

Routing Schemes for Switch-based In-Vehicle Networks

Shuhui Yang

Purdue University at Calumet
yang246@purdue.edu

Wei Li and Zhiwei Xu

Institute of Computing Technology,
liweili@ict.ac.cn, zxu@ict.ac.cn,

Wei Zhao

University of Macau
weizhao@umac.mo

Abstract—In this paper, we study in-vehicle digital network systems. We propose a switch-based architecture for in-vehicle networks and focus on the critical related issue: routing schemes. We note that in-vehicle networks are fundamentally different from many other switch-based networks (e.g., the Internet). This is because within an in-vehicle network, messages are not unicast or multicast as in the Internet, but many-to-many cast. We analyze several routing schemes. We investigate if routing schemes that have already been proposed for Internet-like networks, may be feasible for our in-vehicle networks. We also propose a new algorithm that is specifically designed for many-to-many cast messages as an initialization for further research on routing schemes for switch-based in-vehicle networks.

I. INTRODUCTION

A vehicle is inherently a distributed information system, as the operation of a vehicle depends on the collaborations of functional components in different locations in the vehicle [7] [13] [12]. In modern vehicles, digital networking technology has been exploited for in-vehicle communication due to its cost-effectiveness, space efficiency, and flexibility [8] [CVV05]. Digitized control has realized a variety of functions, including lights, wipers, doors/windows, and motor control.

In general, the “control-by-wire” feature [25] is now being realized in the design of the next generation vehicle thoroughly, for both minor functionalities (e.g. light-by-wire) and mission-critical ones (e.g. break-by-wire, engine-by-wire). An integrated “x-by-wire” system [25] has also been expected. The rapid development of vehicle capabilities introduces new challenges and demands for automotive network systems [NHB05]. For example, the amount of transmitted information is expected to increase significantly with the deployment of additional electronic and digital components (e.g., use of audio and video). In addition, applications such as *collision warning* and *avoidance* and *intelligent transportation systems* (ITS) [14] also require the support of novel network technology [11]. Therefore, real-time, reliability, scalability, and fault-tolerance are the basic requirements for in-vehicle digital networks.

Current in-vehicle networks depend on bus-based architecture and will not be able to meet these new requirements. This is because bus-based network architecture lacks scalability and hence has become the inherent drawback of the capability bottleneck capable of harming system performance.

In this paper, we propose a switch-based architecture for in-vehicle networks. With our proposal, an in-vehicle communication network is partitioned into sub-networks. Each of them can be a bus-based network, as in current in-vehicle systems. These bus-based sub-networks are then connected by a backbone that consists of a number of switches. The several unique characteristics of our switch-based network architecture distinguish it from other proposals: Taking advantage of the switch-based backbone, our network can be easily scaled up in terms of bandwidth capabilities and may eliminate a single point of failure if properly configured. As we connect bus-based sub-networks to the switch-based backbone, our system is still compatible with existing vehicle control units that usually connect to bus-based networks. The cost of the network can be kept low since switches in the system require very low hardware complexity.

Like other networks, routing plays a critical role in terms of network efficiency and effectiveness. Routing within our in-vehicle network is particularly challenging, as the messages inside the in-vehicle networks are many-to-many cast, rather than unicast or multicast. We systematically test a set of representative routing techniques that have been proposed for internet-like networks in order to see if they can be adopted for our switch-based in-vehicle networks. We also design a routing algorithm that is particularly feasible for our system as in initialization for future research. Performance of these routing protocols is compared via simulation. One major observation is that while no routing scheme outperforms others in terms of both space efficiency and real-time capability, our newly designed routing scheme performs statistically for both kinds of measurements.

II. PRELIMINARIES

A number of digital in-vehicle network systems such as CAN [6], LIN [1], FlexRay [5] and TTP/C [4] have been developed and adopted in the current automotive industry. They all use a bus-based architecture, but differ in terms of function, performance, and application scope. These networks can be classified into two categories: the event-triggered and the time-triggered networks. CAN and TTP/C are, respectively, representatives for each category. The event-triggered architecture (e.g., CAN) is designed for asynchronous (priority-driven) communications. On the other hand, time-triggered architecture (such as TTP/C, FlexRay) is

designed to support synchronous communication. In [3], these two types of architectures are compared.

