Ninth ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing

Specifying and Verifying Timing Properties of a Time-triggered Protocol for In-vehicle Communication

Bo Zhang
Department of Informatics
Technische Universitaet Muenchen, Germany
bo.zhang@in.tum.de

Abstract

In order to achieve predictability, time-triggered communication systems have been proposed for use in in-vehicle applications, in particular for safety-critical applications. Timing properties play a crucial role in a time-triggered system as activities in such a system are all triggered by the passage of time. In this paper, we propose techniques for specifying and verifying the timing properties of the FlexRay protocol, an emerging time-triggered communication protocol for automotive systems. Essential timing properties of the protocol are defined and proved. Mechanical support with the theorem prover Isabelle/HOL is also considered.

1. Introduction

Recently, there is a visible trend in automotive industry that more and more functionalities in cars are provided by software solutions instead of the traditional mechanical and/or hydraulic ones. Software solutions impose a demanding challenge on the automotive industry: an advanced communication control system combining multiple sensors, actuators and ECUs is indispensable. Moreover, many applications in automotive systems are inherently safety critical, for instance, the x-by-wire applications [2], where traditional mechanical and hydraulic control systems are replaced by electronic control systems, e.g. brake-by-wire and steer-by-wire. Requirements for the communication control system include high data rate, deterministic behaviors and a high degree of fault tolerance among others. The FlexRay communication protocol has been developed with the intention to fulfill these requirements. The FlexRay protocol can be regarded as a representative of the state-of-the-art safety critical real-time protocol for automotive systems.

The FlexRay protocol is time-triggered in the sense that activities in the system are triggered by the progress of time. For such systems, timing constraints become obvious and must be ensured. For instance, the sending and receiving of a frame must be within a fixed period; communication delay must be bounded.

It is generally agreed that, for safety-critical real-time systems, a careful formal analysis of the mechanisms and algorithms involved is beneficial. As an ongoing work on the formal analysis of the FlexRay communication protocol, we present our formal analysis of the timing properties of the protocol in this paper. Contributions of the paper include:

- We propose a time-triggered model for the FlexRay system. The model is a formal description of the system under consideration. Focus of the model is on the essential communication mechanism of the protocol. We abstract from inessential details of the protocol in order to facilitate the analysis.
- Essential timing properties on the protocol are formally defined. Those properties are described using predicate logics, including both first-order logics and higher-order logics.
- We perform arithmetic reasoning on the system. Mechanical support with the theorem prover Isabelle/HOL is also showed.

The paper is organized as follows: in section 2 we introduce the FlexRay communication protocol; our time-triggered model for the FlexRay system is presented in section 3; we specify and verify the timing properties in section 4; finally, we discuss related work and give the conclusion in section 5.

2. The FlexRay Communication Protocol

FlexRay is a communication system intended to meet the need of automotive control applications. In particular, it targets the safety critical and high dependable systems. At the core of the FlexRay system is the FlexRay communication protocol [3].

Network Topology A FlexRay system consists of a set of nodes interconnected by a communication network. The network can be configured as a single channel or dual channel bus network, a single channel or dual channel star network, or in various hybrid combinations of both bus and star topologies.

Node Architecture A typical FlexRay node or ECU consists of a host processor, the FlexRay communication controller, a bus driver and an optional bus guardian. Two bus drivers and two bus guardians may be included if the communication controller is connected to two communication channels.

Protocol Services A number of protocol services is afforded by the FlexRay protocol. They can be logically organized as several concept layers as shown in Figure 1.

