

FORMAL MODELLING AND ANALYSIS OF FAULT TOLERANCE PROPERTIES IN THE TIME-TRIGGERED ARCHITECTURE

Holger Pfeifer, Friedrich W. von Henke

Universität Ulm, Fakultät für Informatik, Abt. Künstliche Intelligenz

Address: James-Franck-Ring, D-89081 Ulm, Germany

Phone: +49 731 50-24121, Fax: +49 731 50-24119, e-mail: {pfeifer|vhenke}@informatik.uni-ulm.de

Abstract: The Time-Triggered Architecture is a distributed computer architecture for the implementation of highly dependable real-time systems specifically targeting embedded applications, such as digital control systems in the automotive and avionics domain. We have formally modelled and analysed various aspects of the underlying communication protocol TTP/C and its fault tolerance properties. This paper provides an overview of these analyses from a broader perspective and describes the relationships between the individual items. The algorithms implementing the basic protocol services of TTP/C are heavily intertwined and pose challenging problems for formal analysis. This is true not only with regard to the construction of formal proofs, but also for the development of the formal models themselves. We argue that an adequate structuring of the models and proofs along different levels of abstraction is necessary to enable the formal modelling of the central protocol algorithms, make their analysis feasible, and resolve the mutual dependencies among the services.

Keywords: Formal Methods, Deductive Analysis, Safety-Critical Systems, Time-Triggered Architecture

1. INTRODUCTION

The Time-Triggered Architecture (TTA) (Kopetz, 1995, 1998a; Kopetz and Bauer, 2003) is a distributed computer architecture for the implementation of highly dependable real-time systems. Envisioned applications of TTA are embedded digital control systems in the automotive or avionics domain (Heiner and Thurner, 1998). In such safety-critical applications, the dependability of the system becomes a prime issue since failure may have catastrophic consequences. Embedded systems of this kind typically involve distributed, real-time computations and requirements of fault tolerance; this makes their analysis and the verification that they behave as required inherently difficult. To obtain the kind of reliability required for highly safety-critical applications mere testing alone is usually insufficient. In this regard, formal analysis can provide an additional source of confidence; in fact, it has been argued (Butler and Finelli, 1993; Rushby, 1995) that the necessary level of confidence in correct behaviour cannot be achieved without a careful formal analysis of the mechanisms and algorithms involved.

The Time-Triggered Protocol TTP/C consti-

tutes 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. Several aspects of TTP/C and related protocols have been formally modelled and analysed, including clock synchronisation (Pfeifer et al., 1999), group membership (Katz et al., 1997; Pfeifer, 2000; Bouajjani and Merceron, 2002), the startup procedure (Merceron et al., 1998; Steiner et al., 2004), and fault tolerance properties of the communication system featuring central guardians (Pfeifer and von Henke, 2004).

In this paper we provide a broader overview of the results of our work of formally modelling and analysing the fault tolerance properties of TTP/C we have carried out in recent years. It is not our intention to provide in-depth descriptions of the formal details of these analyses, but rather focus on the modelling approaches taken to connect the correctness arguments constructed for the individual items. In this regard, we particularly emphasise the importance of abstraction and genericity in the design and structuring of the

formal models and proofs. Abstraction is a fundamental prerequisite for the feasibility of formal analysis, as it allows for both structuring the models by the provision of abstract interfaces and hiding details unnecessary or irrelevant for the formal analysis and the demonstration of critical properties. An adequate structure of formal models allows one to concentrate on particular aspects of a TTA system, such as the behaviour of the communication network, or the properties of protocol algorithms like group membership. Different items can then be analysed separately from each other, assuming certain properties of other models where necessary.

For instance, the two central algorithms of TTP/C, clock synchronisation and group membership mutually depend on each other. The algorithms have previously been analysed in isolation (Pfeifer et al., 1999; Pfeifer, 2000) and the question arises whether the combination of the arguments for the individual analyses is sound. By carrying out these analyses at appropriate levels of abstraction, where the formal models provide abstract interfaces to exchange information or other requirements between the analyses, the cyclic dependency of the individual proofs can be resolved to obtain a single, integrated proof of correctness for both algorithms.

