A Study of Communication Protocols in Distributed Real-time systems

Joakim Eriksson  
Mechatronics/Robotics program  
Mälardalen University  
Västerås, Sweden  
jen06011@student.mdh.se

Luciano Hermansen  
Mechatronics/Robotics program  
Mälardalen University  
Västerås, Sweden  
lnh06001@student.mdh.se

ABSTRACT
Distributed real-time systems of today are growing more and more complex. Especially in the automotive industry the need for reliable and fault-tolerant communication buses is increasing. To achieve real-time communication there are two paradigms, event triggered and time triggered communication. In this paper we will present the two main protocols used in vehicles today, each representing one of the paradigms. We will give a description of the architecture and functionality of the Controller Area Network (CAN) and the Time Triggered Protocol (TTP). We will also present an equation for response time analysis in event triggered systems such as CAN.

Categories and Subject Descriptors
C.2.2 [Computer communication networks]: Network architecture and design – distributed networks.  
C.2.2 [Computer communication networks]: Network protocols - protocol architecture

General Terms
Performance, Reliability.

Keywords
Real-Time communication, CAN, TTP

1. INTRODUCTION
The vehicles of today are getting more and more complex electronic components. To assure the functionality of these components the manufacturers need reliable embedded systems to handle the data from the sensors and provide information to the actuators. Many of the systems put high demands on the time it can take to handle the data, e.g. the airbag in a car. Hence, both hardware and software must show a time-predictable behavior.

To provide this kind of predictability in the system the manufacturers of modern vehicles mainly use real-time systems. A real-time system can provide the tools needed to predict how long the handling of data takes. By assigning priorities and deadlines to all the tasks in the system it is possible to control when they are executed and in which time frame. But with a growing complexity of the electronic system, e.g. greater number of sensors and multiple control units, the need for a time predictable communication system is increasing.

Up until the middle of the 20th century the electronic systems of the vehicles were quite simple. Connections between different parts of the electrical system were made up by wires that directly connected the nodes to each other. But with a growing number of nodes in the system the electrical wires would soon create a bird's nest of wires in the car. These systems would soon be impossible to handle and they would add a lot of weight to the car. Therefore, a common communication bus was needed.

Today there are several types of communication buses that implement different techniques for transmitting data. In real-time communication there are two paradigms, event triggered communication, e.g. the CAN bus, and time triggered communication, e.g. the TTP. There are even some hybrids that support both time triggered and event triggered communication, e.g. the FlexRay protocol and Time Triggered CAN (TTCAN) [9]. Extensive research is conducted in this field and there are several different protocols that support these three approaches.

In this paper we will provide a description of the two most used communication protocols that we believe are representative for both the time triggered approach and the event triggered approach. In section two we will present an overview of the CAN protocol and in section three we describe the TTP. In section four we will provide a summary and conclusions of the paper.

2. CONTROLLER AREA NETWORK
CAN is the most used network protocol in vehicles today. It was originally developed by Bosch and Intel in Germany in 1983 for vehicle applications. CAN is an event triggered bus of broadcast type, i.e. the network consist of shared medium that can be connected to numerous nodes. Accordingly, each of the nodes has access to all the information in the network at any time [9].

The basic idea of the CAN bus is that each of the nodes ready to transmit data at any time will transmit the frame one bit at the time. After the bit has been transmitted the node reads the bus to see if the value of the bus is the same as the value just sent. To assure the correct value is received the bus is read three times. The CAN bus act like an AND-gate, i.e. if at least one of the nodes transmit a ‘0’ to the bus, the value of the bus will also be ‘0’. This feature is an important part of the arbitration mechanism that decides which node should get access to the bus. In most CAN systems the dominant value is ‘0’ and ‘1’ is recessive but there are systems where ‘1’ is the dominant bit, although not very common [10].

Since the value of the bus must be the consistent at every point of the bus at any given time, the length of the bus cannot be greater than the length corresponding to the spread of half a bit. This implies that the length of the bus corresponds to transmission speed in the system [10]. CAN can be configured to run at bitrates
up to 1 MBit/s and the speed is fixed in the system, i.e. the bit rate cannot be set at runtime [1].

Properties of the CAN bus [1].

