International Journal of Advanced Science and Research
ISSN: 2455-4227; Impact Factor: RJIF 5.12
Received: 30-10-2020; Accepted: 16-11-2020; Published: 04-12-2020
www.allsciencejournal.com
Volume 5; Issue 5; 2020; Page No. 41-47

Design and development of can and flex ray protocols for real-time systems

S Ravichandran[1], KGS Venkatesan[2]
[1] Professor, Computer Science Department, Annai Fathima College of Arts and Science, Madurai, Tamil Nadu, India
[2] Professor, Department of CSE, Megha Institute of Engineering and Technology for Women, Hyderabad, Telangana, India

Abstract
Today, electronics are an indispensable part of modern vehicles. Cars of moderate class contain more than 70 electronic control units connected by different data buses and gateways. The communication protocols are characterized in event-triggered and time-triggered mechanism. Several event-triggered protocols LIN, CAN, J1850, etc. and TTP, SPIDER, Byte flight, Lon Works, Flexray, etc. are the time-triggered protocol have been in the market. In this paper the most highly preferred protocol CAN is compared with the newest Flexray. CAN has a good performance in terms of bandwidth utilization, but is not considered deterministic enough to be used in safety critical applications which require high reliability and dependability. TTP being developed with extensive research for highly dependable systems i.e., hard real-time systems but is inefficient in terms of network utilization and lacks in flexibility. Flexray is a hybrid protocol taking advantages of both event-triggered and time-triggered approaches providing dynamic and static communication depending on the severity of link usage in the network. The advantages and constraints of the protocols in the network designer's perspective are described.

Keywords: electronic control units; real-time system; event-triggered, flexray protocol, can protocol and time-triggered mechanism

Introduction

With the introduction of electronics in automotive systems its use has increased exponentially. Nowadays, variety of electronic devices like microcontrollers, sensors, and actuators are used in modern cars to replace mechanical and hydraulic components. Some examples of this are engine management systems, anti-lock braking systems (ABS), automatic transmissions and central locking. These electronic control modules typically get their inputs from switches and sensors, compute using the received data and then use actuators to enforce the outputs. These electronic control units (ECUs) require exchange of information among each other for execution of their tasks. An Example, to change gear the transmission ECU requests the engine ECU to reduce torque, the transmission ECU then informs the gear shift actuator to change gear. Once the gear change has been made the transmission requests the engine to increase the torque again. Most of these ECUs are interconnected using some sort of communication interface. In today's cars of moderate class, more than 70 ECUs exchange up to 2500 signals.

If all the devices and sensors of the vehicle were to be connected together using point-to-point wiring, the cable networks in cars would grow to lengths of several miles. This would add to the overall cost and reliability problems of vehicle.

As the number of ECUs increase, the need for faster and more reliable communication is required. To overcome this problem a networking system has to be designed, probably using a serial bus system to connect the various control systems. Different types of in-vehicle networks have been developed. Currently, the most widely used network is the Controller Area Network (CAN).

The ultimate goal of the electronics development in the automotive industry has been x-by-wire, taking the name from the aircraft industry's fly-by-wire concept. X-by-wire includes technologies such as brake-by-wire and steer-by-wire. The expansion of in-vehicle networking provides many system-level benefits over previous mechanical means, including: fewer wires required for each function, which reduces the size of the wiring harness and improves system cost, weight, reliability, serviceability and installation time; additionally functions can be added by making software changes, allowing greater vehicle content flexibility; common sensor data available on the network so it can be shared, eliminating the need for multiple sensors. However, the event-triggered communication protocols in use today are poorly suited to x-by-wire systems which require periodic data exchange with low jitter i.e., a protocol being very fast, deterministic, and fault-tolerant that could satisfy the speed, reliability, and safety.

Several time-triggered technologies such as time-triggered CAN (TTCAN), time-triggered protocol (TTP), and Flexray, have been designed to provide predictable medium access at a higher available bandwidth. An example of in-vehicle network for a typical car is shown in figure 1 below.

