

Modular Formal Analysis of the Central Guardian in the Time-Triggered Architecture^{*}

Holger Pfeifer and Friedrich W. von Henke

Fakultät für Informatik, Universität Ulm
D-89069 Ulm, Germany
{pfeifer|vhenke}@informatik.uni-ulm.de

Abstract. We present a modular formal analysis of the communication properties of the Time-Triggered Protocol TTP/C based on the guardian approach. The guardian is an independent component that employs static knowledge about the system to transform arbitrary node failures into failure modes that are covered by the rather optimistic fault hypothesis of TTP/C. Through a hierarchy of formal models, we give a precise description of the arguments that support the desired correctness properties of TTP/C. First, requirements for correct communication are expressed on an abstract level. By stepwise refinement we show that the abstract requirements are met under the optimistic fault hypothesis, and how the guardian model allows a broader class of failures be tolerated.

1 Introduction

The Time-Triggered Architecture (TTA) [1–3] is a distributed computer architecture for the implementation of highly dependable real-time systems. In particular, it targets on embedded control applications, such as *by-wire* systems in the automotive or aerospace industry [4]. For these safety-critical systems fault tolerance is of utmost importance. The Time-Triggered Protocol TTP/C constitutes the core of the communication level of the Time-Triggered Architecture. It furnishes a number of important services, such as atomic broadcast, consistent membership and protection against faulty nodes, that facilitate the development of these kinds of fault-tolerant real-time applications. However, these protocol mechanisms rely on a rather optimistic fault hypothesis and assume that a fault is either a reception fault or a consistent send fault of some node [5]. In order to extend the class of faults that can be tolerated a special hardware component, the so-called *guardian*, is introduced [6]. A guardian is an autonomous unit that protects the shared communication network against faulty behaviour of nodes by supervising their output. The original bus topology of the communication network employed local bus guardians, which were placed between the nodes and the bus. In the more recent star topology, central guardians are used in the hub of each star. The guardian makes use of static knowledge available in

^{*} This research was supported by the European Commission under the IST project NEXT TTA (IST-2001-32111).

a TTA-based system to transform arbitrary node failures into those that are covered by the optimistic fault hypothesis. For example, the time when a given node sends messages is statically determined in a TTA system and known *a priori*. The guardian can hence control the correct timing of messages by granting access to the network only during a node’s pre-defined slot.

The goal of this work is to formally model TTP/C guardians and analyse their fault tolerance properties. In particular, we aim at describing the benefits of the guardians by giving a precise specification of the assumptions on which the derivation of the properties is based. Formal analysis can provide an additional source of confidence in correct behaviour of a system, which is particularly important in the context of safety-critical systems. Several aspects of TTP/C and related protocols have therefore been formally modelled and analysed, including clock synchronisation [7], group membership [8–10], and the startup procedure [11, 12]. A detailed overview of formal analysis work for the Time-Triggered Architecture is given by Rushby [13]. While so far the protocol algorithms of the time-triggered protocol have been the focus of the formal analyses cited above, we concentrate in this paper on the communication properties of TTP/C, thereby complementing and extending previous work.

To describe the behaviour and properties of the communication network and the guardians we develop various formal models, which are organised in a hierarchical fashion. We start by specifying the desired correctness properties of the communication in an abstract form. Subsequently, in a process of stepwise refinement, more detail is added to this initial abstract model. On the next level of the hierarchy, we consider a TTP/C system without guardians. We show that in this case the strong, optimistic fault hypothesis is necessary to guarantee correct communication. Another model then introduces guardians and specifies their behaviour. At this level we demonstrate that the optimistic assumptions can be relaxed, which leads to a fault hypothesis that covers a broader class of faults. The development of the models is in the spirit of, and builds on the work on modelling TTP-related aspects that has been carried out previously [7, 9, 14]. Specifically, it continues the use of the PVS specification and verification system [15] to both specify the model and the properties to be verified, and develop formal proofs that the model satisfies the stated properties. Previous work has demonstrated the suitability of PVS for this type of tasks.

The paper is organised as follows. In Sect. 2 we give a brief overview of the main aspects of the Time-Triggered Architecture. Section 3 describes the structure of the models and motivates their organisation. Details of the components of the formal models are elaborated in Sect. 4. Finally, we conclude in Sect. 5.

2 Brief Overview of the Time-Triggered Architecture

In this section we only briefly describe the main aspects of the Time-Triggered Architecture to the extent that is required for this paper. For more detailed presentations we refer to [3, 16, 17]. In a Time-Triggered Architecture system a set of *nodes* are interconnected by a real-time communication system. Nodes consist of

the host computer, which runs the application software, and the communication controller, which accomplishes the time-triggered communication between different nodes. The nodes communicate via replicated shared media, the communication *channels*. There are two common physical interconnection topologies for TTA. Originally, the channels were replicated, passive buses, while in the more recent star topology the nodes are connected to replicated central star couplers, one for each of the two communication channels.

