Performance Evaluation of Control Networks: Ethernet, ControlNet, and DeviceNet

Feng-Li Lian[1], James R. Moyne[2], and Dawn M. Tilbury[1]
1: Department of Mechanical Engineering and Applied Mechanics
2: Department of Electrical Engineering and Computer Science
University of Michigan, Ann Arbor, MI 48109-2125
{fengli, moyne, tilbury}@umich.edu

Introduction

The traditional communication architecture for control systems is point-to-point, which has been successfully implemented in industry for decades. However, expanding physical setups and functionality push the limits of the point-to-point architecture. Hence, a traditional centralized point-to-point control system is no longer suitable to meet new requirements, such as modularity, decentralization of control, integrated diagnostics, quick and easy maintenance, and low cost [1]. Network systems with common-bus architectures, called *Networked Control Systems* (NCSs), provide several advantages such as small volume of wiring and distributed processing. Especially with respect to manufacturing systems, NCS architectures that utilize processing power at each node allow for modularization of functionality and standard interfaces for interchangeability and interoperability. Object-based device models separate generic, device-type specific from manufacturer-specific functionality. Many different network types have been promoted for use in control systems. In this article, we compare three of them: Ethernet bus (CSMA/CD), Token passing bus (e.g., ControlNet) and Controller Area Network (CAN) bus (e.g., DeviceNet).

Distributed systems contain a large number of devices interconnected together to perform the desired operations over a large area of coverage; examples include industrial automation, building automation, office and home automation, intelligent vehicle systems, and advanced aircraft and spacecraft [2-6]. *Data networks* or *control networks* may be used, depending on the information exchanged. *Data networks* are characterized by large data packets, relatively infrequent bursty transmission, and high data rates; they generally do not have hard real-time constraints. *Control networks*, in contrast, must shuttle countless small but frequent packets among a relatively large set of nodes to meet the time-critical requirements. The key element that distinguishes control networks from data networks is the capability to support real-time or time-critical applications [7].

With NCSs, decisions and control functions (including signal processing) can be distributed among controllers on the network. However, when designing a NCS, a new constraint must be accommodated — the limited bandwidth of the communication network. The effective bandwidth of a network is defined as the maximum amount of meaningful data that can be transmitted per unit time, exclusive of headers, padding, stuffing, etc. and the network bandwidth is equal to the number of raw bits transmitted per unit time. Four factors affect the availability and utilization of the network bandwidth: the sampling rates at which the various devices send information over the network, the number of elements that require synchronous operation, the data or message size of the information, and the medium access control sublayer protocol that controls the information transmission [8]. Therefore, to achieve the timing constraints and guarantee the performance of NCSs, network traffic op-

timization and controller design for the devices connected to the network must be analyzed.

The performance metrics of network systems that impact the requirements of control systems include access delay, transmission time, response time, message delay, message collisions (percentage of collision), message throughput (percentage of packets discarded), packet size, network utilization, and determinism boundaries. For control systems, candidate control networks generally must meet two main criteria: bounded time delay and guaranteed transmission; that is, a message should be transmitted successfully within a bounded time delay. Unsuccessfully transmitted or large time-delay messages from a sensor to an actuator, for example, may deteriorate system performance or make systems unstable. Several protocols have been proposed to meet these requirements for control systems. They include Ethernet (IEEE 802.3:CSMA/CD), Token Bus (IEEE 802.4), Token Ring (IEEE 802.5), and CAN (CSMA/AMP). Control networks are typically based on one of two medium access protocols, CAN (used by Smart Distributed System (SDS), DeviceNet [9], and CAN Kingdom) and the Token Ring or Bus (used by Process Field Bus (PROFIBUS) [10], Manufacturing Automation Protocol (MAP) [11], ControlNet [12], and Fiber Distributed Data Interface (FDDI) [13]).

In this article, we consider how each of these types of control networks could be used as a communication backbone for a networked control system connecting sensors, actuators, and controllers. A detailed discussion of the medium access control sublayer protocol for each network is provided. For each protocol, we study the key parameters of the corresponding network when used in a control situation, including network utilization, magnitude of the expected time delay, and characteristics of time delays. Simulation results are presented for several different scenarios, and the advantages and disadvantages of each network are summarized.

Control Network Basics

In this section, we discuss the medium access control (MAC) sublayer protocol of three candidate control networks: Ethernet based networks, ControlNet (a token-bus based network), and DeviceNet (a CAN based network)[1]. The MAC sublayer protocol, which describes the protocol for obtaining access to the network, is responsible for satisfying the time-critical/real-time response requirement over the network and for the quality and reliability of the communication between network nodes [14]. Our discussion and comparison thus focuses on the MAC sublayer protocols.

[1] Note that Ethernet is not a complete protocol solution but only a MAC sublayer definition, whereas ControlNet and DeviceNet are complete protocol solutions. Following popular usage, we use the term "Ethernet" to refer to Ethernet-based complete network solutions.

Ethernet (CSMA/CD)

Ethernet uses the Carrier Sense Multiple Access with Collision Detection (CSMA/CD) mechanism for resolving contention on the communication medium. The CSMA/CD protocol is specified in the IEEE 802.3 network standard and is described briefly as follows [15], [16], [17]. When a node wants to transmit, it listens to the network. If the network is busy, it waits until the network is idle; otherwise it transmits immediately. If two or more nodes listen to the idle network and decide to transmit simultaneously, the messages of these transmitting nodes collide and the messages are corrupted. While transmitting, a node must also listen to detect a message collision. On detecting a collision between two or more messages, a transmitting node stops transmitting and waits a random length of time to retry its transmission. This random time is determined by the standard Binary Exponential Backoff (BEB) algorithm. The time before trying again is randomly chosen between 0 and $(2^i - 1)$ slot times, where i denotes the ith collision event detected by the node and one slot time is the minimum time needed for a round-trip transmission. However, after 10 collisions have been reached, the interval is fixed at a maximum of 1023 slots. After 16 collisions, the node stops attempting to transmit and reports failure back to the node microprocessor. Further recovery may be attempted in higher layers [17].

The Ethernet frame format is shown in Fig. 1 [17], [18]. The total overhead is 26 bytes. The Data Packet Frame size is between 46 and 1500 bytes. There is a nonzero minimum data size requirement because the standard states that valid frames must be at least 64 bytes long, from Destination Address to Checksum (72 bytes including Preamble and Start of Delimiter). If the data portion of a frame is less than 46 bytes, the Pad field is used to fill out the frame to the minimum size. There are two reasons for this minimum size limitation. First, it makes it easier to distinguish valid frames from "garbage." When a transceiver detects a collision, it truncates the current frame, which means that stray bits and pieces of frames frequently appear on the cable. Second, it prevents a node from completing the transmission of a short frame before the first bit has reached the far end of cable, where it may collide with another frame. For a 10-$Mbps$ Ethernet with a maximum length of 2500 m and four repeaters, the minimum allowed frame time or slot time is 51.2 μs, which is the time required to transmit 64 bytes at 10 $Mpbs$ [17].

Advantages: Because of low medium access overhead, Ethernet uses a simple algorithm for operation of the network and has almost no delay at low network loads [11]. No communication bandwidth is used to gain access to the network compared with the token bus or token ring protocol. Ethernet used as a control network commonly uses the 10 $Mbps$ standard (e.g., Modbus/TCP); high-speed (100 $Mbps$ or even 1 $Gbps$) Ethernet is mainly

used in data networks [17], [22].

Disadvantages: Ethernet is a nondeterministic protocol and does not support any message prioritization. At high network loads, message collisions are a major problem because they greatly affect data throughput and time delay which may be unbounded [11], [20]. The Ethernet capture effect existing in the standard BEB algorithm, in which a node transmits packets exclusively for a prolonged time despite other nodes waiting for medium access, causes unfairness, and results in substantial performance degradation [21]. Based on the BEB algorithm, a message may be discarded after a series of collisions; therefore, end-to-end communication is not guaranteed. Because of the required minimum valid frame size, Ethernet uses a large message size to transmit a small amount of data.

Several solutions have been proposed for using Ethernet in control applications. For example, every message could be time-stamped before it is sent. This requires clock synchronization, however, which is not easy to accomplish, especially with a network of this type [22]. Various schemes based on deterministic retransmission delays for the collided packets of a CSMA/CD protocol result in an upper-bounded delay for all the transmitted packets. However, this is achieved at the expense of inferior performance to CSMA/CD at low to moderate channel utilization in terms of delay throughput [14]. Other solutions also try to prioritize CSMA/CD (e.g., LonWorks) to improve the response time of critical packets [23]. Using switched Ethernet by subdividing the network architecture is another way to increase its efficiency [17].

ControlNet (Token Passing Bus)

MAP, PROFIBUS, and ControlNet are typical examples of token-passing bus control networks. These are deterministic networks because the maximum waiting time before sending a message frame can be characterized by the token rotation time. The token bus protocol (IEEE 802.4) allows a linear, multidrop, tree-shaped, or segmented topology [11]. The nodes in the token bus network are arranged logically into a ring, and, in the case of ControlNet, each node knows the address of its predecessor and its successor. During operation of the network, the node with the token transmits data frames until either it runs out of data frames to transmit or the time it has held the token reaches the limit. The node then regenerates the token and transmits it to its logical successor on the network. If a node has no message to send, it just passes the token to the successor node. The physical location of the successor is not important because the token is sent to the logical neighbor. Collision of data frames does not occur, as only one node can transmit at a time. The protocol also guarantees a maximum time between network accesses for each node, and the protocol has provisions to

regenerate the token if the token holder stops transmitting and does not pass the token to its successor. Nodes can also be added dynamically to the bus and can request to be dropped from the logical ring.