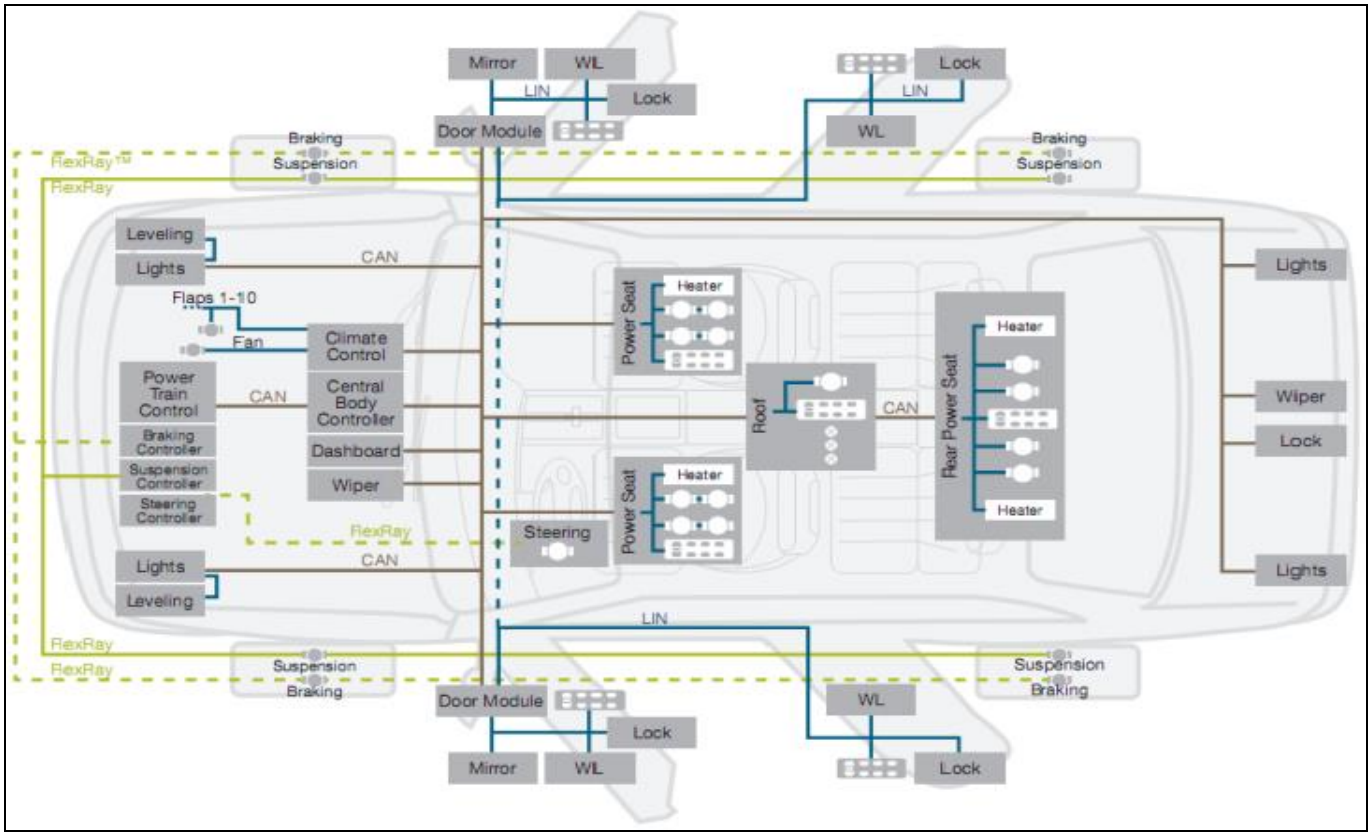

Fig: 1. In-vehicle network example

Time-triggered protocol (TTP) has been developed by Technical University of Vienna after two decades of extensive research. Messages using TTP are statistically schedule based on the progression of time. It has an advantage that it can precisely control the message transmission and reception time. This characteristic makes it suitable for safety critical applications. However there are three drawbacks to see: it is inefficient in terms of network utilization and periodic message response time and the other being lack of flexibility.