Existing in-vehicle networks become the bottleneck of performance due to their bus-based architecture. Work has been done to upgrade the bus-based network architecture. The basic idea has been to partition a network into sub-networks (each of them is still a bus for the sake of compatibility) and then to interconnect the sub-networks. Two types of approaches have been proposed for the interconnection.

The first, called *backbone approach*, is to use a backbone network for interconnection, That is, the backbone network interconnects all functional sub-networks. Information exchanged among different sub-networks is performed by this backbone. In [18], IEEE 1394 networks are used as backbones to link all other networks such as CAN and TTP/C. In [21], a FlexRay network is used as a backbone to interconnect several sub-systems. In [SBF05], an Ethernet is used to interconnect several CAN bus. A common feature of these methods is their use of bus architecture as a backbone. Consequently, we refer these approaches as the *bus-based backbone approach*.

The second one is referred to as the *gateway approach*. In [9] and [23], a dedicated computer node is used to interconnect sub-networks. The gateways can be implemented at a different layer of OSI model. In [10], design and performance of CAN-to-CAN bridges have been discussed. In [15], [19], [20], and [22], different types of gateways which interconnect different in-vehicle network protocols have been introduced.

The gateway approach is efficient when the system is small, i.e., there are only a few sub-networks. However, it is prone to single point failure and may become a performance bottleneck. The backbone approach overcomes these problems but must be properly configured in order to provide the required communication services. However, bus-based backbone still has obvious drawbacks due to the inherit properties of bus technology. When the amount of data exchanged increases, a single bus may not be capable of meeting the real-time and bandwidth requirements. Specific network structures (such as hierarchical structure in [SJW08]) were proposed to increase the capability of the backbone. However, these methods still suffer from the performance bottleneck and lack of scalabilities.

III. SWITCH-BASED NETWORK ARCHITECTURE AND ITS MODEL

A. Proposed network architecture

As mentioned in the previous section, while existing in-vehicle networks are diverse and correspond to different protocols, they typically have similar bus-based topology, implying a broadcast mode for message transmission. In these bus-based in-vehicle networks, nodes are usually called *Electronic Control Units (ECUs)*. An ECU is composed of both hardware and software. Usually, ECUs are directly

connected to sensors and/or actuators. According to their different functions, ECUs send or receive different types of messages. In early generations of in-vehicle networks, only ECUs on the same bus communicate to each other.

However, as demand for automotive electronic systems increases, sub-systems may adopt different in-vehicle networks and information needs to be exchanged among them. The way to connect sub-networks is a critical issue on which we will focus in this paper.

Figure 3.1 shows the network architecture we propose. The network consists of a number of sub-networks which may be CAN, FlexRay, TTP/C or other bus-based sub-networks. Each sub-network may contain one or more ECUs (not shown in the figure). Sub-networks are then connected by a switch-based backbone. This backbone relays messages among the sub-networks.

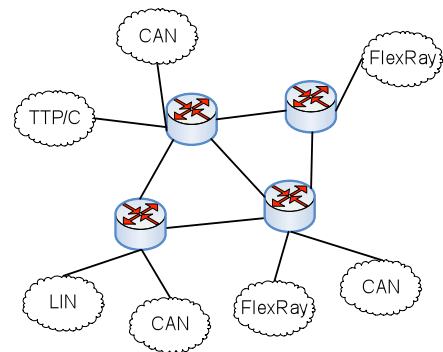


Figure 3.1: Switch-based Network Architecture

A switch used in our network performs functions of message receiving and buffering, routing table look-up, protocol/format conversion, and message relay. Thus, a switch typically consists of the following major components: *Computing Resources*: include basic hardware such as processors, memories, storage, etc. These resources must meet the constraints on cost, reliability, and real time. *Network Interface Controllers (NIC)*: for different sub-networks, a switch needs to use different types of network interface controllers (e.g., CAN controller or FlexRay controllers). NICs control I/O ports of the switch, which interfaces to sub-networks or other switches. *Routing Table*: contains the necessary information for performing routing. Section IV will provide a detailed discussion on this subject.

In comparison with other approaches, this switch-based network architecture has the following advantages:

- (1) Scalability and adaptability: Switch-based networks can be easily adjusted to meet increasing demands and to adapt to different situations.
- (2) Reliability: Redundancy can be easily introduced in order to eliminate the single point of failure, hence increasing reliability.
- (3) Low cost: With proper design, both manufacturing and maintenance costs can be kept at a minimum.