- Prioritization of messages.
- Guarantee of latency
- Configuration flexibility
- Multicast reception with time synchronization
- System-wide data consistency
- Multimaster. Any node may transmit over the bus if it’s free.
- Error detection and signaling
- Automatic retransmission of corrupted messages as soon as the bus is idle again.
- Distinction between temporary errors and permanent failures of nodes and autonomous switching off of defect nodes

2.1 Data handling in CAN

In a network of broadcast type frames from two nodes cannot be sent simultaneously. Hence, any node that wants to transmit over the network has to read the bus to see if it is currently used by another node. If another node is currently transmitting a frame it simply has to wait until the transmission is complete. When the bus is idle all nodes that want to transmit a frame competes to use the network. To determine which node should get access to the bus if several nodes wants to transmit simultaneously the Carrier Sense Multiple Access with Collision Resolution (CSMA/CR) protocol is used.

The CSMA/CR protocol is a protocol that allows several nodes to try and use the network simultaneously if it is idle. Each frame has an identifier which also works as the priority of the frame. The identifier with the highest priority will get access to the network and can continue with transmitting the data. Hence, each of the identifiers in the system has to be unique [10].

When a node is ready to send a frame over the network it uses the CSMA-mechanism described in figure 1. It waits until the bus is idle and then it sends out the start of frame bit. After the start of frame bit has been sent, the node sends out the first identifier bit and then reads the bus. If the bus value corresponds to the value just sent, the node sends the next identifier bit. This procedure is repeated until all the identifier bits have been sent and then the rest of the frame can be sent [10].

If the node detects a difference in the value sent and the value of the bus somewhere along the way, it means that another node with higher priority is also currently trying to use the bus and the arbitration is aborted. Accordingly, frames with high priority have dominant bits close to the MSB, i.e. the highest priority is “00……..00” in systems where ‘0’ is the dominant value. When a node has aborted the arbitration it goes back to waiting for the bus to be idle and do the procedure again [9].

![Figure 1 Flowchart for the CSMA-mechanism](image)

2.2 Frame-types in CAN

In CAN there are four different types of frames; Data frame, Remote frame, Error frame and Overload frame. These four frame-types can be used accordingly to what needs to be broadcasted on the network. There are two versions of Data and Remote frames in CAN, the standard version with an 11 bit identifier field and the extended version where the identifier field has 29 bits. Extended frames can be used in systems where a large number of different data has to be handled [1].

2.2.1 Data and Remote frames

As the name indicates, Data frames are used to transmit data from a transmitter to a receiver. The Data frame consists of seven
different bit fields: start of frame field, arbitration field, control field, data field, CRC-field, ACK-field and end of frame field [1].

Remote frames are used when a node need data from another node, e.g. a control unit need information from a sensor. It consists of the start of frame field, arbitration field, control field, CRC-field, ACK-field and end of frame field [1].

- The start of frame field is a one bit field that is sent out to indicate the start of the frame. When the transmission of the frame is started, a dominant bit is sent out on this bit.
- The arbitration field consists of an 11 bit identifier field. If extended frames are used this field consists of 29 bits. The last bit of the arbitration field is the Remote Transmission Request (RTR) bit which determines whether the frame is a Remote frame or a Data frame. The RTR bit is set to be dominant when transmitting data. When requesting data the RTR bit is set to recessive.
- The control field contains information of how much data the frame transfers. It also contains two bits that are reserved for future expansion.
- In the data field the data of the frame is stored. It can contain up to eight bytes.
- The Cyclic Redundancy Check (CRC) field is used to verify that all the data has been received. It consists of the CRC-sequence and the CRC-delimiter.
- In the Acknowledgement (ACK) field all recipients that have successfully received the message, signs for the message. If the data has been successfully received, the recipient sends out a dominant bit in this slot. The ACK-field contains the ACK-slot and the ACK-delimiter.
- The last part of the frame is the end of frame field. It contains seven recessive bits and mark that the transmission has been completed.

After a frame has been transmitted there is an Inter Frame Space (IFS). During IFS no Data- or Remote frames are allowed to be transmitted [1].

2.2.2 Error and Overload frames

When a node detects an error it transmits an error frame. It can either send out an error active flag or an error passive flag. The error active flag consists of six consecutives dominant bits and the error passive flag consists of six recessive bits. The Error frame consists of two fields, the error flag and the error delimiter [1].

The Overload frame is sent out for two reasons:

1. A receiver requires a delay before the next Data- or Remote frame is transmitted.
2. Dominant bit has been detected during IFS.

Similar to the Error frame, the Overload frame consists of two fields, the overload flag and the overload delimiter. The overload flag also consists of six consecutive dominant bits. In contrast to the Error frame, the Overload frame can only be transmitted at the expected IFS or the first bit after detecting a dominant bit according to condition two [1].