Nowadays, either an event-triggered or a time-triggered mechanism is required for message transmissions in the vehicle network, and in some cases, both of them are required at the same time in complex control systems. A hybrid type of protocol has evolved called Flexray communication protocol which allows transmitting both event-triggered and time-triggered messages on the same bus, thus taking the advantages of both approaches. In this paper, is selected the most commonly used protocol CAN which is an event-triggered technology and the newest protocol in the market, Flexray, for comparison. The section II describes the history and background of the two protocols. The section III will present an overview of how the two protocols work. A number of different comparisons between the two protocols is made in section IV. Finally, the real-time demands are discussed in section V followed by conclusions.

Related Work

Controller Area Network

It is a serial bus system, which was developed by Robert Bosch in 1980's for automotive applications. The design was simple, efficient and robust communication network. The CAN protocol is internationally standardized in ISO 11898-1 and comprises the data link layer and components of the physical layer of the 7-layer ISO-OSI reference model. CAN, which is now available from more than 50 semiconductor manufacturers in hardware. The main principles of CAN are:

- Multi-master: Any node may send if the bus is idle.
- Guarantee of latency times: It is possible to calculate the worst case time for a message to be sent and reached at the destination node.
- Configuration flexibility: Nodes can be added to the network without change in hardware or software.
- Prioritization of messages: Conflicts are avoided by prioritizing the messages.
- Broadcast communication: Every transmitted message is received in all nodes.

Error detection and correction: The faulty messages are retransmitted and if found to be permanent fault, the node is autonomously switched off.

Flexray

The Flexray consortium emerged after BMW and DaimlerChrysler realized that available solutions did not meet their future needs for data throughput and determinism. In September 2000, they joined forces with Free scale and Philips and formed the Flexray consortium to establish Flexray as the de facto industry standard. The first vehicle to use Flexray protocol was BMW X5 in 2006 but the BMW 7-series was introduced in 2008 fully utilizing the protocol. The main principles of Flexray are:

- Deterministic communication
- Support for on-demand communication (not to interfere with the deterministic communication)
- Scalable fault-tolerance.
- Dual channel data rate of 10 Mbps.

- Support for composability
- Predictable behavior at absence of node or presence of error conditions.
- A network wide consistent view of time with a known accuracy of all nodes.
- Distributed computing through a global time clock
- As backbone network, working in conjunction with already established systems (such as CAN, LIN etc.)

Problem Statement

CAN Protocol

A typical vehicle can contain two to five separate CAN networks operating at different transmission rates. A low speed CAN used for non-critical applications like seat and window movement controls operate at less than 125 kbps. They have an energy saving sleep mode in which nodes stop their oscillators until a CAN message awakens them. A higher speed CAN interconnect the more real-time critical functions such as engine management, anti-lock brakes, and cruise control. Although capable of a maximum baud rate of 1Mbps, the electromagnetic shielding required makes it costly.

The CAN protocol defines the Data Link Layer and parts of the Physical Layer. The protocol specifies a 5V differential electrical bus as the physical interface. Most of the layers of the ISO/OSI protocol stack are implemented by the software developer. In the physical layer following properties are discussed:

- Bit Encoding/Decoding
- Bit Timing and synchronization
- Physical Medium
- Data Rate vs. Bus Length

The CAN protocol uses Non-Return-to-Zero (NRZ) bit coding. With NRZ bit coding the signal level remains constant over the bit time so just one time slot is required for the representation of a bit. Bit stuffing is applied by inserting a complementary bit after five bits of equal value. At the receiving end, the receiver has to un-stuff the stuff-bits so that the original data content is processed.

CAN uses synchronous bit transmission which enhances its transmitting capacity. To enable the receiver to correctly read the messages, continuous resynchronization is required. Phase buffer segments are therefore inserted before and after the sample point within a bit interval. The CAN protocol regulates bus access by bit-wise arbitration, so the signal propagation from sender to receiver and back to the sender must be completed within one bit time. There are two types of synchronization used: hard synchronization at the start of a frame and resynchronization within a frame.

