters. An example of such an application service is the precrash system [6] of a car, which requires the interplay of the the vehicle dynamics subsystem and the comfort subsystem.

In addition to these dependencies that are inherent in the application services, dependencies can also arise

through implementation choices driven by economic considerations. In order to reduce the hardware cost, resources can be shared across clusters. For example, automotive temperature sensors are used in the body system for heating, ventilation and air conditioning. Furthermore, many of the components in other automotive clusters use temperature compensation for improving the accuracy of sensors [7].

Typically multi-cluster electronics systems are *mixed-criticality systems*. For example, clusters in the aerospace domain range from non safety-critical passenger entertainment systems to safety-critical flight related subsystems, such as engine management and flight control [8]. In order to accommodate the differences w.r.t. criticality, avionic application subsystems are assigned risk classes [9]. Likewise, automotive application subsystems are associated with Safety Integrity Levels (SILs) [10] in order to express the impact on safety w.r.t. malfunction and function availability.

The tremendous differences in the development and certification costs between different criticality levels are also a major reason for modular system architectures with multi-cluster electronics systems. Using modular certification [11], each cluster can be individually certified to the appropriate level of criticality. Modular certification allows to reduce cost and to focus assurance effort on the most critical parts of a system.

However, in multi-cluster electronics systems with gateways, a prerequisite for modular certification are fault and error containment mechanisms that prevent failures in a non safety-critical cluster from affecting the correct operation of safety-critical ones. The challenge is to construct gateways with the ability to prevent faulty messages from propagating between clusters.

In previous work, we have proposed a generic framework for gateways based on a real-time database [12]. We introduced gateways that contain structured collections of time-sensitive variables associated with timing information (called real-time databases) in order to separate clusters. The real-time databases are accessed via network adaptors that establish the interface to the interconnected networks. The proposed framework includes a code generation tool that produces a middleware layer for forwarding data from a real-time database to the different networked components, as well as for collecting data from networked components in order to update time-sensitive variables in the real-time database.
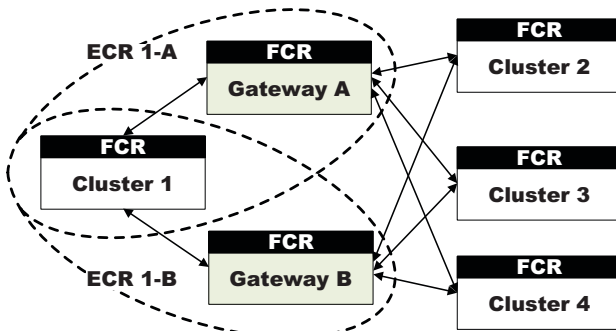
Figure 1. Fault Containment Regions (FCRs) and Error Containment Regions (ECRs)

This paper extends the gateway framework in order to establish fault and error containment for timing and value message failures. Thereby, we establish an effective basis for the construction of multi-cluster mixed-criticality systems. The proposed fault and error containment mechanisms build upon the real-time database and the network adaptors. They have been implemented in a prototype setup with a Time-Triggered Ethernet cluster and a Controller Area Network (CAN) cluster. Using this prototype setup, experiments have investigated the effectiveness and overhead (e.g., additional latencies) of the fault and error containment in the gateways.

The coupling of networks based on database has also been addressed in several existing solutions. For example, the proxy architecture described in [13] supports the interconnection of two heterogeneous networks with a *reliable multicast proxy* containing a database called the *data store*. Another example of a solution using a database for the interconnection of networks is the distributed object-oriented real-time database system called BeeHive [14]. A third example is an architecture for avionic systems-of-systems based on real-time publish/subscribe and High-Level Architecture (HLA) [15]. In contrast to the presented work, existing solutions do not address mixed criticality systems by introducing mechanisms for fault and error containment w.r.t. timing and value message failures.

The paper is structured as follows. Section II gives an overview of a distributed embedded real-time systems containing multiple clusters that are interconnected by gateways. In particular, the requirements concerning fault and error containment are given. The description of the proposed gateways with a real-time database and network adaptors is the focus of Section III. Section IV introduces fault and error containment mechanisms based on these gateways. The mechanisms were evaluated using the experiment setup described in Section V. The results of the experiments are presented in Section VI. The discussion and interpretation of the results follows in Section VII. The paper finishes with a conclusion and information about future work in Section VIII.

II. FAULT AND ERROR CONTAINMENT IN MULTI-CLUSTER REAL-TIME SYSTEMS

This section discusses the requirements regarding fault and error containment in a distributed real-time system
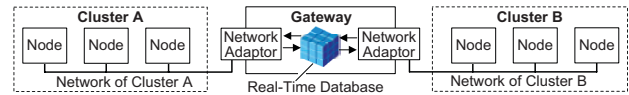

Figure 2. Gateway based on Real-Time Database

encompassing multiple clusters.

A. Fault Containment

A Fault Containment Region (FCR) is defined as a subsystem that operates correctly regardless of any arbitrary logical or electrical fault outside the region [16]. The notion of FCRs is a key concept for reasoning about the behavior of a system in the presence of faults. The knowledge about the immediate impact of a fault can serve as the starting point for the reliability analysis of a system. In addition, fault-tolerance mechanisms such as triple modular redundancy require replicas to be assigned to independent FCRs.

When reasoning about the composition of clusters using gateways, we can regard each cluster and each gateway as a FCR as depicted in Figure 1. Of course, a finer granularity concerning FCRs (i.e., regarding individual nodes within a cluster as FCRs) is useful when implementing fault-tolerant clusters, e.g., using redundant nodes. However, from the point of view of the gateways we are only interested in the composition of complete clusters. Thus, we can abstract from the inner structure of the clusters.

B. Error Containment

Although an FCR can restrict the immediate impact of a fault, fault effects manifested as message failures can propagate across FCR boundaries. Possible failure modes [17] of an FCR are timing message failures and value message failures. In case of a *timing message failure* the message send instants are in violation of the temporal specification of the cluster. A *value message failure* occurs, if the contents of a message do not comply with the specification. These incorrect message contents can encompass the message name and/or the message data.

For this reason the system must also provide *error containment* [16] to avoid error propagation by the flow of erroneous messages. The error detection mechanisms must be part of different FCRs than the message sender [18]. Otherwise, the error detection mechanisms may be impacted by the same fault that caused the message failure.

