

An Efficient Implementation of the SM Agreement Protocol for a Time Triggered Communication System

2010-01-2320

Published
10/19/2010

Markus Jochim and Thomas M. Forest
General Motors

Copyright © 2010 SAE International

ABSTRACT

FlexRay is a time triggered automotive communication protocol that connects ECUs (Electronic Control Units) on which distributed automotive applications are executed. If exact agreement (e.g. on physical values measured by redundant sensors on different ECUs) must be reached in the presence of asymmetric communication faults, a byzantine agreement protocol like Signed Messages (SM) can be utilized. This paper gives examples of how byzantine faults can emerge in a FlexRay-based system and proposes optimizations for a FlexRay-specific implementation of the SM protocol. The protocol modifications allow for a reduction in the number of protocol messages under a slightly relaxed fault model, as well as for a reduction in the number of messages to be temporarily stored by the ECUs.

INTRODUCTION

Distributed automotive applications run on several ECUs (Electronic Control Units) that are connected to the vehicle's sensors and actuators and communicate with each other via automotive bus systems like CAN [1] or FlexRay [2].

For some safety relevant automotive applications like drive-by-wire applications the utilization of a byzantine agreement protocol is an option. Fig. 1 gives an abstract example of an application that runs on four ECUs (nodes) connected via two redundant FlexRay channels. Three of these nodes are connected to redundant sensors that operate within a certain tolerance. For our example we will assume that the actual physical value is 5.0 and we will consider all sensor values between 4.5 and 5.5 to be correct in the sense of "acceptably precise".

Depending on the application it may not be sufficient to guarantee that each fault free node will choose an individual value between 4.5 and 5.5 (approximate agreement) but it may be necessary that all fault free nodes agree on the exactly the same value v ($4.5 \leq v \leq 5.5$) (exact agreement).

Exact agreement on a single value is desirable whenever the nodes must take a consistent threshold decision that is based on the sensed value. Exact agreement can also be desirable if the nodes accumulate state information that diverges from the state information stored in the other nodes over time (e.g. by integrating acceleration to calculate a velocity value) [3]. A realistic example for a threshold decision is a mode change from a normal operation mode to a degraded mode that must be performed by either all nodes or by none of the nodes. For our example we will assume that the nodes will perform a mode change to a degraded mode if the value chosen is greater than 4.9.

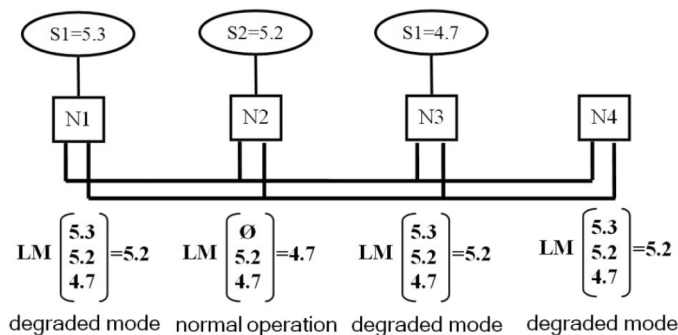


Figure 1. Approximate agreement

If each node N1, N2 and N3 just sends out a single broadcast message to distribute its local sensor value, a single byzantine fault in N1 may result in the scenario depicted in Fig. 1. The

message sent by N1 was asymmetrically received by N2, N3 and N4 such that N2 did not receive any value from N1 whereas N3 and N4 received the correct value (benign byzantine fault). If each node executes, for example, the lower median function on the vector of available values the nodes N1, N3 and N4 will perform a mode switch, whereas N2 will not perform a mode switch.

In our example exact agreement on a single sensor value among the fault free nodes even in presence of a single byzantine fault can be achieved, if each of the nodes N1, N2 and N3 distributes its local sensor value by taking the role of a commander in one of three parallel executions of the classic SM protocol (signed messages) [4]. The three parallel executions will result in the exact same 3-element consistency vector available to all fault free nodes. Since this vector may contain at most one faulty sensor value, this faulty value can be masked out by choosing the lower median of all three values. Even if the median selects the value from the faulty node this value is obviously acceptable since it is neither smaller than the smallest fault free value nor greater than the greatest fault free value in the vector.