The distinguishing characteristic of time-triggered systems is that all system activities are initiated by the passage of time [18]. The autonomous TTA communication system periodically executes a time-division multiple access (TDMA) schedule. Thus, access to the network is divided into a series of intervals, called *slots*. Every node exclusively owns certain slots in which it is allowed to send messages. The times of the periodic sending actions are determined at design time of the system, and a static scheduling table, stored at each communication controller, contains the send and receive instants. It thus provides common knowledge about message timing to all nodes. A complete cycle during which every node has access to the network exactly once is called a *TDMA round*.

Messages are used as a life-sign of the respective sender, and whenever a node receives a correct frame on at least one of the channels it considers the sender correct. Correctness of a frame is determined by each receiving node according to a set of criteria. A node considers a frame correct, if it is well-timed, i. e. arrives within the boundaries of the TDMA slot, the physical signal obeys the line encoding rules, the frame passes a CRC check, and the sender and receiver agree on the distributed protocol state, the so-called *C-state*. One of the desired correctness properties of TTP/C is that all correct nodes always agree on whether or not a message is considered correct.

The Time-Triggered Protocol is designed to provide fault tolerance. In particular, the protocol has to ensure that non-faulty nodes receive consistent data despite the presence of possibly faulty nodes or a faulty communication channel. The provision of fault tolerance is based on a number of assumptions about the types, number, and frequency of faults. Altogether, these assumptions constitute the so-called *fault hypothesis*. The main assumption for the algorithms implemented in TTP/C is that a fault manifests itself as either a reception fault or a consistent send fault of some node [5]. In particular, the TTP/C services rely on transmission faults being consistent. That is, messages must be received correctly by either all non-faulty nodes or none. Moreover, nodes are assumed not to send messages outside their assigned slots. With respect to faults of the communication network, it is assumed that the channels cannot spontaneously create correct messages, and that messages are delivered either with some known bounded delay or never. With regard to the frequency and number of faults, TTP/C assumes that only one node becomes faulty during a TDMA round, and that there is at most one faulty node or one faulty channel at a time. However, the Time-Triggered Architecture can tolerate a broader class of faults by intensively using the static knowledge present in the TDMA schedule. This allows to transform arbitrary failure modes of nodes into either send or receive faults that

can be tolerated by the protocol. The guardians monitor the temporal behaviour of the nodes and bar a faulty node from sending a message outside its designated slots. Thus, timing failures are effectively transformed into send faults.

Moreover, guardians also protect against so-called *slightly-off-specification* (SOS) faults, which are a particular class of *Byzantine* faults. A component is called SOS-faulty if it exhibits only marginally faulty behaviour that appears correct to some components, but faulty to others. A slightly-off-specification timing fault could occur if the transmission of a node terminates very close to the end of its scheduled transmission interval; thus, some receivers might accept the message while others might consider it mistimed. Because the duration of a particular transmission is known beforehand, the guardian can prevent such a cut-off scenario. A node must begin its transmission during a pre-defined period of time after the start of its slot, otherwise the guardian would terminate the right to access the communication network. Thus, the guardian can effectively prevent cut-off SOS faults if the transmission interval is chosen long enough to ensure that a transmission fits the interval whenever it is started in time. Specifically, TTP/C guardians protect against SOS faults in the line encoding of frames at the physical layer, SOS timing faults, transmission of data outside the designated sending slots, masquerading of nodes, and transmission of non-agreed critical state information [6].

3 Bird's Eye View of the Formal Models

The overall goal of modelling the communication network is to provide a concise description of the arguments that support the following three main correctness properties of the TTP/C communication:

- *Validity*: If a correct node transmits a correct frame, then all correct receivers accept the frame.
- *Agreement*: If any correct node accepts a frame, then all correct receivers do.
- *Authenticity*: A correct node accepts a frame only if it has been sent by the scheduled sending node of the given slot.

Once these properties are established, they can be exploited in subsequent analyses of protocol algorithms. This is preferable, since it is generally more feasible to base an analysis on properties of a supporting model or theory, rather than on the mere definitions of the model itself.

In order to facilitate the deduction, the formal proofs of these properties are decomposed into a series of smaller steps, and a hierarchy of corresponding models has been developed. Each of the single models focuses on a particular aspect of the communication. Altogether, we have identified the following four suitable model layers:

- General specification of the reception of frames.
- Channels without guardians, requiring a strong fault hypothesis.