For erroneous messages emitted by a cluster, the gateway constitutes the second FCR. We use the term *Error Containment Region (ECR)* in order to refer to this connection of a cluster producing messages with a gateway that performs error detection and error containment. Figure 1 shows an example depicting the FCRs in a system with four clusters and two gateways. Two ECRs are concerned with the containment of message failures produced by the first cluster.
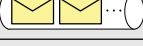
Information Semantics	Real-Time Image	Meta Information	
		Variable	Description
Real-Time Image 1 with State Semantics		t_{update}^1	most recent update instant
		d_{offset}^1	temporal accuracy offset
		b_{req}^1	update request indication
Real-Time Image 2 with State Semantics		t_{update}^2	most recent update instant
		d_{offset}^2	temporal accuracy offset
		b_{req}^2	update request indication
Real-Time Image *k* with Event Semantics		b_{req}^{k+1}	update request indication
		n_{queued}^{k+1}	number of queued elements
Real-Time Image *k+1* with Event Semantics		b_{req}^{k+2}	update request indication
		n_{queued}^{k+2}	number of queued elements

Figure 3. Real-Time Database

III. GATEWAYS BASED ON A REAL-TIME DATABASE

A real-time system can be modeled using a set of real-time entities [19], which are significant state variables located in the environment or the computer system. The current value of a real-time entity is called a *real-time image* and can be sent within a message on a network. Redirection of information through a gateway occurs when a real-time image contained in a message is required by another cluster connected to the gateway.

The mechanisms described in this paper build upon a gateway model as introduced in [12]. A gateway according to this model maintains a *real-time database* with real-time images associated with timing information. Gateways recombine real-time images acquired from one network into messages for another network, while converting between different temporal and syntactic specifications and resolving naming incoherences.

The gateway contains for each accessed cluster a so-called *network-adapter*, which implements the communication protocol of the network of the cluster and performs information exchanges between the network and the real-time database (see Figure 2). Since the validity of the real-time images is time-dependent, the network adaptors are responsible for repeatedly updating the real-time images in the real-time database.

In the following, we will summarize the operation and specification of the gateways based on a real-time database. More detailed information can be found in [12], [20].

1) Real-time Database: The real-time database stores real-time images for the information exchange between the network adaptors. The updating and the storage of real-time images depends on the data semantics [19, p. 103] (see Figure 3). Real-time images with event semantics are queued, whereas real-time images with state semantics are stored in variables that are overwritten whenever a more recent value becomes available. The underlying idea behind these two storage modes is that in case of state semantics old values can be overwritten with newer values. State information represents an absolute value (e.g., speed is 30ms^{-1}) and applications are often only interested in the most recent value of a real-time image. Event information, on the other hand, relates to the occurrence of an event and needs to be processed exactly-once.

In addition to the data of the real-time images, the real-time database stores meta-information about real-time images. The point in time of the most recent update t_{update} is a dynamic attribute associated with each real-time image with state semantics. t_{update} is set to the current time, whenever a network adaptor overwrites the real-time image in the real-time database.

Due to the dynamics of real-time entities, which change their state as time progresses, the validity of real-time images is time-dependent. The temporal accuracy d_{acc} [19, p. 103] denotes how long a real-time image will still remain a valid image of the respective real-time entity in case no update of the real-time image occurs in the meantime. The age of a real-time image at its time of use introduces an error in the value domain, which is determined by the rate of change of the underlying real-time entity. Depending on the dynamics of a real-time entity and the maximum acceptable error in a given application, a gateway is allowed to store a real-time image only for a limited duration before the real-time image is invalidated by the progression of time.

The real-time database captures for each real-time image with state semantics the *temporal accuracy offset* d_{offset}. This static attribute determines the temporal accuracy immediately after an update of the real-time image. Using the temporal accuracy offset, the temporal accuracy can be computed at time t_{now} as follows:

$$d_{acc} = \max(d_{offset} - (\underbrace{t_{now} - t_{update}}_{\text{age of real-time image}}), \ 0)$$

Furthermore, the real-time database contains boolean *update request indications* in order to support on-demand communication activities. For a real-time image with state or event semantics, the respective update request indication b_{req} denotes whether a new real-time image needs to be transferred into the real-time database. By setting the update request indication, a network adaptor can demand real-time images from other network adaptors. A network adaptor receiving messages from a network can initiate receptions conditionally, based on the value of the update request indication.

For every real-time image with event semantics, the real-time database contains a dynamic attribute with the number of queued elements n_{queued}. This attribute denotes the number of elements of the real-time image that are currently queued in the real-time database.

Furthermore, conversion functions can be specified, which serve for the automatic transformation between different syntactic representations of real-time images. Each conversion is associated with a triggering real-time image and a real-time image that shall store the result of the transformation. Upon a change of value of the triggering real-time image, the conversion function computes a new value for the targeted real-time image.

2) Network Adaptors: The network adaptors are specified using state machines with timing constraints and gateway-specific operations [20]. A network adaptor is a state machine with local variables, clock variables,

locations, and edges. An edge interconnects two locations of the network adaptor and can be associated with a guard and an action:

$$\text{guard} \rightarrow \text{action}$$

The guard expresses a boolean condition that defines whether the edge can be taken. Actions are used to define assignments to variables, as well as interactions with the real-time database and networks. Variables are used to capture the internal state of the network adaptor. In particular, variables can store messages and real-time images. In an action, variables can serve as the source for the transfer of a real-time image into the real-time database or the transmission of a message on a network. In analogy, variables can serve as the destination when executing an action to read a real-time image from the real-time database or to receive a message from a network.