The message frame format of ControlNet is shown in Fig. 2 [12]. The total overhead is 7 bytes, including preamble, start delimiter, source MAC ID, cyclic redundancy check (or CRC), and end delimiter. The Data Packet Frame, namely Lpacket or Link Packet Frame, may include several Lpackets that contain the size, control, tag, data, and an individual destination address with total frame size between 0 and 510 bytes. The size field specifies the number of byte pairs (from 3 to 255) contained in an individual Lpacket. Each byte pair must include the size, control, tag, and link data fields.

The ControlNet protocol adopts an implicit token-passing mechanism and assigns a unique MAC ID (from 1 to 99) to each node. As in general token-passing buses, the node with the token can send data; however, there is no real token passing around the network. Instead, each node monitors the source MAC ID of each message frame received. At the end of a message frame, each node sets an "implicit token register" to the received source MAC ID + 1. If the implicit token register is equal to the node's own MAC ID, that node may now transmit messages. All nodes have the same value in their implicit token registers, preventing collisions on the medium. If a node has no data to send, it just sends a message with an empty Lpacket field, called a null frame.

The length of a cycle, called the Network Update Time (NUT) in ControlNet or Token Rotation Time (TRT) in general, is divided into three major parts: scheduled, unscheduled, and guardband, as shown in Fig. 3. During the scheduled part of a NUT, each node can transmit time-critical/scheduled data by obtaining the implicit token from 0 to S. During the unscheduled part of a NUT, each node from 0 to U shares the opportunity to transmit non-time-critical data in a round-robin fashion until the allocated unscheduled duration is expired. When the guardband time is reached, all nodes stop transmitting, and only the node with lowest MAC ID, called the "moderator," can transmit a maintenance message, called the "moderator frame," which accomplishes the synchronization of all timers inside each node and publishing of critical link parameters such as NUT, node time, S, U, etc. If the moderator frame is not heard for two consecutive NUTs, the node with the lowest MAC ID will begin transmitting the moderator frame in the guardband of the third NUT. Moreover, if a moderator node notices that another node has a lower MAC ID than its own, it immediately cancels its moderator role.

Advantages: The token bus protocol is a deterministic protocol that provides excellent throughput and efficiency at high network loads [11], [14]. During network operation, the

token bus can dynamically add nodes to or remove nodes from the network. This contrasts with token ring case, where the nodes physically form a ring and cannot be added or removed dynamically [11]. Scheduled and unscheduled segments in each NUT cycle make ControlNet suitable for both time-critical and non-time-critical messages.

Disadvantages: Although the token bus protocol is efficient and deterministic at high network loads, at low channel traffic its performance cannot match that of contention protocols. In general, when there are many nodes in one logical ring, a large percentage of the network time is used in passing the token between nodes when data traffic is light [14].

DeviceNet (CAN Bus)

CAN is a serial communication protocol developed mainly for applications in the automotive industry but also capable of offering good performance in other time-critical industrial applications. The CAN protocol is optimized for short messages and uses a CSMA/Arbitration on Message Priority (CSMA/AMP) medium access method. Thus the protocol is message-oriented, and each message has a specific priority that is used to arbitrate access to the bus in case of simultaneous transmission. The bit-stream of a transmission is synchronized on the start bit, and the arbitration is performed on the following message identifier, in which a logic zero is dominant over a logic one. A node that wants to transmit a message waits until the bus is free and then starts to send the identifier of its message bit by bit. Conflicts for access to the bus are solved during transmission by an arbitration process at the bit level of the arbitration field, which is the initial part of each frame. Hence, if two devices want to send messages at the same time, they first continue to send the message frames and then listen to the network. If one of them receives a bit different from the one it sends out, it loses the right to continue to send its message, and the other wins the arbitration. With this method, an ongoing transmission is never corrupted.

In a CAN-based network, data are transmitted and received using Message Frames that carry data from a transmitting node to one or more receiving nodes. Transmitted data do not necessarily contain addresses of either the source or the destination of the message. Instead, each message is labeled by an identifier that is unique throughout the network. All other nodes on the network receive the message and accept or reject it, depending on the configuration of mask filters for the identifier. This mode of operation is known as multicast.

DeviceNet, which is based on the CAN specification, is a relatively low-cost communication link connecting devices to a network; it has received considerable acceptance in device-level manufacturing applications. The DeviceNet specification is based on standard CAN (11-bit identifier only[2]) with an additional application and physical layer specification

[2] The CAN protocol supports two message frame formats: standard CAN (version 2.0A, 11-bit identifier) and

[9], [19].

The frame format of DeviceNet is shown in Fig. 4 [9]. The total overhead is 47 bits, which includes start of frame (SOF), arbitration (11-bit Identifier), control, CRC, acknowledgment (ACK), end of frame (EOF), and intermission (INT) fields. The size of a Data Field is between 0 and 8 bytes. The DeviceNet protocol uses the arbitration field to provide source and destination addressing as well as message prioritization.

Advantages: CAN is a deterministic protocol optimized for short messages. The message priority is specified in the arbitration field (11-bit identifier). Higher priority messages always gain access to the medium during arbitration. Therefore, the time delay of transmission of higher priority messages can be guaranteed.

Disadvantages: The major disadvantage of CAN compared with the other networks is the slow data rate (maximum of 500 $Kbps$). Thus the throughput is limited compared with other control networks. The bit synchronization requirement of the CAN protocol also limits the maximum length of a DeviceNet network. CAN is also not suitable for transmission of messages of large data sizes, although it does support fragmentation of data that is more than 8 bytes.

Comparative Analysis of Three Control Networks

Aspects of three typical control networks — Ethernet, ControlNet, and DeviceNet — are compared in Figs. 5 and 6. The parameters used in these figures are listed in Table 1.

As shown in Fig. 5, the transmission time for DeviceNet is longer than the others because the of the lower data rate (500 $Kbps$). Ethernet requires less transmission time on larger data sizes (>20 bytes) compared with the others. Although ControlNet uses less time to transmit the same size of small data, it needs some amount of time (NUT) to gain access to the network.

Fig. 6 shows the data coding efficiency of three control networks versus the data size. The data coding efficiency is defined by the ratio of the data size and the message size (i.e., the total number of bytes used to transmit the data). For small data sizes, DeviceNet is the best among these three types and Ethernet is the worst. For large data sizes, ControlNet and Ethernet are better than DeviceNet (DeviceNet is only 58% efficient, but ControlNet and Ethernet are near 98% efficient for large data size transmission). For control systems, the data size is generally small. Therefore, the above analysis suggests that DeviceNet may be preferable in spite of the slow data rate. Before making that decision, however, the average and total time delay and the throughput of the network must be investigated.

extended CAN (version 2.0B, 29-bit identifier).

The discontinuities seen in Figs. 5 and 6 are caused by data fragmentation (i.e., the maximum size limitation per message). The maximum data sizes are 1500, 504, and 8 bytes for Ethernet, ControlNet, and DeviceNet, respectively. The flat portion of the Ethernet plots for small data sizes is due to the minimum data size requirement (46 bytes).

Timing Analysis of Control Networks

In this section, we characterize the time delay of the three control networks by studying their timing parameters. Fig. 7 shows a general timing diagram of the initialization and ending of the task of sending a message over the network. The time delay of a message, T_{delay}, is defined as the difference between the time when the source node wants to send a message, $\mathcal{T}_{src}$, and the time when the destination node receives this message, $\mathcal{T}_{dest}$, i.e., $T_{delay} = \mathcal{T}_{dest} - \mathcal{T}_{src}$.

The total time delay can be broken into three parts, representing the time delays at the source node, on the network channel, and at the destination node, as shown in Fig. 7. The time delay at the source node includes the preprocessing time, T_{pre}, which is the sum of the computation time, T_{scomp}, and the encoding time, T_{scode}, and the waiting time, T_{wait}, which is the sum of the queue time, T_{queue}, and the blocking time, T_{block}. Depending on the amount of data the source node must send and the traffic on the network, the waiting time may be significant. The network time delay includes the total transmission time of a message and the propagation delay of the network. This will depend on message size, data rate, and the length of the network cable. The time delay at the destination node only includes the postprocessing time, T_{post}, which is the sum of the decoding time, T_{dcode}, and the computation time, T_{dcomp}, at the destination node. The time delay can be explicitly expressed by the following equations:

$$\begin{aligned} T_{delay} &= \mathcal{T}_{dest} - \mathcal{T}_{src} \\ &= T_{pre} + T_{wait} + T_{tx} + T_{post}, \\ &= \underbrace{T_{scomp} + T_{scode}}_{T_{pre}} + \underbrace{T_{queue} + T_{block}}_{T_{wait}} + \underbrace{T_{frame} + T_{prop}}_{T_{tx}} + \underbrace{T_{dcode} + T_{dcomp}}_{T_{post}}. \end{aligned} \quad (1)$$

Because the preprocessing and postprocessing times are typically constant compared with the waiting time and the transmission time, and are a limitation of computer processing parameters rather than network physical and protocol parameters, we will ignore them in the following discussion. The queueing time, T_{queue}, is the time a message waits in the buffer at the source node while previous messages in the queue are sent. It depends on the blocking time of previous messages in queue, the periodicity of messages, and the processing load. Although T_{queue} is difficult to analyze, we include it in our simulations. In some control

applications, old messages are discarded, effectively setting T_{queue} to zero; however, this strategy may require a nonstandard network protocol. In the following subsections, we will analyze the blocking time, frame time, and propagation time for each of the three candidate control networks.