In general SM runs in $F_{max} + 1$ rounds to tolerate any number of $F_{max} \leq m$ traitors, with m being the total number of generals. Agreement protocols like OM (oral messages) that do not make use of signatures need at least $3F_{max} + 1$ generals to tolerate F_{max} traitors. These protocols are not considered practical for automotive applications since important cases that involve three ECUs (agreement between two loyal nodes in presence of a single traitor) cannot be addressed.

Since the implementation of a membership service can be based on an available agreement service, the need for membership service also motivates the implementation of an agreement protocol. Examples for automotive use cases for a membership service are given in [5] and [6].

This paper uses the terminology introduced by Lamport, Shostak and Pease [4] where a faulty node is a traitor, a node that is fault free is said to be loyal, a node that initiates an agreement protocol in order to distribute a local value is called a commander (or commanding general), the nodes that must agree on the commanders value are called lieutenants (or lieutenant general), and both commanders and lieutenants are considered to be generals.

The remainder of this paper is structured as follows:

In the next section we will give some basic information on the FlexRay system and analyze how the following assumptions made in [4] can be met by a FlexRay system:

A1. A message that is sent is delivered correctly.

A2. The receiver of a message knows who sent it.

A3. The absence of a message can be detected.

After that we give some FlexRay specific examples of benign and malicious byzantine faults. This is necessary since it is not obvious how byzantine faults can actually emerge in a broadcast communication system. Lamport's paper does not deal with such questions since it is based on the assumption of point-to-point communication, where a faulty sender may send out different values to different receivers.

Then after giving some fundamental information on the SM protocol and introducing the concept of an acceptable message we analyze the number of messages that must be stored during the execution of the SM protocol. Since memory in the ECUs is a cost factor we propose a modification of the protocol that reduces the number of stored messages.

The total bandwidth required in a communication system is always important. For that reason we also present a protocol modification that reduces the number of messages by slightly relaxing the original assumptions on the fault model.

Finally we present examples of the potential for improvements using the protocol modification, discuss the possibility of combining the modifications, and provide a brief summary

PROPERTIES OF THE FLEXRAY SYSTEM

This section serves two purposes. It gives a brief introduction to the FlexRay [2] system and analyzes assumptions A1 through A3 in the FlexRay context.

FlexRay Consortium is a group of automotive-related companies working together to develop an automotive communication protocol suitable for present and future in-vehicle control applications. The core partners of the consortium include automotive OEMs, an automotive system supplier, and semiconductor manufacturers.

FlexRay is an example of a time-triggered communication protocol - communications patterns are determined by the passage of time rather than by contention arbitration as in a protocol like CAN. The basic notion of FlexRay communication is built around a Time Division Multiple Access (TDMA) scheme. Communication proceeds in repetitive cycles, which are subdivided into slots. Each slot consists of a defined period of time in which a single device is given exclusive access to the communication media. The boundary between the slots (and thus the mechanism for determining media access) is determined by the passage of time.

The nodes participating in a FlexRay system share a common notion of time. This notion of time, referred to as the global time, arises as a result of a fault-tolerant clock synchronization algorithm that ensures that, to within a bounded precision, all of the nodes in the system consider an event (such as the start of a slot, or the start of a cycle) to occur at the same point in time. This common notion of time is what facilitates the use of a TDMA scheme to control media access.

FlexRay provides two different media access methods that can be used to send messages in the system, and the FlexRay cycle provides separate portions, called segments, devoted to each of these media access types. In the static segment, the media access mechanism could be considered “pure TDMA” - the segment is divided into a number of equally sized slots, and each of these slots is assigned for transmission to exactly one node. When the global time reaches the time of the slot if the node that owns the slot has a transmission ready it sends it in the slot. If it has no transmission it sends a special “null frame” that contains no information. In normal operation each slot contains only one message, and all messages sent in the static segment have the same length (in terms of the number of bytes that make up the message).