B. Routing issues

In a switch-based network, routing is usually realized by using routing tables embedded in switches. That is, each switch has a routing table. A routing table has a number of entries. In the simplest form, each entry identifies a message and indicates a related output port. When a message arrives, the routing table is looked up in order to locate the entry that identifies the message. The message is then delivered to the output port indicated in the entry for transmission.

A particular challenge to routing messages in our in-vehicle networks is the addressing mode of messages. Messages in in-vehicle networks are not uni-cast, any-cast, or multi-cast, but *many-to-many cast*. That is, messages are grouped into types. For each type of message, there are typically multiple senders and multiple receivers [TB94]. A message from any sender of a given type must be delivered to all the receivers designated by this type.

An issue presenting itself is how to design an efficient and effective routing scheme for many-to-many cast messages in switch-based networks. One solution is to use the widely adopted routing algorithms in the existing switch-based networks, such as the Internet. For example, we can use the traditional unicast or multicast techniques to realize routing of the many-to-many cast message. However, performance analysis (provided in a later section) demonstrates that these traditional routing algorithms, when applied to the switch-based in-vehicle network, may not be feasible with respect to efficiency and/or effectiveness issues. Thus, we may have to design a routing scheme that is particularly suitable for switch-based in-vehicle networks.

C. Network model and topology

Before we formally present routing schemes, let us introduce some notations for the network and its topology. Let S be a given in-vehicle system and $S = (V, M)$, where V is the set of switches in the system. $V = \{v_1, v_2, \dots, v_n\}$ where v_i is switch i . Each switch has a fixed number of ports, described as $port(v_i) = \{1, 2, \dots\}$. Each port is an interface connecting to another switch or a sub-network. M is the set of message types in the system. $M = \{m_1, m_2, \dots, m_m\}$ where $m_i = (i, S_i, R_i, l_i, D_i, f_i)$, in which i is the message type identifier, $1 \leq i \leq m$; $S_i \subseteq V$ is the sender set of the message of type i ; $R_i \subseteq V$ is the receiver set; l_i is the message length; D_i is the deadline for this type of messages; f_i is the message generation period.

Without loss of generality, let us assume that the switches are identical in our system. This assumption simplifies analysis though our results and can be easily extended to the general case, as discussed in [26].

Note that the senders and receivers of a message refer to switches, rather than ECUs. This is because when a message reaches a switch to which the destination ECU belongs, the switch will simply broadcast the message to the associated sub-network (which should be a bus, (e.g., CAN bus)) with no further routing operation. All ECUs on the bus of the sub-network will be able to receive the message, but only those that are indeed the destinations of this message type will accept it.

D. Simple Topology

Topology has an impact on the performance of networks. However, for Internet, a system designer cannot control its topology. Consequently, the designer cannot make any assumption on network topology. Our in-vehicle network is different. The network is relatively small. Every node that needs to be connected is known in design time. Therefore, it is possible for the system designer to select an appropriate network topology to favor the corresponding routing scheme and consequently to achieve desired performance. We introduce notations related to network topology.

Definition 1. *Switch graph* $G = (V, E)$ of an in-vehicle network system is a graph containing switch nodes and links between switches, where V is the set of all nodes (i.e., switches) and E is the set of all links.

Figure 3.2 (a) shows the switch graph of the system in Figure 3.1. Figure 3.2 (b) shows another example of switch graph. Note that the switch graph corresponds to the system backbone.

For any two nodes v_i and $v_j \in V$, let V_{ij} denote the set of nodes on the shortest path between v_i and v_j in the switch graph.

Definition 2. A subgraph $G' = (V', E')$ of a switch graph G is *shortest-path completed* (spp) if for any two nodes v_i and $v_j \in V'$,

- 1) $V_{ij} \subseteq V'$, and
- 2) if $(v_i, v_j) \in E$, $(v_i, v_j) \in E'$.

Definition 3. An *spp-routing ring* of G is a shortest path completed subgraph of G . The *spp-routing ring* has a ring topology. The number of nodes on the ring is its *ring length*.

In the switch graph shown in Figure 3.2 (b), the entire switch graph is an spp-routing ring with length of 4.

In Figure 3.2 (a), $\{a, b, c\}$ and edges among them form an spp-routing ring with length of 3. Also, $\{a, d, c\}$ and edges among them form another spp-routing ring with length 3.

