Network Protocols for Networked Control Systems

Chapter · January 2005

DOI: 10.1007/0-8176-4404-0_28

CITATIONS
20

READS
135

3 authors, including:

Feng-Li Lian
National Taiwan University
140 PUBLICATIONS **2,392** CITATIONS

SEE PROFILE

Network Protocols for Networked Control Systems

F.-L. Lian,[1] J. R. Moyne,[2] and D. M. Tilbury[2]

[1] Electrical Engineering Department, National Taiwan University, No. 1, Sec. 4, Roosevelt Road, Taipei, 106, Taiwan fengli@ntu.edu.tw
[2] Mechanical Engineering Department, University of Michigan, 2350 Hayward St., Ann Arbor, MI 48103, U.S.A. {moyne,tilbury}@umich.edu

1 Introduction

Control systems with networked communication, called *networked control systems* (NCSs), provide several advantages over point-to-point wired systems such as improvement in reliability through reduced volume of wiring, simpler systems integration, easier troubleshooting and maintenance, and the possibility for distributed processing. There are two types of communication networks. *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 [19].

The change of communication architecture from point-to-point to common-bus, however, introduces different forms of time delay uncertainties between sensors, actuators, and controllers. These time delays come from the time sharing of the communication medium as well as the extra time required for physical signal coding and communication processing. The characteristics of time delays may be constant, bounded, or even random, depending on the network protocols adopted and the chosen hardware. This type of time delay could potentially degrade a system's performance and possibly cause system instability.

Thus, the disadvantages of an NCS include the limited bandwidth for communication and the delays that occur when sending messages over a network. In this chapter, we discuss the sources of delay in common communication networks used for control systems, and show how they can be computed and analyzed.

Several 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 method of synchronization between requesters and providers (such as polling), the data or message size of the information, physical factors such as network length, and the medium access control sublayer protocol that controls the information transmission [7]. There are three main types of medium access control used in control networks: random access with retransmission when collisions occur (e.g., Ethernet and most wireless mechanisms), time-division multiplexing (such as master-slave or token-passing), and random access with prioritization for collision avoidance (e.g., Controller Area Network (CAN)). Within each of these three categories, there are numerous network protocols that have been defined and used. For each type of 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 type are summarized. The focus is on one of the most common protocols in each category; the analysis for other protocols in the same category can be addressed in a similar fashion.

A survey of the types of control networks used in industry shows a wide variety of networks in use; see Table 1. The networks are classified according to type: random access (RA) with collision detection (CD) or collision avoidance (CA), or time-division multiplexed (TDM) using token-passing (TP) or master-slave (MS).

Table 1. Worldwide most popular fieldbuses [18]. Note that the totals are more than 100% because many companies use more than one type of bus. Wireless was not included in the original survey, but its usage is growing quickly.

Network	Type	Users	Application domain
Ethernet	RA/CD	50%	Various
Profibus	TDM/(TP and MS)	26%	Process control
CAN-based	RA/CA	25%	Automotive, process
Modbus	TDM/MS	22%	Various
ControlNet	TDM/TP	14%	Plant bus
ASI	TDM/MS	9%	Building systems
Interbus-S	TDM/MS	7%	Manufacturing
Fieldbus Foundation	TDM/TP	7%	Chemical industry
Wireless (e.g., IEEE 802.11)	RA/CA	Unknown	Various

2 Control Network Basics

In this section, we discuss the medium access control (MAC) sublayer protocol of three types of control networks. We focus our discussion on one of the common networks of each type: Ethernet (including hub, switch, and wireless varieties, which will be defined later), ControlNet (a token-passing network), and DeviceNet (a CAN-based network).[3] 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 [8]. Our discussion and comparison thus focus on the MAC sublayer protocols.

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. 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 the COS/cyclic connection seems most appropriate from a traditional control systems point of view, strobe and poll are commonly used in industrial control networks [4].

2.1 Ethernet networks (CSMA)

Ethernet generally uses the carrier sense multiple access (CSMA) with CD or CA mechanisms for resolving contention on the communication medium. There are three common flavors of Ethernet: (1) hub-based Ethernet, which is common in office environments and is the most widely implemented form of Ethernet, (2) switched Ethernet, which is more common in manufacturing and control environments, and (3) wireless Ethernet.