Blocking Time

The blocking time, which is the time a message must wait once a node is ready to send it, depends on the network protocol and is a major factor in the determinism and performance of a control network. It includes waiting time while other nodes are sending messages and the time needed to resend the message if a collision occurs.

Ethernet Blocking Time

We first consider the blocking time for Ethernet, which includes time taken by collisions with other messages and the subsequent time waiting to be retransmitted. The BEB algorithm described earlier indicates a probabilistic waiting time. An exact analysis of expected blocking time delay for Ethernet is very difficult [24]. At a high level, the expected blocking time can be described by the following equation:

$$E\{T_{block}\} = \sum_{k=1}^{16} E\{T_k\} + T_{resid}, \tag{2}$$

where T_{resid} denotes the residual time that is seen by node i until the network is idle, and $E\{T_k\}$ is the expected time of the kth collision. $E\{T_k\}$ depends on the number of backlogged and unbacklogged nodes as well as message arrival rate at each node. For the 16th collision, the node discards this message and reports an error message to the higher level processing units [17]. It can be seen that T_{block} is not deterministic and may be unbounded due to the discarding of messages.

ControlNet Blocking Time

In ControlNet, if a node wants to send a message, it must wait to receive the token from the logically previous node. Therefore, the blocking time, T_{block}, can be expressed by the transmission time and token rotation time of previous nodes. The general formula for T_{block} can be described by the following equation:

$$T_{block} = T_{resid} + \sum_{j \in \mathcal{N}_{noqueue}} T_{token}^{(j)} + \sum_{j \in \mathcal{N}_{queue}} min(T_{tx}^{(j,n_j)}, T_{node}) + T_{guard}, \tag{3}$$

where T_{resid} is residual time needed by the current node to finish transmitting, $\mathcal{N}_{noqueue}$ and $\mathcal{N}_{queue}$ denote the sets of nodes with messages and without messages in the queues,

respectively, and T_{guard} is the time spent on the guardband period, as defined earlier. For example, if node 10 is waiting for the token, node 4 is holding the token and sending messages, and nodes 6, 7, and 8 have messages in their queues, then $\mathcal{N}_{noqueue} = \{5, 9\}$ and $\mathcal{N}_{queue} = \{4, 6, 7, 8\}$. Let n_j denote the number of messages queued in the jth node and let T_{node} be the maximum possible time (i.e., token holding time) assigned to each node to fully utilize the network channel; for example, in ControlNet $T_{node} = 827.2\ \mu s$ which is a function of the maximum data size, overhead frame size, and other network parameters. T_{token} is the token passing time, which depends on the time needed to transmit a token and the propagation time from node $i-1$ to node i. ControlNet uses an implicit token, and T_{token} is simply the sum of T_{frame} with zero data size and T_{prop}. If a new message is queued for sending at a node while that node is holding the token, then $T_{block} = T_{tx}^{(j,n_j)}$, where j is the node number. In the worst case, if there are N master nodes on the bus and each one has multiple messages to send, which means each node uses the maximum token holding time, then $T_{block} = \sum_{i \in \mathcal{N}_{node} \setminus \{j\}} \min(T_{tx}^{(i,n_i)}, T_{node})$, where the min function is used because, even if it has more messages to send, a node cannot hold the token longer than T_{node} (i.e., $T_{tx}^{(j,n_j)} \leq T_{node}$). ControlNet is a deterministic network because the maximum time delay is bounded and can be characterized by (3). If the periods of each node and message are known, we can explicitly describe the sets $\mathcal{N}_{noqueue}$ and $\mathcal{N}_{queue}$ and n_j. Hence, T_{block} in (3) can be determined explicitly.

DeviceNet Blocking Time

The blocking time, T_{block}, in DeviceNet can be described by the following equation [25]:

$$T_{block}^{(k)} = T_{resid} + \sum_{\forall j \in \mathcal{N}_{hp}} \lceil \frac{T_{block}^{(k-1)} + T_{bit}}{T_{peri}^{(j)}} \rceil \ T_{tx}^{(j)}, \tag{4}$$

where T_{resid} is residual time needed by the current node to finish transmitting, $\mathcal{N}_{hp}$ is the set of nodes with higher priority than the waiting node, $T_{peri}^{(j)}$ is the period of the jth message, and $\lceil x \rceil$ denotes the smallest integer number that is greater than x. The summation denotes the time needed to send all the higher priority messages. For a low-priority node, while it is waiting for the channel to become available, it is possible for other high-priority nodes to be queued, in which case the low-priority node loses the arbitration again. This situation accumulates the total blocking time. The worst-case T_{resid} under a low traffic load is:

$$T_{resid} = \max_{\forall j \in \mathcal{N}_{node}} T_{tx}^{(j)}, \tag{5}$$

where $\mathcal{N}_{node}$ is the set of nodes on the network. However, because of the priority-arbitration mechanism, low-priority node/message transmission may not be deterministic or bounded under high loading.

Frame Time

The frame time, T_{frame}, depends on the size of the data, the overhead, any padding, and the bit time. Let N_{data} be the size of data in terms of bytes, N_{ovhd} be the number of bytes used as overhead, N_{pad} be the number of bytes used to pad the remaining part of the frame to meet the minimum frame size requirement, and N_{stuff} be the number of bytes used in a stuffing mechanism (on some protocols). The frame time can then be expressed by the following equation:

$$T_{frame} = [N_{data} + N_{ovhd} + N_{pad} + N_{stuff}] \times 8 \times T_{bit}. \quad (6)$$

The values N_{data}, N_{ovhd}, N_{pad}, and N_{stuff}[3] can be explicitly described for the Ethernet, ControlNet, and DeviceNet protocols [24].

Propagation Time

The propagation time, T_{prop}, depends on the signal transmission speed and the distance between the source and destination nodes. For the worst case, the propagation delays from one end to the other of the network cable for these three control networks are: $T_{prop} = 25.6$ μs for Ethernet (2500 m), $T_{prop} = 10$ μs for ControlNet (1000 m), and $T_{prop} = 1$ μs for DeviceNet (100 m). The length in parentheses represents the typical maximum cable length used. The propagation delay is not easily characterized because the distance between the source and destination nodes is not constant among different transmissions. For comparison, we will assume that the propagation times of these three network types are the same, say, $T_{prop} = 1$ μs (100 m). Note that T_{prop} in DeviceNet is generally less than one bit time because DeviceNet is a bit-synchronized network. Hence, the maximum cable length is used to guarantee the bit synchronization among nodes.

Case Studies

In this section, we define critical network parameters, and then study two cases of networked control systems: a control network system with 10 nodes, each with 8 bytes of data to send every period, and an SAE vehicle example with 53 nodes [25]. Matlab[4] is used to simulate the MAC sublayer protocols of the three control networks. Network parameters such as the number of nodes, the message periods, and message sizes can be specified in the simulation model. In our study, these network parameters are constant. The simulation

[3] The bit-stuffing mechanism in DeviceNet is as follows: if more than 5 bits in a row are '1', then a '0' is added and vice versa. Ethernet and ControlNet use Manchester biphase encoding, and, therefore, do not require bit-stuffing.

[4] Matlab is a technical computing software developed by the Mathworks Inc.

program records the time delay history of each message and calculates network performance statistics such as the average time delay seen by messages on the network, the efficiency and utilization of the network, and the number of messages that remain unsent at the end of the simulation run.

For control network operation, the message connection type must be specified. Practically, there are three types of message connections: strobe, poll, and change of state (COS)/cyclic. In a *strobe* connection, the master device broadcasts a strobed message to a group of devices and these devices respond with their current condition. In this case, all devices are considered to sample new information at the same time. The time delay between sampling at the source device and receiving at the destination device is the sum of the transmission time and the waiting time at the source node. In a *poll* connection, the master sends individual messages to the polled devices and requests update information from them. Devices only respond with new signals after they have received a poll message. *COS/cyclic* devices send out messages either when their status is changed (COS) or periodically (cyclic). Although COS/cyclic seems most appropriate from the traditional control systems point of view, strobe and poll are commonly used in industrial control networks [9].

Based on these different types of message connections, we consider the following three releasing policies. The first policy, which we call the "zero releasing policy," assumes every node tries to send its first message at $t = 0$ and sends a new message every period. This type of situation occurs when a system powers up and there has been no prescheduling of messages or when there is a strobe request from the master. The second policy, namely, the "random releasing policy," assumes a random start time for each node; each node still sends a new message every period. The possible situation for this releasing policy is the COS or cyclic messaging where no pre-schedule is done. In the third policy, called "scheduled releasing policy", the start-sending time is scheduled to occur (to the extent possible) when the network is available to the node; this occurs in a polled connection.

In addition to varying the release policy, we also change the period of each node to demonstrate the effect of traffic load on the network. For each releasing policy and period, we calculate the average time delays of these ten nodes and the efficiency and utilization of the three different control networks; we also record the number of unsent and failure or discarded messages of each network. Further, we examine the effect of node numbering on the network performance. We then compare the simulation results to the analytic results described above. For ControlNet and DeviceNet, the maximum time delay can be explicitly determined. For Ethernet, the expected value of the time delay can be computed using the BEB algorithm once the releasing policy is known.