The following grammar describes a guard:

rtimage   := [a-z][A-Za-z0-9_]*
message   := [a-z][A-Za-z0-9_]*
variable   := [a-z][A-Za-z0-9_]* | variable[expression] | variable.variable
expression := (expression *ARITH* expression) | d_{acc}(rtimage)
                  | variable | *CONST*
bool   := expression *COMP* expression | (bool ∧ bool) | (bool ∨ bool)
                  | ¬bool | n_{queued}(rtimage) | b_{req}(rtimage) | avail(message)
guard   := bool

A guard is a boolean composition (i.e., disjunction, conjunction, negation) of constraints on variables and real-time images. Constraints can be specified using a binary comparison operator ($COMP \in \{\leq, <, =, \neq, >, \geq\}$) on mathematical expressions. An *expression* defines a function on the values of variables by means of arithmetic operations ($ARITH \in \{+, -, *, /\}$). Variables can be arrays and hierarchically structured. Furthermore, constants (i.e., *CONST* in $\mathbb{Z}$), the temporal accuracy interval $d_{acc}(rtimage)$, the update request indication $b_{req}(rtimage)$, and the number of queued elements $n_{queued}(rtimage)$ of a real-time image can be used in an expression. $avail(message)$ denotes whether a message is available at a port to a network, i.e., whether the message can be received with a $rcv()$ operation (see below).

The action attribute of a transition adheres to the following grammar:

action   := action;action|variable=expression|communication|ε
communication :=req(rtimage) | push(rtimage)|pull(rtimage)|
                       rcv(message) | snd(message)

Besides the assignment of an expression to a variable, communication operations support interactions with networks and the real-time database. The operation $req(name)$ causes the setting of the update request indication of the real-time image *rtimage* in the real-time database. The operation $push(rtimage)$ causes the transfer of the real-time image *rtimage* from a variable of the automaton into the real-time database. Inversely,
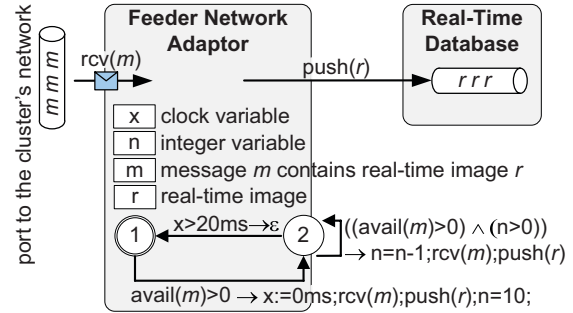

Figure 4. Temporal partitioning in feeder network adaptor using minimum interarrival time with bursts

$pull(rtimage)$ fetches a real-time image from the real-time database and writes it into a variable.

$rcv(message)$ performs the reception of message when the transition is taken. The reception means that the message is read from a network and written to automaton variables of the same name. The operation $snd(message)$ performs the transmission of a message. Prior to passing the message to the network, the current values of a corresponding set of the automaton variables are used to set the contents of the message.

IV. FAULT AND ERROR CONTAINMENT THROUGH THE GATEWAY

In the following, we will explain the achieving of fault and error containment by mechanisms for *temporal* and *spatial partitioning* [21] using gateways with a real-time database and network adaptors. For the description of the error containment mechanisms, we distinguish two types of network adaptors based on the interaction with the real-time database:

- **Feeder network adaptor.** These network adaptors receive messages from a network and write real-time images contained in these messages into the real-time database.
- **Retrieval network adaptor.** These network adaptors read from the real-time database in order to construct messages and transmit them on a network.

A. Temporal Partitioning

Temporal partitioning is concerned with the containment of the effects of *timing message failures* (cf. fault assumptions in Section II). Through temporal partitioning, the gateways ensure predefined temporal properties (e.g., bounded latencies) of the networks of the interconnected clusters despite the redirection of messages. In particular, temporal partitioning limits the effects that can be induced by a node of one cluster on the temporal properties of the messages transmitted by nodes on the network of another cluster.

We can distinguish three layers of defense for timing message failures:

a) Feeder Network Adapter: The feeder network adaptor uses temporal a priori knowledge to discard messages that represent message timing failures. The types of detectable timing failures depend on the corresponding
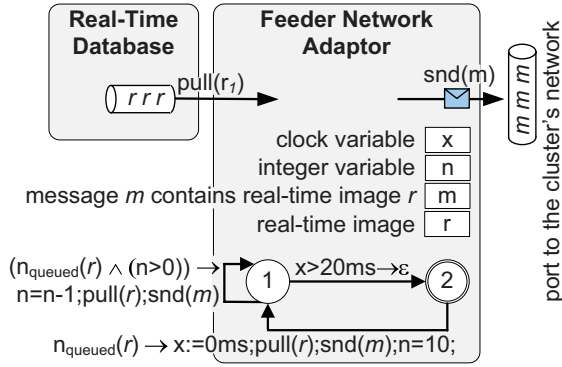
Figure 5. Temporal partitioning in retrieval network adaptor using minimum interarrival time with bursts

Figure 6. Spatial Partitioning in Feeder Network Adaptor (m is a time-triggered Ethernet message containing a real-time image $r = \langle Preamble, Startdelimiter, DestinationMAC, SourceMAC, ...\rangle$

communication protocol. For time-triggered communication protocols (e.g., FlexRay [5], TTP [22]), the network adaptor can detect deviations of the actual message transmission instants compared to the preplanend instants in the time-triggered communication schedule. For event-triggered communication protocols (e.g., CAN [2]), the amount of a priori knowledge on the message timing determines the achievable error containment. For example, message interarrival times and message service times can be monitored and enforced. Figure 4 depicts an example for temporal partitioning in the feeder network adaptor for an event-triggered protocol. The state machine in the network adaptor permits bursts with 10 message receptions in immediate succession and enforces an inactivity interval of 20ms between any two such bursts.

b) Retrieval Network Adapter: To realize temporal partitioning, the retrieval network adaptor can use temporal a priori knowledge for controlling the relaying of real-time images from the real-time database to the destination network. For example, a retrieval network adapter can relay fewer real-time images than contained in the real-time database in order to prevent the overloading of the network (i.e., violation of minimum interarrival times). Figure 5 depicts an example for temporal partitioning in the retriever network adaptor. The state machine of the network adaptor permits a burst of 10 pull operations in immediate succession and enforces an inactivity interval of 20ms between any two such bursts.

c) Real-Time Database: The real-time database supports separate message buffers for different real-time images [19]. Hence, a timing message failure resulting in an overload of a particular real-time image at the real-time database does not affect the other real-time images. This is particulary important for buffers of real-time images with event information, which are realized as queues in the real-time database. Although the queue dedicated to the real-time image with the message overload will be subject to an overflow and message omissions, all other buffers will continue to provide exactly-once semantics.