One of the cheapest and most common physical medium used is twisted wire pair. The two lines are driven by the nodes with a differential signal.

Depending in the size of the propagation delay segment the maximum possible bus length at a specific data rate can be determined.

Table: 1 BIT Rate vs. Bus Length

Bit Rate (KBPS)	Bus Length (M)
1000	30
500	100
250	250
125	500
62.5	1000

In the data link layer the following operations are discussed by the protocol:

- Message Framing
- Arbitration
- Error Detection and Handling

Once the data has been accepted from the processor, it is then bundled into a predefined structure called a frame (figure 2) by the CAN controller.

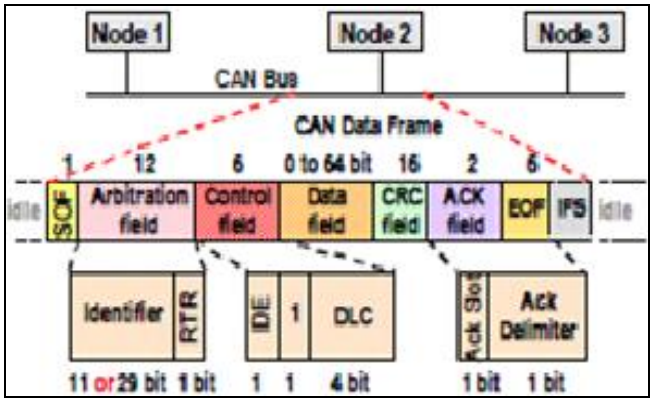

Fig: 2 CAN Frame Format

The CAN protocol defines four different types of frames:

- Data frame: It is generated by a CAN node when the node wishes to transmit data. This is received by all other nodes on the bus.
- Remote frame: It is generated by a destination CAN node to request data from another node on the network.
- Error frame: It is generated by a node when it detects a protocol error.
- Overload Frame: It is generated if a node wishes to request more time to process received information

The CAN communication protocol uses a CSMA/CD process. If two or more units starts to transmit simultaneously the Bitwise Arbitration will determine which unit may continue sending. Since each unit has a unique identifier included at the start of every message, that identifier is used for determining the priority of the unit by having more or less dominant bits in it. The units that are trying to transmit a message will do so by sending one bit at a time and monitoring the value on the bus, if the unit sends a recessive bit and reads a dominant bit the unit will stop transmitting and wait for the bus to be free and retransmit. This will result in the most dominating bits of that unit will be able to transmit its message.

CAN nodes can transition from functioning as a normal node, to shutting down completely based on the severity of the errors detected. This feature is called Fault Confinement.

Flexray Protocol

Flexray is a dual channel, high speed protocol with data rates of up to 20Mbps (10Mbps per channel). It is fault-tolerant and deterministic, and is aimed at advanced applications such as x-by-wire. For safety critical applications, messages can be transmitted simultaneously on both channels giving a built in redundancy to the network. If one channel gets damaged, transmission will continue without interruption on the other channel. This is essential if drive-by-wire applications are to be implemented, so that, in the event of a failure on a channel the driver would still be

able to have full control over the vehicle's brakes and steering. Data is transmitted on the Flexray bus in both timed and event driven manner. Each message is divided into static segment and the dynamic segment. The static segment is defined during the configuration of the application and transmits the data on a TDMA basis. The dynamic segment of the message handles data on an event triggered basis.

The protocol defines parts of the physical layer, data link layer, presentation layer and application layer of the OSI model in the context of a Flexray communication controller.