Average Time Delays

For a given running time, say $T_{max} = 10\ sec$, we can calculate the average time delays from each node for each network.

$$T_{delay}^{avg} = \frac{1}{N} \sum_{i \in \mathcal{N}_{node}} [\frac{\sum_{j=1}^{M^{(i)}} T_{delay}^{(i,j)}}{M^{(i)}}], \tag{7}$$

where N is the total number of nodes, $\mathcal{N}_{node}$ is the set of nodes, and $M^{(i)}$ is the number of messages requested at node i. We assume all messages are periodic; thus the total number of messages is equal to the total running time divided by the period of messages (i.e., $M^{(i)} = \lfloor T_{max}/T_{peri}^{(i)} \rfloor$, where $\lfloor x \rfloor$ denotes the largest integer number less than x). The average delay can be computed for the entire network, as shown, or for the ith node.

Network Efficiency

We will define the efficiency of a network, P_{eff}, as the ratio of the total transmitting time to the time used to send messages, including queueing time, blocking time, and so on; that is,

$$P_{eff} = \frac{\sum_{i \in \mathcal{N}_{node}} \sum_{j=1}^{M^{(i)}} T_{tx}^{(i,j)}}{T_{delay}^{sum}}. \tag{8}$$

Therefore, $P_{eff} \rightarrow 1$ denotes that all the time delay is due to the transmission delay and the network performance is good. On the other hand, $P_{eff} \rightarrow 0$ means that most of the time delay is due to message contention or collision.

Network Utilization

The utilization of a network, P_{util}, is defined by the ratio of the total time used to transmit data and the total running time; that is,

$$P_{util} = \frac{\sum_{i \in \mathcal{N}_{node}} \sum_{j=1}^{M^{(i)}} (T_{tx}^{(i,j)} + T_{retx}^{(i,j)})}{T_{max}}, \tag{9}$$

where $T_{retx}^{(i,j)}$ is the time taken to retransmit the (i,j)th message. P_{util} describes the percentage of effective bandwidth used by the nodes or, conversely, the utilization of the network. If $P_{util} \rightarrow 0$, there is sufficient bandwidth left on the network for other purposes. If $P_{util} \rightarrow 1$, the network is saturated, and we have to redesign the network layout or reassign the traffic load. Note that for Ethernet, under high loading condition, P_{util} can approach 1. However, effective data communication can approach zero (i.e., $P_{eff} \rightarrow 0$) because P_{util} is dominated by $T_{retx}^{(i,j)}$.

Number of Unsent Messages

Control systems need required information to be transmitted successfully and immediately. If the information transmission on a network induces lost messages, the system performance may deteriorate or even become unstable. Hence, it is very important to evaluate a network protocol according to the number or the possibility of unsent messages.

Stability of Networked Control Systems

The stability of a network itself is defined by the number of messages in the queue of each node. If this number is larger than a certain constant or tends to infinity as time increases, the network is said to be unstable (even though we assume infinite buffer size). On the other hand, the stability of a networked control system should be defined by the performance of both the network and the control system. That is, when the networks are unstable (i.e., increasing queue lengths), and the network-induced time delays degrade the control systems sufficiently, the networked control system can become unstable. However, note that it is possible to have a system with an unstable network but a stable control system and vice versa. In this study, we only consider the stability of a network.

If sensors sample the data faster than the network can transmit the data, then the network will be saturated and data will be queued at the buffer (unless it is discarded). In designing a networked control systems, both the effective bandwidth and the sensors' sampling rate must be considered. Although high sampling rates improve the control performance in traditional control systems, they also induce high traffic loads on the network medium. High traffic loads increase the message time delays and can degrade the control performance. A preliminary case study on the tradeoff between sampling time and network traffic has been performed [26].

Case Study 1: 10 Nodes, Equal Periods

In this case study of 10 nodes, with equal periods and data sizes, we examine nine different periods with a total simulation time of 0.5 *sec*. The periods were chosen based on the total time to send all 10 messages over the different networks; 576, 320, and 2200 μs, for Ethernet, ControlNet, and DeviceNet, respectively. The three releasing policies (zero, random, scheduled) were also examined. The simulation results for a message period of 5000 μs are summarized in Table 2 and Figs. 8 – 10.

When the message period is longer than the total message transmission time, the performance indices are similar for all three networks. Because of the different network data rates, however, the message period at which saturation occurs is different for the three networks. When the period is shorter than the total transmission time (i.e., the traffic load is

larger than the availability of network), the utilization approaches 100% and the time delay increases (it will become unbounded as the simulation time goes to infinity). Therefore, the network becomes unstable.

The average time delays for the three releasing policy are shown in Figure 8. The zero releasing policy has the longest average delays in every network because all nodes experience contention when trying to send messages. Although the Ethernet data rate is much faster than DeviceNet, the delays due to collisions and the large required message size combine to increase the average time delay for Ethernet in this case. For a typical random releasing policy, average time delays are reduced because not all nodes try to send messages (or experience network contention) at the same time, although some contention still exists. The scheduled releasing policy makes the best use of each individual network; the average time delay of these releasing policies is only the transmission time. When the networks are not saturated, the average time delay is just equal to the frame time (i.e., the time taken to transmit a message frame). In this case, all three networks maintain a constant time delay.

For the two deterministic control networks (i.e., DeviceNet and ControlNet), if the network is not saturated (i.e., $P_{util} \leq 100\%$), there are no messages queued at the buffer. However, for Ethernet, there are always some messages queued at the buffer, and, moreover, some messages are discarded due to the BEB algorithm. We also notice that the average time delay of each message is not constant, even though the network is not saturated. The discarded messages and nonconstant time delay may make Ethernet unsuitable in this loading situation for control applications.

Table 2 compares the performance of three releasing policies for a message period of 5000 μs. The network efficiency is low for the zero and random releasing policies, but can reach 100% in the scheduled releasing policy if the traffic load is not saturated. With a message period of 5000 μs, none of the networks are saturated, and the network utilization of ControlNet, and DeviceNet is the same for the three releasing policies. However, for Ethernet, the network utilization in the zero and random releasing policies is different from that in the scheduled releasing policy because, when messages collide, there are multiple transmissions for one message.

Again using the 5000 μs period, we also demonstrate the time delay history of the messages sent by three different nodes on the network (nodes 1, 5, and 10 in Fig. 9). DeviceNet is the only network that can guarantee constant time delay for all three releasing policies; this is due to the priority arbitration mechanism and the periodicity of messages. Hence, the qualitative performance of DeviceNet is independent of releasing policy in this case (i.e. when the network is not saturated). As the traffic load increases (high frequency of messages), only higher priority nodes can gain access to the network and maintain bounded

transmission time delays, but low-priority messages cannot access the network and remain unsent. Using the scheduled and random releasing policies, the Ethernet message collision probability is low at low message rates. Hence, Ethernet generally has a constant time delay in low traffic loads. However, when collisions occur as in the zero or random releasing policy, the time delay is not constant and is difficult to predict, as shown in Figs. 9(a) and (b). Although ControlNet only exhibits a constant time delay under scheduled releasing policy conditions, it always guarantees a bounded time delay when the traffic load is not saturated, no matter what the releasing policy, as shown in Fig. 9.

The delays experienced by all the messages on the network are combined in Fig. 10. Again, with a message period of 5000 μs, none of the networks are saturated. In Ethernet, shown in Fig. 10(a), the zero and random releasing policies demonstrate the nondeterministic property of time delay in Ethernet, even though the traffic load is not saturated. Fig. 10(b) shows that message time delay of ControlNet is bounded for all releasing policies; we can estimate the lower and upper bounds based on the formulae derived in the timing analysis section. Due to the asynchronicity between the message period and the token rotation period, these time delays exhibit a linear trend with respect to the message number. The simulation results for DeviceNet, shown in Fig. 10(c), demonstrate that every node in DeviceNet has a constant time delay which depends only on the message number. The estimated mean time delay (1091) for Ethernet in Fig. 10(a) is computed for the case of the zero releasing policy from (2) and the variance is taken as twice the standard deviation. This estimated value is close to the simulated value (1081) given in Table 2. The maximum and minimum time delays for ControlNet and DeviceNet are computed from (3), (4), and (6).

Case Study 2: SAE Benchmark

Next we consider a benchmark application from the SAE [25]. We simulate the specified traffic flow on three control networks. In this example, there are 53 nodes that belong to seven subsystems, namely, the batteries, the vehicle controller, the inverter/motor controller, the instrument panel display, the driver inputs, the brakes, and the transmission control. Not all nodes have the same period, but each node has its own constant period of messages needed to be sent. Because the minimum data element in all three networks is one bytes, a message of 1 or 8 bits has the same transmission time. To study the impact of data size on network performance, we increase the data size at each node by a factor of 8, e.g., changing node 1 from 8 bits in [25] to 8 bytes. The data size and period of each node are shown in Table 3. We also apply the three releasing policies (zero, random, and scheduled) as discussed earlier. The total simulation time is 10 *sec*. The simulation results are shown in Figs. 11 and 12, and the summary is given in Table 4.

Table 4 illustrates three cases based on three different node numbering assignments: the node numbering in case I is ordered based on the functional groups, and the other two cases are ordered by message periods (case II is decreasing and case III is increasing). Note that the total transmission times for all 53 messages are 3052.8, 1366.4, and 8470 in Ethernet, ControlNet, and DeviceNet, respectively.