B. Spatial Partitioning

As introduced in the fault model, a value message failure occurs in case the contents of a message do not comply with the specification. Two important classes of
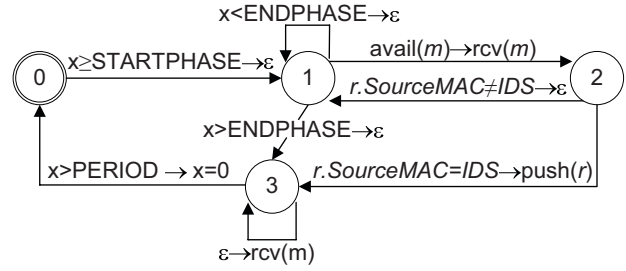
message failures are the *loss of temporal accuracy* and *faulty message names*.

The loss of temporal accuracy occurs if real-time images (contained in a message) have been invalidated by the progression of time. In such a case, receivers can fail due to operating on outdated inputs.

In case of a faulty message name, the following types of spatial interference can occur depending on the types of names contained in a message:

- **Invalid source address:** Such a behavior is also called a masquerading failure. Masquerading is defined as the sending or receiving of messages using the identity of another sender without authority [23].
- **Invalid destination address:** The anticipated receiver does not receive a message, whereas the message can have a harmful effect at unintended receivers (e.g., overwriting of other messages, creation of processing load).
- **Message identifier:** The message can be handled in an incorrect manner by the receiver (e.g., temperature value is mistaken for a speed value).

In analogy to temporal partitioning, the spatial partitioning mechanisms for the above failures can occur in the feeder network adapter, the retrieval network adaptor, and the real-time database.

a) Feeder Network Adapter: The spatial partitioning mechanisms of a feeder network adaptor are based on state machines that encode checks on the contents of received messages. A feeder network adaptor discards messages with wrong contents, i.e., the network adaptor does not insert the real-time images into the real-time database. Figure 6 is an example of a feeder network adapter implementing a simple name constraint of a time-triggered Ethernet network [24].

The state machine waits in the initial state '0' until the start phase of the periodic message has been reached. Thereafter, the automaton waits for the arrival of a message in state '1'. If the message arrives, a transition to state '2' follows and the source address of the message is compared to a predefined node address (*IDS*). In case of a match the message is placed in the real-time database (using a push operation). State '3' is reached either after the reception of a correctly named message or in case the end phase of the enabled time interval has been reached. After the end of the period, the state machine returns to
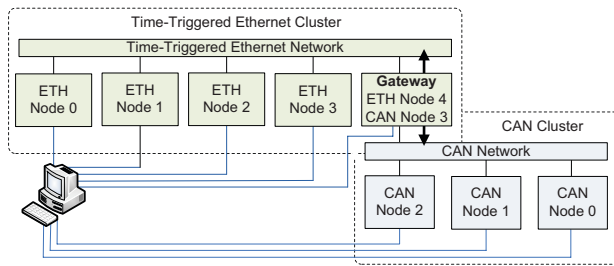
Figure 7. Prototype Setup

the initial state.

Of course, other protocols can require more complex state machines, e.g., multiple receive and send operations as part of a request/reply protocol, before the contents of a message can be checked and forwarded to the real-time database.

b) Real-Time Database: The real-time database can employ a conversion function in order to convert between different syntactic representations (cf. Section III). In case a triggering real-time image is modified through a push operation, a new value for a target real-time image is computed. For the purpose of spatial partitioning, conditions on the contents of the triggering real-time image can be employed for deciding on an update of the target real-time image. Thus, in case of an incorrect message name in the triggering real-time image, no update of the second real-time image is performed.

c) Retrieval Network Adapter: The retrieval network adapter can perform the detection and containment for value failures of the real-time images retrieved from the real-time database. In order to prevent the redirection of real-time images that have been invalidated by the progression of time, the retrieval network adapter can use the temporal accuracy d_{acc}. As discussed in Section III, the temporal accuracy of a real-time image at a specific time can be computed from the most recent update instant in conjunction with the temporal accuracy offset.

V. EXPERIMENTS

This section describes the prototype setup and the configurations that have been used to evaluate the fault and error containment of the proposed gateways.

A. Prototype Setup

The evaluation of the fault and error containment of the presented gateways has been performed with a prototype setup containing two clusters. As depicted in Figure 7, a gateway establishes the link between a cluster with a time-triggered Ethernet network and a cluster with a CAN network.

1) Time-Triggered Ethernet Cluster: Time-triggered Ethernet [24], [25] is a protocol that unifies real-time and non-real-time traffic into a single coherent communication protocol. This protocol supports two types of time-triggered messages, namely time-triggered periodic messages (TTpM) and time-triggered sporadic messages (TTsM). TTpMs are transmitted periodically from the sender node to the receiver nodes in a reserved Time

Division Multiple Access (TDMA) slot. Each TTpM has a constant period and phase in a time-triggered communication schedule. TTpMs are typically used to transmit state information and each correctly received message overwrites the previous instance at each receiver.

TTsMs, on the other hand, are intended to transport sporadic event information. A TTsM is transmitted only if the local message has been updated by the sender's host application since the last transmission of the message. TTsMs have a constant period and phase (like the TTpM) and can theoretically be transmitted in every such period (when the message contents change after every transmission).

The *time-triggered Ethernet cluster* consists of a set of five node computers (Soekris Engineering net4801[1]), each of which contains the 586 class processor SC1100 clocked at 266 MHz. The software within a node encompasses a real-time Linux variant as the operating system and a software implementation of time-triggered Ethernet with 100 Mbps. The communication schedule uses a period of approx. 2 ms.

2) CAN Cluster: CAN is an event-triggered communication protocol, which uses carrier sense, multiple access with collision avoidance (CSMA/CA) for medium access control [2]. CAN employs *data frames* for the transmission of CAN message objects. A data frame contains a unique identifier, which identifies the message object and denotes the message priority. In addition, a node can transmit a *remote frame* to request the dissemination of a data frame with a specific identifier. Collisions on the CAN bus are resolved by bit wise arbitration of the message identifiers. The CAN protocol itself does not enforce temporal constraints on the transmission behavior of the nodes, although many CAN-based applications use timing models on top of CAN (e.g., interarrival times or periodic messages) [26].

