Chapter XV
In–Vehicle Network Architecture for the Next–Generation Vehicles

Syed Masud Mahmud
Wayne State University, USA

ABSTRACT

New types of communication networks will be necessary to meet various consumer and regulatory demands as well as satisfy requirements of safety and fuel efficiency. Various functionalities of vehicles will require various types of communication networks and networking protocols. For example, drive-by-wire and active safety features will require fault tolerant networks with time-triggered protocols to guarantee deterministic latencies. Multimedia systems will require high-bandwidth networks for video transfer, and body electronics need low-bandwidth networks to keep the cost down. As the size and complexity of the network grows, the ease of integration, maintenance and troubleshooting has become a major challenge. To facilitate integration and troubleshooting of various nodes and networks, it would be desirable that networks of future vehicles should be partitioned, and the partitions should be interconnected by a hierarchical or multi-layer physical network. This book chapter describes a number of ways using which the networks of future vehicles could be designed and implemented in a cost-effective manner. The book chapter also shows how simulation models can be developed to evaluate the performance of various types of in-vehicle network topologies and select the most appropriate topology for given requirements and specifications.
INTRODUCTION

Early vehicles used dedicated point-to-point connections for inter-module communications. As the number of modules and features increased in vehicles, the wiring system became bulky, complex, expensive and difficult to install and maintain. As a result, over the years, evolution of in-vehicle communications via a serial bus took place. A serial bus can replace all the dedicated point-to-point wiring between modules. Thus, it significantly reduces wiring complexity as well as weight of the vehicle making the vehicle more fuel efficient. A serial bus system is also scalable, meaning that more modules can be connected to the bus at any time without requiring any changes at other modules. Since different functions of a vehicle need different data rates, such as power-train needs higher data rate than body electronics, the use of a single serial bus for the entire vehicle may not be the best choice to design the in-vehicle communication network. As high-speed electronic components are more expensive than low-speed electronic components, it would be desirable and cost effective to partition the serial bus into two buses: a high-speed serial bus and a low-speed serial bus. The high-speed bus can be used for power-train and the low-speed bus can be used for body electronics. This is the reason why currently most vehicles have at least two serial buses: one for power-train and the other for body electronics, as shown in Figure 1. Since there are some mutually required data between the two partitions of the in-vehicle network, the partitions are interconnected by a gateway device as shown in Figure 1. The two partitions need not use the same type of bus though they can. Since the LIN (Local Interconnection Network) bus is less expensive and LIN protocol (Uplap, 2004) is less complex than the corresponding CAN (Controller Area Network) bus and CAN protocol (Motorola, 1998), nowadays many vehicles are considering using a low-speed LIN bus for body electronics and a high-speed CAN bus for power-train. The gateway device which interconnects a CAN bus and a LIN bus converts messages from one protocol to another protocol when the messages need to go from one partition to another partition.

As the government and consumer demands are increasing for many features such as safety, fuel efficiency, comfort, navigation, entertainment, telematics, multimedia and many more, future vehicles will have many more partitions. Each partition will take care of a particular type of application. All the partitions will need

Figure 1. An in-vehicle networking system with two partitions
to be interconnected for exchanging mutually required information as well as for diagnostic purposes. Like early vehicles, dedicated point-to-point connections among various partitions will increase the complexity of connectivity, as shown in Figure 2. Figure 2 shows an in-vehicle networking system with four partitions. Each has its own bus system. Each pair of partitions is interconnected by a gateway device. For example, Partition-1 and Partition-2 are connected by the gateway device G12. Similarly, Partition-1 and Partition-4 are connected by the gateway device G14. The number of gateway devices increases exponentially with the increase in the number of partitions. For N partitions, the number of gateway devices required for dedicated connections among the various partitions is $\frac{N!}{2 \times (N - 2)!}$. Thus, dedicated connections among the partitions will be very complex and cost prohibitive.

As the number of partitions in a vehicle is going to increase, the automakers must come up with less complex and less expensive techniques for interconnecting the partitions.

The main objective of this book chapter is to show various cost-effective interconnection mechanisms among the partitions of an in-vehicle networking system. The advantages, disadvantages and the feasibility of various interconnection mechanisms using the current technology are also explained in this book chapter.

**BACKGROUND**

There is little doubt that the automobile is the most revolutionary invention in the history of transportation since the wheel. Over the last 100 years, automobiles have gone through numerous phases of improvements. At the dawn of the twentieth century, transportation of people and goods in timely and cost effective manner was the main motivation behind developing automobiles. Safety, fuel efficiency, comfort, entertainment, greenhouse effect, etc. were not the major issues for those early automobiles. Thus, the first generation automobiles were mostly mechanical devices, and they had very few electrical and electronic components. As a result, interconnecting those few electrical and electronic components was not a major problem. After that, various new features were being added to automobiles out of necessity. Initially, the necessity of safety and fuel efficiency was the main driving force behind improvements of automobiles.
Since the 1960’s, there have been significant improvements in vehicle safety. The introduction of safety features such as seat belts, air bags, crash zone, lighting system and new vehicle structures has dramatically reduced the rate of crashes, injuries and fatalities. However, in spite of these impressive improvements, according to U.S. Department of Transportation there were nearly 45,000 fatalities in transportation accidents in the United States in 2004, of which 95 percent involved highway motor vehicles (USDOT, 2006). By continuing with the passive safety technologies, it is difficult to achieve significant further gains in reducing crash costs. Thus, engineers and researchers have started developing active safety technologies to further reduce the costs involved with vehicle crashes. As a result, more electrical and electronic components are being added to vehicles.