The Flexray physical layer is discussed with the following principles:

- Network Topologies
- Transmission medium
- Signal levels and bit representation
- Bit coding and decoding
- Synchronization

It can be configured as a single-channel or dual-channel bus network, each node on the network can be connected to either or both of the channels. This flexibility in configuration may be used to increase bandwidth and/or introduce redundancy in to the system to increase its level of fault tolerance. The Flexray protocol specification does not define cable types to be used, but does stipulate their electrical specifications. The medium in use for Flexray busses may be shielded or unshielded cables, as long as they provide the following characteristics: Impedance of 80-100Ω at a frequency of 10MHz, maximum line delay of 10ns/m and a maximum cable attenuation of 82dB/km at a frequency of 5MHz. The bus communicates using two signals Bus plus (BP) and Bus minus (BM). The differential voltage between the signals V_{diff}, is used to design the bus.

$$V_{diff} = V_{BM} - V_{BL}$$

The decoding process samples the incoming data at eight times the rate of the bit clock. These samples are forwarded to a majority voting process, which analyses the last five samples received. If at least three samples are HIGH the process outputs a value of HIGH for that bit, otherwise it outputs a value of LOW.

This voting process is used to suppress glitches in the received signal, provided that the duration of the glitch is less than three samples.

Flexray is a time triggered networking system. All nodes must be synchronized for successful and accurate communication.

The clocks of the communication controllers in the network, however, can be influenced by temperature and voltage fluctuations, or production tolerances of the oscillator. Virtual reference clock is established using a distributed fault-tolerant clock synchronization algorithm.

Hence each node individually synchronizes itself to the network by observing the timing of transmitted synchronization frames from the other nodes.

The Flexray data link layer describes the following main processes:

- Message framing
- Communication cycle
- Static segment
- Dynamic segment
- Protocol operation control
- Controller host interface

The Flexray frame consists of three segments; these are the header segment, the payload segment, and the trailer segment as shown in figure 3.

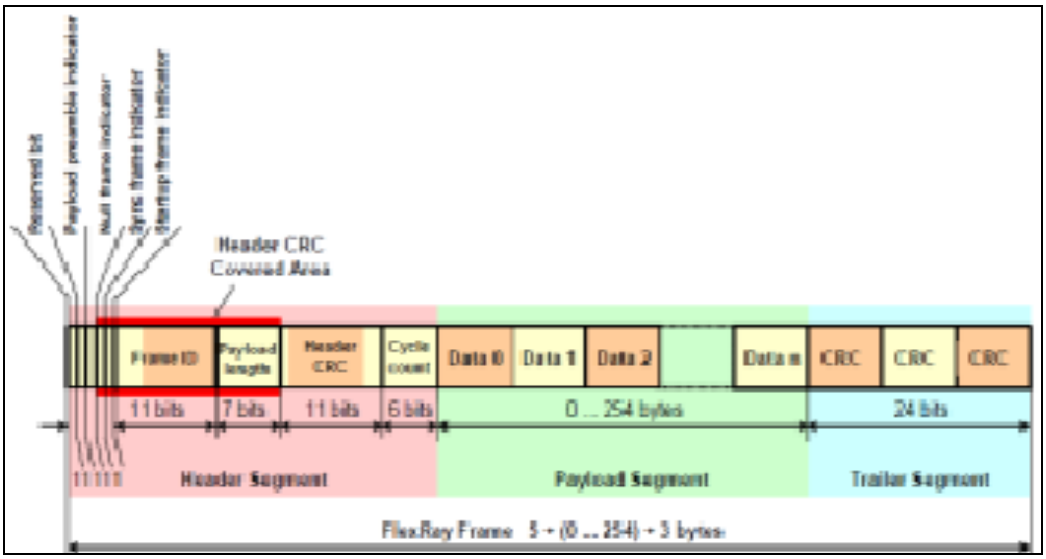

Fig: 3 Flexray Frame Format

The header segment consists of 5 bytes and contains information such as:

- Some information about the purpose and the content of the frame, like if it should be used for clock synchronization, if the payload section contains any valid data, if the frame should be used as a startup message to some node, or if it contain network management information.
- The frame ID which determines in what slot the frame should be transmitted.
- The length of the payload segment.
- A cyclic redundancy check for the header, which is a type of hash function used to detect accidental data changes

The payload frame segment consist of 0 to 254 bytes of data depending of the size of the message a node wishes to send, the payload segment may contain a network management

vector. The frame trailer segment is a 24 bit CRC that is calculated over the two previous segments, the header and the payload parts of the frame in an attempt to find errors in these fields.