Moreover, abstract interfaces of the models also provide a certain degree of genericity, which enables one to express the commonalities of a wide range of designs in a coherent way. For example, there are two common physical interconnection topologies for TTA, the bus and the star. In order to protect the shared communication network against faulty behaviour of nodes, autonomous units, so-called *guardians*, supervise the nodes' 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. For a given analysis of some protocol aspect of TTP/C one could ask how sensitive the models are to changes in the topology of the communication network. Ideally, one would like to develop models that allow proofs to be established for either variant of the guardians. Using generic models, one can capture the essence of the functionality of a guardian and establish properties independently from an actual implementation. Subsequently, the generic guardian models can be refined both to a central guardian-based view and to a model for lo-

cal bus guardians, which inherit the properties of the generic model. Thus, properties established for one type of network topology can easily be carried over to another.

The paper is organised as follows. In the next section, we give a brief overview of the main aspects of the Time-Triggered Architecture. The subsequent section describes the structure of a formal modelling framework we have developed to analyse the various aspects and components of TTA. Then we elaborate how to employ the framework to formally analyse the fault tolerance properties of the most critical algorithms of TTA in Sect. 4. Finally, we conclude in Sect. 5.

2. BRIEF OVERVIEW OF TTA

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 (Kopetz and Bauer, 2003; Bauer et al., 2001; TTTech, 2003). In a Time-Triggered Architecture system a set of *nodes* are interconnected by a real-time communication system. Nodes consist of the host computer running the application software, and the communication controller, which accomplishes the time-triggered communication between nodes. The nodes communicate via replicated shared media. 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 (Kopetz, 1998b). 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, called the *message descriptor list (MEDL)*, stored at each communication controller, contains the send and receive instants. It thus provides common knowledge about message timing to all nodes.

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 (Bauer et al., 2002). 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. 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.

It is also due to the industrial demand to minimise costs that the various services provided by the Time-Triggered Protocol are tightly integrated. For instance, there are no distinct phases for exchanging messages for clock synchronisation or message acknowledgement; instead, those services are realized as side effects of ordinary, scheduled message exchanges. This is accomplished by utilising the global information about the communication structure that is stored in the MEDL. As the intended system behaviour is thus known to all processors, important information can be derived indirectly from the messages. For example, the MEDL contains information about the duration of a given slot or the identity of the sending processor. This knowledge can be exploited by the clock synchronisation service to gain information about the processors' local clock readings. As the delivery time of a message is known beforehand to all processors, the difference between the expected and the actually observed arrival time can be used as an estimate of the deviation of the sending processor's clock from that of the receiver. This information is used to periodically calculate correction terms to adjust the processor's local clock and thus keep it in synchrony with the other clocks of the cluster.

The group membership service uses the messages of processors as life-signs. For example,

a receiving processor can determine that a message is missing immediately after the anticipated arrival time has passed. Similarly, the successful reception of a message is a sufficient condition for the sending processor to be considered active.

As the mentioned protocol services rely on consistent message transport, there exist mutual dependencies between these services. For example, it is obvious that the clocks of the processors must be synchronised in order to allow for correct message transport. Consequently, the clock synchronisation service is a prerequisite for group membership. On the other hand, for clock synchronisation processors need to analyse the reception times of messages that are correctly received. Since processors can only communicate with members of their own group, clock synchronisation requires the availability of group membership. This extreme integration of the various protocol services makes formal analysis even more necessary, but also more difficult.

3. FORMAL MODELS FOR TTA

The central part of our formal analysis work for TTA is represented by a formal framework for modelling and analysing the Time-Triggered Architecture and the services provided by the Time-Triggered Protocol. One of the goals in the design of the framework was to be able to integrate previous work on the formal analysis of fundamental algorithms of the Time-Triggered Protocol, such as the group membership algorithm (Pfeifer, 2000) or the clock synchronisation algorithm (Pfeifer et al., 1999), into the framework without major modifications. In particular, we have aimed at providing suitable interfaces to allow for a seamless integration. To this end, we developed a series of generic and abstract formal models that allow for the various aspects of TTA be analysed individually without compromising the need to integrate the individual analyses.

A natural way to structure the framework is to create several formal models according to the architectural design of a TTA system. Consequently, our the formal modelling framework for TTA consists of two major classes of models: first, generic models that describe the general behaviour of the *nodes* in a TTA system, and second, generic models that focus on the fault-tolerant properties of the *communication network* of a TTA system. Formal models for the analysis of the fault-tolerant properties of the particular protocol algorithms of

TTP/C can then be based on these two sets of generic models.

In the remainder of this section we explain the main aspects of the generic models in our framework.