The energy crisis pushed the automotive industries to make their vehicles more fuel efficient. Fuel efficiency requires efficient burning of fuel inside engine cylinders which eventually requires optimal air-fuel ratio, optimal spark timing and optimal control of valves. As a result, carburetors were replaced by computer controlled electronic fuel injection systems; spark timing is controlled by computers; mechanical valves are being replaced by electrical valves; and many more, which eventually added more electrical and electronic components in vehicles.

As the electrical and electronic components grew in vehicles, the dedicated point-to-point interconnection among these components became very bulky, complex and expensive. As a result, the concept of multiplexing or in-vehicle networking evolved since early 1980’s. In fact, one of the first automotive papers that addressed the need for vehicle multiplexing was published as early as 1976 (Bell, 1976). Multiplexing uses a serial bus, and all electrical and electronic components are connected to this bus via microcontrollers. The microcontrollers take active role in exchanging information among various electrical and electronic components of the vehicles. The microcontrollers send information using messages. The messages are created by transmitting microcontrollers and decoded by receiving microcontrollers based on a particular communication protocol used by vehicles. In the 1980’s, automotive companies were developing their own protocols and there was no standardization in networking protocols among various companies. As a result, jobs of chip manufacturers, suppliers and mechanics were becoming difficult because chip manufacturers had to make different chips for different protocols, suppliers had to make different parts to make them compatible with different protocols and the mechanics had to use different diagnostic computers for different protocols. Recently almost all automotive companies both in US and Europe have started using Controller Area Network (CAN) protocol for in-vehicle networking. Thus, nowadays almost every high end vehicle has at least one CAN network.

The Society of Automotive Engineers (SAE) has defined three basic categories of in-vehicle networks based on network speed and functions: Class A, Class B and Class C networks.

- Class A is a low-speed network with a bit rate of 10 Kbits/sec or less. Class A network is to be used for convenience features such as entertainment, audio, trip computer, etc. Class A functions require inexpensive and low-speed communication, and typically utilize generic UARTs (Universal Asynchronous Receiver/Transmitter).
- Class B is a medium-speed network. Data rate for Class B network is between 10 Kbits/sec and 125 Kbits/sec. The intended application for Class B network is general information transfer such as instrument cluster, vehicle speed, legislated emissions data, etc. In the US, the SAE adopted J1850 as the standard protocol for Class B networks. SAE J1850 has two basic versions. The first is a 10.4 Kbits/sec VPW (Variable
Pulse Width) type which uses a single bus wire. The second is a 41.6 Kbits/sec PWM (Pulse Width Modulation) type which uses a two-wire differential bus.

- Class C is a high-speed network, and it is to be used for real-time control applications such as power-train, vehicle dynamics, anti-lock brakes, etc. The data rate for Class C network is 125 Kbits/sec to 1 Mbits/sec. CAN protocol has been selected for Class C applications.

Continuous regulatory demands for fuel efficiency, safety and emission control are forcing the automakers to implement advanced safety and engine management systems. These advanced systems will require new types of networks and networking protocols. The demands for improved fuel efficiency require drive-by-wire systems that eliminate camshafts, power-sapping belt drives and pumps, and a great deal of unnecessary weight. The goal of drive-by-wire is to replace as many mechanical systems as possible by equivalent electronic and electrical systems. Electric brakes will replace hydraulics, increasing safety, decreasing weight, lowering operating cost and eliminating the use of environmentally hazardous fluids. Steering columns will disappear and will be replaced by electric steering, improving driver safety and making it easy to provide left-hand and right-hand drive models. Similarly, as the demand for other new features such as telematics, entertainment, multimedia, pre-crash warning, driver assistance systems, remote diagnostic, software update, etc. increases, the complexity of in-vehicle communication networks increases as well. Depending upon the new functionality of vehicle electronic modules, different sets of modules will require different types of new networks. For example, drive-by-wire and active collision avoidance systems need fault tolerant networks with time-triggered protocols such as TTP/C (TTTech, 2002), TTCAN (Fuehrer, 2001) and FlexRay (FlexRay, 2004), to guarantee critical services with deterministic latencies. Airbag systems need a protocol that can react very quickly. For airbag systems there are various protocols, such as Safe-by-Wire (Lupini, 2004), Bosch Siemens Temic (BST), DSI and Byteflight (Lupini, 2004), which are being investigated. Multimedia systems need high-bandwidth networks such as MOST and IntelliBus (Lupini, 2004; Firlit, 2004), to transfer video files. Wireless personal area networks with voice activated control are necessary for convenience such as turning on/off lights, wipers, radio, heat and air conditioner without taking hands off the steering wheel (Mahmud, 2006). Bluetooth or Wi-Fi protocol could be used for wireless personal area network. Telematic systems need technologies such as Wi-Fi or Ultrawideband (UWB). Thus, it is apparent that future vehicles will have various types of networks with many partitions. Interconnecting these partitions is necessary for distributed control across various partitions and ease of diagnosis. Since automotive industry is very cost sensitive, interconnecting these partitions will be a challenging task especially when the number of partitions is large. Major studies and in-depth analysis are necessary to determine optimal topology for interconnecting all the in-vehicle network partitions, to figure out optimal bandwidth required from different types of network partitions for keeping latencies under certain bounds, and to select appropriate communication protocols for different network partitions. Various topologies for interconnecting different in-vehicle network partitions are investigated and presented in the next section of this book chapter.