Communication cycles are executed periodically, and are of constant time duration. Within one communication cycle Flexray offers the choice of two access schemes. These are a static time division multiple access (TDMA) scheme, and a dynamic mini-slotting based scheme. The network idle time is a communication free period which concludes each communication cycle and is used by each node to calculate and apply clock correction. If the message data comprise b words, then the frame size f in bit evaluates to

$$f = b.\ 16\ bit + (b.\ 4\ bit + O_F) = b.\ 20\ bit + O_F$$

Where the framing overhead is b. 4 bit+ O_F

Flexray nodes uses a hierarchy for time representation, much like to represent time in hours, minutes, and seconds the Flexray nodes represent time as cycles, macroticks, and microticks as shown in figure 4. Each control unit calculates the microticks on their own, it is the smallest parts of the Flex Ray timing hierarchy and the length of a microtick may vary from different nodes. The macroticks are synchronized by sending so called sync-frames among the connected nodes, based on these messages calculations are done within each node to determine the number of microticks for their next macrotick.

Within a certain tolerance level the macroticks should be identical among the connected nodes. A slot consists of an integer number of macroticks, and this number should be the same for all nodes in the network, and it should also remain the same in every cycle.

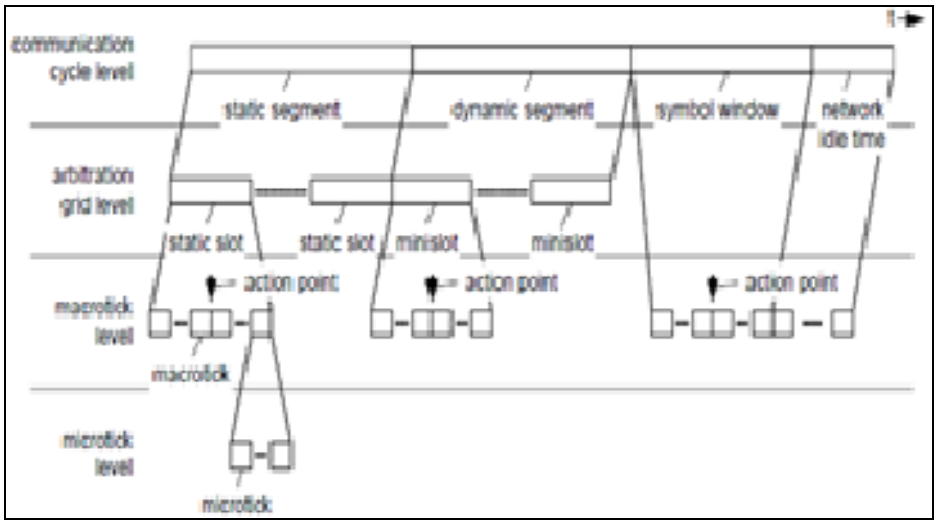

Fig 4: Flexray Communication Cycle

The static segment of the communication cycle atleast contains two static slots used for scheduling time triggered messages and reserved for synchronous communication. This segment contains a configurable number of static slots. All static slots consist of the same number of macroticks. The length of the static slot must be configured to handle the amount of data which is to be transmitted in one communication cycle. The segment timing is exactly the same on both channels.

The dynamic segment is used for event based messages, here the devices compete for bandwidth using a priority driven scheme. The size of the communication slots in the dynamic segment may vary to accommodate frames of different length, but data will only be sent if there is enough time left in the dynamic segment.

The main purpose of the protocol operation control is to react to commands from the host processor, or to protocol conditions such as errors. The controller host interface manages the data and control flow between the host processor and the Flexray protocol engine within each node. It contains two main interface blocks, the protocol data interface and the message data interface.