The comparison of case I simulation results for the three control networks is shown in Fig. 11. For the zero releasing policy, Ethernet has a larger message time delay value and standard deviation among all 53 nodes. This is due to the high probability of message collision with the zero releasing policy. For DeviceNet, as expected, there is a linear trend for message time delay over the message node numbers. However, for ControlNet, the average message time delay is nearly the same over all 53 nodes. This is because the medium access control in ControlNet is based on the token rotation mechanism, and every node gains access to the network medium at a deterministic period. Fig. 11 also shows lower and consistent variability of time delays for ControlNet and DeviceNet compared to that of Ethernet.

For the random releasing policy, ControlNet still demonstrates a constant time delay over all 53 nodes, but Ethernet and DeviceNet do not. However, for DeviceNet, nodes with small node numbers still have a shorter message time delay than nodes with large node numbers. The average message time delays of the random releasing policy are shorter than those of the zero releasing policy, as shown in Table 4. Ethernet still has high variability of time delay compared to ControlNet and DeviceNet.

For the scheduled releasing policy, Ethernet demonstrates outstanding results because of its high transmission speed and zero possibility of message collisions. Because the total transmission time of all 53 messages is 8470 μs in DeviceNet, the last few nodes with short message periods (e.g., nodes 39–53) have higher variability of message time delays, especially, nodes 39, 42, 43, and 49, which have large data sizes and shorter message periods. However, the results for ControlNet are similar for all releasing policies. For multiperiod systems, ControlNet is less sensitive to the releasing policies, and it is very difficult to coordinate the system parameters and maintain the minimum time delay, as we did in the 10-node simulation.

In this SAE example setting, shown in Table 3, there are six different message periods. To study the effect of node numbering, we regroup these nodes by their message periods. We first let those nodes with shorter message periods have lower node numbers and find that the nodes within one group have similar performance for the ControlNet and DeviceNet systems but vary between node groups. However, there is not much difference in performance between node groups for Ethernet. Note that the simulation results within one group are similar to the 10-node example we studied in the first part of this section.

Fig. 12 shows the time delay characteristics of two releasing policies in DeviceNet for three cases of node numbering. Because of the priority mechanism on node numbers, the time delay profile exhibits a linear trend over node numbers. However, there is little variance for the ordered by increasing message period case in the zero releasing policy. This is because high frequency nodes are high priority nodes and have little difficulty accessing the network medium. For the scheduled releasing policy, both the average time delay and the variance are smaller than those in the zero releasing policy. The last few nodes of the decreasing message period numbering case have large time delay variance with both releasing policies because they are lower priority nodes with high frequency messages and always lose the contention battle when one arises. The gap between nodes 33 and 34 due to the fact that the total transmission times from nodes 1 to 33 is more than 5000 μs and node 34 always loses the contention to those nodes with higher priority. A similar situation occurs between nodes 51 and 52.

Conclusion and Future Work

The features of three candidate control networks — Ethernet (CSMA/CD), ControlNet (Token Bus), and DeviceNet (CAN) — were discussed in detail. We first described the medium access control mechanisms of these protocols, which are responsible for satisfying both the time-critical/real-time response requirement over the network and the quality and reliability of communication between devices on the network. These timing parameters, which will ultimately influence control applications, are affected by the network data rate, the period of messages, the data or message size of the information, and the communication protocol. For each protocol, we studied the key performance parameters of the corresponding network when used in a control situation, including network utilization, magnitude of the expected time delay, and characteristics of time delays. Simulation results were presented for several different scenarios. The timing analyses and comparisons of message time delay given in this article will be useful for anyone designing a networked control system.

The evaluation study described here is based on the characteristics and requirements of control systems. The characteristics are small data sizes and periodic messages, and the requirements include guaranteed transmission and bounded time delay. The suitability of network protocols for use in control systems is greatly influenced by these two criteria. Although Ethernet has seen widespread use in many data transmission applications and can support high data rates up to 1 *Gbps*, it may not be suitable as the communication medium for some control systems when compared with deterministic network systems. However,

because of its high data rate, Ethernet can be used for aperiodic/non-time-critical and large data size communication, such as communication between workstations or machine cells. For intra-workstation cell communication with controller, sensors, and actuators, deterministic network systems are generally more suitable for meeting the characteristics and requirements of control systems. For control systems with short and/or prioritized messages, DeviceNet demonstrates better performance. The scheduled and unscheduled messaging capabilities in ControlNet make it suitable for time-critical and non-time-critical messages. ControlNet is also suitable for large data size message transmission.

Our future efforts will focus on controller design for networked control systems, which can differ significantly from the design of traditional centralized control systems. This work will include conducting experimental studies of control networks for control applications. We plan to use this analysis, along with performance evaluation of candidate control networks presented here, as the basis for future message scheduling and control algorithm designs for networked control systems.

Acknowledgments

The authors wish to acknowledge the contributions of John Korsakas and the support of the National Science Foundation Engineering Research Center for Reconfigurable Machining Systems at the University of Michigan under grant EEC95-92125, the Open DeviceNet Vendor Association, and ControlNet International.

References

[1] L.H. Eccles, "A smart sensor bus for data acquisition," *Sensors*, vol. 15, no. 3, pp. 28–36, 1998.

[2] S. Biegacki and D. VanGompel, "The application of DeviceNet in process control," *ISA Transactions*, vol. 35, no. 2, pp. 169–176, 1996.

[3] L.D. Gibson, "Autonomous control with peer-to-peer I/O networks," *Sensors*, vol. 12, no. 9, pp. 83–90, Sep. 1995.

[4] A. Ray, "Introduction to networking for integrated control systems," *IEEE Control Systems Magazine*, vol. 9, no. 1, pp. 76–79, Jan. 1989.

[5] G. Schickhuber and O. McCarthy, "Distributed fieldbus and control network systems," *Computing & Control Engineering*, vol. 8, no. 1, pp. 21–32, Feb. 1997.

[6] D. Song, T. Divoux, and F. Lepage, "Design of the distributed architecture of a machine-tool using FIP fieldbus," in *Proceedings of IEEE Int'l Conf. on Application-specific Systems, Architectures, Processors*, Los Alamitos, CA, pp. 250–260, 1996.

[7] R.S. Raji, "Smart networks for control," *IEEE Spectrum*, vol. 31, no. 6, pp. 49–55, June 1994.

[8] Y. Koren, Z.J. Pasek, A.G. Ulsoy, and U. Benchetrit, "Real-time open control architectures and system performance," *CIRP Annals - Manufacturing Technology*, vol. 45, no. 1, pp. 377–380, 1996.

[9] *DeviceNet Specifications*, Boca Raton, Florida, Open DeviceNet Vendors Association, 2.0 edition, 1997.

[10] G. Cena, C. Demartini, and A Valenzano, "On the performances of two popular fieldbuses," in *Proceedings of IEEE International Workshop on Factory Communication Systems*, Barcelona, Spain, pp. 177–186, Oct. 1997.

[11] J.D. Wheelis, "Process control communications: Token Bus, CSMA/CD, or Token Ring?" *ISA Transactions*, vol. 32, no. 2, pp. 193–198, July 1993.

[12] *ControlNet Specifications*, Boca Raton, Florida, ControlNet International, 1.03 edition, 1997.

[13] S. Saad-Bouzefrane and F. Cottet, "A performance analysis of distributed hard real-time applications," in *Proceedings of IEEE International Workshop on Factory Communication Systems*, Barcelona, Spain, pp. 167–176, Oct. 1997.

[14] S.A. Koubias and G.D. Papadopoulos, "Modern fieldbus communication architectures for real-time industrial applications," *Computers in Industry*, vol. 26, no. 3, pp. 243–252, Aug. 1995.

[15] D. Bertsekas and R. Gallager, *Data Networks*, Englewood Cliffs, NJ, Prentice-Hall Inc., second edition, 1992.

[16] B.J. Casey, "Implementing Ethernet in the industrial environment," in *IEEE Industry Applications Society Annual Meeting*, Seattle, WA, vol. 2, pp. 1469–1477, Oct. 1990.

[17] A.S. Tanenbaum, *Computer Networks*, Upper Saddle River, NJ, Prentice-Hall Inc., third edition, 1996.

[18] H. Bartlett and J. Harvey, "The modeling and simulation of a pick and place computer-integrated manufacturing (CIM) cell," *Computers in Industry*, vol. 26, no. 3, pp. 253–260, Aug. 1995.

[19] G. Paula, "Building a better fieldbus," *Mechanical Engineering*, pp. 90–92, June 1997.

[20] V.K. Khanna and S. Singh, "An improved 'piggyback Ethernet' protocol and its analysis," *Computer Networks and ISDN Systems*, vol. 26, no. 11, pp. 1437–1446, Aug. 1994.

[21] K.K. Ramakrishnan and H. Yang, "The Ethernet capture effect: Analysis and solution," in *19th Conference on Local Computer Networks*, Minneapolis, MN, pp. 228–240, Oct. 1994.

[22] J. Eidson and W. Cole, "Ethernet rules closed-loop system," *InTech*, pp. 39–42, June 1998.

[23] J.R. Moyne, N. Najafi, D. Judd, and A. Stock, "Analysis of Sensor/Actuator Bus Interoperability Standard Alternatives for Semiconductor Manufacturing," in *Sensors Expo Conference Proceedings*, Sep. 1994.

[24] F.-L. Lian, J.R. Moyne, and D.M. Tilbury, "Performance evaluation of control networks: Ethernet, ControlNet, and DeviceNet," Technical Report: UM-MEAM-99-02, *http://www.eecs.umich.edu/~impact*, Feb. 1999.

