In-Vehicular Communication Networking Protocol

Renu Sharma

Indiana University, Purdue University

Indianapolis, IN

rensharm@iupui.edu

ABSTRACT

Today's modern vehicles contain a complex symbiosis of intelligent electronic systems and integrated mechanical structures. In-vehicle networks provide an efficient method of communication between the various electronic components in an automobile. In-vehicle communication contains a very complex structure because safety is a high priority. The communication protocols discussed in this paper are Controller Area Network (CAN), FlexRay, and Local Interconnect Network (LIN). CAN is popular for medium speed applications, FlexRay is used in high speed applications, and LIN is mostly used in low speed applications.

***Keywords* - CAN (Controlling Area Network), FlexRay, LIN (Local Interconnect Network)**

I. INTRODUCTION

Vehicle communications have been growing rapidly as the Electronic Control Modules (ECMs) started interacting with other ECMs. As we refer existing technologies in automotive, vehicles contains minimum 30+ ECMs. Some luxury vehicles claim for 90+ ECMs. This makes automotive communication protocols more challenging and need rapid improvement as users and systems designer. All ECMs are demanding more information to be available in-vehicle. We need diffident bus communication based on type of ECMs. Type of ECMs will decided by its criticality of Application and that makes communication more complicated. In following section/s describe about few major automotive communication protocol. Each has its own importance in vehicle based on its Advantages. Some protocols are expensive in terms of controller or very complicated for implementation than other but it provided more robust and efficient data flow in-vehicles.

CAN and LIN are amongst the most popular protocols in-vehicle and have existed as standardized. It is expected that LIN and CAN networks will continue to dominate vehicle communication over the next decade. FlexRay is in market from last decade and getting its popularity by own. Ethernet and MOST are demand for today's OEM as tomorrow's technology. Those protocols needs to be more standardized and need enhancement for automotive. Ethernet and MOST is not described in this paper. The rest of the paper is organized as follows. Section II describe about CAN protocol along with its layered architecture. Section III gives information about FlexRay protocol. LIN protocol is described in detail in section IV. Section V contains vehicle architecture in detail. Section VI and VII provides advantages, disadvantages, application and described protocol comparison. Conclusion is in last section of the paper section VIII.

II. CAN - CONTROLLER AREA NETWORK

The Controller Area Network (CAN) is a serial asynchronous multi-master communication bus defined by ISO and SAE standards. Developed by Bosch in the 80's for the automobile industry CAN protocol can handle data rates up to 1 Mbit/s. It is largely immune to electrical faults and can detect errors. The CAN protocol describes how information is exchanged between individual nodes of the network. Based on the OSI reference model, it defines the two lowest layers: the data layer and the physical layers. Two versions of the CAN are defined today: variant A specifies 11-bit message identification (ID), variant B specifies a 29-bit ID with a theoretical maximum of more than 536 million IDs. Every CAN node always sends a message to all systems on the bus.

CAN has the following properties

- Prioritization of messages
- Guarantee of latency times
- Configuration flexibility
- Multicast reception with time synchronization
- System wide data consistency
- Multi-master
- Error detection and error 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

Layered Architecture of CAN

Figure 1 shows the layered architecture of CAN according to OSI reference model.

The Physical Layer - It defines how signals are actually transmitted on the bus. Hence it specifies the description of Bit Timing, Bit Encoding, and Synchronization.

Data Link Layer - DLL is divided into two sub-layers; first one is Media Access Control (MAC) and other one Logical Link Control (LLC).

Media Access Control - The MAC sub-layer represents the core of the CAN protocol. It presents messages received from the LLC sub-layer and accepts messages to be transmitted to the LLC sub-layer. The MAC sub layer is responsible for Message Framing, Arbitration, Acknowledgement, Error Detection and Signaling [1]. It's supervised by a management entity called Fault Confinement which is self-checking mechanism for distinguishing short disturbances from permanent failures.

Logical Link Control - The LLC sub-layer is concerned with Message Filtering, Overload Notification and Recovery Management.

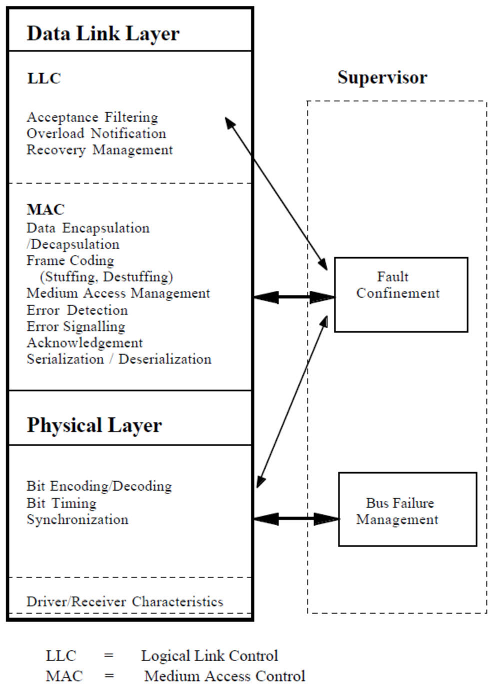


Figure 1 - Layered Architecture of CAN according to the OSI Reference Model [1]

Transmitted and received information is sent on the bus in fixed formats. There are four different frame types in CAN. Data frame, Remote Frame, Error Frame and Overload Frame [1]. Data Frame is used to transmit up to 8 bytes of data. Remote Frame is used to request a data frame from specific node. By sending a Remote Frame a node requiring data may request another node to send the corresponding data frame. The data frame and the corresponding Remote Frame are named by the same message Identifier. Error Frame is used to indicate error to the entire network (Not under CPU) control. Overload Frame is used to create an extra delay between frames [1].