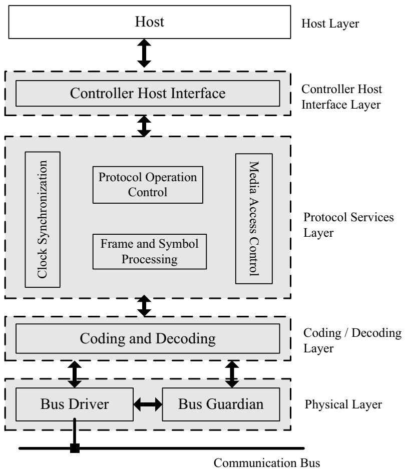

Figure 1. Logical Layers of FlexRay

Media Access Control As one of the important services of the protocol, Media Access Control(MAC) defines the crucial communication mechanism in a FlexRay system. Media access control in FlexRay is based on a repeated communication cycle. A communication cycle contains the static segment, the dynamic segment, the symbol window and the network idle time. Both the static segment and the network idle time are mandatory. The dynamic segment and the symbol window are optional. In the static segment, the Time Division Multiple Access(TDMA) scheme is used, and the mini-slotting based scheme is used in the static segment. The symbol window is a communication period in which a symbol can be transmitted on the network. The network idle time is a communication-free period in which non-communication activity, e.g. execution of the clock synchronization algorithm, can be performed.

3 The Time-triggered Model

In this section, we define a time-triggered model for the FlexRay system. We describe the time-triggered model from the following aspects:

- Clocks
- System Schedule

3.1 Clocks

A time-triggered system consists of a set of nodes, and we use $Node$ to denote this set. Each node is equipped with a physical clock. Two views of clocks are involved: *real time* and *clock time*. We use a dense time domain *realtime* to model real time and another one *Clocktime* to denote clock time. We assume standard arithmetic operators are defined on both domains. Lower case letters are used to denote *realtime* and capital letters are used for *Clocktime*. Real time is used as a reference time and it is not necessarily observable from nodes. Clock time is a node's local approximation of the reference time, and it is defined as a function which maps realtime to Clocktime:

$$C_p : realtime \rightarrow Clocktime.$$

The interpretation is that $C_p(t)$ is the reading of p's local clock at real time t. The subscript p is used to distinguish different nodes' clocks.

A physical clock is not perfect and it cannot always proceed with the same speed. However, the drift rate of a clock is normally bounded. The drift rate of a clock is defined as how fast the clock may diverge. We assume that the drift rate of any non-faulty local clock is always bounded by a known constant ρ. The value of ρ is typically small, e.g. ordinary quartz clocks generally drift in the degree of 10^{-6} seconds per second. The value of ρ is normally given by the manufacturer.

Assumption 1 (Bounded Drift Rate). *For any non-faulty clock p, there exists a ρ such that:*

$$\forall p \in Node, \forall t_1, t_2 \in realtime, t_1 < t_2 :$$
$$(1-\rho)(t_2 - t_1) \leq C_p(t_2) - C_p(t_1) \leq (1+\rho)(t_2 - t_1) \quad (1)$$

ρ is called the bounded drift rate.

We further assume that a non-faulty clock is monotonic.

Assumption 1a (Monotonicity). *For any non-faulty clock p, we have $t_1 \leq t_2 \Longrightarrow C_p(t_1) \leq C_p(t_2)$.*

In fact, the **Monotonicity** assumption can be derived from the **Bounded Drift Rate** assumption as from the **Bounded Drift Rate** assumption we have

$$C_p(t_2) \geq C_p(t_1) + (1-\rho)(t_2 - t_1).$$

Since both $(1-\rho) \geq 0$ and $(t_2 - t_1) \geq 0$ hold, we have $C_p(t_2) \geq C_p(t_1)$.

We assume the correctness of the clock synchronization algorithm: the local clocks of any two nodes are closely maintained. This can be achieved by the clock synchronization mechanism of FlexRay[3].

Assumption 2 (Clock Synchronization). *Let δ be the precision of the clock synchronization, we have*

$$\forall p, q \in Node, t \in realtime : \mid C_p(t) - C_q(t) \mid \leq \delta \quad (2)$$

The precision δ is also a constant, and its value is application-dependent.

3.2 System Schedule

One of the distinct properties of a time-triggered system is that the system activities are statically defined with respect to their occurring time. Each node maintains a schedule which states when the node should take certain actions, e.g. sending and receiving messages. A global schedule may be viewed as the composition of individual schedules. All nodes of the system have the same view of the global schedule. We define a function $sched$ to indicate the beginning of a slot as measured by the nodes' local clocks.

$$sched : Slot \longrightarrow Clocktime$$

The interpretation of $sched(i)$ is that $sched(i)$ is the start time of slot i as measured by a node's local time. The duration of a slot is constant, thus we use R to denote it, and R is measured by local clock time. Obviously, for any non-faulty node, the following equality holds.

$$sched(i+1) - sched(i) = R \tag{3}$$

Let "0" be the first slot, and given any non-faulty node, obviously the following formula holds.

$$sched(i+1) = sched(0) + i * R \tag{4}$$

4 Specification and Verification of the Timing Properties

In this section, we first define the timing properties of the FlexRay protocol, and subsequently, we prove the correctness of those properties.