However, in Figure 3.2 (a), $\{a, b, c, d\}$ with edges $\{(a, b), (b, c), (c, d), (d, a)\}$ does not form an spp-routing ring since the shortest path from b to d is not included and it is not the shortest path completed. Thus, our readers should not be confused between a ring and an spp-routing ring. An spp-routing ring is a ring sub-graph in a graph, but a ring sub-graph may not necessarily be an spp-routing ring.

Definition 4: An in-vehicle network has *simple topology* if, in its switch graph, the length of every spp-routing ring is no more than 3.

For a given set of nodes, we can use the following two-step method to construct a network with simple topology: to form a ring structure, and then to connect nodes on the ring in a zigzag manner. More sophisticated design may take into account predetermined requirements, such as direct connection between some nodes.

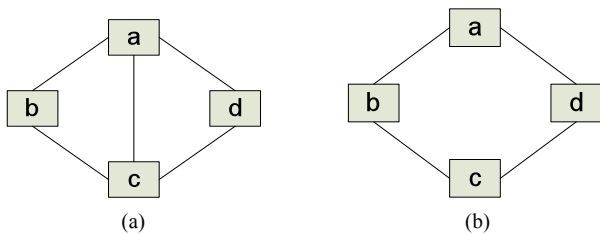


Figure 3.2: (a) Switch Graph of the System in Figure 3.1 (b) Another Example of Switch Graph

IV. ROUTING SCHEMES

Note that no matter which routing algorithm is used, upon the arrival of a message, the operation carried out by a switch is similar: it will first look up the routing table for an entry that identifies the message and then transmit the message over the output port(s) indicated in that entry. Different routing schemes differ at the routing tables they use. That is, although we present a routing scheme, it is not necessary to describe the operations by which a message is routed but the algorithms by which the related routing tables are constructed.

Recall that in our in-vehicle network, naturally, messages may be grouped according to their types. Consequently, upon the arrival of a message, a switch may have to transmit the message over multiple output ports.

A. Routing by unicast

The most common routing approach in a switched network is unicast, where each message is associated with a sender and a receiver. The routing is based on the receiver's address.

In this case, a preprocessing is necessary. When a message of type i is generated, $|R_i|$ copies of the message are duplicated and each will have a different receiver address. Then, these messages are routed according to the unicast routing approach. We refer to this routing approach as *Unicast routing*.

B. Routing by source-based multicast

Source-based multicast routing [24] is one of the multicast routing approaches that have been well studied. Using this technique, many-to-many cast can be easily implemented as follows: each sender of a message type has a multicast tree spreading to every receiver of the message. A message of many-to-many cast from the sender will be transmitted over the multicast tree by using the source-based multicast.

C. Routing by core-based multicast

Core-based multicast is another multicast approach [24]. The basic idea is as follows: a multicast tree is first constructed among the receivers of a message, with the root as the core. Then, each source has a path connecting to this core using unicast routing.

The core-based multicast routing approach is hybrid and consists of two phases. The first is a unicast process and the second is multicast. A preprocessing is also necessary: a message upon its generation must be encapsulated into a unicast format with the core node of this message type as its destination. When the message reaches the core, it will be routed as a multicast message based on its message type. Hence, additional processing is necessary on the core node. Each switch then contains two routing tables, one for the unicast phase (U-table) and one for the multicast (M-table).

D. A new routing scheme

In all previous schemes, message type, together with sender or destination, is used to identify an entry in the routing table, which results in relatively large routing tables. As mentioned before, switches in in-vehicle network have small number of ports. If we can use message type and the input port, where the message comes from, to identify a message, entries can be merged, and hence allowing the routing scheme to achieve a balanced performance.

Messages of the same type and arrival at the same input port of a switch are routed to the same output ports. In this case, an arrival message only needs to be identified with its message type and input port. As this scheme depends on input port to identify a message, we call it input-port based routing scheme. Algorithm 1 is the procedure for routing table construction.

Algorithm 1. Input-port based Routing Scheme

1. For each message type, h
2. For each sender, i ,
3. For each receiver j ,
4. Find the shortest path between i to j ,
5. Fill in the routing tables by adding a new entry of (entry#, h , input port#, output port#),
6. Merge the entries of the same input ports,
7. End.

We say a routing scheme is *correct* if it guarantees (1) delivery of messages to their destination(s) and (2) loop-less in routing.