On the network any node can transmit when the bus is idle. If two are more nodes starts transmitting in the same frame, then bus access conflict is resolved by bit wise arbitration. If two or more units start transmitting messages at the same time, the bus access conflict is resolved by bitwise arbitration using the message Identifier [1]. The mechanism of arbitration guarantees that neither information nor time is lost. If a data frame and a remote frame the same identifier are initiated at the same time, the data frame overcomes the remote frame. During arbitration every transmitter compares the level of the bit transmitted with the level that is monitored on the bus. If these levels are equal the unit may continue to send. When a 'recessive' level is sent and a 'dominant' level is monitored the unit has lost arbitration and must withdraw without sending one more bit. It means highest priority (lower ID) message wins the bus access. Transmitting nodes which lose arbitration becomes the receivers. Nodes which lose arbitration will automatically re-transmit the frame [1].

There are two different formats which differ in the length of the Identifier field: Frames with the number of 11 bit Identifier are called as Standard Frames [1]. On the other hand frames containing 29 bit Identifier are called as Extended Frames.

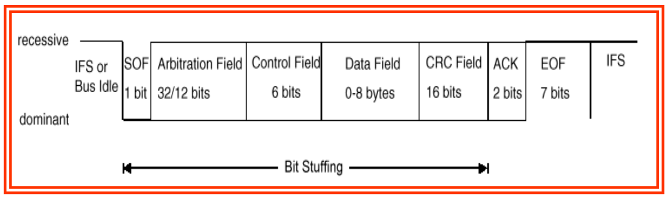


Figure 2 - CAN Data Frame [1]

Data Frame - The above figure 2 shows CAN data frame. Data field consist of seven different bit fields: Start of frame, Arbitration field, Control field, Data field, CRC field, ACK field and end of frame [1]. Data field can be length zero. Start of frame (SFO) marks data frames and remote frames. It consists of single dominant bit. Station is allowed to start transmission only when the bus is idle. Arbitration field is different for Standard Format and Extended Format Frames. In Standard Format the arbitration filed consists of the 11 bit identifier and the RTR-BIT. The identifier bits are denoted as ID-28 ... ID-18.Extended Format the arbitration field consists of the 29 bit identifier, the SRR-Bit, the IDE-Bit, and the RTR-BIT. The identifier bits are denoted ID-28 ....ID-0.

Standard Format – The below figure 3 shows 11 bit identifier. The Identifier's length is 11 bits corresponds to its base id in extended format. These bits are transmitted in the order from ID-28 to ID-18.7 bits from ID-28 to ID-22 need not to be recessive [1]. RTR bit i.e. remote transmission request bit is present in standard format as well as extended format. RTR bit has to be 'dominant' in data frame and within remote frame it should be 'recessive'. IDE (Identifier Extension bit) bit in standard format is 'dominant'. Control field consist of 6 bits. Frames in Standard Format include the data length code, the IDE bit, which is transmitted 'dominant' and the reserved bit r0 [1].

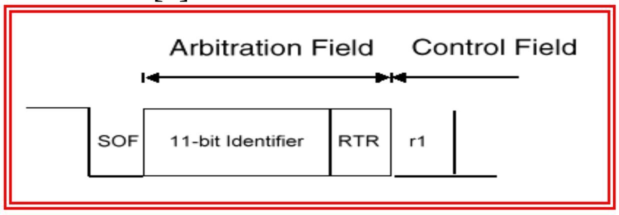


Figure 3 - 11 bit Identifier CAN 2.0A [1]

Extended Format – Figure 4 shows 29 bit identifier CAN 2.0 B. Here in extended format identifier's length is 29-bit. Extended format is divided into two sections i.e. Base ID and Extended ID [1]. Base ID consist of 11-bits. It is transmitted in the order of ID-28 to ID-18. It is same as that of standard identifier format. Base ID defines extended frames base priority. Extended ID consists of 18-bits which are transmitted in the order of ID-17 to ID-0. Base ID is transmitted first followed by IDE bit and SRR bit. Extended ID is transmitted after the SRR bit [1].