Due to the complexity of the protocol, we only consider a generic architecture of the protocol. The architecture is generic in the sense that the following general abstractions are made:

- We consider a generic bus topology with one communication channel.
- The bus guardian is decentralized. In other words, each communication controller has a bus guardian.
- We assume the reliability of the physical communication medium.
- We assume the correctness of the wake-up and the start-up process.

4.1 Timeliness in FlexRay

As we have mentioned, the FlexRay protocol uses a TDMA schema within the static segment. One communication cycle is divided into a finite number of slots with the same time duration. Slots are statically assigned to individual nodes. During one slot the scheduled node has exclusive access to the communication bus. However, the node does not use the whole slot duration for message transmission. The duration of one slot includes three parts as shown in Figure 2: pre-transmission, transmission and post-transmission. It is only during the transmission phase that a scheduled node transmits the message. The pre-transmission and post-transmission are defined in order to ensure that

- The sending of a message and the corresponding reception of the message always occur within one slot.
- There is no overlap between two consecutive transmissions due to the imperfect clock synchronization.

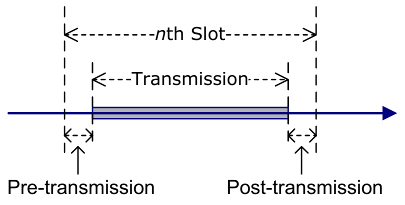

Figure 2. Parameters of a Slot

The duration of the pre-transmission and post-transmission is different for the sender, its bus guardian and the receivers. Without loss of generality, we assume node p is the sender of slot n, g is p's bus guardian and q is one of the receivers of the same slot. We use $\alpha_p(n)$ to represent the duration of node p's pre-transmission and $\beta_p(n)$ to represent the duration of node p's post-transmission.

For the sake of convenience, we refer to the transmission phase of a slot as *sending window* for the sender, *relay window* for the bus guardian, and *reception window* for the receiver respectively(Figure 3).

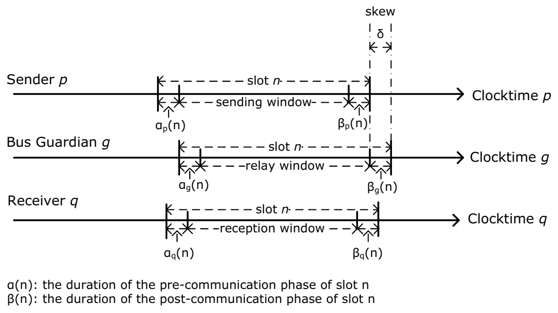

Figure 3. Sending Window, Relay Window and Reception Window

We specify timing constraints on the system as relationships between these parameters, which will be presented in the subsequent sections. These timing constraints are intended to ensure the temporal correctness of the FlexRay protocol as defined in the following.

Theorem 1 (Correct Relay). *If a non-faulty communication controller sends a message, its non-faulty bus guardian relays the message.*

Theorem 2 (Consistency). *If a bus guardian relays a message, then all non-faulty communication controllers receive the message.*

Theorem 3 (Non-interference). *There is no transmission overlap of two consecutive slots between two non-faulty nodes.*

In the next section, we prove one of the theorems – the Theorem 1 (Correct Relay). It should also be noted that we use backward reasoning in our proofs. We choose backward reasoning due to the consideration that some parameter relationships are not explicitly specified in the FlexRay specification. Our intention is to show which kind of constraints on the parameters are mandatory in order to ensure the correct timing behavior. After the proofs, the essential timing constraints are derived. Moreover, due to space reasons, the other two theorems are not discussed here. However, they can be proved in a similar manner.