Hub-based Ethernet (CSMA/CD)

Hub-based Ethernet uses hub(s) to interconnect the devices on a network. When a packet comes into one hub interface, the hub simply broadcasts the packet to all other hub interfaces. Hence, all of the devices on the same network

[3]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. These include industrial Ethernet solutions such as Modbus/TCP, PROFINET, and EtherNet/IP.

receive the same packet simultaneously, and message collisions are possible. Collisions are dealt with utilizing the CSMA/CD protocol as specified in the IEEE 802.3 network standard [1, 2, 21].

This protocol operates as follows: when a node wants to transmit, it listens to the network. If the network is busy, the node 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 retransmission time 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 [21].

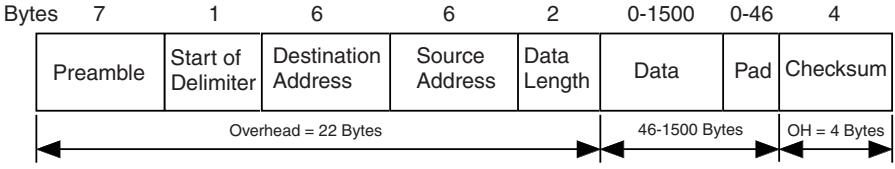


Fig. 1. Ethernet (CSMA/CD) frame format

The Ethernet frame format is shown in Fig. 1 [21]. The total overhead is 26 (=22+4) 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 the 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 Mbps [21].

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 [24]. 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 [21].
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 delays may become unbounded [24]. 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 substantial performance degradation [20]. 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 this form of 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 [6]. 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 [8]. Other solutions also try to prioritize CSMA/CD (e.g., LonWorks) to improve the response time of critical packets [14]. To a large extent these solutions have been rendered moot with the proliferation of switched Ethernet as described below. On the other hand, many of the same issues reappear with the migration to wireless Ethernet for control.

Switched Ethernet (CSMA/CA)

Switched Ethernet utilizes switches to subdivide the network architecture, thereby avoiding collisions, increasing network efficiency, and improving determinism. It is widely used in manufacturing applications. The main difference between switch-based and hub-based Ethernet networks is the intelligence of forwarding packets. Hubs simply pass on incoming traffic from any port to all other ports, whereas switches learn the topology of the network and forward packets to the destination port only. In a star-like network layout, every node is connected with a single cable to the switch. Thus, collisions can no longer occur on any network cable.

Switches employ the cut-through or store-and-forward technique to forward packets from one port to another, using per-port buffers for packets

waiting to be sent on that port. Switches with cut-through first read the MAC address and then forward the packet to the destination port according to the MAC address of the destination and the forwarding table on the switch. On the other hand, switches with store-and-forward examine the complete packet first. Using the cyclic redundancy check (CRC) code, the switch will first verify that the frame has been correctly transmitted before forwarding the packet to the destination port. If there is an error, the frame will be discarded. Store-and-forward switches are slower, but will not forward any corrupted packets.

Although there are no message collisions on the networks, congestion may occur inside the switch when one port suddenly receives a large number of packets from the other ports. Three main queueing principles are implemented inside the switch in this case. They are first-in-first-out (FIFO) queue, priority queue, and per-flow queue. The FIFO queue is a traditional method that is fair and simple. However, if the network traffic is heavy, the quality of service cannot be guaranteed. In the priority queueing scheme, the network manager reads some of the data frames to distinguish which queues will be more important. Hence, the packets can be classified into different levels of queues. Queues with high priority will be processed first followed by queues with low priority until the buffer is empty. With the per-flow queueing operation, queues are assigned different levels of priority (or weights). All queues are then processed one by one according to priority; thus, the queues with higher priority will generally have higher performance and could potentially block queues with lower priority.

Examples of timing analysis and performance evaluation of switched Ethernet can be found in [9, 23].

Wireless Ethernet (CSMA/CA)

Wireless Ethernet, based on the IEEE 802.11 standard, can replace wired Ethernet in a transparent way since it implements the two lowest layers of the International Standards Organization (ISO)/Open Systems Interconnection (OSI) model. Besides the physical layer, the biggest difference between 802.11 and 802.3 is in the medium access control. Unlike wired Ethernet nodes, wireless stations cannot "hear" a collision. A collision avoidance mechanism is used but cannot entirely prevent collisions. Thus, after a packet has been successfully received by its destination node, the receiver sends a short acknowledgment packet (ACK) back to the original sender. If the sender does not receive an ACK packet, it assumes that the transmission was unsuccessful and retransmits.