**INTERCONNECTION OF NETWORK PARTITIONS**

**Issues, Controversies and Problems**

Multiplexing or in-vehicle networking evolved due to the need for reducing wiring harness, ease of integration, ease of diagnosis and cost
savings. Most of the today’s vehicles have only two network partitions as mentioned earlier in this book chapter and also shown in Figure 1. Interconnecting two partitions is a trivial problem. We need only one gateway device to connect the two partitions. The gateway device must support protocols of both partitions. The gateway device will do protocol conversions when messages need to go from one partition to the other partition. Nowadays, microcontrollers are available that support various types of interfaces. For example, PIC18F4685 microcontroller supports SPI, I²C, RS-485, RS-232, LIN and CAN interfaces. Similarly, PIC18F97160 microcontroller supports SPI, I²C, RS-485, RS-232, LIN and Ethernet interfaces.

Thus, a microcontroller that supports both interfaces of the two partitions, shown in Figure 1, can be used as the gateway device between the two partitions.

As the number of partitions grows, dedicated point-to-point connections among various partitions will require many gateway devices, and beyond a certain point it would be cumbersome and cost prohibitive. One of the challenging issues will be the selection of an appropriate topology to interconnect various partitions of the in-vehicle network.

**Selection of a Topology**

In this subsection, a number of multi-level bus-based network topologies are discussed and suggested for future in-vehicle networks. A hierarchical bus-based network provides a cost-effective interconnection solution technique for partition-based systems (Mahmud, 1994). Since interconnection using a hierarchical bus requires less number of gateway devices, one logical approach for interconnecting four partitions of a vehicular network will be the use of a second level bus or an Level 2 bus (L2 bus), as shown in Figure 3 (Mahmud, 2005). Figure 3 shows that each partition has its own bus called the Level 1 bus or L1 bus. Different partitions have different types of L1 buses. For example, one partition may have a CAN bus as its L1 bus, and another partition may have a LIN bus as its L1 bus. An L1 bus of a partition is connected to the L2 bus by a gateway device. For example, the L1 bus of Partition-1 is connected to the L2 bus using the gateway device G1, as shown in Figure 3. Thus, the number of gateway devices required to connect N partitions to the L2 bus is N. Whereas for dedicated point-to-point connections among various partitions, the number of required gateway devices is \( \frac{N!}{2 \times (N - 2)!} \).

Hence, the complexity of connections shown in Figure 3 is much less than that shown in Figure 2. Figure 3 shows an additional unit called the Diagnostic Gateway. Dealer will connect its diagnostic computer (diagnostic module) to the vehicle via this diagnostic gateway. Thus, for N partitions in a system, total number of connections to the L2 bus including the diagnostic gateway is \((N + 1)\). For a dedicated point-to-point connection system, there has to be a gateway device between the diagnostic module and each partition. Thus for a system with N partitions and a diagnostic module, dedicated point-to-point connections will require \( \frac{(N + 1)!}{2 \times (N - 1)!} \) gateway devices. It is clear
that a two-level bus system, which contains an L1 bus for each partition and only one L2 bus for the entire system, is a cost-effective way of interconnecting all the partitions. The bit rate of the L2 bus should be at least equal to the total bit rate necessary for inter-partition communications for all partitions. Normally not too much inter-partition data needs to be exchanged among various partitions. Thus, a Class B or even a Class A bus may be good enough. However, the final selection decision must be made after an extensive analysis and simulation of the system.

**Required Bandwidth for the L2 bus:** An approximate analysis of the required bandwidth for the L2 bus can be determined as follows. Assume that the system has \( N \) partitions. Let \( d_i \) be the traffic density (bits/sec) on the L1 bus of Partition-\( i \). Nowadays, most messages for a vehicular system are periodic. We would like to make it clear that periodic messages are not to be confused with time-triggered messages. Time slots are reserved for time-triggered messages but not for periodic messages. Periodic messages need to go through an arbitration process to get the bus. If all periodic messages and their periods for a particular partition are known, then the value of \( d_i \) can be estimated. Let \( M_i \) be the number of messages in Partition-\( i \), and \( l_{i,j} \) be the length of Message-\( j \) of Partition-\( i \) with period \( t_{i,j} \). The value of \( d_i \) can then be written as:

\[
    d_i = \sum_{j=1}^{M_i} \frac{l_{i,j}}{t_{i,j}}
\]

(1)

Let \( f_i \) be the fraction of Partition-\( i \)'s messages that will go to the L2 bus. Then the total traffic density on the L2 bus is

\[
    d_L = \sum_{i=1}^{N} d_i f_i = \sum_{i=1}^{N} \left( \sum_{j=1}^{M_i} \frac{l_{i,j}}{t_{i,j}} \right) f_i
\]

(2)