In order to ensure that all nodes see a message in the same slot, the duration of a static slot must be larger than the time required to actually transmit the messages sent in the slot. A receiver will only accept a message if it both begins and ends within the slot. In theory it is possible to size the static slots such that more than one message could fit in the slot such that, under fault conditions, a faulty node could send more than one message in a slot. This possibility can be prevented at design time by limiting the duration of the static slots to less than twice the minimum transmission time of the individual messages sent in static segment. This paper assumes the FlexRay cycle is designed to prevent more than one message in a slot.

The assignment of slots to nodes is done at system design time, and it does not change during the operation of the protocol. As a result, at any given point in the static segment all nodes in the system know who is allowed to transmit. This allows, for example, additional protection to be implemented in the form of a bus guardian that ensures that only the node allowed to transmit in a given slot is actually able to transmit in the slot. Further, the fixed relationship of slots to nodes ensures that nodes always have access to the bandwidth they are allocated at design time, and they always are granted this access at the same predefined point in time in each cycle. These characteristics allow the design of highly predictable systems. This preallocation of bandwidth, combined with the possibility of the use of bus guardians to ensure that other nodes cannot interfere with a node's transmissions, partially ensure property A1. Finally, since all nodes in the system know in advance when a node is supposed to transmit

information, receiving nodes can easily detect the absence of a message simply by checking at the end of the slot whether or not the information has been received (property A3).

The second media access method available in FlexRay supports dynamic allocation of bandwidth depending on whether or not a node has a message to transmit. This is done using a minislotted method similar to the method used in the byteflight communication protocol [7]. This media access method, which is used in the dynamic segment of the FlexRay cycle, does not use a fixed TDMA procedure but rather changes the sizes of the slots depending on whether a node transmits a message. If a node transmits the slot expands until transmission is complete. If a node does not transmit the slot “collapses” into a small period of time. As a result, the timing of the messages sent in the dynamic segment varies based on whether or not other messages are actually transmitted (i.e., transmission times are not known a priori), and this limits the ability to protect media access using bus guardians. Further, since the duration of the dynamic segment is fixed, it is not guaranteed that a node that wants to access the bus will actually be able to (if, for example, nodes that come before it in the dynamic segment have already used up all available time). As a result, it is expected that the applications that would likely use an agreement protocol would not be likely to use the dynamic segment. The remainder of this paper assumes that all communication is done in the static segment.

FlexRay was targeted for fault tolerant applications. Since such applications generally require redundancy, the protocol has built in support for physical layer redundancy. This is done by providing two distinct physical channels, labeled “channel A” and “channel B”. Each channel uses an identical but independent media access mechanism, and in the static segment the slot sizes and slot boundaries are the same for both channels. Slots can be assigned to nodes independently on channels A and B, i.e., a given slot number can be assigned to one node on channel A and a different node on channel B. Obviously it is also possible to assign a slot to a node on both channels, and this is often done on systems using channel level redundancy.

This paper assumes that all nodes involved in the agreement mechanism are connected to both channels. Furthermore, it is assumed that the channel level redundancy that is part of the FlexRay protocol is used to transmit critical data (i.e., data is transmitted by sending identical copies of the data on channel A and channel B), and that this redundancy is sufficient such that a node that wants to send information is able, with a high degree of confidence, to be certain that at least one of the copies of the message is received by other nodes in the system, thus ensuring the remainder of property A1. If channel redundancy is not considered sufficient it is possible

to augment this with temporal redundancy (i.e., send a message more than one time in a cycle).¹

The format of FlexRay messages is fairly complicated (refer to [2] for details). Of particular note are the two CRC's that are part of the message, the header CRC and the frame CRC. Both of these will be discussed again in the next section, but for the time being we will focus on the header CRC. In order for a message to be considered valid by a receiver it must have a valid header CRC, which is a function of the slot identifier. The transmitter does not compute the header CRC - it is configured into the transmitter at design time. As a result, only the intended transmitter will be able to generate a message that would be considered valid by the receivers of a system. Combined with the fact that all receivers have a priori knowledge of the relationship between the slot and the sender (and the content of the data in the slot) it is easy to see that property A2 is met by FlexRay. The strength of this property can also be enhanced by the use of bus guardians (which further ensure that only the legitimate owner of the slot is able to transmit in the slot).