The collision avoidance mechanism in 802.11 works as follows. If a network node wants to send while the network is busy, it sets its backoff counter to a randomly chosen value. Once the network is idle, the node waits first for an interframe space and then for this backoff time before attempting to send. If another node accesses the network during that time, it must wait again for

another idle interval. In this way, the node with the lowest backoff time sends first. Certain messages (e.g., ACK) may start transmitting after a shorter interframe space, thus they have a higher priority. Collisions may still occur because of the random nature of the backoff time; it is possible for two nodes to have the same backoff time.

Several refinements to the protocol also exist. Nodes may reserve the network either by sending a request to send (RTS) message or by breaking a large message into many smaller messages (fragmentation); each successive message can be sent after the smallest interframe time. If there is a single master node on the network, the master can poll all the nodes and effectively create a TDM contention-free network.

2.2 TDM networks

Time-division multiplexing can be accomplished in one of two ways. In a master-slave network, a single master polls multiple slaves. Slaves can only send data over the network when requested by the master. In this way, there are no collisions, since the data transmissions are carefully scheduled by the master. In a token-passing network, there are multiple masters, or peers. The token bus protocol (e.g., IEEE 802.4) allows a linear, multidrop, tree-shaped, or segmented topology [24]. The node that currently has the token is allowed to send data. When it is finished sending data, or the maximum token holding time has expired, it "passes" the token to the next logical node 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. Most token-passing protocols guarantee a maximum time between network accesses for each node, and most also have provisions to regenerate the token if the token holder stops transmitting and does not pass the token to its successor. In many cases, nodes can also be added dynamically to the bus and can request to be dropped from the logical ring.

ASI, Bitbus, and Interbus-S are typical examples of master-slave networks, while Profibus and ControlNet are typical examples of token-passing networks. Each peer node in a Profibus network can also behave like a master and communicate with a set of slave nodes during the time it holds the token. These are deterministic networks because the maximum waiting time before sending a message frame can be characterized by the token rotation time. 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.

ControlNet

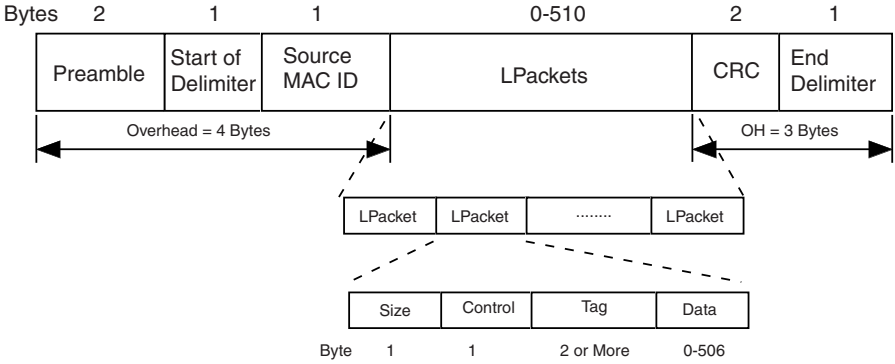


Fig. 2. The message frame of ControlNet (token bus)

The ControlNet protocol is used here as a case study of the operation of a typical token-passing network. The message frame format of ControlNet is shown in Fig. 2 [3]. The total overhead is 7 bytes, including preamble, start delimiter, source MAC ID, CRC, and end delimiter. The data packet frame, also called the link packet (Lpacket) frame, may include several Lpackets that contain size, control, tag, and data fields with total frame size between 0 and 510 bytes. The individual destination address is specified within the tag field. The size field specifies the number of byte pairs (from 3 to 255) contained in an individual Lpacket, including 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 the 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 an NUT, each node can transmit time-critical/scheduled data by obtaining the implicit token from 0 to S. During the unscheduled part of an NUT, nodes 0 to U share 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 the 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 the 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.