Both the Error flags and the Overload flag disrupt the law for bit stuffing (see section 2.2.3) which will cause the rest of the nodes in the system to send out an Error or Overload flag. When the nodes have sent out the respective flag they will start sending the error or overload delimiter which both consists of eight consecutive recessive bits. When eight consecutive recessive bits have been detected the Error or Overload frame is completed.

2.2.3 Bit Stuffing

To prevent the system from faulty interpreting a message as an Error frame or Overload frame, the method of bit stuffing is used. Bit stuffing prevents the frames from containing five or more consecutive bits of identical value. When five consecutive identical bits are discovered in the bit stream an extra bit of the opposite value is inserted. When the frame is received, the bit stuffing is reversed and the extra bits are removed from the bit stream [10]. Example 1 describes a possible scenario.

Example 1:
The data field of a frame contains the following value:

\[ 1111 \ 1000 \ 0100 \ 0000 \]

To avoid the six consecutive identical values the method of bit stuffing is applied which gives the following result:

\[ 1111 \ 1000 \ 0100 \ 0100 \ 0000 \]

The red marked bits have been inserted in the bit stream and the system will not interpret the message as an Error or Overload frame. Note that the first inserted '0' is also taken into account when deciding the number of identical bits in the bit stream, even though it is not a part of the data.

The method of bit stuffing is applied to the start of frame field, arbitration field, control field, data field and the CRC sequence field of the Data frames and Remote frames. The remaining fields of the frames are of such a form that bit stuffing is not necessary. By assigning identifiers that does not contain more than four consecutive identical bits, the method will not be used on the identifier field either [1].

2.3 Response time Analysis in CAN

Since CAN is an event triggered bus, i.e. all nodes will try to send their messages as soon as they have a message to send, the real-time properties of the system must be sustained. Just like a system running on a single CPU, a response time analysis can be performed prior to system start to verify that the system is schedulable. The equation used to calculate the response time is similar to the equation used on single CPU systems and is divided into two steps [10].

\[ w_{t+1} = w_t + B_t + \sum_{j \in h_p(t)} \left[ \frac{w_t + J_t}{T_j} \right] C_j \quad (1) \]
\[ R_t = J_t + w_t + C_t - \frac{\tau_{bit}}{2} \quad (2) \]

The first equation (equation 1) is an iterative equation that is used to calculate the response time of the first bit in the frame. It is
iterated until \( W_i^{n+1} = W_i^n \) which gives the response time for the first bit in the frame. The terms in the equation represents:

- \( \tau_{bit} \) is the transmission speed of one bit over the bus. The transmission time is dependent on the speed of the bus.
- \( B_j \) is the time the message can be blocked from using the bus. Since the bus is non-preemptive, the maximum blocking time is the transmission time of one frame.
- \( \sum_{\forall j \in hp(i)} \) is the sum of all delays caused by other messages with higher priority.
- \( J_j \) is the jitter higher priority messages experience due to other tasks running on their respective nodes.
- \( T_j \) is the periodicity of higher priority tasks.
- \( C_j \) is the transmission time for frames with higher priority.

The second part (equation 2) gives the response time for the entire frame.

- \( R_i \) is the total response time for the entire frame.
- \( J_i \) is the jitter experienced from other tasks running on the same node.
- \( w_i \) is the response time for the first bit.
- \( C_i \) is the transmission time of the frame.

When calculating the transmission time for the entire frame, it is only the worst case transmission time that is interesting. For a standard frame with an 11 bit identifier field the transmission time can be calculated with equation 3.

\[
C_i = \left( 47 + 8n + \left\lfloor \frac{34 + 8n - 1}{4} \right\rfloor \right) \tau_{bit}
\]  

(3)

The different parts of the equation represent the number of control bits + the data bits + the number of bits inserted in the bit stream by the bit stuffing mechanism. The equation assumes that the identifiers are constructed in such a way that bit stuffing is not necessary on this field. It also assumes that bit stuffing has to be performed after every fourth bit in the rest of the fields described in section 2.2.3.

3. TIME TRIGGERED PROTOCOL

TTP is based on the Time Triggered Architecture (TTA). The driving force in a TTA is the time generated at a global time base with a known precision. The global time base is used to synchronize the nodes which makes the communication easier and to guarantees the timeliness of a real-time application. A time triggered system differs from an event triggered system in such way that all the functions and the systems behavior are predefined [7].

TTP is developed for systems with very high requirements on reliability [10]. It was developed and designed at the Vienna University of Technology in the 80’s and has since 1998 been further developed by TTTech.