FlexRay supports a variety of network topologies. The most simple topology is a passive bus where each node is directly connected to both channel A and channel B or to only one of the two channels. In practice the active star topology is of particular relevance. In this topology every node has an individual link to a central active star device. A FlexRay star is functionally similar to an Ethernet hub. Whenever a node sends a message, the star will forward the message to all other nodes connected to it. Star topologies are sometimes used in safety related applications since the stars have certain error detection capabilities that help to isolate faults that may occur on a single branch only (e.g. a short of a bus line to ground). Furthermore stars make it possible to run FlexRay networks over longer distances. Examples of single channel and dual channel star topologies will be shown in the next section.

Similar to several other communication protocols like CAN, the physical layers used by the channels of the FlexRay protocol have broadcast behavior. As a result, activity that is presented by a node onto a channel of the network is normally received by all the other nodes connected to that channel. Specifically, with FlexRay it is not required to send individual point-to-point transmissions to all receivers of a message - a single transmission on the physical layer would suffice as it would be received by all other nodes. This is somewhat different than the assumption made in [4], which assumes independent point-to-point connections between the nodes in the system. In the protocols described by Lamport et al, the messages sent by loyal generals to each of the receivers are intended to be identical - as a result, there is no

handicap imposed by broadcast transmission - to the contrary, the fact that in general all messages are broadcast actually makes it more difficult for certain types of byzantine faults to exist in a FlexRay network. This topic will be discussed in more detail in the next section.

BYZANTINE FAULTS IN FLEXRAY

We begin by describing various faults that can occur in a FlexRay system. Obviously a faulty node can send incorrect information to the other nodes in the system. Assume that a faulty node can show byzantine behavior which means that a faulty process may not only send out faulty messages ("wrong values") but it may also send out different messages to different receiving nodes. This of course also includes that it may "forget" to send some or all of its messages to some or all receiving nodes. If a faulty node sends different values to different receiving nodes we define this as a malicious byzantine fault (MBF). If the node sends a single value to some of the receiving nodes and does not send a value to some other receiving nodes we talk about a benign byzantine fault (BBF). Referring to terminology used in [4] we can associate a faulty node (general) that shows byzantine behavior with the term "traitor" and terms like "traitorous general", "traitorous commander" or "traitorous lieutenant" and a fault free node (general) with terms like "loyal general", "loyal commander" or "loyal lieutenant".

As discussed in the previous section, the physical layers used by FlexRay have broadcast properties (i.e., under normal circumstances it is assumed that a message sent on the network is received by all other nodes connected to the network). Initially this may seem at odds with the characteristics of a byzantine fault (where different receivers have different notions of what was transmitted on the network). In this section it is shown that such faults can indeed happen in FlexRay systems, providing examples of mechanisms that can introduce various types of byzantine faults in a system.

First, consider the behavior of the FlexRay channel. Obviously a fault in a channel can prevent information from being transferred through the network. If the faulty network results in only some nodes receiving the information, while other nodes do not receive any information this would result in a benign byzantine fault. An example of a fault that could have this behavior is for the transmitter to send signals on the FlexRay network that are slightly off specification (SOS) such that the drive voltages are slightly below the required minimum voltage levels. As the various receivers will each have different sensitivity levels it is possible that some nodes will be able to receive the signal while other nodes will not. This would result in a BBF for the transmission of the node in question.

¹If this is done the analysis in the following sections must be modified to take into account the possibility of a traitor introducing more than two values into the system.