- Channels with guardians, requiring only a weaker fault hypothesis.
- Different network topologies: local bus guardians and central guardians.

Each of the models contributes a small step towards proving the desired correctness properties. The steps themselves are each based on a set of assumptions or preconditions. Put in an abstract, and maybe also a slightly over-simplified way, on each model layer i one establishes a theorem of the form

$$assumptions_i \Rightarrow properties_i .$$

The idea is to design the different models in such a way that the properties on one level establish the assumptions on the next. Ultimately, the models are integrated and the reasoning is combined, yielding a chain of implications of roughly the following kind:

$$\begin{aligned} assumptions_0 &\Rightarrow properties_0 \\ &= \text{ or } \Rightarrow \\ &assumptions_1 \Rightarrow properties_1 \\ &= \text{ or } \Rightarrow \\ &assumptions_2 \Rightarrow \dots \Rightarrow properties_f . \end{aligned}$$

The final properties, $properties_f$, correspond to the desired main correctness properties of the TTP/C communication as specified above, while the initial assumptions, $assumptions_0$, describe what constitutes the basic fault hypothesis.

We are going to briefly summarise the main aspects of the four model layers. At the bottom, the model describes the reception of frames by the nodes. Here, the various actions that nodes take in order to judge the correctness of the received frame are formalised. This amounts to considering the transmission time and the physical encoding rules of the frame, and the outcomes of the CRC check and the C-state agreement check, respectively [17]. The main correctness properties of the communication network are then expressed in terms of these notions. The assumptions of this model layer concern requirements about the functionality of the communication channels. In particular, they describe properties of the frames that a channel transmits, such as physical encoding or delivery times, and reflect the hypothesis about possible faults of the communication network. In essence, this model establishes a proposition that informally reads as follows:

$$general_channel_properties \Rightarrow Validity \wedge Agreement \wedge Authenticity . \quad (1)$$

On the next level, we model the transmission of frames through channels that are not equipped with guardians. The goal is then to derive the assumptions of the basic model, as covered by the expression $general_channel_properties$. However, in order to do so, a strong hypothesis on the types of possible faults of nodes is necessary. This strong fault hypothesis requires, for instance, that even a faulty node does not send data outside its sending slot, and nodes never send correct frames when they are not scheduled to do so. Using our informal notation, we can sketch the reasoning at this model layer as follows:

$$strong_fault_hypothesis \Rightarrow general_channel_properties . \quad (2)$$

Guardians are employed to transform arbitrary node faults into faults that are covered by the strong fault model. Thus, the strong fault hypothesis can be replaced with weaker assumptions on the correct behaviour of the guardians. The functionality and the properties of the guardians are formally specified in the third model of the hierarchy, where the following fact is established:

$$\textit{weaker_fault_hyp.} \wedge \textit{generic_guardian} \Rightarrow \textit{general_channel_properties} . \quad (3)$$

Ideally, we would have liked to demonstrate directly that – together with the guardian properties – the weak form of the fault hypothesis implies the strong one. However, it turned out to be rather challenging to accomplish a formal proof for this fact and hence we had to revert to reasoning according to (3).

The model of the guardians is generic, as it does not, for instance, stipulate the type of guardian to be used in the communication network. The final level of our hierarchy models each of the two typical topologies of a TTP/C network: the bus topology and the star topology. In the former, each node of the network is equipped with its own local bus guardian, one for each channel, while in the latter the guardians are placed into the central star-coupling device of the channels. On this model layer we show that the properties of the guardians are independent from the choice of a particular topology, given that both the local bus guardians and the central guardians implement the same algorithms. Hence, we establish the following facts:

$$\textit{local_bus_guardian} \Rightarrow \textit{generic_guardian} . \quad (4)$$

$$\textit{central_star_guardian} \Rightarrow \textit{generic_guardian} . \quad (5)$$

The hierarchic arrangement of the models for the communication network allows for a concise description of the dependencies of the three main correctness properties. On the basic level the fundamental prerequisites are described that are necessary for the desired correctness properties to hold, while the subsequent levels express what must be assumed from the nodes and guardians, respectively, to satisfy these prerequisites. In particular, the treatment precisely explains the benefits of introducing guardians into the communication network.

4 Formal Analysis of TTP/C Communication

In this section we present the main details of the formal models for the communication network according to the hierarchy that has been set out in the previous section. The presentation is in the style of a mathematical transcription of PVS modules that have been developed. However, due to space restrictions, only verbal explanations of proofs can be provided. For a comprehensive description we refer to an accompanying technical report [19].