[25] K. Tindell, A. Burns, and A.J. Wellings, "Calculating Controller Area Network (CAN) message response times," *Control Engineering Practice*, vol. 3, no. 8, pp. 1163–1169, Aug. 1995.

[26] F.-L. Lian, J.R. Moyne, and D.M. Tilbury, "Control performance study of a Networked Machining Cell," To appear in *Proceedings of American Control Conference*, Chicago, Illinois, June 2000.

Nomenclature

All variables defined in this article are time-varying;

N_{data}	:	data size (bytes)
N_{ovhd}	:	overhead size (bytes)
N_{pad}	:	padding frame (bytes)
N_{stuff}	:	stuffing frame (bytes)
P_{eff}	:	network efficiency (%)
P_{util}	:	network utilization (%)
T_{block}	:	blocking time at the source node
T_{bit}	:	bit time
T_{dcode}	:	decoding time at the destination node
T_{dcomp}	:	computing time at the destination node
T_{delay}	:	time delay of a message
T_{frame}	:	frame time on the network medium
T_{guard}	:	guardband period
T_{max}	:	maximum simulation time
T_{node}	:	maximum token holding time
T_{peri}	:	message period
T_{retx}	:	retransmission time on the network medium
T_{post}	:	postprocessing time at the destination node
T_{pre}	:	preprocessing time at the source node
T_{prop}	:	propagation time on the network medium
T_{queue}	:	queueing time at the source node
T_{resid}	:	residual time
T_{scode}	:	encoding time at the source node
T_{scomp}	:	computing time at the source node
T_{tx}	:	transmission time on the network medium
T_{token}	:	token passing time from one node to the other
T_{wait}	:	waiting time at the source node
$\mathcal{N}_{hp}$	:	the set of high-priority nodes in DeviceNet
$\mathcal{N}_{node}$	:	the set of nodes on the network
$\mathcal{N}_{noqueue}$	:	the set of nodes without messages queued
$\mathcal{N}_{queue}$	:	the set of nodes with messages queued
$\mathcal{T}_{dest}$	:	time instant at destination node
$\mathcal{T}_{src}$	:	time instant at destination node

Table 1.
Typical System Parameters of Control Networks

	Ethernet	ControlNet	DeviceNet
Data rate[a] ($Mbps$)	10	5	0.5
Bit time (μs)	0.1	0.2	2
Max. length (m)	2500	1000	100
Max. data size ($byte$)	1500	504	8
Min. message size ($byte$)[b]	72[c]	7	47/8[d]
Max. number of nodes	>1000	99	64
Typical Tx speed (m/s)	coaxial cable: 2×10^8		

a: typical data rate; b: zero data size;
c: including the preamble and start of delimiter fields;
d: DeviceNet overhead is 47 bits.

Table 2.
Simulation Result of Three Releasing Policies with Message Period of 5000 μs (10-node case).

Releasing Policies	Zero	Random	Scheduled
Average time delay (μs)			
Ethernet	1081	172	58
ControlNet	241	151	32
DeviceNet	1221	620	222
Efficiency (%)			
Ethernet	5.3	33.0	100
ControlNet	13.3	21.1	100
DeviceNet	18.2	35.8	100
Utilization (%)			
Ethernet	34.5	16.4	11.5
ControlNet	6.4	6.4	6.4
DeviceNet	44.4	44.4	44.4

Table 3.
A Benchmark Application of SAE

The data size (byte) and message period (ms) of the 53 nodes. Not all nodes have the same period, but each node has its own constant period of messages needed to be sent. These 53 nodes can be grouped into six groups by their message periods, which are 5 ms (8 nodes), 10 ms (2 nodes), 20 ms (1 node), 50 ms (30 nodes), 100 ms (6 nodes), and 1000 ms (6 nodes).

Node	**Data Size** (byte)	**Period** (ms)	**Node**	**Data Size** (byte)	**Period** (ms)	**Node**	**Data Size** (byte)	**Period** (ms)
1	8	100	21	2	1000	41	1	50
2	8	100	22	3	50	42	8	5
3	8	1000	23	1	50	43	8	5
4	8	100	24	1	50	44	1	50
5	8	1000	25	1	50	45	1	50
6	8	100	26	1	50	46	1	50
7	8	5	27	1	50	47	1	50
8	8	5	28	1	50	48	1	50
9	8	5	29	8	10	49	8	5
10	8	100	30	8	10	50	2	50
11	8	5	31	2	50	51	8	50
12	8	100	32	8	5	52	1	50
13	1	1000	33	1	1000	53	8	50
14	4	50	34	8	50			
15	1	50	35	1	50			
16	1	50	36	2	1000			
17	2	50	37	1	50			
18	1	20	38	1	50			
19	1	50	39	7	50			
20	3	50	40	1	50			

Table 4.
Simulation Results of SAE Example: Three Node Numbering Cases

Node Numbering	**Case I Subsystem Ordering**			**Case II Decreasing Message Period Ordering**			**Case III Increasing Message Period Ordering**		
Releasing Policy	Zero	Rand.	Sch'd	Zero	Rand.	Sch'd	Zero	Rand.	Sch'd
Average time delay (μs)									
Ethernet	3268	122	58	3377	103	58	3362	80	58
ControlNet	558	417	439	555	406	458	562	407	444
DeviceNet	2176	428	453	2021	305	568	2202	253	386
Efficiency (%)									
Ethernet	1.7	47.2	100.0	1.8	55.8	100.0	1.8	71.8	100.0
ControlNet	5.6	7.5	7.1	5.6	7.7	6.8	5.6	7.7	7.0
DeviceNet	9.1	46.2	43.7	9.8	64.8	34.8	9.0	78.1	51.2
Utilization (%)									
Ethernet	47.6	16.6	14.5	47.5	16.3	14.5	47.5	15.4	14.5
ControlNet	7.9	7.9	7.9	7.9	7.9	7.9	7.9	7.9	7.9
DeviceNet	49.7	49.7	49.7	49.7	49.7	49.7	49.7	49.7	49.7

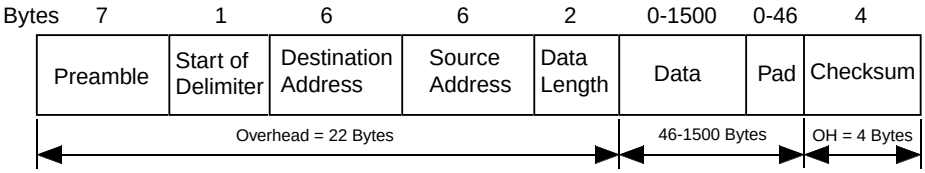


Figure 1. Ethernet (CSMA/CD) frame format.

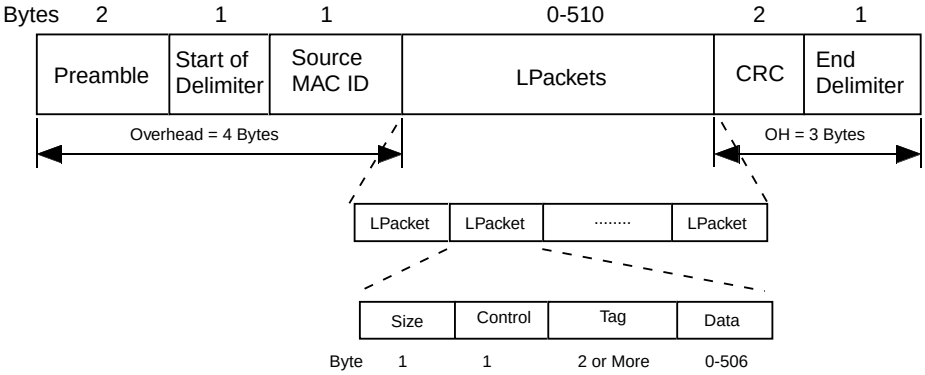


Figure 2. The message frame of ControlNet (Token Bus).

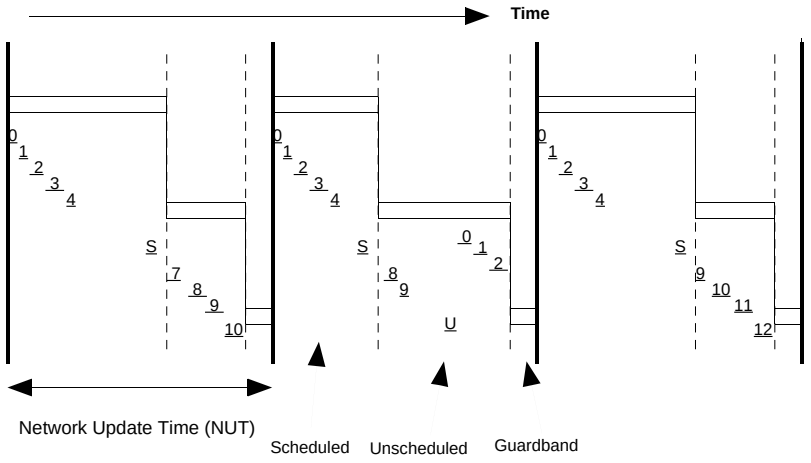


Figure 3. Medium access during scheduled, unscheduled, and guardband time.