4.2 Proof of the Theorem 1

Informally, the aim of the proof is to show that the relay window of the bus guardian covers the sending window of the sender. The proof is decomposed into two parts. The first part shows that within one slot the opening of the relay window is no later than that of the sending window, and the second part shows that the closing of the relay window is no earlier than that of the sending window.

Proof(the first part) The aim of the first part is to show that the real time when the bus guardian opens its relay window should be not later than the real time when the sender opens its sending window. The start of the sending window, i.e. the start of the message transmission, is $sched(n) + \alpha_p(n)$ for sender p of slot n. Let t be the real time when the sender p begins its sending window of slot n, we have

$$C_p(t) = sched(n) + \alpha_p(n). \tag{5}$$

Obviously we require that at real time t the bus guardian should already open its relay window. Otherwise, the message can not be correctly relayed. Therefore, the goal of the theorem becomes to show that the following formula holds:

$$C_g(t) \geq sched(n) + \alpha_g(n). \tag{6}$$

From the clock synchronization assumption (2) we know

$$|C_p(t) - C_g(t)| \leq \delta, \tag{7}$$

which is the same as

$$-\delta \leq C_p(t) - C_g(t) \leq \delta. \tag{8}$$

According to the definition (5) the following formula holds

$$C_g(t) \geq sched(n) + \alpha_p(n) - \delta. \tag{9}$$

From the formula (6) and (9), the following constraint is derived

$$\alpha_p(n) \geq \alpha_g(n) + \delta. \tag{10}$$

Proof(the second part) The aim of the second part is to show that the real time when the bus guardian closes its relay window should be not earlier than the real time when the sender closes its sending window. As we have stated, the duration of any slot in FlexRay is a constant R. The end of a slot marks the beginning of its succeeding slot. Given a slot n, the end of its transmission phase can be computed by $sched(n + 1) - \beta_p(n)$, where $\beta_p(n)$ is the duration of the post-transmission of slot n for the sender p. The duration of the post-transmission of slot n for the bus guardian g is denoted as $\beta_g(n)$. Let t' be the real time at which the sender p closes its sending window, we have

$$C_p(t') = sched(n + 1) - \beta_p(n). \tag{11}$$

Obviously t is the latest real time at which the sender of slot n closes its sending window. Obviously, we require that the bus guardian should keep its relay window open until real time t'. Thus, the goal of the proof is to show that the following formula holds:

$$C_g(t') \leq sched(n + 1) - \beta_g(n). \tag{12}$$

From the clock synchronization assumption (2) we know

$$|C_p(t') - C_g(t')| \leq \delta. \tag{13}$$

which is the same as

$$-\delta \leq C_p(t') - C_g(t') \leq \delta. \tag{14}$$

Due to the definition of (**??**) we have the following formula

$$C_g(t') \leq sched(n + 1) - \beta_p(n) + \delta. \tag{15}$$

From (12) and (15), we can derive the following constraint

$$\beta_p(n) \geq \beta_g(n) + \delta \tag{16}$$

Discussion of the Constraints The preceding proofs show that in order to ensure that the relay window of a bus guardian covers the sending window of its corresponding sender(a communication controller), both constraint (10) and (16) should hold.

Constraint (10) means that the pre-transmission phase of the sender should be longer than the pre-transmission phase of the bus guardian plus the parameter δ. Note that δ can be viewed as the precision of the clock synchronization algorithm. The preliminary FlexRay bus guardian specification [4] suggests the following two configurations with respect to the parameters involved: $\alpha_g(n) = \delta$ and $\alpha_p(n) \geq 2\delta$. These clearly make the inequality (10) valid.

Constraint (16) means that the duration of the post-transmission for the sender should be longer or equal to the duration of the post-transmission phase for the bus guardian plus the parameter δ. The FlexRay bus guardian specification does not explicitly specify this constraint. However, due to the fact $\beta_g(n) = \delta$, we can derive the following constraint for the duration of the post-transmission with respect to the sender

$$\beta_p(n) \geq 2 * \delta. \tag{17}$$

We have presented our proofs of the first theorem in this section. Parameter constraints are derived from the proofs. In the next section, we will check the proofs mechanically by the use of the theorem prover Isabelle/HOL.