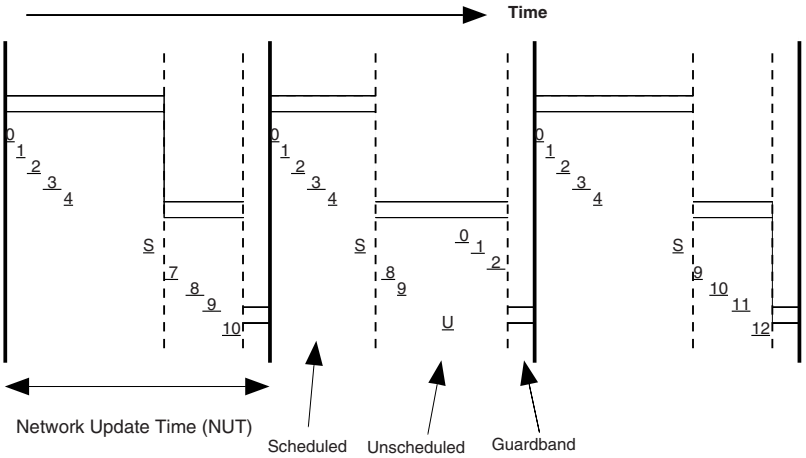


Fig. 3. Medium access during scheduled, unscheduled, and guardband time

Advantages: The token bus protocol is a deterministic protocol that provides excellent throughput and efficiency at high network loads [8,24]. During network operation, the token bus can dynamically add nodes to or remove nodes from the network. This contrasts with the token ring case, where the nodes physically form a ring and cannot be added or removed dynamically [24]. 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 [8].

3 CAN-Based Networks: DeviceNet

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 (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 is an example of a technology based on the CAN specification that has received considerable acceptance in device-level manufacturing applications. The DeviceNet specification is based on the standard CAN (11-bit identifier only)[4] with an additional application and physical layer specification [4, 17].

The frame format of DeviceNet is shown in Fig. 4 [4]. 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. Higher priority messages always gain access to the medium during arbitration. Therefore, the transmission delay for higher priority messages can be guaranteed.

[4]The CAN protocol supports two message frame formats: standard CAN (version 2.0A, 11-bit identifier) and extended CAN (version 2.0B, 29-bit identifier).

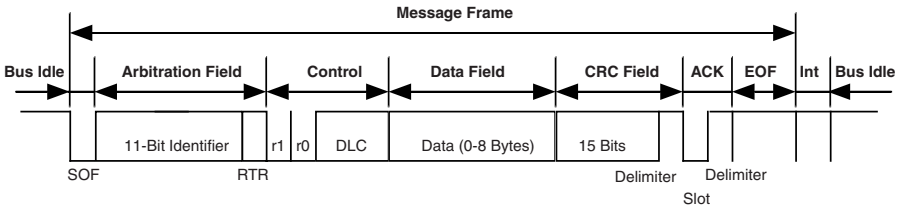


Fig. 4. The message frame format of DeviceNet (standard CAN format)

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.

4 Timing Components

The important time delays that should be considered in an NCS analysis are the sensor-to-controller and controller-to-actuator end-to-end delays. In an NCS, message transmission delay can be broken into two parts: device delay and network delay. The device delay includes the time delays at the source and destination nodes. The time delay at the source node includes the preprocessing time, T_{pre}, and the waiting time, T_{wait}. The time delay at the destination node is only the postprocessing time, T_{post}. The network time delay includes the total transmission time of a message and the propagation delay of the network. The total time delay can be expressed by the following equation:

$$T_{delay} = T_{pre} + T_{wait} + T_{tx} + T_{post}. \tag{1}$$

The key components of each time delay are shown in Fig. 5 and will be discussed in the following subsections.

4.1 Pre- and postprocessing times at source and destination nodes

The preprocessing time at the source node is the time needed to acquire data from the external environment and encode it into the appropriate network data format. There may be one processor performing both functions, or multiple processors; we define the total elapsed time required as the pre- or postprocessing time. This time depends on the device software and hardware characteristics. In many cases, it may be assumed that the preprocessing time

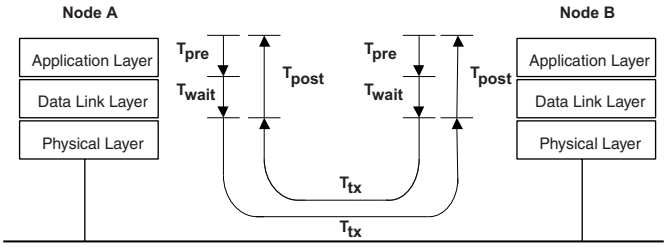


Fig. 5. A timing diagram showing time spent sending a message from a source node to a destination node

is constant or negligible. However, this assumption is not true in general; in fact, there may be noticeable differences in processing time characteristics between similar devices, and these delays may be significant.

The postprocessing time at the destination node is the time taken to decode the network data into the physical data format and output it to the external environment.

4.2 Experimental investigation of pre- and postprocessing times

In practical applications, it is very difficult to identify each individual timing component discussed above. Instead, by monitoring the time-stamped traffic of the request-response messaging on a DeviceNet network, we can show the characteristics of processing times, i.e., the sum of the preprocessing and postprocessing times of one device.