The *CAN cluster* of the prototype setup uses single board computers (Soekris Engineering net4521) as nodes. Each node is equipped with a PCMCIA-based CAN interface card [27] containing the CAN controller SJA1000. The CAN network has been configured for a bandwidth of 500 kbps, which is commonly used, e.g., in high-speed automotive CAN networks.

3) Gateway: The *gateway* is a node that belongs to both clusters. This node is equipped with a time-triggered Ethernet interface and a PCMCIA-based CAN interface card. The node executes code that has been automatically generated from formal gateway specifications [12], [20]. Section V-B will give an overview of the gateway specifications used in the prototype setup, focusing on the state machines of the network adaptors. The full description of the UML models that were used as input for the code generation is omitted due to space constraints.

In addition to the automatically generated code, the node serving as the gateway contains drivers for CAN and time-triggered Ethernet. These drivers establish the
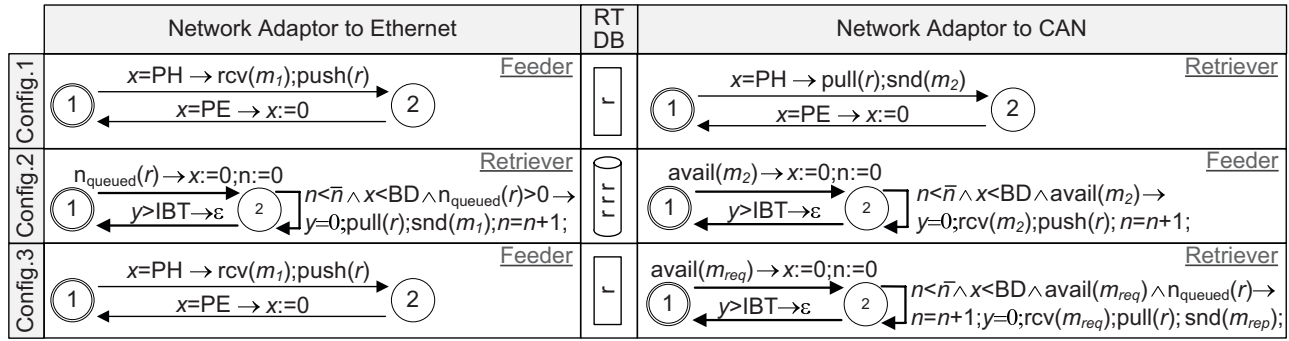
---

[1]www.soekris.com

Network Adaptor to Ethernet	RT DB	Network Adaptor to CAN
Config.1 — *Feeder*: $1 \to 2$: $x=PH \to \mathrm{rcv}(m_1); \mathrm{push}(r)$; $2 \to 1$: $x=PE \to x:=0$		**Config.1** — *Retriever*: $1 \to 2$: $x=PH \to \mathrm{pull}(r); \mathrm{snd}(m_2)$; $2 \to 1$: $x=PE \to x:=0$
Config.2 — *Retriever*: $1 \to 2$: $n_{\mathrm{queued}}(r) \to x:=0; n:=0$; $2 \to 1$: $y>IBT \to \varepsilon$; loop at 2: $n<\bar{n} \wedge x<BD \wedge n_{\mathrm{queued}}(r)>0 \to y:=0; \mathrm{pull}(r); \mathrm{snd}(m_1); n=n+1;$		**Config.2** — *Feeder*: $1 \to 2$: $\mathrm{avail}(m_2) \to x:=0; n:=0$; $2 \to 1$: $y>IBT \to \varepsilon$; loop at 2: $n<\bar{n} \wedge x<BD \wedge \mathrm{avail}(m_2) \to y:=0; \mathrm{rcv}(m_2); \mathrm{push}(r); n=n+1;$
Config.3 — *Feeder*: $1 \to 2$: $x=PH \to \mathrm{rcv}(m_1); \mathrm{push}(r)$; $2 \to 1$: $x=PE \to x:=0$		**Config.3** — *Retriever*: $1 \to 2$: $\mathrm{avail}(m_{req}) \to x:=0; n:=0$; $2 \to 1$: $y>IBT \to \varepsilon$; loop at 2: $n<\bar{n} \wedge x<BD \wedge \mathrm{avail}(m_{req}) \wedge n_{\mathrm{queued}}(r) \to n=n+1; y:=0; \mathrm{rcv}(m_{req}); \mathrm{pull}(r); \mathrm{snd}(m_{rep});$

Figure 8. Configuration of Network Adaptors (x, y . . . clock variables; n . . . integer variable; n, BD, IBT, PE, PH . . . constants; $m_1, m_2, m_{\mathrm{req}}, m_{\mathrm{rep}}$ . . . messages; r . . . real-time image contained by $m_1, m_2, m_{\mathrm{rep}}$)

gateway-specific operations, such as `snd`, `rcv` and `avail`.

B. Gateway Specification for Three Configurations

In the following, three configurations will be explained that were used in the experiments: periodic, unidirectional sporadic and client/server communication. These three configurations represent three important communication paradigms and are supported by the CAN and Ethernet networks in the clusters of the prototype setup. In all three configurations, the sending node includes in each message a sequence number and a time stamp of the transmission instant. The receiving node determines the transmission latency using the time stamp contained in the message and the current time. Using the sequence number, the receiver recognizes lost or duplicate messages.

In all three configurations, the messages are used by the receivers to update a state variable with time-dependent validity. We assume an exemplary temporal accuracy offset of 20 ms in the experiments in order to be able to describe the temporal accuracy of this variable.

1) Configuration for Periodic Communication: In this configuration, the gateway redirects a periodic message from the Ethernet network to the CAN network. The network adaptors are depicted in the first row in Figure 8. The feeder network adaptor receives messages with a predefined period *PE* and phase *PH*. Whenever the clock variable indicates that the phase has been reached, the transition to state 2 is taken, receiving a message and pushing the contents into the real-time database. The real-time database stores a state variable, the contents of which are periodically overwritten by the feeder network adaptor. After the period has elapsed (i.e., the clock variable is equal to the period), the transition back to the initial state 1 is taken. The state machine of the retrieval network adaptor has a similar structure. In state 1, the retrieval network adaptor waits for the clock variable becoming equal to the phase, before reading the value of the state variable from the real-time database and sending a message. Thereafter, the period needs to elapse before the state machine progresses back to state 2.