4.3 Verification in Isabelle/HOL

In this section, we mechanically verify the informal proofs we have presented in the previous section. The proof assistant Isabelle/HOL is used.

Isabelle [6] is a generic theorem prover, and is designed for interactive reasoning in a variety of formal systems. Isabelle/HOL is the specialization of Isabelle for Higher Order Logic(HOL). Isar [5, 13] is an extension of Isabelle with structured proofs, and it makes the proofs both human-readable and machine-checkable.

Proofs in Isabelle are organized as theories. A theory in Isabelle can be viewed as a named collection of types, functions and theorems. We first declare our theory. Note that our theory extends the Isabelle built-in theory *Complex-Main*.

theory *FlexRayTiming* **imports** *Complex_Main* **begin**

We first declare the types used within our proofs. Node and slot are declared as natural numbers, and real is used for both real time and clock time.

types
```
  node = nat
  slot = nat
  realtime = real
  Clocktime = real
```

Constants can be easily defined in Isabelle. For instance, the precision of the clock synchronization is declared as a constant.

consts
  δ :: *real* — Precision of the clock synchronization

Pre-transmission and post-transmission phase of a slot are defined. These parameters correspond to α and β as used in the informal proofs. As definitions of these parameters are different for the sender, the bus guardian and the receiver, "S", "G" and "R" are used to indicate the sender, the bus guardian and the receiver respectively.

consts
  pre_S :: real — Pre-transmission phase of the sender

  pre_G :: real — Pre-transmission phase of the bus guardian
  post_S :: real — Post-transmission phase of the sender

  post_G :: real — Post-transmission phase of the bus guardian

We further declare the three functions used in the proofs.

consts
  non_faulty :: "[node, realtime] ⇒ bool" — The definition of non-faulty
  C :: "[node, realtime] ⇒ Clocktime" — The definition of clock time
  sched :: "slot ⇒ Clocktime" — The scheduled start time of a slot

The beginning time and the closing time of the sending window, relay window and reception window for a given slot are defined as follows.

constdefs
```
  Sending_begin :: "slot ⇒ Clocktime"
  "Sending_begin n ≡ sched n + pre_S"
  Sending_end :: "slot ⇒ Clocktime"
  "Sending_end n ≡ sched (n + 1) - post_S"
  Relay_begin :: "slot ⇒ Clocktime"
  "Relay_begin n ≡ sched n + pre_G"
  Relay_end :: "slot ⇒ Clocktime"
  "Relay_end n ≡ sched (n + 1) - post_G"
```

Assumptions on the system are defined as axioms. For instance, the clock synchronization assumption is defined as follows.

axioms
```
  clock_syn: "⟦ non_faulty p t; non_faulty q t ⟧
  ⟹ abs(C p t - C q t) ≤ δ "
```

With those definitions, goal of the proofs is to show that the following two theorems hold.

theorem *"⟦ non_faulty p t; non_faulty g t; pre_G = δ; pre_S ≥ 2 * δ; C p t = Sending_begin n ⟧ ⟹ C g t ≥ Relay_begin n"*

theorem *"⟦ non_faulty p t; non_faulty g t; post_G = δ; post_S ≥ 2 * δ; C p t = Sending_end n ⟧ ⟹ C g t ≤ Relay_end n"*

end

Isabelle's standard simplification tactics can sufficiently handle the arithmetic reasonings used in our proof. The complete proof script is presented in the Appendix.

5. Related Work, Discussion and Conclusion

Related Work and Discussion Pop et al. [8] presented a worst-case based technique for the timing analysis of the FlexRay protocol. Focus of their paper was on the timing analysis of both the static segment and the dynamic segment of the FlexRay protocol. As most critical protocol services of the FlexRay protocol are not available for the dynamic segment, e.g. bus guardian and clock synchronization services, our focus is on the static segment. Bauer et al. [1] proposed a central bus guardian approach for Time-Triggered Architecture(TTA) [12]. They showed an informal transmission window timing analysis of the TTA. Subsequently, Rushby [10] formally verified the transmission window timing of the TTA in PVS [9]. Rushby [11] also showed an overview of formal verification of the TTA. Pfeifer's thesis [7] gave a detailed verification of the TTA, which included some timing related aspects of the TTA protocol.