In the experimental setup, there is only one master and one slave connected to the network and the master continuously polls this slave. Referring to Fig. 5, let Node A be the master and Node B be the slave. Here, there is no other network traffic other than the request-response messages between the master and slave, i.e., $T_{wait} = 0$, and the request-response frequency is set low enough that no messages are queued up at the sender buffer. By monitoring the message traffic on the network medium and time-stamping each message, we can further calculate the processing time of each request-response, i.e., $T_{post} + T_{pre}$, after subtracting the transmission time.

Fig. 6 shows the histogram of 400 samples of four typical DeviceNet device processing times [11]. The devices are standard I/O types, such as those used for limit switches. The (right) solid and (left) dashed lines are the maximum and minimum values of the processing times, respectively. The histogram plots indicate the nondeterministic processing times of different network devices and their variance. Devices 1 and 3 have a similar functionality of discrete inputs/outputs, but different numbers of input/output modules. Device 3 provides several augmentable modules and hence has more processing units and a higher computation load. Device 1, on the other hand, has only one unit. Device 2 has a fairly consistent processing time, i.e., a low variance. Note that

the smallest time that can be recorded is 1 μs. The uniform distribution of processing time at Device 4 is due to the fact that it has an internal sampling time which is mismatched with the request frequency. Hence, the processing time recorded here is the sum of the actual processing time and the waiting time inside the device. Device 4 also provides more complex functionality and has a longer processing time than the others.

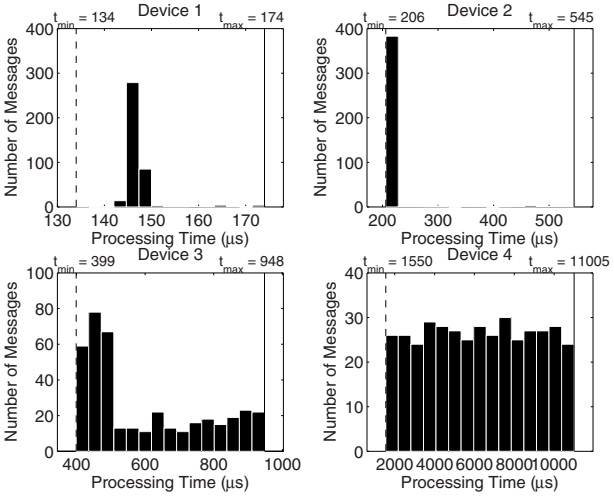


Fig. 6. Processing time histogram of four typical DeviceNet devices

A key point that can be taken from the data presented in Fig. 6 is that the device processing time can be substantial in the overall calculation of T_{delay}. In fact, this delay often dominates over network delays. Thus, in designing NCSs, device delay and delay variability should be considered as important factors when choosing components.

4.3 Transmission time on network channel

The transmission time is the most deterministic parameter in a network system because it only depends on the data rate, the message size, and the distance between two nodes. The formula for transmission time can be described as follows. $T_{tx} = T_{frame} + T_{prop}$. T_{frame} is the time required to send the packet across the network, and T_{prop} is the propagation time between any two devices. Since the typical transmission speed in a communication medium is 2×10^8 m/s, the propagation time T_{prop} is negligible on a small scale. In 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.

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 the 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:

$$T_{frame} = [N_{data} + N_{ovhd} + N_{pad} + N_{stuff}] \times 8 \times T_{bit}. \tag{2}$$

The values N_{data}, N_{ovhd}, N_{pad}, and N_{stuff}[5] can be explicitly described for the Ethernet, ControlNet, and DeviceNet protocols, see [10].

4.4 Waiting time at source nodes

A message may spend time waiting in the queue at the sender's buffer and could be blocked from transmitting by other messages on the network. Depending on the amount of data the source node must send and the traffic on the network, the waiting time may be significant. The main factors affecting waiting time are network protocol, message connection type, and network traffic load. For example, consider the strobe message connection in Fig. 7. If Slave 1 is sending a message, the other 8 devices must wait until the network medium is free. In a CAN-based DeviceNet network, it can be expected that Slave 9 will encounter the most waiting time because it has a lower priority on this priority-based network. However, in any network, there will be a nontrivial waiting time after a strobe, depending on the number of devices that will respond to the strobe.

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.

[5]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.