3.1 Generic models for node behaviour

The models that describe the general behaviour of the nodes in a TTA system are intended to form the common ground for the analysis of the most critical properties of TTP/C. The models abstract from various details in the execution of the TTP/C protocol. In essence, the behaviour of a node is reduced to describing the actions it takes in a single slot of the TDMA schedule, that is, the sending and receiving of messages, and the update of its internal state according to the message received. Therefore, there are three entities that describe the behaviour of a TTA node:

- its current internal state,
- the message it sends in a slot, if any,
- the message it receives in a slot.

The values of these entities at a given time are determined on the one hand by the protocol algorithms the node is executing, and on the other hand by the environment. While a node can control its own internal state and the messages it is sending, the environment is responsible for the messages a node can receive. Moreover, the environment can cause a node to become faulty. Consequently, we need two objects to model what it means for a node to execute (an algorithm of) the Time-Triggered Protocol:

- a state-transition function `trans`,
- a message-generation function `msg`.

Since it is our goal to develop a generic model that can be used to describe different algorithms of the Time-Triggered Protocol, these two items become parameters of our model. In order to formally model a given protocol algorithm such as, for instance, group membership, one has to provide concrete values to these two parameters.

With regard to the concrete modelling of the actions a node takes, that is, its state transitions and the generation of messages, one has to decide at which granularity these actions should be modelled. A more abstract description is likely going to facilitate formal analysis, while a more detailed model would allow for the analysis of different kinds of properties that might not be expressible within the abstract model. As these models

are going to be used to analyse the protocol algorithms of TTP/C, one has to answer the question of what are the appropriate levels of abstraction for these algorithms. A natural level of abstraction is given by the so-called *untimed synchronous system model* (Lynch, 1996). On this level, one assumes that all nodes execute their state transitions synchronously in a lock-step fashion. This corresponds to nodes that employ perfectly synchronised clocks. This view is taken for most of the informal protocol specification document (TT-Tech, 2003), since it allows a convenient description of the protocol algorithms in terms of the slots in the TDMA schedule. At this level of abstraction, the state transitions of nodes can be modelled by a simple recursive function on the slot numbers. Figure 1 shows the relevant parts of a formal module (in the specification language of the PVS system (Owre et al., 1998)) that describes the state of a node p in a given slot from the untimed, synchronous system model view. Note that the state transition function `trans` and the message generation function `msg` are parameters of the PVS theory. Thus, the theory captures the essence of the synchronous behaviour of a systems and can be applied to model a given synchronous algorithm, such as the group membership algorithm, by instantiating the parameters with concrete values.

```
synchronous_system
[ (...)
, initstate : [Proc -> InitState]
, msg       : [Proc, State -> Message]
, trans     : [Proc, State, Message
              -> State]
] : THEORY
BEGIN
(...)
state(slot)(p) =
  IF slot = 0
  THEN initstate(p)
  ELSE LET s = state(slot-1)(p),
        m = rcvd(slot-1)(p)
        IN trans(p,s,m)
ENDIF
END synchronous_system
```

Fig. 1: PVS theory for untimed synchronous system model

However, not all properties are expressible at this level. In particular, this refers to everything that specifically involves time, such as a clock synchronisation service, since the timing aspect is abstracted away in the synchronous system model. Hence, one also needs a more detailed description to model the timing behaviour of nodes. In comparison to the model for the untimed, synchronous

level the timed model additionally introduces entities that formalise the local clocks of the nodes, and the state updates and generation of messages are now described in a more fine grained way on the level of the ticks of these clocks, rather than in terms of slots. On the other hand, the functions describing the state transitions the message generation are again only parameters of the model in order to provide a generic interface for various concrete algorithm instances.

Obviously, if different aspects of the communication protocol TTP/C are modelled and analysed at different levels of abstraction, using different formal models, the question arises how the different model layers relate to each other. In fact, the untimed synchronous system model is a sound abstraction of the time-triggered system model provided that the clocks of the processors are synchronised within a small bound. This relationship can be captured generically (Rushby, 1997, 1999), i. e. it is independent of the particular algorithm that is modelled at the synchronous level. Rushby's proof allows properties, which have been established on the more abstract level of synchronous systems, to be transformed, or refined, to the concrete time-triggered system level. Moreover, it should be noted that the validity of this generic refinement proof depends on the presence of a clock synchronisation service, because only if the clocks of the node are synchronised tightly enough can the internal states of the nodes as expressed on the abstract level be faithfully translated onto the timed system level. Depending on the precision of the clock synchronisation, one obtains bounds on the instant at which a node must begin its message broadcast, and on the interval during which a node must wait for messages to arrive. Qualitative constraints relating the various timing entities involved can be extracted from the formal refinement proof.

