TEMPORAL BOUNDS FOR TTA : VALIDATION

K. Godary, I. Augé-Blum, A. Mignotte
CITI Lab. / INSA Lyon - 21 av. J. Capelle 69621 Villeurbanne, France
karen.godary@insa-lyon.fr, isabelle.auge-blum@insa-lyon.fr, anne.mignotte@insa-lyon.fr

Abstract: In the context of real-time fault-tolerant architecture, as TTA (Time-Triggered Architecture), the temporal validation of the system behavior is very important. Indeed, the fault-tolerant mechanism execution must respects several temporal constraints. To validate the mechanism behaviors, and to give their maximum execution time (temporal bound), we propose here a temporal validation methodology for TTA. This methodology uses the UPPAAL tool, based on the timed automata and the model-checking analysis. This methodology allows us to extract the temporal bounds of the TTA services.

Keywords: Time-Triggered Architecture, Embedded real-time networks, Fault tolerance, Temporal validation, UPPAAL modeling.

1. INTRODUCTION

Motivation. TTA (Time-Triggered Architecture) (Kopetz, 1998, TTP, 2002) is a real-time network architecture for embedded systems. It is mainly used for automotive embedded applications. This architecture is based on two primary concepts : time triggered protocol and fault tolerance. These concepts are performed by a number of services defined in TTA specifications. Furthermore, the services are defined by using basic algorithms such as : synchronization, membership fault detection, or reintegration.

An application using TTA will require temporal guarantees on the temporal behavior of the architecture. The execution time of the algorithms and services has an influence on the behavior of the application level. Therefore, they have to be validated within a specific fault environment. For a specific service, its maximum execution time (its temporal bound), must never be superior to a fixed deadline. In this article we are going to define these bounds and describe the validation process to obtain them.

State of the art. TTA specifications definition (TTP, 2002) allows extraction of basic temporal bounds. Obviously, these bounds are not validated by the specifications definition. While it is quite easy for elementary services to extract their bounds on simplified hypothesis, it is more difficult for regular TTA services. For instance, the clock precision is never taken into account, nor complex fault hypotheses.

On one hand, formal proofs (Pfeifer, 2003, Rushby, 2002, Bouajjani and Merceron, 2002) validate basic algorithms. For example in (Pfeifer, 2000), the membership algorithm is formally verified with the PVS theorem prover. This article proves the convergence of the algorithm. Yet, it does not define its temporal bound.

On the other hand, another approach is developed in (Bouajjani and Merceron, 2002), which gives a parametric proof of clique avoidance and membership correctness. These algorithms can be modeled by graphs, used to prove their temporal bounds. In the study, a bound is given for the detection of one fault, with simplified hypotheses. However this study is based on graph theory, a method too specific to be applied to all TTA algorithms.

All these formal methods are applicable on basic algorithms. Nevertheless they are difficult to be used on combination of services, or for the whole architecture. However the proven properties can be used as hypotheses in several other validation methodologies including the one presented in this article.

Other approaches produce deadline verification at a higher level. In (Caspi et al., 2003) for instance, the design of the whole system is based on a LUSTRE model, including temporal information and constraints, mapping on the hardware architecture, and the related scheduling. Temporal constraints are validated during scheduling. In this context, the TTA bus is modeled by a TDMA (Time-Division Multiple Access) round. Nevertheless, the deadline of a message transmission on the bus do not take into account the presence of faults nor fault tolerance. Such an approach would take fault tolerance into account if the temporal information are bounded durations (message transmission for instance) including faults. This is the aim of our approach.

Our approach is based on UPPAAL (Larsen et al., 1997). This tool has already been used for verification of other protocols (Jensen et al., 1996, Lönn and Pettersson, 1997, Lindahl et al., 1998). For instance, the study in Lönn and Pettersson, 1997 is similar to our method : the TDMA protocol is modeled in Uppaal, some properties are verified, and a parameterized deadline bound is extracted. But this study did not verify the protocol in the fault tolerance context, nor the whole architecture (it did not consider the applicative levels).