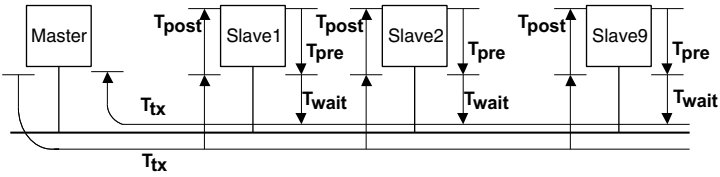


Fig. 7. Waiting time diagram

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 in Section 2.1 indicates a probabilistic waiting time. An exact analysis of expected blocking time delay for Ethernet is very difficult [10]. At a high level, the expected blocking time can be described by the following equation:

$$E\{T_{block}\} = \sum_{k=1}^{16} E\{T_k\} + T_{resid}, \tag{3}$$

where T_{resid} denotes the residual time 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 the 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 [21]. 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 times and token rotation times of previous nodes. The general formula for T_{block} can be described by the following equation:

$$T_{block} = T_{resid} + \sum_{j \in \mathcal{N}_{noqueue}} T_{token}^{(j)} + \sum_{j \in \mathcal{N}_{queue}} \min(T_{tx}^{(j,n_j)}, T_{node}) + T_{guard}, \tag{4}$$

where T_{resid} is the residual time needed by the current node to finish transmitting, $\mathcal{N}_{noqueue}$ and $\mathcal{N}_{queue}$ denote the sets of nodes without messages and with 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 $T_{node} = 827.2$ μs, 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 that 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 (4). 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 (4) can be determined explicitly.

DeviceNet blocking time

The blocking time, T_{block}, in DeviceNet can be described by the following equation [22]:

$$T_{block}^{(k)} = T_{resid} + \sum_{\forall j \in \mathcal{N}_{hp}} \lceil \frac{T_{block}^{(k-1)} + T_{bit}}{T_{peri}^{(j)}} \rceil \, T_{tx}^{(j)}, \tag{5}$$

where k is the iteration index of obtaining steady-state T_{block}, T_{resid} is the 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 node, and $\lceil x \rceil$ denotes the smallest integer number that is greater than or equal to x. The summation denotes the time needed to send all the higher priority messages. While a low priority node 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

$$T_{resid} = \max_{\forall j \in \mathcal{N}_{node}} T_{tx}^{(j)}, \tag{6}$$

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

Fig. 8 shows experimental data of the waiting time of nine identical devices on a DeviceNet network. These devices have a very low variance of processing time. We collected 200 pairs of messages (request and response). Each symbol denotes the mean, and the distance between the upper and lower bars equals two standard deviations. If these bars are over the limit (maximum or minimum), then the value of the limit is used instead. It can be seen in Fig. 8 that the average waiting time is proportional to the node number (i.e., priority). Also, the first few devices have a larger variance than the others, because the variance of processing time occasionally allows a lower priority device to access the idle network before a higher priority one.

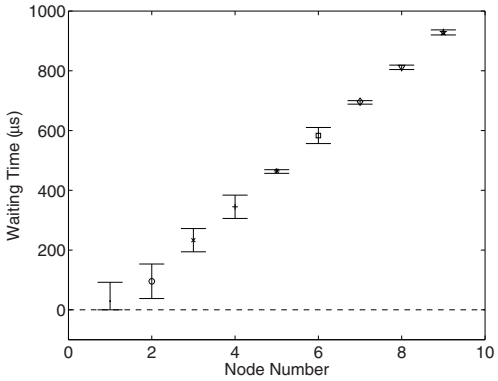


Fig. 8. Nine identical devices with strobed message connection

5 Network Comparisons

In this section, comparisons are drawn between the three types of control networks using the three networks that have been discussed in detail: Ethernet, ControlNet (token bus), and DeviceNet (CAN). The parameters for these networks are shown in Table 2. After summarizing the theoretical and simulation results for these three networks, we show some experimental results for time delays and throughput in wireless Ethernet.

5.1 Data transmission

One method for comparing control networks is by the time taken to transmit data and the efficiency of the data transmission.

As shown in Fig. 9(a), the transmission time for DeviceNet is longer than the others because of the lower data rate (500 Kbps). Ethernet requires less transmission time on larger data sizes (>20 bytes) compared with the others.

Table 2. 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.