2) Configuration for Unidirectional Sporadic Communication: The gateway redirects sporadic messages with bursts in this configuration. Figure 9 shows the timing model of the sporadic messages with bursts. A burst has
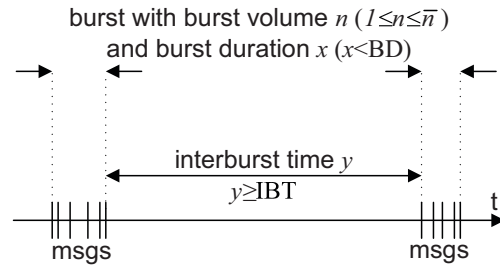

Figure 9. Timing Model

a volume n between 1 and $\bar{n}$ messages. These n messages must occur in a time interval x that has a maximum burst duration of *BD*. After the burst, a burst interarrival time has to elapse before any subsequent messages. *BIT* is a lower bound for the burst interarrival time y.

The feeder and retrieval network adaptors in the second row of Figure 8 contain state machines that enforce this timing model. The feeder network adaptor interacts with the CAN network. When a message is available at the port, the clock variable is reset to measure the time until the end of the maximum burst duration. Within the burst (in state 2) up to $\bar{n}$ messages are received and their contents are pushed into the real-time database. Since sporadic message are frequently associated with event information, the real-time database contains a queue to store real-time images with support for exactly once processing. After the reception of each message, a clock variable is reset in order to capture the silence time since the last message. When the silence time is longer than the burst interarrival time *BIT*, the state machine transits back into state 1 and is ready for a new burst.

Due to the state machine of the network adaptor, the insertion of information into the queues of the real-time database occurs as specified in the timing model even in case the messages arriving from the network violate the temporal constraints.

The state machine of the retrieval network adaptor implements the same timing model. However, instead of receiving messages from the network during a burst, the retrieval network adaptor dequeues messages from the real-time database and forwards them to the Ethernet network. Even if more messages are present in the real-time database than may be disseminated according to the timing model, the retrieval network adaptor will enforce the transmission of messages on the Ethernet network in compliance with the timing model.

Configuration	Cluster-Internal Communication of CAN Cluster						Cluster-External Communication					
	No Redirection		Redirection of all msgs.		Redirection of timely msgs. (partitioning)		No Redirection		Redirection of all msgs.		Redirection of timely msgs. (partitioning)	
	Max. Latency	Min. Temporal Accuracy	Max. Latency	Min. Temporal Accuracy	Max. Latency	Min. Temporal Accuracy	Max. Latency	Min. Temporal Accuracy	Max. Latency	Min. Temporal Accuracy	Max. Latency	Min. Temporal Accuracy
Periodic	8,8 ms	11,2 ms	∞	0 ms	15,2 ms	4,8 ms	∞	0 ms	12,9 ms	7,1 ms	11,3 ms	8,7 ms
Sporadic	9,3 ms	10,7 ms	∞	0 ms	16 ms	4 ms	∞	0 ms	12,1 ms	7,9 ms	10,6 ms	9,4 ms
Client/Server	11,8 ms	8,2 ms	∞	0 ms	17,2 ms	2,8 ms	∞	0 ms	14,3 ms	5,7 ms	15,8 ms	4,2 ms

Figure 10. Observed latency and temporal accuracy for each mode (no redirection, redirection of all messages, redirection of timely messages) in the three configurations (periodic, sporadic, client/server). A minimum temporal accuracy of 0 indicates a loss of temporal accuracy in the experiment.

3) Configuration for Client/Server Communication: In this configuration the CAN network performs client/server interactions, while periodic communication takes place on the Ethernet network. The third row in Figure 8 shows the respective network adaptors. The retrieval network adaptor of the gateway receives the request messages from the CAN network. After the arrival of such a request message, the retrieval network adaptor sends a reply message on the CAN network with information pulled out of the real-time database. For the request messages, we use the same burst timing model as for the unidirectional sporadic messages. The feeder network adaptor operates like in the configuration for periodic communication. It acquires information from the Ethernet network and pushes the information into the real-time database.

VI. EXPERIMENTAL RESULTS

This section describes the observed timing for the three configurations (i.e., periodic, unidirectional sporadic, client/server) of the prototype. We observed the *cluster-internal communication*, which comprises the messages exchanged between nodes within a cluster. In addition, measurements the *cluster-external communication* are provided. The cluster-external communication consists of messages redirected between clusters via the gateway. For both cluster-internal and cluster-external communication, the message latencies and the temporal accuracy of the state variables updated by the messages have been captured.

For each of the three configurations, we have tested the gateway in three operational modes. The third mode is the most interesting one for the experimental evaluation, while the other two serve as references.

1) *No redirection of messages.* The gateway blocks all messages.
2) *Redirection of all messages.* The gateway redirects all messages without imposing any temporal specification. No temporal partitioning is performed.
3) *Redirection of timely messages.* The gateway redirects only timely messages, which comply with the temporal specification. Thus the gateway performs temporal partitioning by enforcing the temporal specification of the respective network with the network adaptors of Section V-B.

In conjunction with the three operational modes, the temporal behavior of the cluster-internal communication provides evidence regarding the effectiveness of the temporal partitioning. In particular, the timing of the third mode compared to the timing of the second mode without temporal partitioning shows the benefits of temporal partitioning. Furthermore, we can perform a comparison with the timing of the first mode that trivially provides temporal partitioning by performing no redirection at all.

The timing of the cluster-external communication in the three operational modes shows the performance of the gateway and the overhead due to the fault and error containment mechanisms. By comparing the second and the third mode, we see the performance penalty due to blocking untimely messages.

Figure 10 summarizes the experimental results for each mode of operation and each configuration. For the periodic and client/sever configurations, the latencies of the cluster-internal communication are also depicted graphically in Figures 11 and 12.