SRR Bit (Substitute Remote Request): The SRR bit is a recessive bit. It is transmitted in Extended Frames at the position of the RTR Bit in Standard Frames and so substitutes the RTR-Bit in the Standard Frame. Therefore, collisions of a Standard Frame and an Extended Frame, the Base ID of which is the same as the Standard Frame's Identifier, are resolved in such a way that the Standard Frame prevail the Extended Frame [1]. IDE bit is 'recessive' in extended format. Control field is different for standard format and extended format. Frames in Extended Format include the data length code and two reserved bits r1 and r0. The reserved bits have to be sent 'dominant', but receivers accept 'dominant' and 'recessive' bits in all combinations.

DLC-DATA LENGTH CODE (Standard Format as well as Extended Format) the number of bytes in the data field is indicated by the DLC [1]. This data length code is 4 bits wide and is transmitted within the control field.

DATA FIELD - It consists of the data to be transferred within a data frame [1]. It can contain from 0 to 8 bytes, which each contain 8 bits which are transferred MSB first.

CRC FIELD - It consists of CRC sequence and CRC delimiter. The frame check sequence is derived from a cyclic redundancy code best suited for frames with bit counts less than 127 bits (BCH Code) [1].In order to carry out the CRC calculation the polynomial to be divided is defined as the poly-nominal, the coefficients of which are given by the de stuffed bit stream consisting of start of frame, Arbitration field, Control field, Data field (if present) and, for the 15 lowest coefficients, by 0. CRC delimiter consists of a single 'recessive' bit.

ACK FIELD: It is two bits long and contains the ACK slot and the ACK delimiter. In the ACK field the transmitting station sends two 'recessive' bits [1]. A receiver which has received a valid message correctly, reports this to the transmitter by sending a 'dominant' bit during the ACK slot. All stations having received the matching CRC SEQUENCE report this within the ACK slot by super scribing the 'recessive' bit of the transmitter by a 'dominant' bit. ACK delimiter is the second bit of the ACK field and it has to be 'recessive' bit [1].

End of Frame: Each Date frame and Remote frame is delimited by a flag sequence consisting of seven 'recessive' bits.

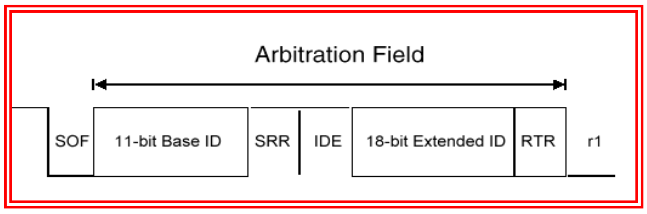


Figure 4 - 29 bit Identifier CAN 2.0B [1]

Remote Frame - Remote frame present in both standard format and extended format. Basically, used for requesting data frames. It consists of six different bit fields. It has Start of frame, Arbitration field, Control field, CRC field, ACK field, end of frame [1]. RTR bit of remote frame is 'recessive'. Identifier and data length is same as that of data frame but there is no Data field present in remote frame. Polarity of RTR bit is important. It specifies whether the transmitted bit is a data frame or it is a remote frame. Remote frame is not present in all controllers.

Error Frame - It is used to indicate an error to the entire network. Error frame consist of Error flag and Error delimiter. In order to terminate an error frame correctly, an 'error passive' node may need the bus to be 'bus idle' for at least 3 bit times [1]. The bus should not be loaded to 100%. Error flags are of two type i.e. Active error flag and Passive error flag. Active error flag is transmitted by active error node. It consists of six consecutive dominant bits. Passive error flag is transmitted by passive error node. It has six consecutive recessive bits. The Error delimiter consists of eight 'recessive' bits. After transmission of an error flag each station sends 'recessive' bits and monitors the bus until it detects a 'recessive' bit. After that it starts transmitting seven more 'recessive' bits [1].

Overload frame - Receiver can generate an overload frame when it is not ready to receive any frames. The overload frame consists of two fields one is overload flags and other one is overload delimiter [1]. There are two kinds of overload conditions which lead to transmission of overload flag. The internal condition of receiver requires delay of next data frame or remote frame. Detection of a dominant bit at first or second transmission required. If a CAN node samples a dominant bit at the last bit of an error delimiter or overload delimiter it will start transmitting an overload frame. Start of an overload frame due to condition is only allowed to be started at the first bit time of an expected intermission, whereas overload frames due to other conditions start one bit after detecting the 'dominant' bit. Overload flag consist of six dominant bits and overload delimiter contains eight recessive bits [1]. Overload delimiter is similar to that of error delimiter. Overload frame is not used by many controllers.

III. FLEXRAY

FlexRay is an automotive network communication protocol developed by FlexRay Consortium. In 2000 BMW, Daimler-Chrysler, Philips and Motorola joined forces to create a new automotive network scheme which would better suit the increasing demand for advanced applications in automation. It was widely known as FlexRay Consortium [7]. There is a great consensus in the automotive industry that FlexRay will become the new standard within vehicles and thus replacing the old CAN-bus. 2006 BMW X5 was the first vehicle built with FlexRay protocol but the first vehicle to fully utilize the FlexRay protocol was the BMW 7-series that had the FlexRay system introduced in 2008[7]. FlexRay consortium dispersed in 2009 and now it is an ISO standard.