The order of the messages transmission is predefined in an offline-schedule which makes the TTP predictable. However, unlike the CAN protocol, the message handling is not priority based. Instead of a priority based access to the bus for messages, TTP uses the principle of a Time Division Multiple Access (TDMA) which divides the time into slots that nodes are assigned to. The nodes are only permitted to transmit data during its predefined TDMA-slot.

The TTP is composable which means that it is possible to integrate new components or subsystems into a larger system without loosing the characteristics of the newly integrated parts themselves [2].

One of the main ideas with the TTP was to replace the existing mechanics and hydraulics in vehicles (used for steering and braking) with electronic systems and computers.

During the development of the TTP, the main objective was to design a protocol that provides all the necessary services required for a fault-tolerant real-time system. Some of the services are listed below [5].

- **Message transport with predictable low latency** – the protocol must have a short maximum execution time which preferably should be independent of the load and the fault conditions. The protocol should be able to handle both event-triggered and time-triggered messages.
- **Fault-tolerance** – since the protocol is aimed for safe-critical systems, it must be able to handle all nodes and channel failures that are listed in the fault hypothesis.
- **Temporary blackout handling** – if the system operation is being interfered by a powerful external disturbance, the protocol must be able to detect and handle the effects of the disturbances immediately.
- **Clock Synchronization** – the most essential function in a time-triggered system is the clock. Therefore, it is important to provide a global time base time with a known precision.
- **Membership service** – provides failure detection of in- and out-coming links. The nodes are aware of which nodes that are functioning and not. The membership service is also needed to provide implementations of atomic multicast protocols and redundancy management protocols.
- **Distributed redundancy service** – the task of a redundancy protocol is to remove non-functioning nodes and replace them with spare nodes or repaired nodes.
- **Support for rapid mode change** – several real-time systems provides a set of executable modes. The
protocol should be able to manage and control rapid mode changes.

- **Minimal overhead** – since one of the intended uses of this protocol is in the automotive industry (in which the system communications have low bandwidth) it is important that the system overhead is minimal.

- **Utmost flexibility without compromising the predictability** – the protocol should be able to deliver utmost flexibility while the predictability of the timeliness is maintained.

### 3.1 The System

The TTP protocol consists of fail-silent nodes which are connected by two replicated buses. A fail-silent node is able to detect if the node itself is non-functional and if that is the case, it will not operate nor communicate with the rest of the system. Hence, the name fail-silent. The purpose of a fail-silent node is that each single node is either delivering the correct functions or nothing at all [10].

Each node can be replicated and grouped into a Fault Tolerant Unit (FTU). All nodes in a FTU are doing the same thing, i.e., executing the same function. The system should be able to run even if one node is non-functional [3], i.e. if one node in a FTU is not operating correctly, another one might do. Hence, the unit is tolerant against failures. The FTU is operational as long as at least one node in the unit is operational [5].

Each node contains a bidirectional communication controller which in turn is connected to the replicated buses. The communication buses only allow one message transmitting at the time. As mentioned, a TDMA protocol is used to manage the access for the nodes to the communication buses. TDMA permits one node at the time to send data. The TDMA protocol is triggered by the global time base given by the TTP. The time-interval when a certain node is allowed to send data is called a TDMA-slot. Each of the nodes is assigned one TDMA-slot. However, if a node needs to send much data, it will be assigned more slots. When the last node has transmitted its data, the first node will get access again to transmit. The time-interval between the first and the last nodes access to the bus is called a TDMA-round, see figure 2. Hence, the maximum waiting time for a node to transmit data is a TDMA-round if each node contains one TDMA-slot [10].

![Figure 2 Example of a TDMA round](image)

#### 3.1.1 The controller state (C-state)

To check if the sender and the receiver are consistent at the sending and the receiving time, a controller state is used. The controller state includes the current operational mode, the time and the membership. The mode has an identification bit of the current operating mode. The time is holding the current global time and the membership contains information about which node has been active and which has not been active recently. To enforce agreement on the controller state, the TTP uses a CRC calculation. If the CRC check disagrees on the sender and the receiver, the message will be discarded [5].