4.1 General Model for the Reception of Frames

In our general formal model, we divide communication between nodes into three phases: the sending of frames by a sending node, the transmission of the frame

on a channel, and the reception of the frame at the receiving nodes. To model the reception of frames we introduce a function $rcvd(n, c, r)$ to denote the frame a receiving node r receives in slot n on channel c . Similarly, $sent(n, c, p)$ denotes the frame that a node p has sent in slot n on channel c , while $transmit(n, c)$ models the frame that is transmitted on channel c in slot n . For a non-faulty node r we assume that it receives the frame that is transmitted on the channel, and that even faulty nodes cannot receive other messages than those transmitted – or nothing, in the case of a reception fault. As for the sending nodes, we assume that in their sending slots non-faulty nodes either send a correct frame on all channels, or do not send any frame on any channel; the latter is required for nodes that are not yet integrated.

In TTP/C, frames can be *null* frames, *valid* frames, or *correct* frames. The status of the frame received by node r on channel c in slot n will be denoted by $frame_status(n, c, r)$. Frames are considered syntactically valid if the frame is transmitted during the receive window of the receiving node, no code violations are observed during the reception, and no other transmission was active within the receive window before the start of the frame. For a frame to be considered correct, it has to pass both the CRC check and the C-state agreement check [5].

Next, we formally state the desired correctness properties introduced in the previous section. The node scheduled to send in slot n is denoted $sender(n)$, while we use the (overloaded) notation \mathcal{NF}^n to denote both the set of non-faulty nodes and the non-faulty channels in slot n ; consequently, $r \in \mathcal{NF}^n$ and $c \in \mathcal{NF}^n$ indicate that node r and channel c are non-faulty in slot n .

Property 1 (Validity). *For all slots n , there exists a channel c such that if the sender of slot n sends a correct frame on c then all non-faulty nodes will receive this frame and assign the status correct to it:*

$$\begin{aligned} \exists c : sender(n) \in \mathcal{NF}^n \wedge sends_correct(n, c, sender(n)) \Rightarrow \\ \forall r \in \mathcal{NF}^n : rcvd(n, c, r) = sent(n, c, sender(n)) \wedge \\ frame_status(n, c, r) = correct \end{aligned}$$

Here, the predicate $sends_correct(n, c, sender(n))$ subsumes what is considered a correct sending action of a node: the sending node sends a non-null frame, does so at the specified time, the frame carries the correct C-state information and the physical signal obeys the line encoding rules.

Property 2 (Agreement). *All non-faulty nodes consistently assign the frame status correct to a frame received on a non-faulty channel c :*

$$\begin{aligned} p \in \mathcal{NF}^n \wedge q \in \mathcal{NF}^n \wedge c \in \mathcal{NF}^n \Rightarrow \\ frame_status(n, c, p) = correct \Leftrightarrow frame_status(n, c, q) = correct \end{aligned}$$

Property 3 (Authenticity). *A non-faulty node r assigns the frame status correct to a frame received on a non-faulty channel c only if it was sent by the scheduled sender of the slot:*

$$\begin{aligned} r \in \mathcal{NF}^n \wedge c \in \mathcal{NF}^n \wedge frame_status(n, c, r) = correct \Rightarrow \\ rcvd(n, c, r) = sent(n, c, sender(n)) \end{aligned}$$

In order to prove that these desired correctness properties hold for our model, several preconditions must be satisfied. We first list these general requirements and subsequently explain their respective purpose and why they are necessary.

General Requirements. The properties *Validity*, *Agreement*, and *Authenticity* can be proved if the following requirements are met:

1. A non-faulty channel either transmits the frame sent by some node, or nothing, or a corrupted frame.

$$c \in \mathcal{NF}^n \Rightarrow (\exists p : \text{transmit}(n, c) = \text{sent}(n, c, p)) \\ \vee \text{transmit}(n, c) = \text{null} \vee \text{corrupted}(\text{transmit}(n, c))$$

2. If the scheduled sender exclusively accesses the channel and sends a correct frame, then this frame is transmitted by the channel.

$$c \in \mathcal{NF}^n \wedge \text{single_access}(n, c) \wedge \text{sends_correct}(n, c, p) \wedge p = \text{sender}(n) \Rightarrow \\ \text{transmit}(n, c) = \text{sent}(n, c, p)$$

3. Channels can only transmit what has been sent by some node.

$$\text{transmit}(n, c) \neq \text{null} \Rightarrow \exists p : \text{sends}(n, c, p)$$

4. In every slot, there is at least one non-faulty channel that is accessed only by a single node.

$$\exists c \in \mathcal{NF}^n : \text{single_access}(n, c)$$

5. The transmission time of a correct frame on a non-faulty channel does not deviate from the sending time by more than some bounded delay d .