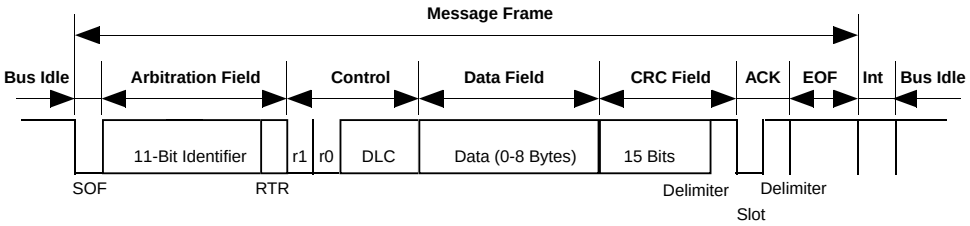


Figure 4. The message frame format of DeviceNet (standard CAN format).

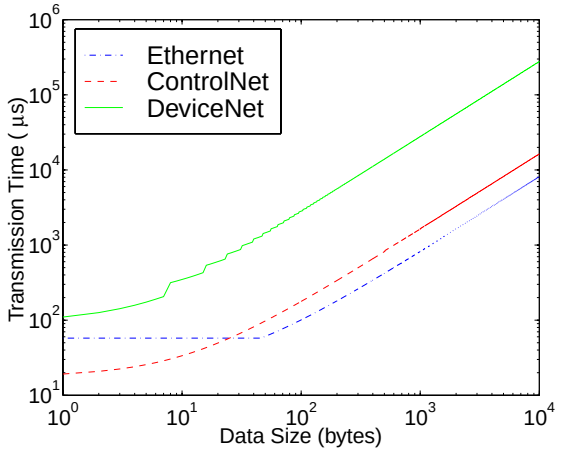


Figure 5. A comparison of the transmission time vs. the data size for the three networks.

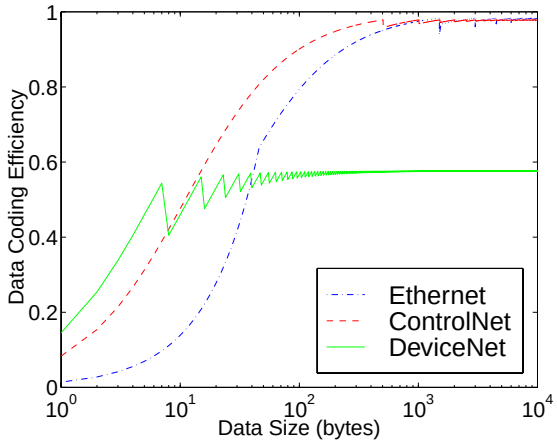


Figure 6. A comparison of the data coding efficiency vs. the data size for each network.

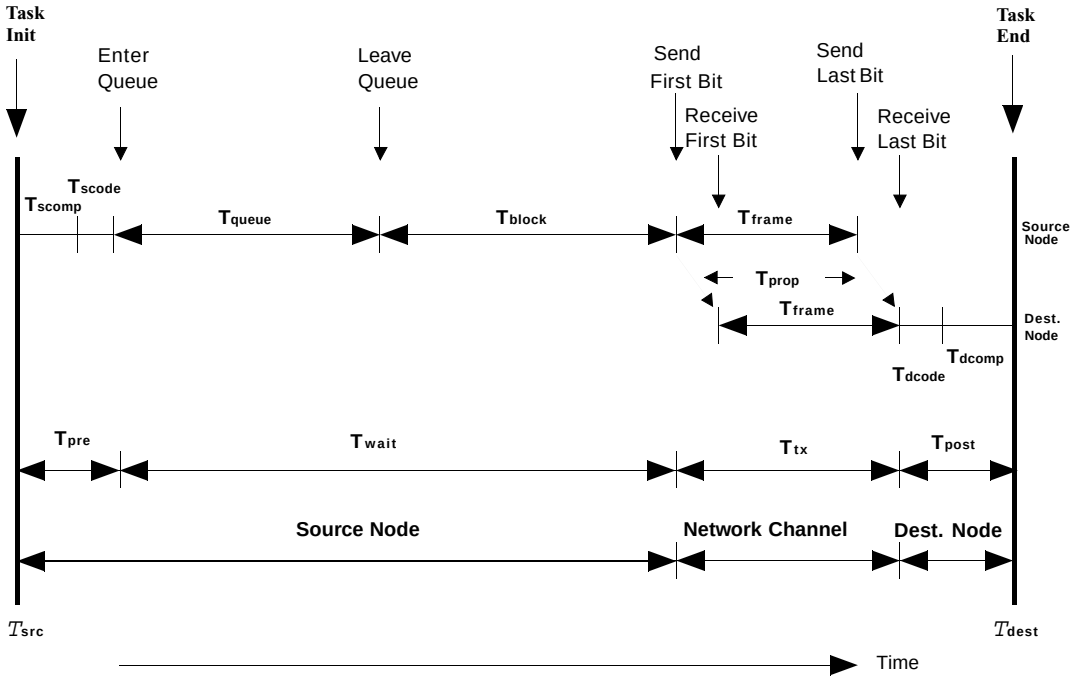


Figure 7. A timing diagram showing time spent sending a message from a source node to a destination node.

$T_{delay} = \mathcal{T}_{dest} - \mathcal{T}_{src} = T_{pre} + T_{wait} + T_{tx} + T_{post}$. Delays occur at the source node due to computation and coding of the message, queueing at the source node, and blocking due to network traffic. Once the message is sent, there is a propagation time delay (due to physical length of network) and a transmission time delay (due to message size). At the destination node, there are again decoding and computation delays before the data can be used.

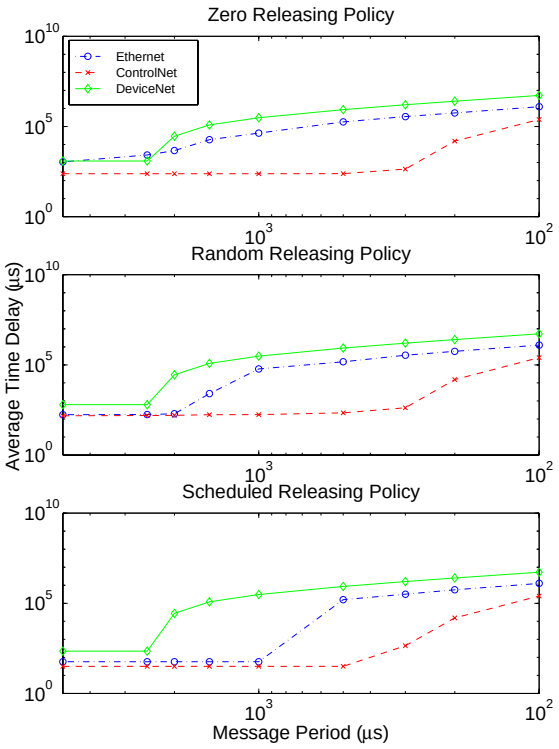


Figure 8. A comparison of the average time delay vs. message period (10-node case).