FlexRay is designed to be faster and more reliable than CAN and TTP, but it is also more expensive [2]. FlexRay is very flexible and fault tolerance communication protocol. It supports high data rate i.e. 10Mbit/s. It supports star and party line bus topology and it has two independent data channels which are used for fault tolerance. FlexRay System or FlexRay Cluster consists of several electronic control units (ECU) each with a bus interface connected to one or two communication channels. The bus operates on a time cycle which is divided into two parts Static segment and the dynamic segment. The static segment is reallocated into slices for individual communication types. It gives stronger real-time guarantee than its predecessor CAN. The dynamic segment operates more like CAN, with nodes taking control of the bus as available, allowing event-triggered behavior [3].

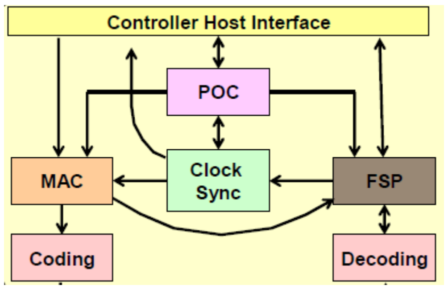


Figure 5 - Basic building block of FlexRay

Figure 5 shows basic building blocks of FlexRay. It shows CHI (controller host interface) POC (protocol operation control) Media access control, clock sync and frame/symbol processing. Controller host interface present between host and microcontroller. It handles configuration and status data. Protocol operation control reacts to host commands and protocol conditions. MAC schedules the bus and writes accesses. It assembles message header. Frame and symbol processing handles received messages and performs timing and error check.

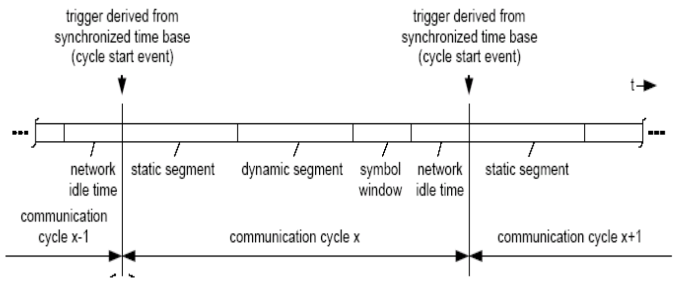


Figure 6 - Communication cycle

Communication Cycle - The above figure 6 shows communication cycle which consist of network idle time, static segment, dynamic segment, symbol window [2].

Static Segment - Communication cycle consist of static segment which has configurable no of static slots. Static slots have identical number of macroticks. Static segment is divided into static slots of equal duration. Slot duration defined as per cluster. Both channels have identical slot timings. Slot has slot ID [2]. For each node, specific time slot is given. 16 slots can be assignable per node. If there is no frame configured then the slot will remain empty.

Dynamic Segment - This segment is optional. It contains configurable number of minislots. All minislots has same no of macroticks. At the start of dynamic segment, all slot counters set to zero. The node can send a message when it has message with ID matching with the slot counter. Once the message sent, all nodes detect a new dynamic slot [2]. If there is no message transmitted all the slot counters increase again. Low value ID gives high priority of being transmitted. Dynamic slot is variable in size.

Symbol window - This is optional segment. It contains macroticks. A single symbol can be sent within a symbol window. Arbitration among different senders is not provided by protocol for the symbol window [2]. If arbitration required for symbol window then it can be achieved with the help of higher level protocol.

Network idle time - Communication cycle contains network idle time [2]. Node calculates and applies clock correction terms during this network idle time.

Frame Format – Following figure 7 shows the FlexRay Frame format. Reserved bit: For a transmitting node, reserved bit should set to logical '0' and for a receiving node no need to set reserved bit.

Payload preamble indicator: In the static segment frame, this bit indicates the presence of a NM vector at the beginning of the payload and in the dynamic segment frame; this bit indicates the presence of a message ID at the beginning of the payload [2].

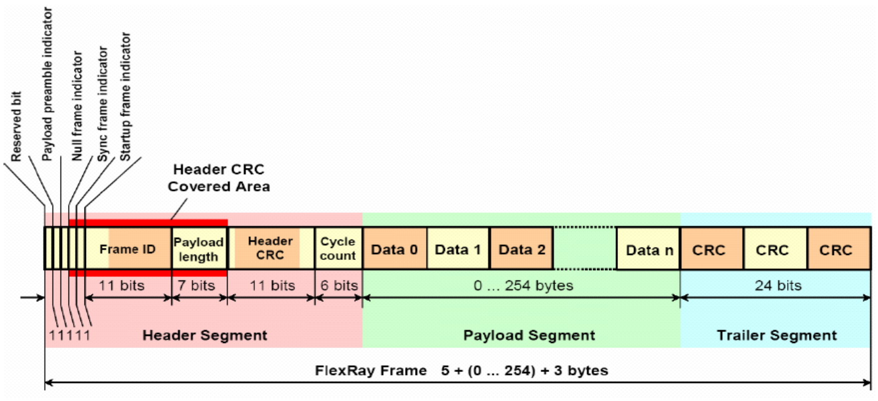


Figure 7 - FlexRay Frame format [2]

Null frame indicator: If the payload segment contains no valid data then this bit is set to zero otherwise set to one if the payload segment contains data.