As the traffic density on a bus increases, the queuing delay for the messages to get the bus increases as well. If \( u \) is the utilization of a particular bus, then the probability that a message will get the bus during its first trial is \( (1 - u) \). The probability that the message will not get the bus during its first trial but will get the bus during its second trial is \( u(1 - u) \). In general, it can be

\[\text{Figure 3. An in-vehicle network with four L1 buses and one L2 bus}\]
shown that the probability that the message will not get the bus during the first \((i - 1)\) trials but it will get the bus during the \(i\)th trial is \(u^{i-1}(1 - u)\). Hence, the average number of trials necessary for the message to get the bus is \(\sum_{i=1}^{\infty} iu^{i-1}(1 - u) = \frac{1}{1-u}\). The average message transmission delay through the bus, including the queuing delay for the bus, is 
\[
\frac{l_{avg}}{(1-u)BW}\]. Where, \(l_{avg}\) is the average length of a message and \(BW\) is the bandwidth of the bus. For example, if \(u = 30\%\), \(l_{avg} = 120\) bits and \(BW = 500\) kbits/sec, then the average message transmission delay including the queuing delay for the bus is 343 \(\mu\)sec. The average queuing delay is 
\[
\frac{ul_{avg}}{(1-u)BW}\]
and it increases very rapidly with the increase in bus utilization \(u\). For example, for \(l_{avg} = 120\) bits and \(BW = 500\) kbits/sec, the average queuing delay is 103 \(\mu\)sec for 30\% bus utilization, and it is 360 \(\mu\)sec for 60\% bus utilization.

In vehicular networks, normally the bus utilization is kept around 30\% or less to reduce contention on the bus so that a message is not significantly delayed to get the bus. Let \(u_{L2}\) be the maximum allowable steady-state utilization of the L2 bus. The required bandwidth for the L2 bus can then be expressed as

\[
BW_{L2} = \frac{d_{L2}}{u_{L2}} = \frac{\sum_{i=1}^{\infty} \left( \sum_{j=1}^{M_i} l_{i,j} \right) f_i}{u_{L2}}
\]  

(3)

Since, for a given system, the values of \(l_{i,j}\) and \(t_{i,j}\) are accurately known, and the value of \(f_i\) can also be accurately determined by investigating all the messages of a partition, the value of the required bandwidth, \(BW_{L2}\) for the L2 bus can be determined for a given value of \(u_{L2}\). However, the actual selection of the L2 bus needs to be made after extensive simulation and analysis of the entire system.

**End-to-End Message Latency for a 2-Level Bus System:** In a 2-level bus system, an inter-partition message will first go through the L1 bus of the source partition, then it will go through the L2 bus, and finally it will go through the L1 bus of the destination partition. Thus, various bus delays and protocol conversion delays will contribute to the total end-to-end message latency from the source node to the destination node. For a particular message, the end-to-end latency due to network components, such as buses and gateway devices, can be expressed as:

\[
T = T_{SL1} + T_{SG} + T_{L2} + T_{DG} + T_{DL1}
\]  

(4)

where, \(T_{SL1}\) is the delay due to the L1 bus of the source partition; \(T_{SG}\) is time required by the source gateway device (the gateway device which connects the source L1 bus to the L2 bus) to convert messages from one protocol to another protocol; \(T_{L2}\) is the delay due to the L2 bus; \(T_{DG}\) is time required by the destination gateway device (the gateway device which connects the destination L1 bus to the L2 bus) to convert messages from one protocol to another protocol; and \(T_{DL1}\) is the delay due to the L1 bus of the destination partition. \(T_{SG}\) and \(T_{DG}\) can also be called as protocol conversion time. Substituting the values of bus delays in Equation (4), the end-to-end message delay can also be expressed as:

(see Box 1.)

\[
T = \frac{l_{avg-SL1}}{(1-u_{SL1})BW_{SL1}} + \frac{l_{avg-L2}}{(1-u_{L2})BW_{L2}} + \frac{l_{avg-DL1}}{(1-u_{DL1})BW_{DL1}}
\]  

(5)

Box 1.
where, $l_{avg \_ SL1}$, $u_{SL1}$ and $BW_{SL1}$ respectively indicate the average message length, utilization and bandwidth of the source L1 bus; $l_{avg \_ L2}$, $u_{L2}$ and $BW_{L2}$ respectively indicate the average message length, utilization and bandwidth of the L2 bus; and $l_{avg \_ DL1}$, $u_{DL1}$ and $BW_{DL1}$ respectively indicate the average message length, utilization and bandwidth of the destination L1 bus. The message delay basically has two components: the sum of the delays due to all buses and the sum of the protocol conversion times for the gateway devices. Thus, the message delay for a two-level bus system can be expressed as:

$$T = \sum_{over-two-L1-buses} \left( \frac{l_{avg \_ bus}}{(1 - u_{bus})BW_{bus}} \right) + \sum_{over-two-gateway \ devices} T_{Protocol\ Conversion} \quad (6)$$

Box 2.

Future vehicles many require many partitions, one for each subsystem of the vehicle. For example, the airbag system will need a partition for itself because there could be as many as 64 nodes for the airbag system (Lupini, 2004). In future, as the number of partitions will grow, various inter-partition connection topologies should be investigated in order to select an appropriate topology for interconnecting partitions. The remaining parts of this book chapter present various topologies for interconnecting partitions.
and L2 buses. The system shown in Figure 4 also has a diagnostic gateway (DG) connected to the L2 bus. For the design shown in Figure 4, total number of connections to the L2 bus including the diagnostic gateway is $\lceil N/2 \rceil + 1$, where $N$ is the number of partitions and the notation $\lceil n \rceil$ means ceiling of $n$. Moreover, for the same number of partitions in the system, the data rate required for the L2 bus of Figure 4 is less than that for the L2 bus of Figure 3. The reason is that the data transfer between the pairs of partitions $(P_i, P_j)$, $(P_3, P_4)$ and $(P_5, P_6)$ can go directly through the gateway devices rather than through the L2 bus. As a result, for the same L2 bus technology, the design shown in Figure 4 can support more partitions than that shown in Figure 3.