The protocol data interface transfers protocol related control, configuration and status data between the host and the Flexray protocol.

The massage data interface transfers messages and message related control, configuration, and status data between the Host and the Flexray protocol.

Implementation

This section will discuss the common features and differences of the two protocols.

Complexity

The Can system is considered to be a low complexity system due to the single channel bus with arbitration method for bus access. The Flexray is considered to be very complex due to use of two different communication accesses. Flexray uses both static and dynamic bus access which makes the whole system very complex.

Safety

The CAN protocol has many features to ensure safety such as bit monitoring, CRC and fault confinement. Its broadcast bus with prioritization of messages ensures deadlines for high priority functions. The CAN bus uses line topology greatly reducing the risk of the system. Since Flexray is supposed to be the next generation communication system for automotive networks, safety is one of the major motives in its development. The systems may be configured in many ways with the two channels, so if one fails the system could keep its functionality. Each node in the Flexray system is assigned a bus guardian which is a safety measure to ensure that the node does not get stuck in an erroneous mode.

Composability

The lack of composability in CAN is because of the broadcast bus with prioritized messages. Nodes that are added or removed have an impact on the temporal behavior of the system. Flexray on the other hand developed with one of the requirement. Flexray has the ability to make the system both static and dynamic with regards to the handling of messages. This makes it easy to add new nodes without having to change the scheduling of the system. Nodes can therefore be tested separately and then integrated into an existing system. Whereas nodes of CAN network cannot be tested separately.

Flexibility

The CAN system provides some form of flexibility but restricts in many ways. New nodes can be added or removed from the system, due to single channel bus network it doesn't allow any variations in system design. Like the name, Flexray is intended to be very flexible system. Due to its two channels it may be configured in different ways like bus, star and hybrid topologies. The standard design of ECU interface makes the system highly flexible.

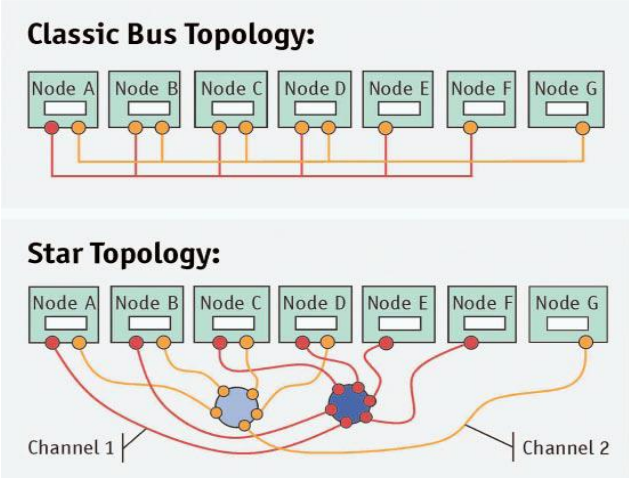

Fig: 5 Flexible Structure of Flexray with Classic Bus or Star Topology

Communication Principles

Flex Ray frames support a data field containing up to 254 bytes with a 40 bits header and a 24bits trailer, while CAN only supports a data field with up to 8 bytes with 44 bits of additional information. This means that Flex Ray may send larger messages with less overhead then CAN, but for smaller messages the overhead is about the same. However, Flexray's static segment use the same size for all slots, shorter messages will be ready before the slot is used up making the bus idle for the rest of the slot. There will also be some idle time in the dynamic segment because of the use of minislots; however that is not as big as in the static segment. CAN does not have that idle time as long as there are messages in the queue. The bandwidth of CAN goes up to 1Mbps while FlexRay can send data with a bit rates up to 10 Mbps on two channels.

Performance Evaluation

In order to build real-time reliable network: the average load on the network should be properly determined such that messages cannot be lost, if lost how often and under what conditions they are lost?

The additional ECUs can be connected as and when required, also determination of number of signals added in due can overload the bus. The designer should be in a position to diagnose these problems keeping in mind the performance and timing of the system. Finally, a suitable integration methodology is required to verify that each connected ECU and the entire network meet the defined timing constraints.