In this article, we propose an approach to find parameterized temporal bounds of TTA behavior, based on specific fault hypotheses. TTA is described in the next section. Our approach uses bounds already defined in TTA specifications or other publications. We extract new bounds by analysis of timed automata models. This approach is defined in the third section. Section 4 gives an illustrating example (the reintegration mechanism). Then section 5 presents experiments and results on our methodology.

2. TTA BASIC DEFINITION

Architecture

TTA (Kopetz, 1998) is composed of a cluster, i.e. a set of nodes connected through a communication network based on TTP/C (Time Triggered Protocol class C) (TTP, 2002). A node is composed of four independent levels : the application level; the real-time operating system (RTOS), compatible with the OSEKTime specifications (OST, 2001); the FTLayer (Bauer and Kopetz, 2000), compatible with the FTCom (Fault Tolerant Communication) specifications (FTC, 2001) (this level is in charge of redundancy, which is the base of fault-tolerance); and the TTP/C controller (TTP, 2002).

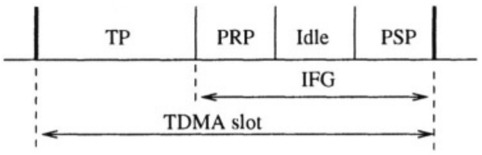

Figure 1. Temporal slot decomposition

Time Triggered Protocol

TTP/C is based on a static and predefined scheduling. It implements a broadcast communication using the TDMA strategy. Moreover, the bus access is guaranteed by autonomous subsystems, bus guardians, which prevent the nodes to emit at the wrong time.

Information about transmitted data and emission dates are stored in a static table : the MEDL (MEssage Descriptor List), included in each node. Then, clock synchronization is necessary. It is based on the comparison between the time of the real arrival of each message and the expected time given by the MEDL.

A slot is the smallest unit of the communication phase. The slots are grouped together in a TDMA round. Each slot is dedicated to the transmission of a node. The usual system phase is a cyclic execution of all TDMA rounds : the cluster cycle. A slot is composed of 4 phases (figure 1):

- *Transmission Phase (TP):* a TDMA slot begins with the transmission of a message on the bus.

- *Post Receive Phase (PRP):* allows the evaluation of received data and the execution of protocol services.

- *Pre-Send Phase (PSP)* : necessary to load the schedule information from the MEDL and to prepare the data transmission of the next slot.

- *Idle phase* : the duration of the PRP and PSP depends on the actions performed. So the Idle phase is used to stretch the slot to the duration fixed by the schedule designer.

The set of phases without transmission on the bus is called Inter-Frame Gap (IFG). It is the set of the three phases PRP, Idle and PSP.

3. METHODOLOGY

Our methodology is based on model validation. The chosen formalism is a specific timed automata implemented in the UPPAAL tool (Larsen et al., 1997). Its associated model-checker enables to check reachability properties by an exhaustive analysis of all the possible behaviors of the system. More discussions on different possible modeling formalisms are in Godary et al., 2004b. For more information on the UPPAAL tool see for instance Pettersson, 1999 or Bengtsson and Yi, 2004).

Methodology definition

Our methodology is composed of two phases. The aim of the first step (illustrated figure 2) is to obtain an abstracted and verified UPPAAL model of TTA. Then, the aim of the second one is to extract parameterized formulae for temporal bounds of mechanisms and services of TTP/C, such as mode change, reintegration (of a TTA node after a fault) or initialization.

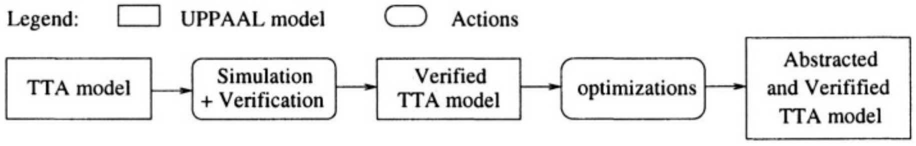

Figure 2. Methodology - step 1

Step 1: Abstracted and verified model (figure 2).