In order to design a system like the one shown in Figure 4, a partition, say $P_i$, should be paired with another partition, say $P_j$, such that inter-partition traffic between the pair $(P_i, P_j)$ will be more than that between any other pair $(P_i, P_k)$ for all $j \neq k$. In other words two partitions can be paired together if they have more functional dependencies. Moreover, the two partitions which are going to be paired together should be physically close to each other. Otherwise, very long wires will be necessary to connect them to the corresponding gateway device which will reduce the benefits of multiplexing.

In the recent past, drive-by-wire or x-by-wire concepts have drawn significant attentions within the automotive industries. As drive-by-wire may become a standard sometimes in the future, various safety critical messages may go through one or more L1 buses. To guarantee on-time delivery of safety critical messages, future in-vehicle networks should be designed such that the system will tolerate bus faults. Otherwise, the safety of vehicles and passengers will be compromised. There are various existing time-triggered protocols for drive-by-wire applications. Some of these protocols are TTCAN, FlexRay and TTP/C. FlexRay and TTP/C support a dual-bus system. In these systems, safety critical messages are sent through both buses at the same time. Thus, safety critical messages can still be delivered if only one bus is at fault at the time of delivery. If the TTCAN protocol is ever going to be used for drive-by-wire applications, then it must also support two buses for tolerating single-bus faults.

Figure 5 shows another network architecture where Partition $P_1$ contains a network with a time-triggered protocol for drive-by-wire applications. $P_1$ uses two buses to tolerate single-bus faults. As a result, the gateway device $G_1$ has four ports: two for two L1 buses of $P_1$, one for one L1 bus of $P_2$ and another one for the L2 bus.

As mentioned earlier, pairing of partitions will be feasible if the two partitions of a pair are close to each other, and they have significant functional dependencies. Otherwise, pairing will not provide any additional benefits to vehicle multiplexing.

**Figure 4: An in-vehicle network with six partitions ($P_1$, $P_2$, . . , $P_6$), three gateways/switches ($G_1$, $G_2$ and $G_3$) and one diagnostic gateway (DG)**
If pairing is not possible due to functional or physical limitations, and if one L2 bus is not enough to connect all partitions of a vehicular network, then two or more interconnected L2 buses can be used. The architecture shown in Figure 6 has two L2 buses. Each L2 bus connects a number of partitions to form a cluster of partitions. One L2 bus connects Partitions P_i through P_j to form one cluster, and another L2 bus connects Partitions P_{i+1} through P_{j+1} to form another cluster. The two clusters are connected to a 3-port gateway device G. The third port of G is connected to the diagnostic gateway DG. In this type of systems, inter-cluster messages will have more end-to-end latencies than inter-partition messages within the same cluster because the inter-cluster messages will have to go through one more L2 bus and one more gateway device G. Since clustering has to be done in such a way that inter-cluster messages are not going to control vehicle dynamics in real-time, some additional latency for inter-cluster messages could be tolerated by the system.

If the in-vehicle networking system is going to have more than two clusters, then the additional clusters can be connected in a cascade manner. Figure 7, shows an in-vehicle networking system with four clusters connected in a cascade manner. The four L2 buses of four clusters are connected in a cascade manner using three gateway devices: GW_1, GW_2, and GW_3. The end-to-end message delay will be maximum when messages need to go from Cluster 1 to Cluster 4 and vice versa. The maximum delay can be expressed as:

\[(\text{see Box 3.)}\]

---

Figure 5. An in-vehicle network with six partitions, three gateways/switches and one diagnostic gateway. Partition P1 has two L1 buses

Figure 6. An in-vehicle network with two clusters of partitions. A gateway device G connects the two clusters and a diagnostic gateway device DG

---
In-Vehicle Network Architecture for the Next-Generation Vehicles

The total number of gateway devices, including the diagnostic gateway, in an in-vehicle network with cascaded clusters is \( m + c \), where \( m \) is the number of partitions, and \( c \) is the number of clusters in the system.

If the inter-cluster message delay between Cluster 1 and Cluster 4 is too high and not acceptable, then another way of interconnecting those clusters would be the use of a third level bus or an L3 bus. Figure 8 shows a system with four clusters interconnected by an L3 bus. The four clusters are connected to the L3 bus using four gateway devices: GW_1, GW_2, GW_3 and GW_4.