$$c \in \mathcal{NF}^n \wedge \text{sends_correct}(n, c, p) \wedge \text{single_access}(n, c) \Rightarrow \\ \exists d : d \leq \text{max_delay} \wedge \text{transmission_time}(f') = \text{send_time}(f) + d \\ \text{where } p = \text{sender}(n), f = \text{sent}(n, c, p), f' = \text{transmit}(n, c)$$

6. If a non-faulty channel transmits a frame with a correct signal encoding, then the frame sent must provide a correct encoding of the sender's C-state.

$$c \in \mathcal{NF}^n \wedge \text{transmit}(n, c) = \text{sent}(n, c, p) \wedge \text{sends}(n, c, p) \wedge \\ \text{signal_encoding_OK}(\text{transmit}(n, c)) \Rightarrow \\ \text{cstate_encoding_OK}(n, \text{sent}(n, c, p), p)$$

7. A non-faulty channel transmits a correctly sent frame only if it originates from the scheduled sender of the given slot.

$$c \in \mathcal{NF}^n \wedge \text{sends_correct}(n, c, p) \wedge \text{transmit}(n, c) = \text{sent}(n, c, p) \Rightarrow \\ p = \text{sender}(n)$$

Requirements 1, 2, and 3 constrain what is considered a correct behaviour of a channel. According to Req. 1, a channel either transmits a frame, or blocks

the transmission completely, or invalidates the frame. The latter two cases particularly allow for the functionality of a guardian. On the other hand, Req. 2 enforces that a non-faulty channel does neither block nor invalidate a correct frame of the scheduled sender. Furthermore, Req. 3 accounts for the fact that channels are passive entities.

The remaining five requirements are necessary to establish formal proofs for the correctness properties. Consider the *Validity* property, which states two things: first, all non-faulty receivers must receive the frame sent by the sender, and, second, all of them must accept this frame. With respect to the first part, we know that all non-faulty nodes receive the frame that channel transmits. Furthermore, Req. 2 states that a non-faulty channel always transmits the frame sent by the sender, provided that there is no other node accessing the channel. To ensure this, Req. 4 is necessary. The expression *single_access* is an abstract parameter of the model, and its concrete interpretation will be defined only in the subsequent refining models layers. As for the second part of *Validity*, we need to demonstrate that all correct nodes receive a non-null, valid frame that passes both the CRC check the C-state agreement check. Non-emptiness of the frame can be proved from Req. 4 and Req. 2, and the fact that the original sender is non-faulty and sends a correct, i. e., non-null, frame. Requirements 5 and again 4 ensure that the received frame is valid, by constraining its timing and its uniqueness on the channel, respectively. C-state agreement is ensured by Req. 6. As for the CRC check, an incorrect CRC checksum is intended to signal a transmission fault. As the channel c under consideration is non-faulty, it is reasonable to assume that no transmission fault occurs on c and consequently that the frame received by a non-faulty receiver passes the CRC check.

The *Agreement* property can be derived in a similar way from the same set of requirements as *Validity*. Here, consistent reception is achieved by assuming that non-faultiness of two nodes comprises that they are synchronised to each other, which ensures that they will receive a frame within their respective receive window consistently. Finally, Req. 7 is necessary to prove *Authenticity*, which requires to show that if a frame is considered correct by a correct receiver r , then node r has in fact received the frame sent by the scheduled sender of the slot. The following two subsections describe under which fault hypotheses these requirements can be met, both for a scenario with and without guardians.

4.2 Strong TTP/C Fault Hypothesis

The general model expresses certain required properties in terms of $sent(n, c, p)$ and $transmit(n, c)$. In a technical sense, these entities are parameters of the model. We now give an interpretation to these parameters for a network without guardians and show that the general requirements are satisfied for these interpretations. In this model, $single_access(n, c)$ is defined true if there are no two different nodes that send on channel c in slot n . The interpretation of $sent(n, c, p)$ and $transmit(n, c)$ is given in an axiomatic style, and the set of axioms essentially constitutes the fault hypothesis of the guardian-free setting.

Again, we first state all of the assumptions of the fault hypothesis and subsequently explain their respective purpose.