It is relatively easy to establish that the unicast, source-based multicast, and core-based multicast routing schemes are correct. It is, however, not trivial to establish the correctness of the input-port based routing scheme. In this new routing scheme, the routing table uses more information. As a matter of fact, a counter-example can be given to show that in certain networks, this scheme may not be correct. We can use the switch graph of Figure 3.2 (b) to illustrate a counter-example. Assume that the network has one message type with all nodes as senders and receivers. That is, we have $M = \{m_1\}$, $m_1 = (I, S_I, R_I, l_I, D_I, f_I)$ and $S_I = R_I = \{a, b, c, d\}$.

Based on the input-port-based scheme proposed, when a node, say node c , receives a message from a neighbor, say node b , c will have to forward the message to the other neighbor, node d since c cannot distinguish whether the message sender is node b or node a (only input port of that mes-

sage is available). Consequently, if the message sender is node a , node d already gets the message from node a directly and node c does not need to forward the message at all. This ambiguousness will lead to an endless routing loop in the network.

Our goal here is to identify certain topological conditions under which the scheme becomes correct. It is our hope that the topological condition is not too constrained and can be easily met in practice.

Theorem 1. *Assume that in a network, input-port-based routing scheme is adopted. If the switch graph is a simple topology (i.e., the maximum length of its spp-routing rings in its switch graph is no more than 3), the scheme is correct.*

Interested readers are referred to [26] for a complete proof. Thus, if in a system, its maximum length of spp-routing rings is no more than 3, the shortest path scheme will be truly effective. The question is whether or not this condition can be easily met. Fortunately, this is the case in practice. This is because in-vehicle networks are usually expected to have a small number (say, less than 6) of switches and they are usually well connected. Consequently, their switch graphs are unlikely to contain a large spp-routing ring. For example, it can be easily proved that the switch graph in Figure 3.2 (a) meets this condition.

V. PERFORMANCE EVALUATION AND COMPARISON

A. Evaluation method and performance metrics

We will evaluate and compare the performance of various routing schemes by simulation. In each run of the simulation, a set of k in-vehicle network systems are evaluated. Let these systems be W_1, W_2, \dots, W_k . Each of these systems is randomly generated with the following characteristics: (1) n : the number of switches in the system. n is usually set as 20, unless otherwise mentioned. (2) p : the number of ports of a switch. p is set to be 6. (3) Network has a randomly generated topology that meets the requirement of simple topology as defined in Section III. (4) m : the number of message types in the system. m is set to be 50 unless otherwise mentioned. (5) The sender set of every message type is randomly generated. The size of the sender set is randomly picked from 1 to n . (6) The receiver set of every message type is randomly generated. The size of the receiver set is also randomly picked from 1 to n . (7) f_i : The generation period of message type i is randomly chosen between 10 and 100. (8) The size of messages is a constant and we assume that it takes one unit time to transmit the message. (9) The deadline of a message is equal to α times of its period. Let $D_i = \alpha f_i$. α is 1.0 unless otherwise indicated.

For system W_i , we collect the following performance data: The first is the worst-case routing table size. The second piece of performance data we seek is related to whether the capability of the system can meet the real-time requirements. To obtain it, the worst-case delay of each type of message needs to be derived. We adopt the method proposed in [17] [2] to analytically derive the worst-case delay.

Obviously, it is preferred in reality that the mean of the worst case routing table size be as small as possible. Small size of routing table would reduce the cost of switches. This is critical in manufacturing commercial vehicles where hardware cost should be kept at a minimum. On the other hand, the real-time success rate measures the system capability of meeting real-time requirements. The higher the success rate, the better the chance that the real-time requirements can be met.

The above settings of a simulation system can be found in many practical systems [16].

B. Simulation results

Figure 6.1 shows the results. Figure 6.1 (a) shows the worst-case routing table size when n , the number of switches in the system varies from 0 to 20 (m being 50). Figure 6.1 (b) is the worst-case routing table size when m , the number of message types, varies from 10 to 100 (n being 20).

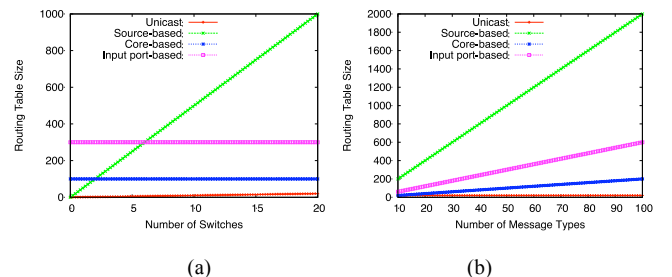


Figure 6.1: (a) Mean of the Worst-Case Routing Table Size vs. Number of Switches. (b) Mean of the Worst-Case Routing Table Size vs. Number of Message Types.