The topology shown in Figure 8 is in fact a hierarchical network topology with three levels. For the network shown in Figure 8, the inter-cluster message delay can be expressed as:

\[
(\text{see Box 4.})
\]

**Box 3.**

\[
T_{\text{Cascade}} = \sum_{\text{over-two-L1-buses and four-L2-buses}} \left( \frac{l_{\text{avg-bus}}}{(1-u_{\text{bus}})BW_{\text{bus}}} \right) + \sum_{\text{over-five-gateway devices}} T_{\text{ProtocolConversion}} \tag{7}
\]

**Box 4.**

\[
T_{\text{Hierarchical}} = \sum_{\text{over-two-L1-buses, two-L2-buses and one-L3-buses}} \left( \frac{l_{\text{avg-bus}}}{(1-u_{\text{bus}})BW_{\text{bus}}} \right) + \sum_{\text{over-four-gateway devices}} T_{\text{ProtocolConversion}} \tag{8}
\]

*Figure 7. An in-vehicle network with four clusters connected in a cascade manner using gateway devices: GW_1, GW_2, and GW_3*
The total number of gateway devices, including the diagnostic gateway, for a 3-level hierarchical in-vehicle network is $m + c + 1$, where $m$ is the number of partitions, and $c$ is the number of clusters in the system.

From Equations (7) and (8), it is seen that the maximum end-to-end message delay for the hierarchical network is less than that for the cascaded network because for the cascaded network with four clusters, in the worst case the messages have to go through six buses, and five gateway devices will have to run protocol conversion algorithms. Whereas, for the hierarchical network with four clusters, the messages have to go through five buses, and four gateway devices will have to run protocol conversion algorithms. For the hierarchical network, the price that we have to pay to get lower message delay compared to the cascaded network, is the cost of an extra gateway device and the cost of an extra bus (the L3 bus). Thus, if little bit longer inter-cluster message delay is not a major issue, then cascaded in-vehicle network will be a cost-effective choice than a hierarchical in-vehicle network.

If Clusters 1 and 4 are physically close to each other, then another better way of forming the network will be to connect the L2 buses of Clusters 1 and 4 by a gateway device as shown in Figure 9. As a result, a ring network is formed at the level of L2 buses.

For the ring network, in the worst-case an inter-cluster message will have to go through two L1 buses, three L2 buses and four gateway devices. Thus, the worst-case message delay can be written as:

---

**Figure 8. A hierarchical in-vehicle network with four clusters interconnected using an L3 bus**

---

**Figure 9. An in-vehicle ring network with four clusters**
In-Vehicle Network Architecture for the Next-Generation Vehicles

Box 5.

\[ T_{\text{Ring}} = \sum_{\text{over-two-L1-buses and-three-L2-buses}} \left( \frac{L_{\text{avg-bus}}}{(1-u_{\text{bus}})B_{\text{bus}}} \right) + \sum_{\text{over-four-gateway devices}} T_{\text{ProtocolConversion}} \]  

(see Box 5.)

From Equations (7), (8) and (9), it is seen that the worst-case inter-cluster message delay for the ring network of Figure 9 is less than that for the cascaded network of Figure 7 but comparable to that for the hierarchical network of Figure 8. However, the ring network will be less expensive than the hierarchical network because it does not need the L3 bus. There is one minor disadvantage of the ring network compared to the hierarchical network. In the hierarchical network, the messages from the diagnostic gateway to a network node will have to go through three buses: the L3 bus, an L2 bus and an L1 bus, and two gateway devices. Whereas, in the ring network, for the worst-case, the messages from the diagnostic gateway to a network node will have to go through three buses: two L2 buses and an L1 bus, and three gateway devices. Since a slight increase in latency during a diagnostic procedure is not a problem, a ring network will be the choice among the all three choices: cascade, hierarchical and ring, provided the physical locations of the clusters enable a ring network to be formed.

SIMULATION MODELS FOR ANALYSIS OF VARIOUS IN-VEHICLE NETWORK TOPOLOGIES

Extensive simulation and analysis should be done before a particular topology is selected for a particular type of vehicle. The performance of various partitions of the in-vehicle network as well as the entire network can be determined by developing simulation models. After executing the simulation models on computers, one can determine the minimum requirements for different partitions of the network architecture. Some of these requirements will include minimum processing powers needed from various electronic modules, minimum bandwidth needed from various buses, minimum computation power and minimum size of buffers needed in various gateways, etc. Computer simulations will also allow us to select the optimal network topology needed to meet the requirements of the next generation vehicles. Types of simulations and techniques of developing simulation models for vehicular networks are described in the following subsections of this book chapter.

Types of Simulations: Computer simulations can be divided into two types (Molloy, 1989): i) time-based simulation and ii) event-based simulation. In a time-based simulation, the program control loop is associated with time. For each execution of the main control loop, the simulation clock advances by a fixed time unit. In the case of an event-based simulation, the execution of the main loop represents a single event. The simulation clock is simply advanced by the amount of time since the last event. For asynchronous systems, the event-based simulation is computationally more expensive and accurate than the time-based simulation. But, for synchronous systems, the time-based simulation is as good as the event-based simulation, and it is computationally less expensive than event-based simulation. CAN protocol is an event-based protocol. Thus, an event-based simulation tool will be the most appropriate tool for analyzing various performance parameters of a CAN bus. Buses that use time-triggered protocols, such as
TTCAN, and FlexRay, carry both time-triggered and event-triggered messages. The time-based simulation can be used to simulate time-triggered messages and the event-based simulation can be used to simulate event-triggered messages. Thus, a combination of both time- and event-based simulations is necessary to simulate a bus like TTCAN or FlexRay.