Finally, the formalisation of the actions that can be taken by the environment is what constitutes the fault model. Here we have to describe the assumptions about what kind of faults can possibly occur to a node, and how the messages sent by a node relate with the messages a node can receive at the other end. This latter point in particular is dealt with separately in the formal models about the communication network, which we are going to describe in the following subsection.

3.2 Models for the communication network

The modelling of the communication network complements the models that deal with the inter-

nal behaviour of the nodes. They describe the actions taken at a node to send and receive messages, and the functionality of the communication channels and their guardians.

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 will 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

$$assumptions_i \Rightarrow properties_i \quad (1)$$

$$\begin{array}{l}
\text{assumptions}_0 \Rightarrow \text{properties}_0 \\
= \text{ or } \Rightarrow \\
\text{assumptions}_1 \Rightarrow \text{properties}_1 \\
= \text{ or } \Rightarrow \\
\text{assumptions}_2 \Rightarrow \dots \Rightarrow \text{properties}_f
\end{array}$$

Fig. 2: Reasoning structure of models for the communication network

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 kind depicted in Figure 2. 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 (TTTech, 2003). 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:

$$\begin{array}{l}
\text{general_channel_properties} \Rightarrow \\
\text{Validity} \wedge \text{Agreement} \wedge \text{Authenticity}
\end{array} \quad (2)$$

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 $\text{general_channel_properties}$. 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:

$$\begin{array}{l}
\text{strong_fault_hypothesis} \Rightarrow \\
\text{general_channel_properties}
\end{array} \quad (3)$$

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:

$$\begin{array}{l}
\text{weaker_fault_hyp.} \wedge \text{generic_guardian} \Rightarrow \\
\text{general_channel_properties}
\end{array} \quad (4)$$

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:

$$\begin{array}{l}
\text{local_bus_guardian} \Rightarrow \text{generic_guardian} \\
\text{central_guardian} \Rightarrow \text{generic_guardian}
\end{array} \quad (5)$$

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.

Moreover, the genericity of the guardian models serves as an interface to the analyses for particular algorithms of the TTP/C protocol. In these analyses one can abstract from the concrete realisation of the guardians and the topology of the communication network, and instead use the properties established for the generic guardian. Thus, correctness arguments for a given protocol algorithm can be instantiated for either guardian variant.

A detailed presentation of the developments described in this subsection is given in (Pfeifer and von Henke, 2004).

4. FORMAL ANALYSIS OF TTA

In this section we describe how the generic modelling framework just described can be employed to analyse various aspects of the protocol TTP/C in a way that allows to connect the correctness arguments developed for the individual protocol algorithms.

4.1 Analysing central algorithms within the general framework

The formal modelling framework for TTA provides suitable interfaces to allow for a seamless integration of analysis of fundamental algorithms of the Time-Triggered Protocol, such as the group membership algorithm (Pfeifer, 2000) or the clock synchronisation algorithm (Pfeifer et al., 1999), which have been performed earlier. For instance, the group membership algorithm is modelled at the untimed, synchronous system level by instantiating the formal parameter `trans` of the PVS theory `synchronous_system` with the definition of the state transition function for the TTP/C membership algorithm. On the other hand, clock synchronisation was analysed at the more concrete timed system level.

The mentioned analyses of critical protocol algorithms have been carried out using a num-

ber of abstractions. For instance, the description of the sending and receiving of messages by the nodes has been kept abstract. Particularly, entities such as a bus guardian have not been modelled explicitly in these models. However, the definitions and proved facts for the basic formal models of the TTA architecture can easily be re-used in the analyses of the protocol algorithms by providing concrete values for the parameters of the basic models. Thus, the analyses of particular protocol aspects can be coupled to the developments concerning the overall architecture.

Moreover, previous analyses have been carried out relative to a strict fault model. By integrating them with the models of the communication network, we can make use of the properties of the central guardians. In this way, the basis for the examination of the protocol algorithms is established, as the adoption of a strict fault model for the communication in the analysis of TTP/C is justified by the reasoning we have carried out on the guardian properties. Altogether, we argue that the correctness properties established in previous analyses of critical protocol services can be carried over to the new star network architecture of the TTA including central guardians.