Figure 6.2 (a) shows the results for the system with network size being 20 and the number of message types being 100. However, the generation period of each message type f_i here is timed by β , which changes from 0.1 to 1 to generate different workloads within the system. Thus, the workload in the system is $\sum_{i=1}^m \beta \times |S_i| / f_i$, and it varies from 100 to 1,000 according to the adjustment of β in Figure 6.2 (a).

Figure 6.2 (b) shows the real-time success rate in terms of a relative deadline, the ratio of deadline over period of a message, α . α varies from 0.5 to infinity. In fact, when α reaches 10, most of the routing schemes can already achieve a 100% success rate.

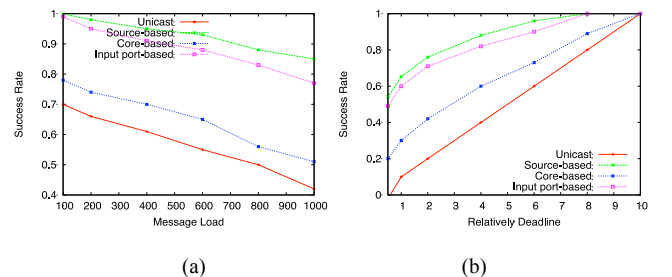


Figure 6.2: (a) Real-Time Success Rate vs. Message Load. (b) Real-Time Success Rate vs. Relative Deadline.

C. Summary of simulation results

Simulation results can be summarized as the following and also in Table 6.1:

(1) There is no scheme that can consistently outperform others. For example, the unicast scheme has the best performance in terms of routing table size but becomes unacceptable in terms of real-time success rate while the source-based multicast scheme is the best in terms of real-time success rate but performs poorly in terms of routing table size. While the import-port based scheme has not become the best in terms of either routing table size or real-time success rate, it is the only acceptable scheme in both cases. This indicates a high probability that this scheme will be used in practical systems.

Table 6.1: Summary of Simulation Results

Routing Scheme	Routing Table Size	Real-Time Success Rate
Unicast	Best	Poor, not acceptable
Source-based Multicast	Poor, not acceptable	Best
Core-based Multicast	Acceptable	Poor, not acceptable
Input-port-based Cast	Acceptable	Acceptable

VI. FINAL REMARKS

In this paper, we proposed switch-based in-vehicle network architecture in order to realize a scalable, reliable, low-cost digital communication in next generation automobile systems. We addressed a critical issue in network design: routing. We analyzed several routing schemes, including those that use existing routing techniques (e.g., unicast and multicast). We also proposed a new scheme that identifies a message in routing by both its type and the input port where the message arrives.

The work presented in this paper is preliminary and many extensions are possible. For example, in addition to simple topology that has been proposed and used in this paper, other topologies may also be of interest and should be explored in order to seek a topology that may optimize the system performance.