**Developing an Event-Based Simulation Tool:** Here we explain how to develop an event-based simulation tool to meet the requirements of specific needs for the next generation vehicles. In a vehicle system, an event is triggered by a sensor or by the driver’s action such as pressing the brake or the gas paddle. The event then goes to a processor queue, as shown in Figure 10. If the processor is busy or other events are waiting in the processor queue, then the event stays in the processor queue. Otherwise, the processor starts working on the event immediately. The event, queued in the processor queue, will stay in the queue until the previous events are taken care of by the processor. The processor spends some time (processing time) on an event. After that, the processor tries to send a message through the bus (say CAN bus) connected to the processor. If the bus is available, then the message is immediately sent to the bus. Otherwise, the message is queued in the transmit buffer of the node. The message will go through the bus after the previous messages have been transmitted. After a certain time (transmit time), the message is queued to the receive buffer of the destination node. If the processor at the receiving node is free, then it will immediately process the message. Otherwise, the processor will first take care of other queued messages and then it will process this new message. Figure 10 shows the entire process of information flow from a sensor to an actuator for a single-bus system.

**Components of the latency of an event:** There is a delay (latency) between the time an event occurs at a sensor or due to a driver’s action, and the time an actuator takes an action. The latency has several components as shown in the following equation.

$$\text{latency} = t_{pq} + t_{ps} + t_{tb} + t_{\text{bus}} + t_{\text{propagation}} + t_r + t_{pd}$$

where,
- $t_{pq}$ = delay at the processor queue
- $t_{ps}$ = processing time at the source node
- $t_{tb}$ = delay at the transmit buffer of the source node
- $t_{\text{bus}}$ = transmit time through the bus
- $t_{\text{propagation}}$ = propagation delay through the bus

*Figure 10. Information flow from a sensor to an actuator*
• $t_{rb}$ = delay at the receive buffer of the destination node
• $t_{pd}$ = processing time at the destination node

In the simulation model, the value of the delay, $t_{pq}$, at the queue of the source processor can be determined by monitoring the arrival and exit times of the event in the processor queue. The value of the processing time, $t_{ps}$, depends on the type of the processor. This time can be expressed as

$$t_{ps} = k_{1s} + k_{2s}N$$  \hspace{1cm} (11)

Where $k_{1s}$ is a constant time, which represents the constant overhead of the processor at the source node in order to prepare a message. This constant overhead may include:

• The time needed by the processor to get out of the main program,
• The time needed by the processor to go to the interrupt service routine (assuming that the transmission of a message is interrupt driven),
• The time needed by the processor to check some status bits and set/clear some control bits of the CAN controller,
• The time needed by the processor for some other house keeping operations.

Thus, for a given processor, the value of $k_{1s}$ can be calculated based on its clock frequency and the number of instructions the processor needs to execute the abovementioned operations. The parameter $k_{2s}$, shown in Equation (11), is another constant, per byte of data in coding the message, and $N$ is the number of data bytes in the message.

The value of $t_{ps}$ can be determined by monitoring the arrival and exit times of the message in the transmit buffer of the source node. The value of $t_{bus}$ can be expressed as

$$t_{bus} = \frac{L}{S}$$  \hspace{1cm} (12)

Where, $L$ is the total number of bits in the message, and $S$ is the speed of the bus in bits/sec. For example, if the message is a CAN message, then $L$ will include the start bit, arbitration bits, RTR bit, control bits, data bits, stuff bits, CRC bits, all delimiter bits and the end of frame bits.

The value of $t_{propagation}$ should be negligible compared to other components of the delay. It depends on the speed of electrical signals through the bus. In free space, electromagnetic signals take approximately one nanosecond to move a foot. Thus, the propagation delay through a bus should be little over one nanosecond per foot. As a result, the value of $t_{propagation}$ should be in the order of few tens of nanoseconds which is very small compared to other components of Equation (10). Thus, the propagation delay through the bus can be ignored for performance analysis.

The value of $t_{rb}$ can be determined by monitoring the arrival and exit times of the message in the receive buffer of the destination node. Like $t_{ps}$, the value of $t_{pd}$ can be expressed as

$$t_{pd} = k_{1d} + k_{2d}N$$  \hspace{1cm} (13)

Where $k_{1d}$ is a constant time, which represents the constant overhead of the processor at the destination node in order to decode a message, and $k_{2d}$ is the additional overhead, per byte of data, in decoding the message.

For a system with multiple partitions, the latency of an event can be expressed as:

(see Box 6.)

The term $t_{gateway}$ indicates the delay at each gateway device on the path of the signal. There are three components of $t_{gateway}$: queuing delays at both the input and output queues of the gateway and the time required for protocol conversion by the gateway device.
Simulation Model: Figure 11 shows the block diagram of an event-based simulation model. In an event-based simulation model, an event is generated by a scheduler. The scheduler generates this event based on some kind of realistic stochastic process. We can develop a stochastic model for a particular event by collecting a large set of real data for that event. For example, the stochastic model of the brake event (pressing the brake paddle by the driver), can be developed by logging the driver’s brake activity over a long period of time and collecting similar data from many vehicles.

After generating an event, the scheduler keeps that event in the event queue, as shown in Figure 11. When an event is dispatched from the event queue, another event is generated. This new event is then sent to the event queue. When a new event enters into the event queue, it is sorted within the event queue using the event-time as the sorting key. The new event is then placed at an appropriate point in the event queue. After dispatching an event from the event queue, the event is then sent to the event processing unit. The event processing unit then simulates the event in exactly the same way as a real system would process it.