Table 2: Comparison of CAN and Flexray

property	can	Flex ray
bandwidth	1mbps	10mbps
no. of channels	1	2
frame data length	0-8	0-254
comm technique	dynamic	static & dynamic with tdma
complexity	low	high
composability	no	high
flexibility	limited	high

Knowing the worst case response time for each message is important for making schedule analysis to see if the tasks will be ready in time or not. Making such analysis is possible with both CAN and Flex Ray. However the way to calculate the response times for the different communication systems are a bit different. CAN use Fixed Priority scheduling, which means that each message got a set priority, and higher prioritized messages will get access to the bus before lower prioritized messages. The advantage with this system is that high prioritized messages will have a quick response time, however low prioritized messages may have to wait a long time before they get bus access. Another disadvantage with fixed priority scheduling is that it is not very predictable. However, it is possible to calculate worst case response times for every message.

Flex Ray have different time slots for messages i.e., scheduled by static cyclic and fixed priority. Static cyclic scheduling got the advantages that it is easy to calculate response times and is very predictable. However since it needs to be scheduled before runtime it got some limitations, like how to schedule very important messages that rarely needs to be sent.

In worst case that kind of message may have to wait for a whole cycle if its slot has passed, or multiple slots have to be assigned to that message but then these slots will pass empty most of the time. Making a static schedule may not be that easy for complex systems.

Using one static section and one dynamic section like Flexray may be an advantage since frequent real-time messages can be sent in the static segment, while uncommon high priority messages and lower prioritized messages can use the dynamic segment.

Conclusions

We have compared the widely used CAN system with the newly Flexray protocol. There are several advantages with using Flexray instead of the older CAN protocol, such as higher bandwidth and more flexibility. However it will take time before Flexray becomes widely used, because of the complexity of the system as compared to CAN. Even though the Flexray protocol is supposed to take CAN's place in automotive systems, there are still certain areas where a CAN system may be better suited. Therefore, systems with Flexray and CAN combination would be an excellent solution.

Acknowledgments
The authors are thankful to N. Navet, A. Albert, G. Leen and D. Heffe man for providing the necessary facilities for the preparation of the paper. Also thanks to IJASR Journal staffs to publish this paper.

References

1. Navet N, Song Y, Simonot-Lion F, Wil wert C. “Trends in automotive communication systems,” *Proc. IEEE*. 2019; 93:1204-1224.
2. Albert A. “Comparision of event-triggered and time-triggered concepts with regard to distributed control systems,” in Embedded World, 2018.
3. Uvesten MPP. “A distributed flexray-based research platform,” Master thesis. Chalmers Univ. of Tech, 2018.
4. Nossal R, Lang R. “Model based system development: An approach to building x-by-wire applications” IEEE Micro. 2019; 22(4):56-63.
5. Leen G, Heffeman D. “TTCAN: A new time-triggered controller area network,” Microprocessor and Microsystems. 2018; 26:77-94.
6. Heller C, Schalk J, Schneele S, Reichel R. “Approaching the limits of flexray,” Network Computing and Applications, 2008. Seventh IEEE International Symp. On, pages 205-210, July 2018.
7. Markovitz R, Temple C. “Flexray = a communication network for automotive control systems,” Factory Communication systems, IEEE International Workshop on, 2017, 207-212.
8. Pat Richards. Microchip Technologies Inc. Understanding Microchips CAN Module Bit Timing. AN754, 2019.
9. Rogers B, Schmechtig S. “Flexray Message Buffers – The host view of a flexray system, 2019.
10. Shapiro J. Flexray: The Next Generation In-Vehicle Network. Evaluation Engineering, March, 2018.
11. K. Pazul “Controller Area Network (CAN) Basics,” AN713, 2017.
12. Kopetz H, Bauer G. “The time-triggered architecture,” Proc. IEEE. 2017; 91(1):112-126.