From the referring document TTP, 2002 a first model of TTA is built in UPPAAL. To guarantee the expected behavior, we check our model with formal verification of behavioral properties (for instance, one can check that two states are never reached at the same instant). More information on the initial model, and some results on this formal verification are given in the section 4

After those simulation and verification, we obtain a verified TTA model.

The problem of this model is combinatorial explosion. Therefore, some abstraction rules have been applied to simplify the analysis method implemented in UPPAAL. This abstracted TTA model keeps the same behavior, but it is easier to analyze. Basically, we have applied the following abstraction rules: reduction of the number of variables, automata, clocks, and interleaving. More details on these rules can be found in Godary et al., 2004a.

Finally, we have obtained an abstracted and verified TTA model.

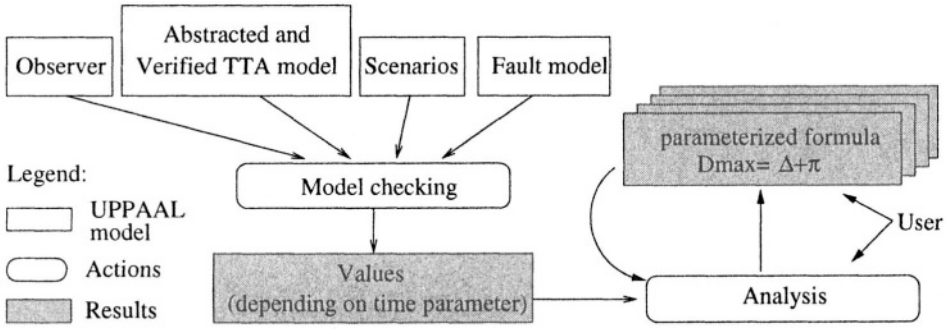
Observer
Abstracted and Verified TTA model
Scenarios
Fault model
Legend:
UPPAAL model
Actions
Results
Model checking
Values (depending on time parameter)
parameterized formula Dmax= Δ+π
User
Analysis

Figure 3. Methodology - step 2

Step 2 : Parameterized formula (figure 3).

Each basic mechanism is expressed in the UPPAAL model: either with an explicit modeling of its behavior, or with a representation of its characteristics. For instance, the synchronization algorithm is represented by the fact that all the local clocks of the nodes are synchronized within a fixed interval π. This abstraction is possible because this algorithm has already been validated (Pfeifer, 2003).

The basic mechanism model is completed with a Test automaton to verify with the model-checker that a deadline value D is never reached. This deadline value is a parameter of the system model. Then different values of D are verified by dichotomy, until the worst-case scenario is found. This worst-case behavior is the temporal bound of the studied mechanism.

Then, the same model is used with different values of parameters (such as clock drift or slot duration). We obtain a set of values depending on system parameters. We analyze them to deduce parameterized formulae for each bound.

Now, our methodology is going to be illustrated in more details with a complete analysis of the reintegration mechanism of TTA. This is a simple example which can help to understand its interest.

4. ILLUSTRATION ON A BOUND : REINTEGRATION

The reintegration service

The reintegration of a node can occur when one of the TTA basic algorithms detects a fault. Then the faulty node transits in passive state, and waits for reintegration. All the nodes however continue to perform TDMA principle, and the non-faulty nodes continue a normal execution.

The faulty node has a slightly different behavior. In the PSP, if the next slot is the one of the faulty node, the faulty node checks if it can transit to active state. The condition for that is that the controller has received at least MIC (Minimum Integration Count) correct frames since it has failed. Then the node reintegrate the cluster (transits to active state) and can send its frame.

The exact bound is validated between the beginning of the reintegration (i.e. the detection of the fault which causes the transition of the faulty node to passive state), and the end of the reintegration (i.e. its transition to active state).

Hypotheses

- **Time-triggered concept :** Some TTA mechanisms are supposed to be correct, as there have already been formally validated in the literature (Pfeifer, 2003, Rushby, 2002, Bouajjani and Merceron, 2002): membership, clique avoidance and clock synchronization algorithms. These algorithms provide the basis of the time-triggered strategy : a global view of the system for all the non faulty nodes. All the nodes are synchronized with a maximal clock drift π.