The stochastic model of an event determines the inter-arrival time of the event, based on the behavior of the device that generates the event. A synchronous device, such as a time-triggered device, generates events after fixed interval of time. Hence, for a time-triggered system, the inter-arrival time is fixed. But, an asynchronous device doesn’t generate events after fixed interval of time. Most physical asynchronous systems behave like a Poisson process. Thus, we can assume that a non time-triggered device will behave like a Poisson process. The stochastic model of an event will have the appropriate values of the parameters that define the characteristics of the Poisson process of the event. As mentioned earlier, these parameters of the stochastic model for an event can be determined by collecting a large set of field data for that particular event. The model shown in Figure 11 can be used to simulate an event-triggered system with an event-triggered bus (say CAN bus).

Since both TTCAN and FlexRay support time- and event-triggered messages, we can develop the simulation models of these buses by building another layer on the top of the event-based simulation model. Figure 12 shows a simulation model of a combined time- and event-based system. Here, the event scheduler is a deterministic as well as a stochastic process. In this case, a global simulation clock is maintained to keep the time-triggered events in sync. The frequency of this global simulation clock (software clock) should be at least equal to the highest frequency clock used in the in-vehicle network architecture.
requirement on the software clock will allow us to determine the latency and other performance parameters at a very high resolution (within a clock cycle of the highest frequency clock of the real system). At every global simulation clock pulse, the event queue must be checked. If the event-time of the event at the head of the event queue is less than or equal to the current value of the global simulation clock, then the event will be dispatched from the event queue. The dispatched event will then be checked to determine whether it is a deterministic event or a non-deterministic event. If it is a deterministic event, then the deterministic scheduling process will be used to schedule the next deterministic event. Otherwise, stochastic scheduling process will be used to schedule the next non-deterministic event. The scheduled event, whether deterministic or non-deterministic, will then be sent to the event queue. The event will be placed in the event queue after sorting it with other events in the event queue. The event-time will be used as the key to sort the events in the event queue.

By developing a simulation model, one can determine various performance parameters of the in-vehicle network. These parameters will include, but not limited to, the latency of different types of message, busload, bandwidth, and throughput.

**FUTURE TRENDS**

There is no doubt that future trends are going to be in the direction of drive-by-wire, active-safety, inter-vehicle communications and vehicle-to-infrastructure communications. Future vehicles will access internet and exchange information with other vehicles and road-side units for various reasons such as collision avoidance, message dissemination, software upgrade in electronic modules, receiving data from content providers, sharing files, etc. Some researchers have proposed architecture for Next Generation Vehicle Network and presented a dynamic discovery service protocol for internet access (Baroody, 2005; Baroody, 2006). If future in-vehicle networks are not properly designed, then vehicles' safety could be compromised or its operation could be degraded due to various types of cyber attacks from external entities. The in-vehicle network that is mainly responsible for controlling vehicle dynamics, must be well protected from outside cyber attacks. Good security software and firewalls must be installed in all gateway devices that exchange vehicle’s dynamic information. The gateway device that connects the telematic unit to the internal network of the vehicle must have a good firewall so that it can filter out all suspicious information that is coming from outside. After extensive research, if it is not possible to come up with a good security solution to protect...
In-Vehicle Network Architecture for the Next-Generation Vehicles

vehicles’ safety-critical modules, then no external traffic should be allowed to enter the safety-critical modules.

For collision avoidance applications, a vehicle will disseminate its dynamic information. However, current in-vehicle networks are not secure. This means that when receiving nodes are picking up CAN messages from a CAN bus, they are not verifying whether or not the messages are authentic. The receiving nodes automatically assume that the messages are valid as long as the messages are free from various types of errors such as CRC error, bit error, stuff error and frame error. Thus, if a person with malicious intent connects a device to the vehicle’s CAN bus and sends messages with wrong information, other nodes will still accept them as valid messages. The problem will occur when these messages with wrong information are picked up by a gateway device and then routed to the vehicle’s telematic unit. The telematic unit will then broadcast these messages with wrong information to other vehicles. As a result, there could be disruption of traffic leading to accidents.

Since vehicle-to-vehicle communication is a peer-to-peer communication, protection of drivers’ or owners’ privacy from peers is going to be another challenging task. Privacy protected secure communication is not an easy task. Good Cyber-Physical Systems at the network level are necessary so that future vehicles will be able to communicate with external entities without getting involved in safety, security and privacy problems.

The issues related to vehicular network security are still at the research stage. The purpose of this book chapter is not to provide any security solutions but to explain the significance of this topic and what needs to be done to safely operate future intelligent vehicles. Many researchers have already started looking into these issues and are trying to propose solutions (Hossain, 2007; Rabadi, 2007; Rabadi, 2008).

CONCLUSION

In this paper, we discussed various types of topologies and protocols that could be used in the future in-vehicle networks. Due to the complexity of the future in-vehicle networks, many partitions of the network will be required. These partitions must be interconnected together for functional dependencies and better diagnostic purposes. A number of network topologies has been presented and analyzed for cost, bandwidth and message latencies. Since future vehicles will be communicating with external entities for various reasons, the book chapter also addressed the issues of security, safety and privacy which should be taken into consideration at the time of designing the in-vehicle network components. Finally, some ideas have been presented in developing simulation models to analyze various types of networks which will ultimately help in selecting a network topology and various network components for a given set of requirements and specifications.

REFERENCES