A. Configuration for Periodic Communication

Figure 11 graphically depicts the latencies of the messages on the CAN network excluding the redirected messages (i.e., only cluster-internal communication). This diagram gives an impression of the impact induced by the redirection of messages on the timing of the cluster-internal communication. When the gateway blocks all messages, the maximum observed latency of the cluster-internal communication is 8.8ms. In case the gateway redirects all messages, the observed latencies increase significantly. 21% of the messages have a latency above 20ms. 7% of the messages are never delivered and subject to omission failures. In case the gateway performs temporal partitioning by enforcing the temporal specification using the network adaptors (cf. Section V-B.1), the maximum observed latency of the cluster-internal communication is 15.2ms and all messages are delivered.

The cluster-external communication (i.e., the messages redirected between clusters) exhibits maximum latencies of 11.3ms without temporal partitioning and 12.9ms with temporal partitioning. The temporal partitioning mechanisms delay and block untimely inter-cluster communication. Due to not redirecting messages immediately, the latencies with temporal partitioning are higher.

B. Configuration for Unidirectional Sporadic Communication

The results for the configuration with unidirectional sporadic communication are similar to the ones of the previous configuration (cf. second row in Figure 10). However, the unpredictability resulting from the randomness of the message interarrival times causes higher latencies and smaller temporal accuracy. Compared to the
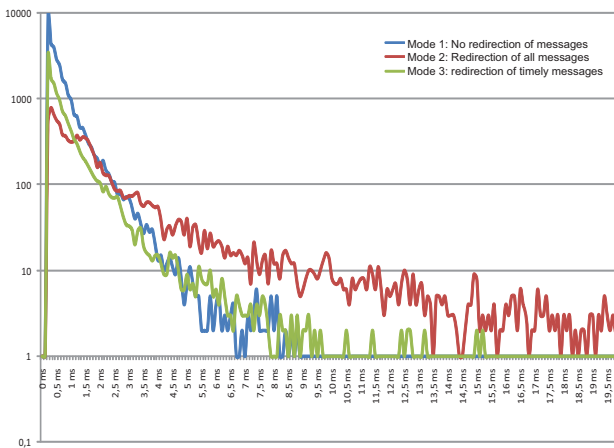
Figure 11. Latencies in configuration for periodic communication: vertical axis denotes number of messages (with logarithmic scaling), message latency is distinguished along the horizontal axis

previous configuration, more nodes try to send messages at the same time, thereby causing higher delays.

When the gateway blocks all messages, the maximum observed latency of the cluster-internal communication is 9.3ms. In case the gateway redirects all messages, the observed latencies increase significantly and 11% of the messages are never delivered in the observation period. In case the gateway performs temporal partitioning by enforcing the temporal specification using the network adaptors, the maximum observed latency of the cluster-internal communication is 16ms and all messages are delivered.

For the cluster-external communication, maximum latencies of 10.6ms were observed without partitioning and 12.1ms with partitioning.

C. Configuration for Client/Server Communication

Figure 12 shows the experimental results for the client/server configuration. Compared to the periodic and unidirectional sporadic communication, this configuration exhibits a higher load on the network since both request and reply messages are exchanged. Request messages exhibit random interarrival times. Without redirection, the maximum latency on the CAN network is 11.8ms. When redirecting messages with temporal partitioning the latency increases to 17.2ms in this configuration. Without temporal partitioning, 29% of the messages are not delivered within the observation time.

Due to the higher latencies in this configuration, the latency of the cluster-external communication also reaches higher values. Without temporal partitioning the maximum latency is 14.3ms. With temporal partitioning the maximum latency is 15.8ms.

VII. DISCUSSION OF RESULTS

The experiments have demonstrated the gateway's ability to prevent one cluster from causing timing failures in another cluster. The gateway enforces a temporal specification that is encoded in the network adaptors using state machine with timing constraints.
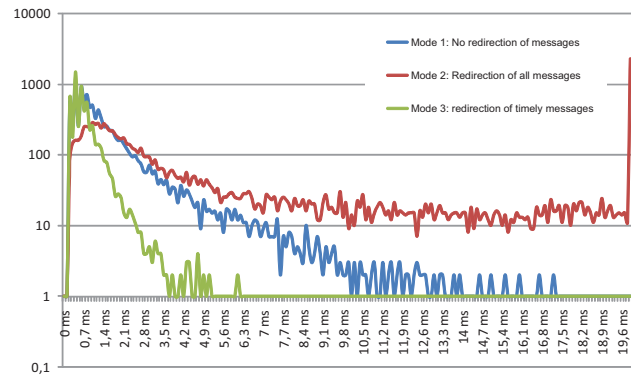

Figure 12. Latencies in configuration for client/server communication

Since messages are redirected to a CAN network, the gateway has an effect on the timing of the cluster-internal communication. The flexibility of CAN, which does not enforce predefined message transmission instants like a time-triggered communication protocol (e.g., FlexRay), limits the capabilities of temporal partitioning. The redirected messages can delay the CAN messages exchanged internally between the nodes of the CAN cluster. However, this effect is bounded due to the temporal constraints of the gateway. The temporal constraints include the periods in the periodic scenario and the minimum interarrival times and burst constraints in the sporadic and client/server configurations. In the example used for the experiments, the temporal partitioning performed by the gateway ensures that the real-time image does not lose its temporal accuracy.

On the other hand, without partitioning mechanisms, i.e., when state machines are missing in the network adaptors that would constrain the redirection of messages, temporal accuracy is lost. In addition, messages exhibit higher higher latencies or are not delivered at all.

VIII. CONCLUSIONS AND FUTURE WORK

Fault and error containment play an important role in mixed criticality systems, where gateways interconnect non safety-critical and safety-critical clusters. Such mixed criticality systems are typical in many applications, e.g., in the automotive and avionic domains.

The presented gateways based on a real-time database and network adaptors provide a solution to this challenge. The real-time database decouples the interconnected clusters and facilitates the containment of timing and value message failures. The real-time database is accessed through network adaptors, which incorporate state machines with timing constraints in order to enforce temporal specifications (e.g., message interarrival times, bandwidth constraints). In addition, network adaptors support the containment of value message failures through checks on message contents (e.g., message identifier, address, temporal accuracy). Furthermore, the real-time database provides information regarding the temporal accuracy of time-sensitive information, thereby enabling the containment of a specific type of value message failures: messages containing values that have been invalidated by the progression of time.