- **Architecture :** The model includes a cluster of 4 nodes with bus guardians in a one bus topology. No redundancy is considered.

- **Fault hypotheses :** In faulty models, we consider that only one fault at a time can occur during one TTA cluster cycle, which corresponds in our model to two TDMA rounds. This is realistic considering the usual fault hypotheses in automotive applications; but it rejects repetitive faults or combination of faults. In non faulty models, the only fault modeled is the one which initiates the reintegration process. It does not interfere with the reintegration itself, and thus not with the bound value.

Model

The model is composed of several automata : one MEDL, one central bus-guardian, one test automaton, one overall behavior manager and a set of automata for each node (the scheduler, the host_ftlayer level and the controller). They are not presented here because of the limited place.

A fault detection cause the starting of the deadline verification : the test automaton (see figure 4) transits to the *Reintegration* state on the *reintegration_being* singnal from the controller. Similary, the end of the reintegration is indicated to the test automaton with the *reintegration_end* signal. If this signal is not received before the end of the deadline (*clock==deadline*), the test automaton transits to the *Error* state. The deadline verification is then the verification of the Fault state reachability.

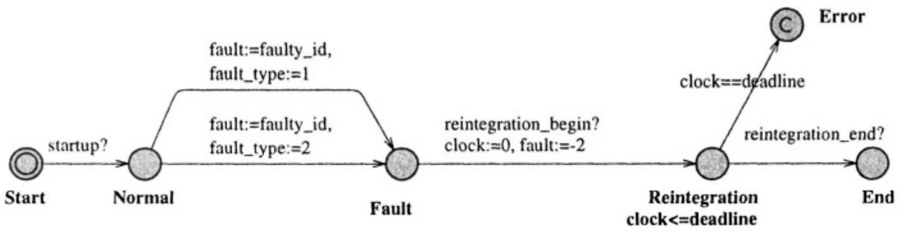

Figure 4. Test automaton

Validation

Verification results. Few results of the reintegration deadline verification are given as example in table 1. These results are the same for all faulty nodes. The CPU times and memory sizes have been measured for the verification of the property of the deadline, in the case the *Error* state is reached.

Table 1. **Reintegration deadline verification**

π	Δ_{slot}	Δ_{TP}	Δ_{PRP}	D_r	CPU time	Memory size
2	300	220	40	1542	2.74	19752KB
4	300	220	40	1544	2.74	19716KB
2	350	220	40	1842	2.85	26948KB
2	400	200	40	2162	2.41	16620KB

Analysis - worst scenario. The figure 5 illustrates the first line of table 1, for the faulty node 2. This worst scenario exists for a fault detected in the PRP of the slot which is two slots before the faulty node one. In this case, at the next node PSP, the reintegration condition (*integration* > *MIC* with *integration* the number of received correct frames) is not fulfilled, and then the node must waits for its next PSP, i.e. one round later.

Analysis - parameterized formulae. In the model, the node transits to passive state only at the end of the PRP. Thus, for the maximal bound, we add Δ_{PRP} at the bound verified on the model: $D_{Reint} = D_r + \Delta_{PRP}$.

The IFG phase duration is calculated by adding the other durations of the slot phases: $\Delta_{IFG} = \Delta_{PRP} + \Delta_{idle} + \Delta_{PSP}$

The parameterized formula is extracted by the worst case analysis of different scenarios, as shown in figure 5. Then we have: $D_r = \Delta_{round} + \Delta_{slot} + \Delta_{idle} + \Delta_{PSP} + \pi$ and $D_{Reint} = \Delta_{round} + \Delta_{slot} + \Delta_{IFG} + \pi$.

This formula is confirmed with all the results of table 1. For example, with the first line of this table : $D_r = 1542 = 5 * 300 + (300 - 220 - 40) + 2$.