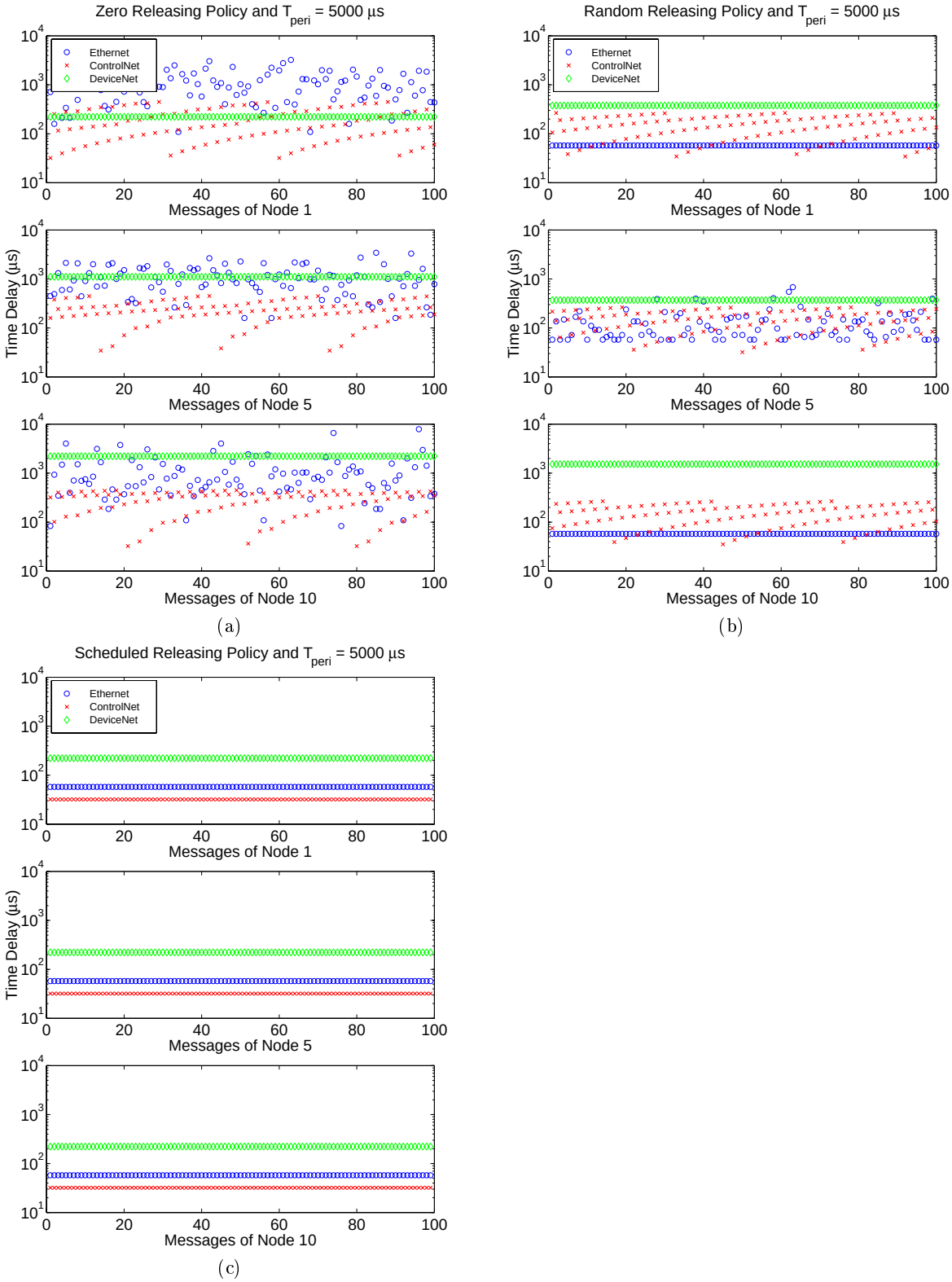
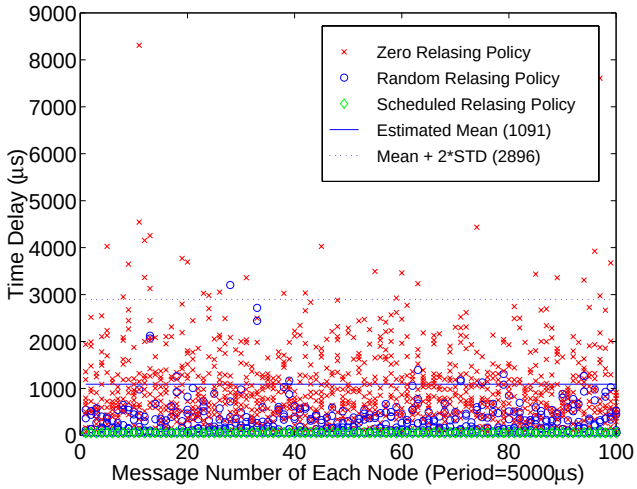
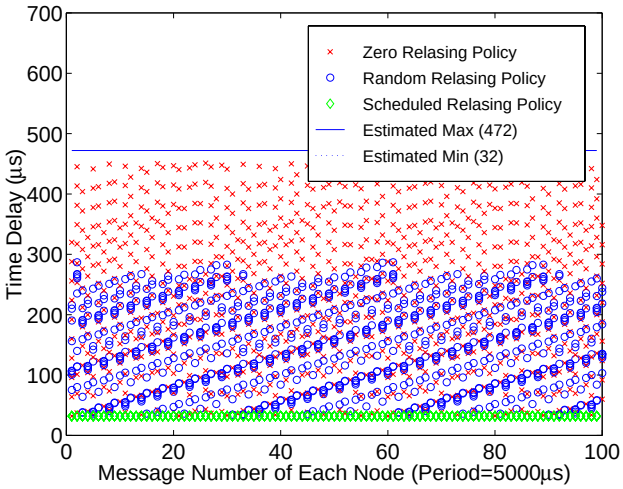


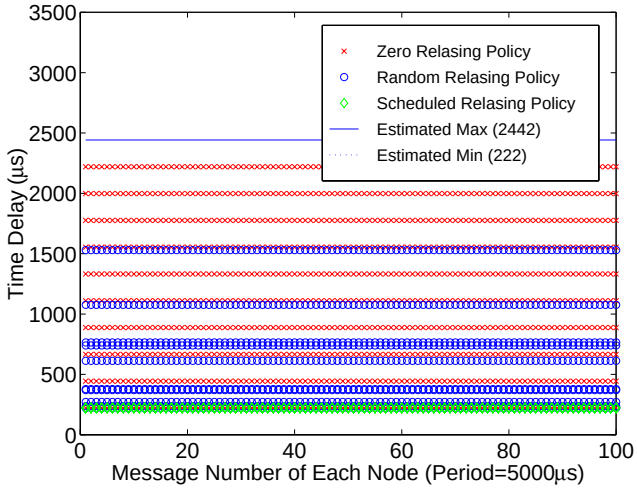
Figure 9. The time delay history of nodes 1, 5, and 10 with a period of 5000 μs and using the three releasing policies (10-node case).



(a) Ethernet



(b) ControlNet



(c) DeviceNet

Figure 10. Message time delay associated with three releasing policies (10-node case).

The estimated mean, maximum, and minimum values are computed from the network analysis for the zero and scheduled releasing policies.

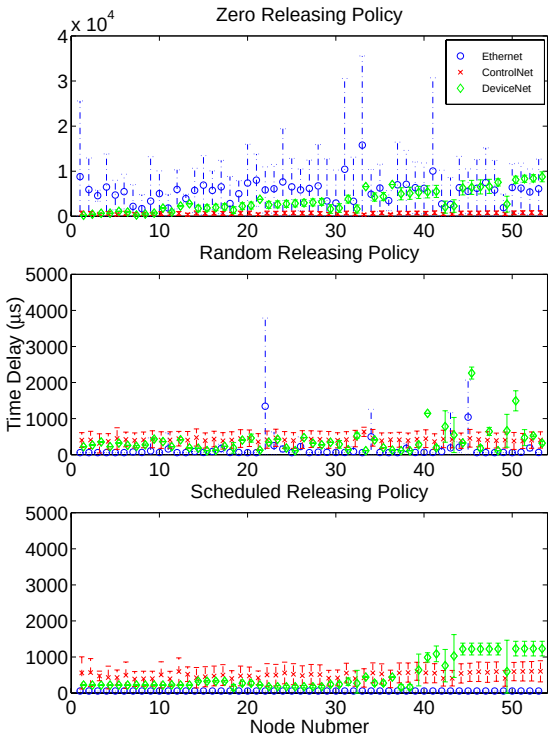
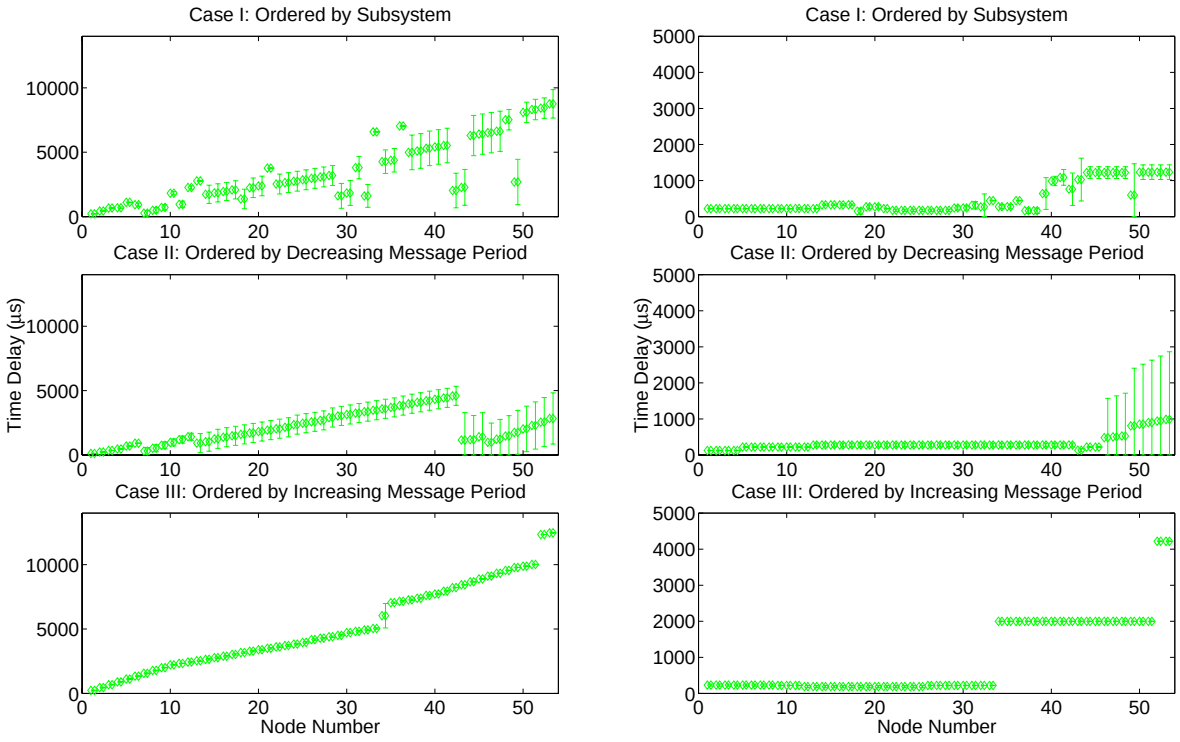


Figure 11. Statistical demonstration of message time delay on three control networks for the SAE example: node numbering by subsystem.

Note: The symbols "circle," "cross," and "diamond" denote the message time delay means of Ethernet, ControlNet, and DeviceNet, respectively; the dashdot, dashed, and solid lines represent the amount of message time delay variance computed from the simulation result as twice the standard deviation of the time delay.



(a) Zero Releasing Policy (b) Scheduled Releasing Policy

Figure 12. Statistical demonstration of message time delay on DeviceNet for the SAE example

Note: The symbol "diamond" denotes the message time delay means; the solid lines represent the amount of message time delay variance computed from the simulation result as twice the standard deviation of the time delay.