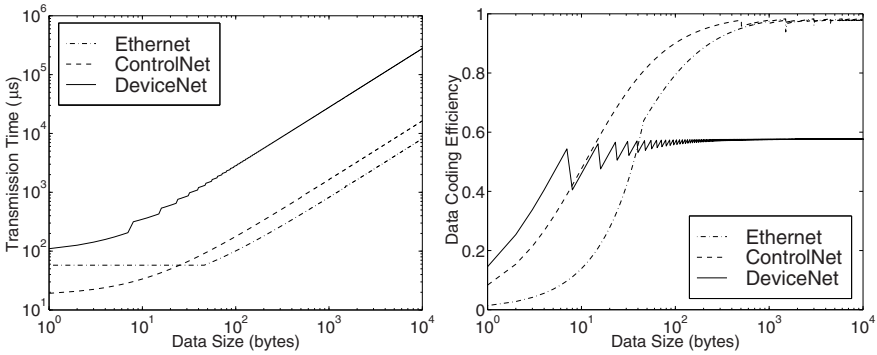
Although ControlNet uses less time to transmit the same amount of data, it needs some time (NUT) to gain access to the network.

The data coding efficiency (see Fig. 9(b)) 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 (due to its large minimum message size). For large data sizes, ControlNet and Ethernet are better than DeviceNet (DeviceNet is only 58% efficient due to its small maximum message size, but ControlNet and Ethernet are near 98% efficient). 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.

5.2 Case study of 10-node NCS

In this section, we use a case study of an NCS to compare the three different control networks. The system has 10 nodes, each with 8 bytes of data to send every period. MATLAB[6] 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 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.

[6]MATLAB is a technical computing software developed by The MathWorks, Inc.



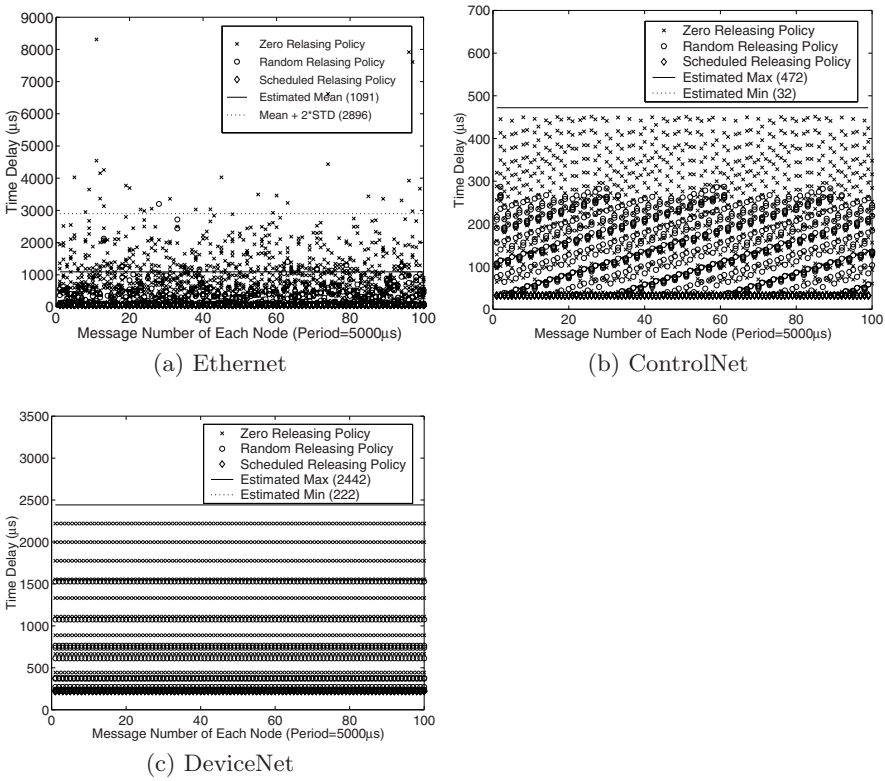
(a) Transmission time (b) Data coding efficiency

Fig. 9. A comparison of transmission time and data coding efficiency versus the data size for three control networks

Based on the three different types of message connections (poll, strobe, and cyclic), we consider the following three releasing policies. The first policy, which we call the "zero releasing policy," assumes that 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, 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 cyclic messaging, where no preschedule 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 or with a well-scheduled cyclic policy.

In addition to varying the release policy, we can also change the period of each node to demonstrate the effect of traffic load on the network. For each releasing policy and period, we simulate the system and calculate the average time delays of these 10 nodes. We then compare the simulation results to the analytic results described in Section 4. 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.

The simulation results for a message period of 5000 μs are summarized in Fig. 10. The zero releasing policy has the longest average delay in every network because all nodes experience contention when trying to send messages. Although the Ethernet data rate is much faster than that of 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



(a) Ethernet

(b) ControlNet

(c) DeviceNet

Fig. 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.

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 time delay of this releasing policy is only the transmission time.