REFERENCES

- [1] Audi AG, BMW AG, Daimler Chrysler AG, Motorola Inc. Volcano Communication Technologies AB, Volkswagen AG, and Volvo Car Corporation, "LIN specification and LIN press announcement," SAE World Congress Detroit, <http://www.lin-subbus.org>, 1999.
- [2] G. Agrawal, B. Chen, W. Zhao, and S. Davari, "Guaranteeing Synchronous Message Deadlines with the Timed Token Medium Access Control Protocol," in IEEE Transactions on Computers, vol. 43, no. 3, pp. 327-339, March 1994.
- [3] A. Albert, "Comparison of event-triggered and time-triggered concepts with regards to distributed control systems," in Proceedings of Embedded World Conference, pp. 235-252, 2004.
- [4] R. Bannatyne, "Time Triggered Protocol: TTP/C," Embedded Systems Programming, pp. 76-86, Mar. 1999.
- [5] J. Berwanger, C. Ebner, and et al., "FlexRay--The Communication System for Advanced Automotive Control Systems," in SAE World Congress, Society of Automotive Engineers Press, no. 2001-01-0676, Apr. 2001.
- [6] Robert Bosch, CAN specification 2.0, Parts A and B, Sep. 1991.
- [7] E. G. Chowanietz, "Automobile Electronics in the 1990s. Part 1: Powertrain Electronics," in Journal of Electronics and Communication Engineering, vol. 7, no. 1, pp. 23-36, 1995.
- [8] J. A. Cook, I.V. Kolmanovsky, D. McNamara, E.C. Nelson, and K.V. Prasad, "Control, Computing and Communications: Technologies for the Twenty-First Century Model T," in Proceedings of the IEEE, vol. 95, no. 2, pp. 334-355, Feb. 2007.
- [9] D. S. Dodge, "Gateways - 101," in the Proceedings of IEEE Military Communications Conference, vol. 1, pp. 532-538, 2001.
- [10] H. Ekiz, A. Kutlu, and E.T. Powner, "Implementation of CAN/CAN bridges in distributed environments and performance analysis of bridged CAN systems using SAE benchmark," in Proceedings of IEEE Southeastcon '97, pp. 185-187, Apr. 1997.
- [11] P. F. Hokayem and C. T. Abdallah, "Inherent Issues in Networked Control Systems: A Survey," in Proceedings of 2004 American Control Conference, pp. 4897-4902, 2004.
- [12] H. Kopetz, "Automotive electronics," in Proceedings of 11th Euromicro Conference on Real-Time Systems, pp. 132-140, 1999.
- [13] J. G. Kassakian, H. Wolf, J. M. Miller, and C. J. Hurton, "Automotive electrical systems circa 2005," in IEEE Spectrum, vol. 33, no. 8, pp. 22-27, Aug. 1996.
- [14] J. McQueen and B. McQueen, Intelligent Transportation Systems Architectures, Boston, USA, Artech House Books, 1999.
- [15] T. Moon, S. Seo, J. Kim, S. Hwang, and J. Jeon, "Gateway system with diagnostic function for LIN, CAN and FlexRay," in Proceedings of International Conference on Control, Automation and Systems, pp. 2844-2849, Oct. 2007.
- [16] N. Navet, Y. Song, F. Simonot-Lion and C. Wilwert, "Trends in automotive communication systems," Proceedings of the IEEE 93(6), 1204-1223, 2005.
- [17] A. Raha, S. Kamat, Wei Zhao, and W. Jia, Admission Control for Hard Real-Time Connections in ATM LANs, in IEEE Proceedings - Communications, 148 (4), pp. 1-12, August 2001.
- [18] M. Rabel, A. Schmeiser, H.P. Grobmann, "Ad-hoc in-car networking concept," in Proceedings of IEEE Intelligent Transportation Systems, pp. 363-368, 13-15 Sept. 2005.
- [19] J. Sommer and R. Blind, "Optimized Resource Dimensioning in an embedded CAN-CAN Gateway," in Proceedings of International Symposium on Industrial Embedded Systems, pp. 55-62, Jul. 2007.
- [20] S. Shaheen, D. Heffernan, and G. Leen, "A gateway for time-triggered control networks," in Microprocessors and Microsystem, vol. 31, no. 1, pp. 38-50, Feb. 2007.
- [21] G. Sung, C. Juan, C. Wang, "Bus Guardian Design for automobile networking ecu nodes compliant with FlexRay standards," in Proceedings of IEEE International Symposium on Consumer Electronics, pp. 1-4, Apr. 2008.
- [22] S. Seo, S. Lee, S. Hwang, and J. Jeon, "Development of Network Gateway Between CAN and FlexRay Protocols For ECU Embedded Systems," in Proceedings of International Joint Conference on SICE-ICASE, pp. 2256-2261, Oct. 2006.
- [23] M. Tokunaga and S. Yoshida, "Advanced Functionality of Vehicle Gateways," SEI Technical Review, no. 55, pp. 55-59, Jan. 2003.
- [24] B. Wang, and J.C. Hou, "Multicast Routing and Its QoS Extension: Problems Algorithms, and Protocols," in IEEE Network, 14(1), January-February 2000.
- [25] C. Wilwert, N. Navet, Y.-Q. Song, and F. Simonot-Lion, "Design of automotive X-by-Wire systems," The Industrial Communication Technology Handbook, R. Zurawski, Ed. Boca Raton, FL: CRC, 2004.
- [26] S. Yang, W. Li, and W. Zhao, "Analysis of Routing Schemes for Switch-Based In-Vehicle Networks," Technical Report, Department of Computer Science, Rensselaer Polytechnic Institute, Aug. 2008.