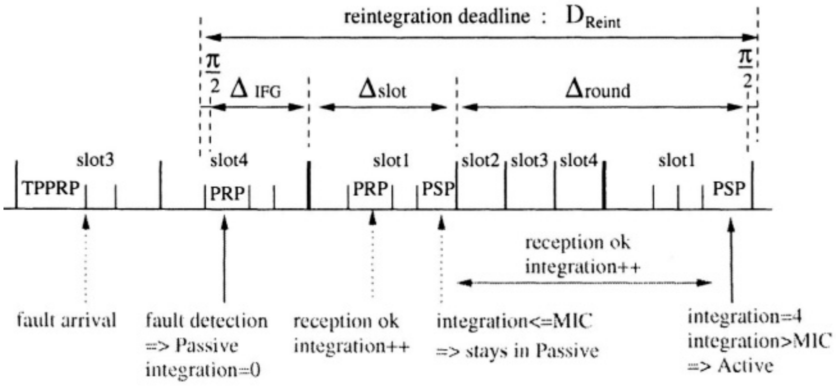

Figure 5. Worst reintegration scenario for node 2

5. EXPERIMENTS AND RESUTLS

Each of the UPPAAL models has been verified proving a number of properties on the UPPAAL model-checker. For instance, the cyclic infinite behavior is guarantee verifying the deadlock property. Another example is the verification of the reachability of the reintegration *Error* state.

This section gives bounds for the TTA services. On one hand, some elementary bounds for TTA algorithms are given by the TTA specification TTP, 2002 : Δ_{cycle} for the clock synchronization algorithm and Δ_{round} for the membership loss detection. Another one is given with formal verification in Bouajjani and Merceron, 2002 : $2 * \Delta_{round}$ for the clique avoidance algorithm.

In the other hand, our methodology was first performed on bounds validation for basic services. These bounds can be seen table 2. The fault hypotheses are the ones fixed in section 4. The communication blackout detection service is the only service which necessarily works under a longer time fault (more than one frame failed).

Table 2. **TTA services bounds**

Services	**Maximal bounds**
Without Faults	
Initialization	$MIC + (N+1) * \tau_{round}$
Reintegration	$\Delta_{round} + \Delta_{slot} + \Delta_{IFG} + \pi$
Reinitialization	$3 * \Delta_{round} + (MIC - 1) * \Delta_{slot} + \Delta_{IFG} + \pi$
Mode change	$\Delta_{cycle} + \Delta_{round} + \Delta_{PSP} + \pi$
With Faults	
Communication blackout check	$\Delta_{round} + \Delta_{slot}$
Reintegration	$\Delta_{round} + 2 * \Delta_{slot} + \Delta_{IFG} + \pi$
Reinitialization	$3 * \Delta_{round} + \Gamma_{int} * \Delta_{slot} + \Delta_{IFG} + \pi$

Another interesting study to do is the bound validation of the composition of several mechanisms. All TTA mechanisms are not independent and the temporal bound of the composition is not the sum of the different mechanism bounds. Specific analyses have to be done to extract these bounds.

This section shows as an example, the composition of both the detection mechanism of membership loss fault, and the reintegration service. The bound is not detailed here, and is done with simple fault hypotheses : the fault causing the membership loss is a symmetric one (it has the same effect in all the nodes), and there is no other faults during the rest of the service execution.

The temporal bound of the membership loss detection can be extract from the specification : Δ_{round}. The one of the reintegration service is given in table 2 : $\Delta_{round} + \Delta_{slot} + \Delta_{IFG} + \pi$. The sum of this two services bounds is then superior two rounds and one slot. But the real temporal bounds is equal to $2 * \Delta_{round}$. This difference is because the sum do not take into account that this is the same node which is concerned with the two mechanisms. And the worst case do not happened for both mechanisms for the same node. Moreover, this bound is still an overvaluation of the real combined bound. Indeed, the membership loss detection bound used is the one given in the specification, and we do not modeled and analyze the worst case in details.

6. CONCLUSION

We defined a methodology to determine temporal bounds for basic TTA services and algorithms. These bounds are parameterized values. They should be used at higher levels for temporal validation of application.