Sync frame indicator: This bit is set to zero then no receiving node shall consider the frame for synchronization [2]. If this bit is set to one then all receiving nodes shall utilize the frame for synchronization if it meets other acceptance criteria.

Startup frame indicator - The startup frame indicator set to one in the sync frames of coldstart nodes [2]. Every coldstart node can transmit exactly one frame per communication cycle and channel with the startup frame indicator set to one.

Frame ID - A frame ID used once on each channel in a communication cycle [2]. It ranges from 1 to 2047.

Payload Length - It is 7 bits. For static segment payload length is fixed and it is different for dynamic segment [2].

Header CRC - It has 11 bits. This is calculated over sync frame indicator, startup frame indicator, frame ID and payload length. For a transmitted frame CC does not calculate header CRC [2]. Header CRC of transmitted frames are calculated offline and then it gives to CC by means of configuration. CCC calculate header CRC of a received frame to check whether CRC is correct or not.

Payload segment (0 - 254 bytes): First 0 to 12 bytes of the payload segment may optionally be used as NM vector in static segment frame transmission [2]. In dynamic segment frame transmission, first two bytes of payload segment may optionally use as message ID field.

NM Vector (optional) - The length of NM Vector shall be configured to the same value for all nodes in a cluster [2]. At the transmitting node the NMVector is written by the host as application data.

Message ID - It is 16 bit and optional. In dynamic segment frame, first two bytes of payload is used as receiver filterable data [2]. Message ID is written at transmitting node with the help of host application data.

FlexRay trailer segment - It contains 24 bit CRC for the frame. Frame CRC is computed over header segment and payload segment of frame. The initialization vectors channel A --- 0xFEDCBA and channel B --- 0xABCDEF.

IV. LIN - LOCAL INTERCONNECT NETWORK

Local interconnect network (LIN) is widely known as low cost automotive networks which complements the existing automotive multiplex networks [4]. It is serial communication protocol. LIN will be the most important factor for the implementation of a vehicle network in order to gain further quality enhancement and cost reduction of vehicles. The standardization will reduce existing low-end multiplex solutions and will cut the cost of development, production, service, and logistics in vehicle electronics. This network is based on master –slave architecture [5]. One network node is chosen to control all communication and this is called as LIN master. LIN master performs role of bus arbiter. LIN master transmits message in the form of token. A token then understood as a request by LIN slave. Slave can respond to token in three ways i.e. it can send data, receive data or ignore data. Token and data together forms a LIN message. Up to 64 messages may be defined in local interconnect network. Each message is available to receive by any node or even by master if it has slave task. Therefore, LIN is called centrally controlled message distribution system.

Main properties of the LIN are:

- Single-master / multiple-slave concept
- Low cost silicon implementation based on common UART/SCI interface hardware, an equivalent in software, or as pure state machine.
- Self-synchronization without quartz or ceramics resonator in the slave nodes
- Deterministic signal transmission
- Low cost single-wire implementation
- Speed up to 20kbit/s.

LIN Frame format – The below figure 8 shows LIN frame format. The frame format is divided into two parts i.e. message header and message response. Message header consists of break, sync and identifier while message response consists of data and checksum [6]. The token is referred as message header. Message header is always sent by master task. Master header is used for synchronization. Message response is sent by slave task.

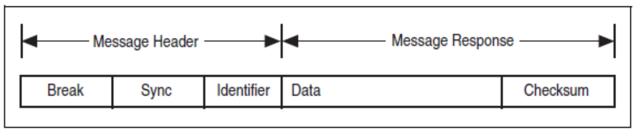


Figure 8 - Frame format [6]

Break: LIN frame starts with break, which has 13 dominant bits (nominal) followed by a break delimiter which is of one bit (nominal) recessive [6]. This is called as a start-of-frame notice to all nodes on the bus.

Sync - The sync field is the second field transmitted by the master task in the header. Sync is defined as the character x55. The sync field allows slave devices that perform automatic baud rate detection to measure the period of the baud rate and adjust their internal baud rates to synchronize with the bus [6].

ID: This is the final field which is transmitted in the header with the help of master task. This field gives identification for each message on network. It also determines which nodes respond to each transmission. A slave task usually listens for frames ID and verifies their respective parties and determines if they are publisher or subscriber [6]. LIN provides 64 IDs out of that 0 to 59 IDs used for signal carrying frames. ID 60 and 61 used for diagnostic data, 62 is used for user-defined extensions while 63 are kept reserved for future protocol enhancements.

Data Bytes: Data bytes field usually transmitted by the slave task in the response [6]. This field consists from one to eight bytes of payload data bytes.