Hypotheses for a Network without Guardians. The requirements of the general model are satisfied for a communication network without guardians, if the following assumptions hold:

1. A non-faulty channel without a guardian will transmit a frame sent by a node p if no other node accesses the channel in the given slot n .

$$c \in \mathcal{NF}^n \wedge \text{sends}(n, c, p) \wedge \neg \exists q : q \neq p \Rightarrow \text{sends}(n, c, q) \Rightarrow \text{transmit}(n, c) = \text{sent}(n, c, p)$$

2. If a channel broadcasts a non-null frame, then there is a corresponding node that has sent this frame.

$$\text{transmit}(n, c) \neq \text{null} \Rightarrow \exists p : \text{sends}(n, c, p)$$

3. The delivery time of a frame on a non-faulty channel does not deviate from the transmission time by more than some bounded delay d .

$$c \in \mathcal{NF}^n \wedge \text{sends}(n, c, p) \wedge f' \neq \text{null} \wedge \neg \text{corrupted}(f') \Rightarrow \exists d : d \leq \text{max_delay} \wedge \text{transmission_time}(f') = \text{sending_time}(f) + d$$

where $f = \text{sent}(n, c, p)$, $f' = \text{transmit}(n, c)$

4. Frames sent must contain a correct encoding of the sender's C-state.

$$\text{sends}(n, c, p) \Rightarrow \text{cstate_encoding_OK}(n, \text{sent}(n, c, p), p)$$

5. Correct frames must only be sent by the scheduled sender of a given slot.

$$\text{sends_correct}(n, c, p) \Rightarrow p = \text{sender}(n)$$

6. Nodes other than the sender of a slot, including faulty ones, will not send data on all non-faulty channels outside their assigned sending slots.

$$p \neq \text{sender}(n) \Rightarrow \neg \forall c \in \mathcal{NF}^n : \text{sends}(n, c, p)$$

7. In every slot, there is at least one non-faulty channel.

$$\exists c \in \mathcal{NF}^n$$

8. There is at most one faulty node in every slot.

$$p \notin \mathcal{NF}^n \wedge q \notin \mathcal{NF}^n \Rightarrow p = q$$

In the setting without guardians, all that can be assumed of the frame transmitted by a channel is that it depends on what is sent by the sending nodes. This is what Hyp. 1 does, and it is sufficient to prove Req. 2. To prove Req. 1 one additionally needs Hyp. 2, which, in fact, is actually also identical to Req. 3.

Because we cannot further constrain the behaviour of the channels, some of the requirements of the general model have to be restated as hypotheses at this level. This is also true to some extent for Hyp. 3, which is equivalent to Req. 5. Note, however, that this by no means just an inadmissible simplification of the matter. On the contrary, these hypotheses are direct formalisations of the strong fault hypothesis of the “raw” TTP/C protocol [6]. The other hypotheses are necessary to prove the other requirements of the general model. Specifically, Hyp. 4 and 5 justify Req. 6 and 7, respectively, while Hyp. 6 to 8 establish Req. 4.

What makes this set of hypotheses strong or optimistic is the fact that assumptions are not restricted to non-faulty nodes, but also encompass faulty ones, cf. Hyp. 4, 5, and 6. In the following subsection these hypotheses will be replaced with weaker ones about the behaviour of *non-faulty* guardians.

4.3 Guardians

In our guardian model, we use $g(c)$ to denote the guardian of channel c . We think of a guardian having incoming links from each of the nodes of the network, and corresponding outgoing links. The task of a guardian is to receive the frames sent by the nodes, analyse them, and relay them to the other nodes according to certain rules. Obviously, these rules would prescribe, among other things, that only the frame of the scheduled sender of a slot is relayed. Hence, to describe the functionality of a guardian we use a function $relay(n, g(c), p)$ that denotes the frame the guardian $g(c)$ relays from node p in slot n . Regarding the interpretation of $transmit(n, c)$, we say that a frame is transmitted on a channel c if there is a node p such that the guardian $g(c)$ of channel c relays that frame for p , and does not relay any frame for all nodes other than p . As the guardian prevents nodes from concurrently accessing a channel, we can define $single_access(n, c)$ to yield true in the guardian model for all slots n . Furthermore, a channel is considered non-faulty if its corresponding guardian is. The following assumptions are made in the guardian model.

Hypotheses for Guardians. The requirements of the general model are satisfied for a network with guardians, if the following hypotheses hold:

1. If the scheduled sender of a slot sends a correct frame, then a correct guardian relays this frame.

$$p = sender(n) \wedge g(c) \in \mathcal{NF}^n \wedge sends_correct(n, c, p) \Rightarrow relay(n, g(c), p) = sent(n, c, p)$$

2. Frames of nodes other than the scheduled sender must not be relayed.

$$p \neq sender(n) \wedge g(c) \in \mathcal{NF}^n \Rightarrow relay(n, g(c), p) = null$$

3. If a sending node does not start to send its frame within the nominal sending window, the guardian closes the window with the effect that a null frame is relayed.

$$p = sender(n) \wedge g(c) \in \mathcal{NF}^n \wedge \neg sending_time_OK(n, sent(n, c, p), p) \Rightarrow relay(n, g(c), p) = null$$