In addition to these considerations, there is another aspect in the analyses of critical protocol services: the algorithms for clock synchronisation and group membership have been analysed separately from each other. While these analyses are valuable in their own right, it is the integrated protocol that is implemented and run. Hence, the question arises whether the results of the isolated analyses still hold for the integration. We therefore investigated how to provide a formal proof that the individual analyses can be combined in a sound way.

4.2 Integration of group membership with clock synchronisation

A closer look at the formal models reveals that clock synchronisation and group membership depend on each other. On the one hand, there is a hierarchical dependency in our treatment: the verification of the group membership algorithm is carried out at the level of a untimed synchronous system, where one assumes that all processors are perfectly synchronised and run in lock-step. This assumption abstracts from a synchronisation mechanism and therefore the proofs for group membership depend on the correctness of the

clock synchronisation service. On the other hand, the clock synchronisation algorithm of TTP/C differs from many standard algorithms in several ways. Most importantly, the way a processor obtains information about the clock readings of other processors is totally integrated into the exchange of data messages. Since a processor only accepts messages from processors that it considers to belong to the same membership set, clock synchronisation actually depends also on group membership. These apparently circular dependencies need to be broken by combining the proofs so that the results previously obtained by analysing the individual services in isolation can also be established for the integrated services.

However, a simple combination of the two verification results fails because the levels of abstraction at which the services are modelled do not match. The clock synchronisation algorithm has been modelled at the more detailed time-triggered system level, whereas the group membership service has been dealt with at the more abstract level of the synchronous system model. As mentioned earlier, the untimed synchronous system model is a sound abstraction of the time-triggered system model provided that the clocks of the processors are synchronised within a small bound.

In the following we provide an outline of the approach to integrate group membership with clock synchronisation. The first step towards an integrated proof is to split up the analysis into a series of successive intervals, corresponding to the synchronisation intervals used in the analysis of the clock synchronisation algorithm. This is the result of the observation that there is a temporal dependency between group membership and clock synchronisation: in order for the synchronisation algorithm to adapt the processors' physical clocks correctly, and thus keeping them synchronised during the following period $i + 1$, it must rely on the membership service having been available during the current synchronisation period i . The goal is then to prove, by induction on the synchronisation interval i , that for all intervals both the clock synchronisation property and the group membership property hold:

$$\forall i : cs_prop(i) \wedge mem_prop(i) \quad (6)$$

The membership part of this theorem is trivially true, since the proof for the group membership property is actually independent of synchronisation intervals: the property holds for all slots and hence also for all intervals (Pfeifer, 2000). As

for the clock synchronisation part, things are a bit more involved. From an abstract point of view, the proof of the induction step of the clock synchronisation property proceeds according to the following formula: If the clock synchronisation property holds in the i th synchronisation interval and some requirements for synchronisation are met, then the clock synchronisation property also holds in the $i + 1$ st interval.

$$cs_req(i) \wedge cs_prop(i) \Rightarrow cs_prop(i + 1) \quad (7)$$

The expression $cs_req(i)$ captures the requirements that are necessary for the synchronisation algorithm to work properly; in particular, it is required all non-faulty nodes must agree on whether or not the message from the broadcaster of a slot was received correctly, and that a message sent by a non-faulty broadcaster is received correctly by all non-faulty receivers. Note how these requirements relate to the correctness properties of TTP/C communication *Agreement* and *Validity*, respectively, described above. Moreover, the requirements directly rely on the availability of the group membership service. Hence, one needs to show that the membership property ensures that the requirements for clock synchronisation can be fulfilled.

$$\forall i : mem_prop(i) \wedge cs_prop(i) \Rightarrow cs_req(i) \quad (8)$$

There is an additional conjunct, $cs_prop(i)$, in the hypothesis of the formula above. This is due to the fact that the membership property and the requirements for clock synchronisation are expressed at different levels of abstraction. As explained earlier, group membership is dealt with at the untimed synchronous system level, where the notion of time is abstracted away. In contrast to this, the clock synchronisation requirements are expressed in terms of the ground model of TTA, which explicitly deals with timing issues. It is therefore necessary to “map” the proof of group membership down to the time-triggered system level. This transformation, however, requires that synchronised clocks are present (Rushby, 1997). This step is the crucial part of the integration proof, as it establishes the relationship between clock synchronisation and group membership.