Checksum: This is transmitted by slave task in response. LIN bus defines use of one of two checksum algorithms which is used to calculate value in the eight-bit checksum field [6]. Classic checksum is calculated by summing the data bytes alone, on the other hand enhanced checksum is calculated by summing the data bytes and the protected ID.

V. VEHICLE ARCHITECTURE

Following figure 9 shows the example for centralized gateway architecture. Advanced vehicle systems require deterministic (time triggered) behavior, which can only be fulfilled by certain protocols (e.g. FlexRay). Other vehicle systems require the communication of events that should not be lost and are supported by CAN-busses. Small sensors and Low priority communication can be handled by Sub system busses (e.g. LIN). In case a gateway has to deal with a high demand regarding data forwarding and possibly also with different types of protocols (e.g. in case of a centralized gateway topology) the performance of the gateway can be compromised in certain circumstances if the system is not properly designed. Therefore, and mainly due to the interaction between the time-triggered and event-triggered buses at low-level (protocol stack and the routing and buffering of data), specific mechanisms should be envisaged to manage this (for instance, using dedicated hardware resources).

In high-end centralized topologies, due to its own intrinsic structure, there is only one big gateway connecting all domains in the architecture. Therefore, it is expected that the gateway be highly loaded, applications and communication services.

The routing and switching of messages is the key concern of the gateway. For high-end vehicle system, a huge number of messages will be transferred, with strict real time requirements. In this scenario it is highly recommended to have strong hardware support, in order to guarantee the gateway functions under all load conditions. In this case, there is only one gateway connecting all domains. Therefore, it is expected that the gateway be highly loaded with applications and communication services.

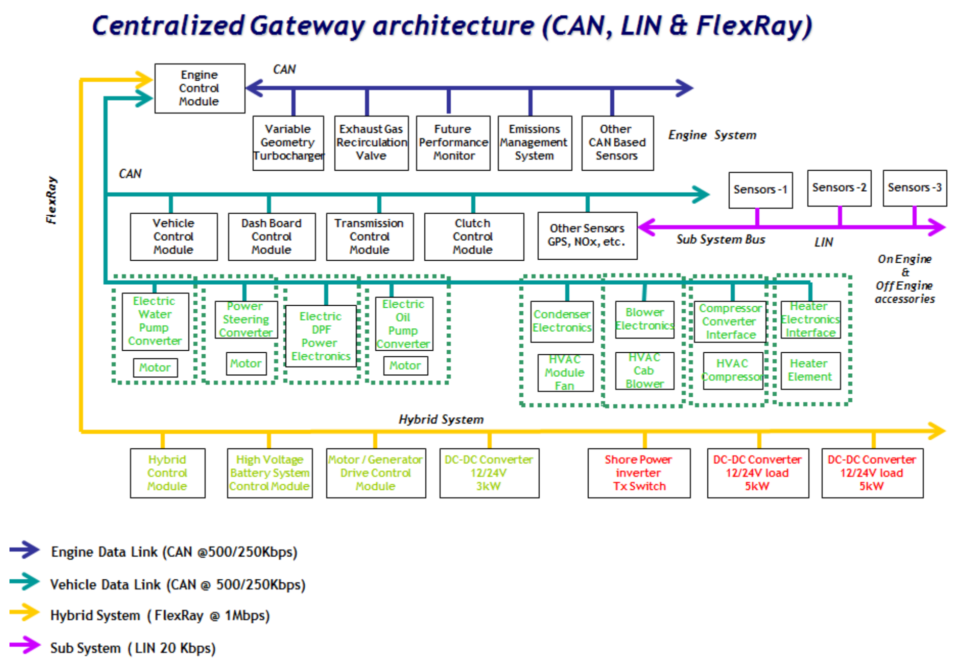

Centralized Gateway architecture (CAN, LIN & FlexRay)
CAN
Engine Control Module
Variable Geometry Turbocharger
Exhaust Gas Recirculation Valve
Future Performance Monitor
Emissions Management System
Other CAN Based Sensors
Engine System
CAN
FlexRay
Sensors -1
Sensors -2
Sensors -3
Vehicle Control Module
Dash Board Control Module
Transmission Control Module
Clutch Control Module
Other Sensors GPS, NOx, etc.
Sub System Bus
LIN
On Engine & Off Engine accessories
Electric Water Pump Converter
Motor
Power Steering Converter
Motor
Electric DPF Power Electronics
Electric Oil Pump Converter
Motor
Condenser Electronics
HVAC Module Fan
Blower Electronics
HVAC Cab Blower
Compressor Converter Interface
HVAC Compressor
Heater Electronics Interface
Heater Element
Hybrid System
Hybrid Control Module
High Voltage Battery System Control Module
Motor / Generator Drive Control Module
DC-DC Converter 12/24V 3kW
Shore Power inverter Tx Switch
DC-DC Converter 12/24V load 5kW
DC-DC Converter 12/24V load 5kW
Engine Data Link (CAN @500/250Kbps)
Vehicle Data Link (CAN @ 500/250Kbps)
Hybrid System (FlexRay @ 1Mbps)
Sub System (LIN 20 Kbps)

Figure 9 - In high-end centralized topologies

Note – All ECMs names are for examples. This Power system and vehicle topology may not be the real implementation of any OEM.