The prototype implementation and the experiments have shown the effectiveness of the gateways for the containment of timing failures in three realistic example scenarios (i.e., periodic, sporadic and client/server communication). In future work, an experimental evaluation of further scenarios (e.g., streaming communication) and an evaluation of the containment of value failures is planned.

REFERENCES

[1] G. Leen and D. Heffernan, "Expanding automotive electronic systems," *Computer*, vol. 35, no. 1, pp. 88–93, Jan. 2002.

[2] *CAN Specification, Version 2.0*, Robert Bosch Gmbh, Stuttgart, Germany, 1991.

[3] *MOST Specification Version 2.2*, MOST Cooperation, Karlsruhe, Germany, Nov. 2002.

[4] *LIN Specification Package Revision 2.0*, LIN Consortium., Sept. 2003.

[5] *FlexRay Communications System Protocol Specification Version 2.1*, FlexRay Consortium. BMW AG, DaimlerChrysler AG, General Motors Corporation, Freescale GmbH, Philips GmbH, Robert Bosch GmbH, and Volkswagen AG., May 2005.

[6] M. Skutekand, M. Mekhaiel, and G. Wanielik, "A precrash system based on radar for automotive applications," in *Proc. of the IEEE Intelligent Vehicles Symposium*, June 2003, pp. 37–41.

[7] N. El-Awar, D. Geer, T. Krellner, and P. Straub, "Automotive temperature sensing," Keystone Thermometrics, Tech. Rep., Sept. 1999.

[8] J. Pouzet and P. Platt, "Flexible airborne architecture," in *Proc. of the NASA ICNS Conference*. Eurocontrol Communication and Surveillance Business Unit, May 2006.

[9] *DO-178B: Software Considerations in Airborne Systems and Equipment Certification*, Radio Technical Commission for Aeronautics, Inc. (RTCA), Washington, DC, Dec. 1992.

[10] *ISO/WD 26262: Road vehicles - Functional safety. Working Draft*, Int. Standardization Organisation, 2007.

[11] J. Rushby, "Modular certification," Computer Science Laboratory SRI International, 333 Ravenswood Avenue, Menlo Park, CA 94025, USA, Tech. Rep., Sept. 2001.

[12] R. Obermaisser, "A model-driven framework for the generation of gateways in distributed real-time systems," in *Proc. of the 28th IEEE Real-Time Systems Symposium*, Tucson, Arizona, USA, Sept. 2007.

[13] Y. Chawathe, S. Fink, S. McCanne, and E. Brewer, "A proxy architecture for reliable multicast in heterogeneous environments," *ACM Multimedia*, pp. 151–159, 1998.

[14] J. Stankovic, S. Son, and C. Nguyen, "The cogency monitor: an external interface architecture for a distributed object-oriented real-time database system," in *Proc. of the Fourth IEEE Real-Time Technology and Applications Symposium*, June 1998, pp. 71–78.

[15] R. Ferguson, B. Peterson, and H. Thompson, "System software framework for system of systems avionics," in *Proc. of the 24th Digital Avionics Systems Conference*, vol. 2, Oct. 2005, pp. 8.A.1–1 – 8.A.1–10.

[16] J. Lala and R. Harper, "Architectural principles for safety-critical real-time applications," in *Proc. of the IEEE*, ser. 1, vol. 82, Jan. 1994, pp. 25–40.

[17] D. Powell, "Failure mode assumptions and assumption coverage," in *Proc. of the 22nd IEEE Annual Int. Symposium on Fault-Tolerant Computing (FTCS-22)*, Boston, USA, July 1992, pp. 386–395.

[18] H. Kopetz, "Fault containment and error detection in the time-triggered architecture," in *Proc. of the Int. Symposium on Autonomous Decentralized Systems*, Pisa, Italy, Apr. 2003.

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

[20] R. Obermaisser, "Formal specification of gateways in integrated architectures," in *Proc. of 6th IFIP Workshop on Software Technologies for Future Embedded & Ubiquitous Systems (SEUS 2008), LNCS 5287*. Capri Island, Italy: Springer, 2008.

[21] J. Rushby, "Partitioning for avionics architectures: Requirements, mechanisms, and assurance," NASA Langley Research Center, NASA Contractor Report CR-1999-209347, June 1999.

[22] H. Kopetz and G. Grünsteidl, "TTP – a protocol for fault-tolerant real-time systems," *Computer*, vol. 27, no. 1, pp. 14–23, Jan. 1994, vienna University of Technology, Real-Time Systems Group.

[23] G. Coulouris, J. Dollimore, and T. Kindberg, *Distributed Systems: Concepts and Design*, 2nd ed. Int. Computer Science Series, Addison-Wesley, 1994.

[24] H. Kopetz, A. Ademaj, P. Grillinger, and K. Steinhammer, "Time-triggered Ethernet protocol specification, version 0.85," Technische Universität Wien, Institut für Technische Informatik, Treitlstr. 1-3/182-1, 1040 Vienna, Austria, Tech. Rep., 2005.

[25] H. Kopetz and et al., "The Time-Triggered Ethernet (TTE) design," in *Proc. of 8th IEEE Int. Symposium on Object-oriented Real-time distributed Computing (ISORC)*, May 2005.

[26] K. Tindell and A. Burns, "Guaranteed message latencies for distributed safety-critical hard real-time control networks," Dept. of Computer Science, University of York, Tech. Rep. YCS229, June 1994.

[27] M. Borovicka, "Design of a Gateway for the Interconnection of Real-Time Communication Hierarchies," Master's thesis, Technische Universität Wien, Institut für Technische Informatik, Vienna, Austria, 2003.

Roman Obermaisser Roman Obermaisser is an assistant professor at the Institute of Computer Engineering (Real-Time Systems Group) at the Vienna University of Technology. He has studied Computer Sciences and has finished his doctoral studies in February 2004. His research interests are system architectures for distributed embedded real-time systems, communication infrastructures of integrated computer systems and time-triggered communication protocols (e.g., FlexRay, TTP or TTEthernet). Roman Obermaisser is the technical coordinator of the FP7-ICT collaborative project GENESYS (GENeric Embedded SYStem Platform) with 23 partners. The GENESYS project has started in January 2008 and is aiming to give birth to novel technologies for embedded computer systems that are of universal importance across individual application domains.