Furthermore, the clocks need to be initially synchronised, that is, the clock synchronisation property must hold during the initial synchronisation interval. We assume that this is the case.

$$cs_prop(0) \quad (9)$$

The proof of the desired property (6) of integrated synchronisation and membership services can be accomplished by an easy induction on the number of synchronisation intervals i using the formulas above. For the inductive step in this proof it is necessary to show that the introduced abstractions can be resolved. On the one hand, this means that the proof of group membership must be transferred from the higher level of abstraction to the level in which the clock synchronisation algorithm is modelled. On the other hand, it must be shown that group membership indeed provides an adequate interpretation of the abstract entities in the synchronisation model such that the presupposed hypotheses for clock synchronisation are satisfied.

The outlined approach deals with the dependencies between group membership and clock synchronisation in two ways of abstraction. First, the dependency of membership on clock synchronisation is resolved by describing the group membership algorithm at the abstract level of untimed synchronous systems. Second, the clock synchronisation algorithm is parameterised by an abstract assumption that captures the essence of what is required from the membership service.

5. CONCLUSIONS

We have described the development of a formal framework to model and analyse critical aspects of the Time-Triggered Architecture in order to provide mathematically substantiated arguments that architecture and algorithms provide certain services and satisfy certain critical properties, in support of the claims that TTA meets the high reliability requirements of safety-critical applications in the automotive and aerospace domain.

The central part of the framework consists of a series of hierarchically organised formal models. Each of the models covers a particular aspect of the various elements of the architecture at an appropriate level of abstraction. The models are divided into two classes: one that describes the behaviour of TTA nodes executing a given protocol algorithm of TTP/C such as group membership, and another that covers the communication among the TTA nodes through the channels of the interconnection network. The models are expressed in both an abstract and generic way. Genericity allows for expressing the commonalities of a wide

range of designs in a coherent way. The models that describe the behaviour of the communication channels, for example, provide a generic treatment of the guardians that allows the model to be refined to either a central guardian-based view, or to a model for local bus guardians.

The analysis of the properties of the communication network of TTA has supported the claim that the functionality of the guardians ensure that arbitrary node failures are converted into fault modes that the TTP/C protocol algorithms can tolerate. Thus, the strong fault hypothesis of TTP/C can thus 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. Furthermore, previous analysis of TTP/C protocol algorithms on group membership and clock synchronisation remain valid also for the star network architecture with central guardians.

We have also described how two flavours of abstraction techniques can provide the required separation of protocol services of TTP/C without compromising the possibility of adequately integrating the different analyses. First, we deal with abstraction of the formal model itself. Here, the formalisation of group membership is lifted to the higher, more abstract level of untimed synchronous systems that allows to express the important aspects of the service without referring to details that are irrelevant to the essential service properties. Second, we consider the abstraction of the proof of a particular correctness property. Here, the dependency of clock synchronisation on group membership is dealt with by parameterising the correctness result relative to abstract assumptions that capture the essence of what is required from the membership service. At this point it is crucial, however, that these assumptions do not require us to include the complete description of these other protocol parts into the formal model of clock synchronisation. Consequently, the assumptions about the environment must be expressed in an abstract way. Of course, the validity of these assumptions must be demonstrated in a second step, but this step is taken only at the time the various separated models are integrated. In this step, the abstractly expressed assumptions

are given concrete interpretations, and proofs must then be given to show that the assumed interface properties do hold indeed for the concrete instantiations.

The benefit of this approach is that there is no need to perform a double or parallel induction to accomplish the proof of the integration theorem. In fact, the two properties can be analysed independently from each other and the interdependencies of the two services can be clearly isolated and resolved separately.

Future work on formally analysing TTA could be aimed at broadening the focus of the analyses. For instance, one needs to examine adequate approaches to close the gap between the rather abstract level of formal models and the concrete realizations of the protocol algorithms in the silicon implementation of TTP/C. It would be worthwhile to investigate how the formal models and the methods used can be extended such that a complete chain of correctness arguments can be supported. This could also contribute towards formal analysis being eventually used as a basis in the certification of the Time-Triggered Protocol.

Moreover, techniques are required to be able to reason about properties of TTA at the system level. To this end, we need to find ways of specifying and analysing a system built on top of TTA, based on the properties that are provided by TTA at its interfaces. Ultimately, this would allow for deducing properties of a TTA application and systems exploiting time-triggered communication from the established properties of the TTP/C protocol.