In the future we expect to complete our UPPAAL models. Indeed, some combination of services have to be bounded. Moreover, other fault hypotheses can be modeled. As bounds are calculated in reasonable CPU time, our methodology could be applied to more complex models.

REFERENCES

(2001). *OSEK/VDX Fault Tolerant Communication - Version 1.0.* http://www.osek-vdx.org/.

(2001). *OSEK/VDX Time-Triggered Operating System - Version 1.0.*

(2002). *Time-Triggered Protocol TTP/C, High-Level Specification Document - 1.0.0.* TTTech Computertechnik AG, Time-Triggered Technology, Vienna, Austria, http://www.ttech.com.

Bauer, G. and Kopetz, H. (2000). Transparent redundancy in the time-triggered architecture. In *Int. Conf. on Dependable Systems and Networks (DSN 2000),* New York, New York.

Bengtsson, J. and Yi, W. (2004). Timed automata : Semantics, algorithms and tools. Uppsala University.

Bouajjani, A. and Merceron, A. (2002). Parametric verification of a group membership algorithm. In *7th Int. Symp. on Formal Techniques in Real-Time and Fault Tolerant Systems (FTRTFT'02),* volume LNCS 2469, pages 311–330, Oldenburg (Germany).

Caspi, P., Curie, A., Maignan, A., Sofronis, C., Tripakis, S., and Niebert, P. (2003). From simulink to scade/lustre to tta: a layered approach for distributed embedded applications. In *Proc. of the 2003 ACM SIGPLAN conference on Language, compiler, and tool for embedded systems,* pages 153–162. ACM Press.

Godary, K., Augé_Blum, I., and Mignotte, A. (2004a). Evaluation of model abstractions for the temporal validation of tta with uppaal. Technical Report RR2004-2, Lab. CITI, INSA Lyon.

Godary, K., Augé_Blum, I., and Mignotte, A. (2004b). Sdl and timed petri nets versus uppaal for the validation of embedded architecture in automotive. Technical Report RR2004-1, Lab. CITI, INSA Lyon.

Jensen, H. E., Larsen, K. G., and Skou, A. (1996). Modelling and analysis of a collision avoidance protocol using spin and uppaal. In *In Proc. of the 2nd SPIN Workshop,* New Jersey, USA. Rutgers University.

Kopetz, H. (1998). The time-triggered architecture. In *IEEE Int. Symp. on Object-Oriented Real- Time Distributed Computing (ISORC'98),* volume LNCS 2469, Kyoto, Japan.

Larsen, K., Pettersson, P., and Yi, W. (1997). UPPAAL in a nutshell. *International Journal on Software Tools for Technology Transfer,* 1(1):134 –152.

Lindahl, M., Pettersson, P., and Yi, W. (1998). Formal Design and Analysis of a Gear-Box Controller. In *Proc. of the 4*th *Workshop on Tools and Algorithms for the Construction and Analysis of Systems,* LNCS. Springer-Verlag.

Lönn, H. and Pettersson, P. (1997). Formal Verification of a TDMA Protocol Startup Mechanism. In *Proc. of the Pacific Rim Int. Symp. on Fault-Tolerant Systems,* pages 235–242.

Pettersson, P. (1999). *Modelling and Verification of Real-Time Systems Using Timed Automata: Theory and Practice.* Phdthesis - technical report docs 99/101, Department of Computer Systems, Uppsala University.

Pfeifer, H. (2000). Formal verification of the ttp group membership algorithm. In *IFIP, Int. Conf. on Formal Description Techniques for Distributed Systems and Communication protocols and Protocol Specification, Testing and Verification, FORTE/PSTV 2000,* Pisa, Italy.

Pfeifer, H. (2003). *Formal Analysis of Fault-Tolerant Algorithms in the Time-Triggered Architecture.* PhD thesis, Universitat Ulm, Germany.

Rushby, J. (2002). An overview of formal verification for the time-triggered architecture. In Damm, W. and Olderog, E.-R., editors, *Formal Techniques in Real-Time and Fault-Tolerant Systems,* LNCS, Oldenburg, Germany. Springer-Verlag.