4. If the signal encoding of the frame sent by a node violates the coding rules the guardian terminates the transmission of the frame prematurely, thus corrupting the frame.

$$p = \text{sender}(n) \wedge g(c) \in \mathcal{NF}^n \wedge \neg \text{signal_encoding_OK}(\text{sent}(n, c, p)) \Rightarrow \text{corrupted}(\text{relay}(n, g(c), p))$$

5. If the C-state encoded in a frame does not correspond to the guardians own C-state, then the guardian aborts the transmission of the frame, and the relayed frame will be corrupted.

$$p = \text{sender}(n) \wedge g(c) \in \mathcal{NF}^n \wedge \neg \text{cstate_encoding_OK}(n, \text{sent}(n, c, p), p) \Rightarrow \text{corrupted}(\text{relay}(n, g(c), p))$$

6. Guardians are passive and can only relay frames that have actually been sent by some node.

$$\text{relay}(n, g(c), p) \neq \text{null} \Rightarrow \text{sends}(n, c, p)$$

7. A guardian transmits a relayed frame with a bounded delay.

$$g(c) \in \mathcal{NF}^n \wedge \text{sends}(n, c, p) \wedge f' \neq \text{null} \Rightarrow \exists d : d \leq \text{max_delay} \wedge \text{transmission_time}(f') = \text{sending_time}(f) + d$$

where $f = \text{sent}(n, c, p)$, $f' = \text{relay}(n, g(c), p)$

8. For all slots, the guardian of at least one of the channels is non-faulty.

$$\exists c : g(c) \in \mathcal{NF}^n$$

9. A faulty guardian fails silently and does not relay any frame.

$$g(c) \notin \mathcal{NF}^n \Rightarrow \text{relay}(n, g(c), p) = \text{null}$$

Hypotheses 1 and 2 describe the basic functionality of a guardian, while the supervising functions are expressed by Hyp. 3, 4, and 5. Note that the only assumption that is made about a faulty guardian is that it fails silently, cf. Hyp. 9. These hypotheses ensure that a non-faulty guardian always transmits a correct frame sent by the scheduled sender of a given slot, cf. Req. 2, and corrupted or null-frames otherwise, see Req. 1. Combining these two facts establish Req. 7. Requirement 3 can be proved from Hyp. 6, while Req. 4 follows from Hyp. 8. Requirement 5 on the bounded delay of transmissions is established by Hyp. 7, and Hyp. 4 and Hyp. 5 are used to prove Req. 6.

4.4 Local vs. Central Guardians

The guardian model described above is generic in the sense that it does not determine the type of guardians used in the network. By refining the interpretation of the denotation $g(c)$, it can be applied to both an interconnection network with a star topology and one with a bus topology. In fact, the generic model can

directly be matched to central guardians, while for an interconnection network that uses the bus topology one introduces a function lbg to denote particular guardian devices, such that $lbg(p, c)$ is the local bus guardian of node p for channel c . Then, the expression $g(c)$ would denote a function that yields for a given node its local bus guardian that controls channel c . For space reasons, however, we have to omit the details.

In essence we can state that, as long as the same algorithms and supervising functions are implemented in either guardian type, both the local bus guardians and the central guardians of a star coupler provide the functionality to satisfy the requirements stated in the basic model and thus ensure that the main correctness properties for the communication of TTP/C hold.

5 Conclusions

The goal of formally analysing aspects of the Time-Triggered Architecture is to provide mathematically substantiated arguments that architecture and algorithms provide certain services and satisfy certain critical properties.

In this regard we have presented a formal analysis of the guardian-based communication of TTP/C. We have developed a series of formal models of the interconnection network that are hierarchically structured and formalise different aspects of the communication of TTP/C nodes at various levels. The basic level provides a precise specification of the desired correctness properties of the TTP/C communication. It states several requirements on both the behaviour of the sending nodes and the channels that must be satisfied in order to guarantee that the correctness properties hold. These requirements serve as an interface of the model. In a process of stepwise refinement we have proved the validity of these properties for TTP/C by showing that the interface requirements hold for the refined model layers. The organisation of the model hierarchy not only facilitates the formal proof by dividing it into manageable steps. It also reflects the structure of what constitutes Time-Triggered Protocol, viz. the communication controllers of the nodes, and the guardians. The former provide the fault-tolerant protocol services on the basis of strong fault assumptions, which, in turn, are guaranteed by the guardians. Thus, one of the benefits of our formal analysis is that the formal models yield a concise formal description of the respective purposes and dependencies of these components, and precisely state the assumptions on a guardian, which previously have been stated only informally [6].