REFERENCES

- G. Bauer, H. Kopetz, and P. Puschner, (2001). Assumption Coverage under Different Failure Modes in the Time-Triggered Architecture. In *Proc. 8th IEEE Intl. Conf. on Emerging Technologies and Factory Automation*, pages 333–341, Oct. 2001.
- G. Bauer, H. Kopetz, and W. Steiner, (2002). Byzantine Fault Containment in TTP/C. In *Proc. Intl. Workshop on Real-Time LANs in the Internet Age*, pages 13–16, June 2002.
- A. Bouajjani and A. Merceron, (2002). 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*, pages 311–330. Springer-Verlag, 2002.
- R. Butler and G. Finelli, (1993). The Infeasibility of Quantifying the Reliability of Life-Critical Real-Time Software. *IEEE Trans. on Software Engineering*, 19(1):3–12, Jan. 1993.
- G. Heiner and T. Thurner, (1998). 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.
- S. Katz, P. Lincoln, and J. Rushby, (1997). Low-Overhead Time-Triggered Group Membership. In *Proc. 11th Intl. Workshop on Distributed Algorithms*, volume 1320 of *LNCS*, pages 155–169. Springer-Verlag, 1997.
- H. Kopetz, (1995). The Time-Triggered Approach to Real-Time System Design. In *Predictably Dependable Computing Systems*. Springer-Verlag, 1995.
- H. Kopetz, (1998). The Time-Triggered Architecture. In *Proc. 1st Intl. Symp. on Object-Oriented Real-Time Distributed Computing*, pages 22–31, Apr. 1998a.
- H. Kopetz, (1998). The Time-Triggered (TT) Model of Computation. In *Proc. 19th IEEE Real-Time Systems Symposium*, pages 168–177, 1998b.
- H. Kopetz and G. Bauer, (2003). The Time-Triggered Architecture. *Proceedings of the IEEE*, 91(1):112–126, Jan. 2003.
- N. Lynch, (1996). *Distributed Algorithms*. Morgan Kaufmann Series in Data Management Systems. Morgan Kaufmann Publishers Inc., San Francisco, 1996.
- A. Merceron, M. Müllerburg, and G. Pinna, (1998). 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*, pages 185–195. Springer-Verlag, 1998.

- S. Owre, J. Rushby, N. Shankar, and D. Stringer-Calvert, (1998). PVS: An Experience Report. In *Applied Formal Methods*, volume 1641 of *LNCs*, pages 338–345. Springer-Verlag, Oct. 1998.
- H. Pfeifer, (2000). Formal Verification of the TTP Group Membership Algorithm. In *Proc. of FORTE XIII / PSTV XX*, pages 3–18. Kluwer Academic Publishers, Oct. 2000.
- H. Pfeifer, D. Schwier, and F. von Henke, (1999). Formal Verification for Time-Triggered Clock Synchronization. In *Proc. of Dependable Computing for Critical Applications 7*, pages 207–226. IEEE Computer Society, Jan. 1999.
- H. Pfeifer and F. W. von Henke, (2004). Modular Formal Analysis of the Central Guardian in the Time-Triggered Architecture. In M. Heisel, P. Liggesmeyer, and S. Wittmann, editors, *Proc. of the International Conference on Computer Safety, Reliability, and Security (SAFE-COMP)*, volume 3219 of *Lecture Notes in Computer Science*, pages 240–253, Potsdam, Germany, Sept. 2004. Springer-Verlag.
- J. Rushby, (1995). Formal Methods and their Role in the Certification of Critical Systems. In R. Shaw, editor, *Safety and Reliability of Software Based Systems (Twelfth Annual CSR Workshop)*. Springer-Verlag, 1995.
- J. Rushby, (1997). Systematic Formal Verification for Fault-Tolerant Time-Triggered Algorithms. In M. D. Cin, C. Meadows, and W. Sanders, editors, *Dependable Computing for Critical Applications – 6*, pages 203–222. IEEE Computer Society, Mar. 1997.
- J. Rushby, (1999). Systematic Formal Verification for Fault-Tolerant Time-Triggered Algorithms. *IEEE Trans. on Software Engineering*, 25(5): 651–660, Sept. 1999.
- W. Steiner, J. Rushby, M. Sorea, and H. Pfeifer, (2004). 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, June 2004.
- TTTech, (2003). Time-Triggered Protocol TTP/C High-Level Specification Document. Available at <http://www.tttech.com/technology/specification.html>, 2003.