VI. Advantages and Applicability

- Very flexible and can be merged with any existing architecture
- Complexity - ECM is connected to 3 networks it means Routing software is complex
- Guaranteed data consistency - FlexRay - Cycle-based, use of TDMA and Minislotting
- Good Latency - FlexRay – by distributed clock synchronization
- Scalable - Due to higher bandwidth, adding of node is easy
- Bandwidth - 10 MBPS
- Deterministic FlexRay – It use static segments
- Fault Tolerance - FlexRay - TDMA slot scheme protected by node local bus guardians
- Error Detection and Handling - FlexRay - 24 bit CRC

Disadvantages can be as mention followed but these are very generic even in existing systems/architecture

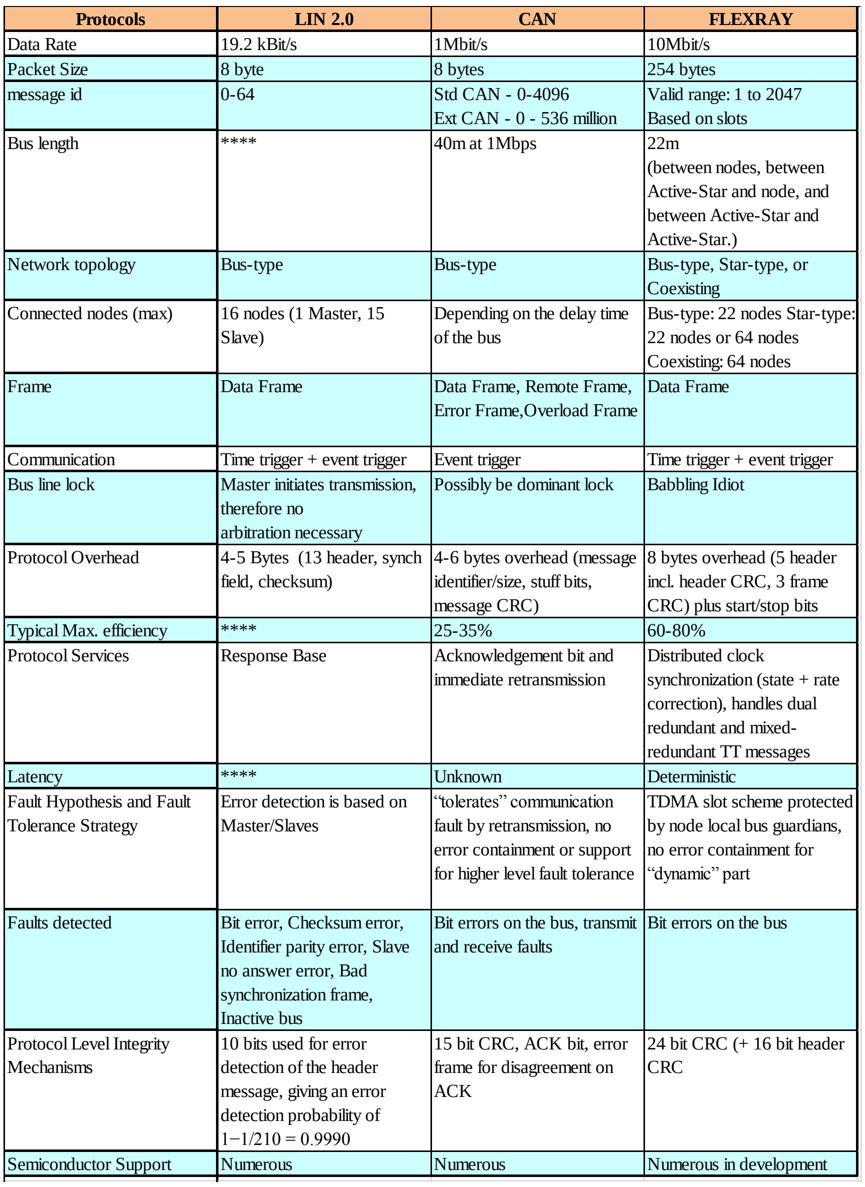
- Software Overhead on Gateway node (ECM)
- High Cost - Manufacturing & Maintenance cost is High

As proposed in above vehicle architecture, FlexRay can be used and backbone network and Sub-nodes will be over CAN. LIN will be used with very limited applications like cosmetic sensors of vehicle. FlexRay and CAN will be connection between major systems Like Active Safety Systems, Passive Safety System, Power Train, Body Control System etc. High speed busses are in demand today; but CAN has its own important in automotive. All OEMs want deterministic behavior of CAN and Speed of FlexRay therefor even FlexCAN is coming to the market. FlexCAN is an embedded network architecture that extends Controller

Area Network (CAN) [9]. It was inspired by FlexRay and the need to provide more deterministic behavior over the CAN network. Its focus is on redundancy at the hardware level, and time-based prioritized communication at the protocol level [9].