#### 3.1.2 Frames

All data that is transmitted on the buses are wrapped into frames. Every single frame can be divided into different fields. A frame consists of a control field, a data field and a CRC field. The control field also called the start of frame (SOF) is in turn divided into three subfields, see figure 3. The first subfield contains information about whether the frame is an I-frame (initializing frame) or an N-frame (normal frame). An I-frame is an initialization frame which is needed to initialize a node whereas the N-frame is a normal frame containing data. The second subfield is the mode change field which holds information about the successor mode. The last field in the frame is the acknowledgment field which is set when frames have been sent from the nodes. The CRC-field verifies if the frames have identical C-states on the sender and the receiver. [5]

![Figure 3 shows how a typical N-frame looks like](image)

#### 3.1.3 Membership Protocol

The membership protocol is used to control if the nodes are active or not. If a node is inactive it is considered to be non-functional. Whenever a node is allowed to transmit data, the membership bit in the controller state is set. If the receiving node is not receiving any correct frames within a certain time, the membership bit will be set to zero to inform the system that the sending node is non-functional. [5]

#### 3.1.4 Temporary blackout handling

A temporary blackout occurs when several nodes fails due to a powerful external disturbance. The TTP is able to detect the temporary blackout by its continuous checking of the membership protocol. If a mayor drop is recorded it will change the mode to a mode called blackout monitoring. When the system has stabilized it eventually switches mode to the blackout recovery mode. [10]

#### 3.1.5 TTP configurations

The TTP system can be configured depending on the desired level of fault-tolerance. Figure 4 shows two typical configurations [10]. The class I configuration uses one active node and two buses. The class II configuration on the other hand, uses two active nodes.
and two buses. The nodes in the class II configuration are grouped together in a FTU.

![Figure 4 TTP configurations](image)

### 3.2 Applications of the TTP

"X-by wire" is a project which aims to replace the traditional mechanics and hydraulics in vehicles by electronics and computers. To allow and be able to implement such a system, very high requirements are put on the reliability of the communication protocol. The x-by wire system must be at least as reliable as the substituted mechanical system [10]. The x in x-by wire stands for any basis of safety related functions, e.g. brake or steer [11]. The idea with a steer-by wire system for instance is to replace the steering wheels mechanics (the steering column) with electronics with no mechanical backup.

X-by-wire systems are frequently used in the aircraft industry and more recently there are plans of developing the technique for use in the automotive industry [4].

### 3.3 TTP/A and TTP/C

There are two different versions of the TTP. TTP/A is a reduced and a low-cost protocol version. TTP/A is suitable for the SAE automotive class A applications [6]. Class A applications are not covering any safety critical feature. The class A applications are covering the "comfort"-features such as the seat controls or the electrical window controls. The data rate is less than 10kbit/s. [9]

TTP/C is the "full" version of the TTP and is more expensive than TTP/A. TTP/C is a high-speed protocol that meets the SAE requirements for class D applications. The class D protocols cover the x-by wire applications [9].

### 4. SUMMARY AND CONCLUSIONS

In this paper, we have introduced the CAN and the TTP communication protocols for embedded real-time systems. CAN and TTP are the two most used communication protocols in the automotive industry today.

The event triggered protocols like CAN provides a priority based message handling and are known for their flexibility. One reason of why the CAN protocol does not meet the requirements for the x-by wire systems is due to its lack of determinism. It is impossible to determine the transmission order of the messages. However, CAN is time-deterministic which means that the latency of the messages can be calculated with response time analysis. CAN is also efficient in its use of the bandwidth because of the priority based message handling, i.e. the messages considered to be the most important are those who gets transmitted [8].

The time triggered protocol offers a deterministic behavior and is dependable and fault tolerant. For safety related applications such as the x-by wire systems the need for a deterministic behavior is essential. One thing that makes the TTP deterministic is that the message-scheduling is made prior to the implementation of the system. Hence, the messages will always be sent in the same order and has a known latency. Another important factor is the level of fault tolerance of the protocol. The TTP offers several services for handling faults and errors, like for instance the FTUs and the membership protocol. One drawback with the time triggered protocol is the inefficient way of transmitting the messages [8]. High priority messages will have to wait on low priority messages since each node is allocated its own TDMA-slot in a TDMA-round.

Depending on the areas in which the protocols are intended to be used, the event triggered protocol is better suited for one and the time triggered protocol for the other. One of the central focused areas today in the automotive industry is the already mentioned x-by wire project which puts very high requirements on the protocols features. The time triggered protocol is one of the competing protocols for setting the standard in the x-by wire systems since it was intended for uses in x-by wire applications.

The advantages that CAN and TTP provide has contributed to the development of the future upcoming communication protocols, e.g. TTCAN and FlexRay. The developers are striving after making their protocols more flexible, dependable, fault-tolerant, and composable.

### 5. REFERENCES