The analysis of the properties of the communication network of TTA supports the claim that the functionality of the guardians ensures that arbitrary node failures are converted into fault modes the TTP/C protocol algorithms can tolerate. Thus, the strong fault hypothesis of TTP/C can be replaced by a weaker, minimal fault hypothesis on the correct behaviour of the guardians, which has two direct advantages. First, applications of TTA can rely on the architecture to tolerate a broad class of faults, and, second, protocol algorithms of TTP/C can be designed for and analysed under the strong fault model, which allows for simpler algorithms and significantly facilitates formal analysis.

References

1. Kopetz, H.: The Time-Triggered Approach to Real-Time System Design. In: Predictably Dependable Computing Systems. Springer-Verlag (1995)
2. Kopetz, H.: The Time-Triggered Architecture. In: Proc. 1st Intl. Symp. on Object-Oriented Real-Time Distributed Computing. (1998) 22–31
3. Kopetz, H., Bauer, G.: The Time-Triggered Architecture. Proceedings of the IEEE **91** (2003) 112–126
4. Heiner, G., Thurner, T.: Time-Triggered Architecture for Safety-Related Distributed Real-Time Systems in Transportation Systems. In: Proc. 28th Intl. Symp. on Fault-Tolerant Computing, IEEE Computer Society (1998)
5. Bauer, G., Kopetz, H., Steiner, W.: Byzantine Fault Containment in TTP/C. In: Proc. Intl. Workshop on Real-Time LANs in the Internet Age. (2002) 13–16
6. Bauer, G., Kopetz, H., Steiner, W.: The Central Guardian Approach to Enforce Fault Isolation in the Time-Triggered Architecture. In: Proc. 6th Intl. Symp. on Autonomous Decentralized Systems. (2003) 37–44
7. Pfeifer, H., Schwier, D., von Henke, F.: Formal Verification for Time-Triggered Clock Synchronization. In: Proc. of Dependable Computing for Critical Applications 7, IEEE Computer Society (1999) 207–226
8. Katz, S., Lincoln, P., Rushby, J.: Low-Overhead Time-Triggered Group Membership. In: Proc. 11th Intl. Workshop on Distributed Algorithms. Volume 1320 of LNCS, Springer-Verlag (1997) 155–169
9. Pfeifer, H.: Formal Verification of the TTP Group Membership Algorithm. In: Proc. of FORTE XIII / PSTV XX, Kluwer Academic Publishers (2000) 3–18
10. Bouajjani, A., Merceron, A.: Parametric Verification of a Group Membership Algorithm. In: Proc. 7th Intl. Symp. on Formal Techniques in Real-Time and Fault-Tolerant Systems. Volume 2469 of LNCS, Springer-Verlag (2002) 311–330
11. Merceron, A., Müllerburg, M., Pinna, G.: Verifying a Time-Triggered Protocol in a Multi-Language Environment. In: Proc. 17th Intl. Conf. on Computer Safety, Security and Reliability. Volume 1516 of LNCS, Springer-Verlag (1998) 185–195
12. Steiner, W., Rushby, J., Sorea, M., Pfeifer, H.: Model Checking a Fault-Tolerant Startup Algorithm: From Design Exploration To Exhaustive Fault Simulation. In: Proc. Conf. on Dependable Systems and Networks, IEEE Computer Society (2004)
13. Rushby, J.: An Overview of Formal Verification for the Time-Triggered Architecture. In: Proc. 7th Intl. Symp. on Formal Techniques in Real-Time and Fault-Tolerant Systems. Volume 2469 of LNCS, Springer-Verlag (2002) 83–105
14. Rushby, J.: Systematic Formal Verification for Fault-Tolerant Time-Triggered Algorithms. IEEE Trans. on Software Engineering **25** (1999) 651–660
15. Owre, S., Rushby, J., Shankar, N., Stringer-Calvert, D.: PVS: An Experience Report. In: Applied Formal Methods. Volume 1641 of LNCS, Springer-Verlag (1998) 338–345
16. Bauer, G., Kopetz, H., Puschner, P.: Assumption Coverage under Different Failure Modes in the Time-Triggered Architecture. In: Proc. 8th IEEE Intl. Conf. on Emerging Technologies and Factory Automation. (2001) 333–341
17. TTTech: Time-Triggered Protocol TTP/C High-Level Specification Document. Available at <http://www.tttech.com/technology/specification.html> (2003)
18. Kopetz, H.: The Time-Triggered (TT) Model of Computation. In: Proc. 19th IEEE Real-Time Systems Symposium. (1998) 168–177
19. Pfeifer, H., von Henke, F.: Modular Formal Analysis of the Central Guardian in the Time-Triggered Architecture. Technical report, Fakultät für Informatik, Universität Ulm, Germany (2004)