VII. VEHICULAR PROTOCOL COMPARISON

The following figure 10 shows the comparison between LIN, CAN and FlexRay protocols. As we refer table, FlexRay 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 and LIN supports 64 bytes of data and 29 bits of header Information. This means that FlexRay may send larger messages with less overhead then CAN, but for smaller messages the overhead is about the same. LIN is very much slower than CAN and FlexRay, LIN is not at all suitable for vehicle critical communication application.



Protocols	LIN 2.0	CAN	FLEXRAY
Data Rate	19.2 kBit/s	1Mbit/s	10Mbit/s
Packet Size	8 byte	8 bytes	254 bytes
message id	0-64	Std CAN - 0-4096 Ext CAN - 0 - 536 million	Valid range: 1 to 2047 Based on slots
Bus length	****	40m at 1Mbps	22m (between nodes, between Active-Star and node, and between Active-Star and Active-Star.)
Network topology	Bus-type	Bus-type	Bus-type, Star-type, or Coexisting
Connected nodes (max)	16 nodes (1 Master, 15 Slave)	Depending on the delay time of the bus	Bus-type: 22 nodes Star-type: 22 nodes or 64 nodes Coexisting: 64 nodes
Frame	Data Frame	Data Frame, Remote Frame, Error Frame,Overload Frame	Data Frame
Communication	Time trigger + event trigger	Event trigger	Time trigger + event trigger
Bus line lock	Master initiates transmission, therefore no arbitration necessary	Possibly be dominant lock	Babbling Idiot
Protocol Overhead	4-5 Bytes (13 header, synch field, checksum)	4-6 bytes overhead (message identifier/size, stuff bits, message CRC)	8 bytes overhead (5 header incl. header CRC, 3 frame CRC) plus start/stop bits
Typical Max. efficiency	****	25-35%	60-80%
Protocol Services	Response Base	Acknowledgement bit and immediate retransmission	Distributed clock synchronization (state + rate correction), handles dual redundant and mixed-redundant TT messages
Latency	****	Unknown	Deterministic
Fault Hypothesis and Fault Tolerance Strategy	Error detection is based on Master/Slaves	"tolerates" communication fault by retransmission, no error containment or support for higher level fault tolerance	TDMA slot scheme protected by node local bus guardians, no error containment for "dynamic" part
Faults detected	Bit error, Checksum error, Identifier parity error, Slave no answer error, Bad synchronization frame, Inactive bus	Bit errors on the bus, transmit and receive faults	Bit errors on the bus
Protocol Level Integrity Mechanisms	10 bits used for error detection of the header message, giving an error detection probability of 1−1/210 = 0.9990	15 bit CRC, ACK bit, error frame for disagreement on ACK	24 bit CRC (+ 16 bit header CRC
Semiconductor Support	Numerous	Numerous	Numerous in development

Figure 10 – Comparison between LIN, CAN and FlexRay

This paper provides all fundamentals of vehicle Network and its protocols.

Note: **** - Some comparison fields are beyond scope of Protocol Specification and can be achieved by the real-time work experience.

VIII. CONCLUSION

In-vehicular networking protocols like CAN, FlexRay, and LIN are described in detail in this paper. In-vehicular networking protocols are very important and gaining popularity due to its complexity. Modern vehicles contain all the complex electronic and mechanical systems. In-vehicle network communications provide efficient methods of communication between electronic components. CAN, FlexRay, and LIN are vehicular networking protocols used in most of the modern vehicles for ECU communications. CAN is popular in medium speed applications (250 kbps and 500 kbps), and FlexRay (10 mbps) is used in high speed applications. We need a less expensive, high speed vehicle protocol with advantages (diagnostics and error correction mechanism) from CAN protocol. Hence, TTCAN and FTCAN will become prominent in the near future.

REFERENCES

[1] CAN Specification 2.0, part B http://www.cancia.org/fileadmin/cia/specifications/CAN20B.pdf/

[2] http://en.wikipedia.org/wiki/FlexRay

[3] How FlexRay works - http://www.cancia.org/fileadmin/cia/specifications/CAN20B.pdf/

[4] LIN 1.3 Specification http://lniv.fe.uni-lj.si/courses/ndes/LIN_Spec_1_3.pdf/

[5] http://ece.eng.wayne.edu/~smahmud/ECECourses/ECE5620/Notes/LIN_Protocol.pdf/

[6] Introduction to the Local Interconnect Network (LIN) Bus overview by National Instruments http://www.ni.com/white-paper/9733/en/#toc2/

[7] Andreas Forsberg, Johan Hedberg, Comparison of FlexRay and CAN-bus for Real Time Communication.

[8] Juan R. Pimentel Kettering University, Jose A. Fonseca University of Aveiro, FlexCAN: A Flexible Architecture for Highly Dependable Embedded applications.