Since both TTA and FlexRay are time-triggered protocol intended for automotive applications, they share similarities. For instance, they both use a TDMA schema for time-triggered communication. However, with respect to the TDMA schema, one of the differences between FlexRay and TTA is that the duration of a slot in FlexRay is a constant, and a node may own a finite number of slots within one communication cycle. However, a node in TTA can have only one slot within one communication cycle, and slots may have variable duration within one communication cycle. Due to these distinguished properties of the FlexRay protocol, our approach has the following characteristics: first, since the duration of a slot is a constant(the constant R in the definition (3)) with respect to a node's local clock, the end of a slot marks the beginning of the succeeding slot. Thus, for a given node, timing relationship between two consecutive slots becomes straightforward. Second, due to the fact that some parameter relationships are not explicitly defined in the FlexRay specification, we intend to derive the essential constraints on the system. Thus, a backward reasoning style is used. After the proofs, parameter relationships are resolved.

Conclusion We have presented a time-triggered model for the FlexRay system. Time granularity of the time-triggered model is with respect to local clocks as measured by individual nodes. Timing properties of the FlexRay protocol are discussed, specified, and verified. Essential timing constraints on the system are derived from the proofs. Mechanical support of our proofs is also included with the help of the proof assistant Isabelle/HOL.

Acknowledgment

The author would like to thank Leonor Prensa Nieto, who has given some helpful tips on an initial discussion of the topic during the author's visit to LORIA, Nancy, France.

References

[1] G. Bauer, H. Kopetz, and W. Steiner. The Central Guardian Approach to Enforce Fault Isolation in the Time-Triggered Architecture. In *Proceedings of the Sixth International Symposium on Autonomous Decentralized Systems (ISADS 03)*, 2003.

[2] E. Dilger, T. Führer, B. Müller, and S. Poledna. The X-By-Wire Concept: Time-Triggered Information Exchange and Fail-Silence Support by New System Services. In *SAE World Congress*, Warrendale, PA, USA, 1998. SAE Press.

[3] FlexRay Consortium. *FlexRay Communication System - Protocol Specification - Version 2.1*, 2005.

[4] FlexRay Consortium. *Preliminary Central Bus Guardian Specification Version 2.0.9*, 2005.

[5] T. Nipkow. Structured Proofs in Isar/HOL. In H. Geuvers and F. Wiedijk, editors, *Types for Proofs and Programs (TYPES 2002)*, volume 2646 of *LNCS*, pages 259–278. Springer, 2002.

[6] T. Nipkow, L. C. Paulson, and M. Wenzel. *Isabelle/HOL — A Proof Assistant for Higher-Order Logic*, volume 2283 of *LNCS*. Springer, 2002.

[7] H. Pfeifer. *Formal Analysis of Fault-Tolerant Algorithms in the Time-Triggered Architecture*. PhD thesis, Universität Ulm, Germany, 2003.

[8] T. Pop, P. Pop, P. Eles, Z. Peng, and A. Andrei. Timing analysis of the flexray communication protocol. In *ECRTS '06: Proceedings of the 18th Euromicro Conference on Real-Time Systems*, pages 203–216, Washington, DC, USA, 2006. IEEE Computer Society.

[9] PVS. PVS Specification and Verification System Homepage, 2006. (see http://pvs.csl.sri.com/).

[10] J. Rushby. Formal Verification of Transmission Window Timing for the Time-Triggered Architecture. Technical report, SRI International, 2001.

[11] J. Rushby. An Overview of Formal Verification for the Time-Triggered Architecture. In W. Damm and E.-R. Olderog, editors, *Formal Techniques in Real-Time and Fault-Tolerant Systems*, volume 2469 of *Lecture Notes in Computer Science*, Oldenburg, Germany, 2002. Springer-Verlag.

[12] TTTech, TTA Group. *Time-Triggered Protocol TTP/C, High-Level Specification Document, Protocol Version 1.1*, November 2003.

[13] M. Wenzel. *The Isabelle/Isar Reference Manual*. Institut für Informatik, TU München, 2005.