In Ethernet, shown in Fig. 10(a), the zero and random releasing policies demonstrate its nondeterministic time delay, even though the traffic load is not saturated. Fig. 10(b) shows that the 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 Section 4. 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 node number. The estimated mean time delay (1091 μs) for Ethernet in Fig. 10(a) is computed for the case of the zero releasing policy from (3), and the variance is taken as twice the

standard deviation. The maximum and minimum time delays for ControlNet and DeviceNet are computed from (4) and (5).

5.3 Wireless Ethernet throughput and delays

In addition to time delays, the difference between the theoretical data rate and the practical throughput of a control network should be considered. For example, raw data rates for 802.11 wireless networks range from 11 to 54 Mbits/sec. The actual throughput of the network, however, is lower due to both the overhead associated with the interframe spaces, ACK, and other protocol support transmissions, and to the actual implementation of the network adapter. Although 802.11a and 802.11g have the same raw data rate, the throughput is lower for 802.11g because its backwards compatibility with 802.11b requires that the interframe spaces be as long as they would be on the 802.11b network. Computed and measured throughputs are shown in Table 3 [5]. The experiments were conducted by continually sending more traffic on the network until a further setpoint increase in traffic resulted in no additional throughput.

Table 3. Maximum throughputs for different 802.11 wireless Ethernet networks. All data rates and throughputs are in Mbit/sec.

Network type	802.11a	802.11g	802.11b
Nominal data rate	54	54	11
Theoretical throughput	26.46	17.28	6.49
Measured throughput	23.2	13.6	3.6

Experiments conducted to measure the time delays on wireless networks are summarized in Table 4 and Fig. 11 [5]. Data packets were sent from the client to the server and back again, with varying amounts of cross-traffic on the network. The send and receive times on both machines were time-stamped. The packet left the client at time t_a and arrived at the server at time t_b; then left the server at time t_c and arrived at the client at time t_d. The sum of the pre- and postprocessing times and the transmission time on the network for both messages can be computed as (assuming that the two nodes are identical)

$$\begin{aligned} 2 * T_{delay} &= 2 * (T_{pre} + T_{wait} + T_{tx} + T_{post}) \\ &= t_d - t_a - (t_c - t_b). \end{aligned}$$

Note that this measurement does not require that the clocks on the client and server be synchronized. Since the delays at the two nodes can be different, it is this sum of the two delays that is plotted in Fig. 11 and tabulated in Table 4.

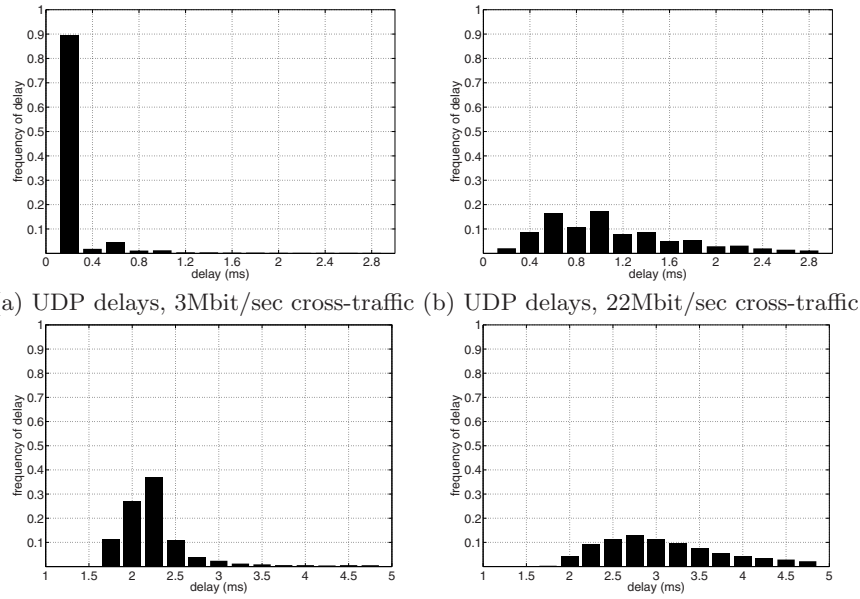
Two different types of data packets were considered: User Datagram Protocol (UDP), and object linking and embedding (OLE) for Process Control (OPC). UDP is a commonly used connectionless protocol that runs on top of Ethernet, often utilized for broadcasting. UDP packets carry only a data load of 50 bytes. OPC is an application-to-application communication protocol primarily utilized in manufacturing to communicate data values. OPC requires extra overhead to support this application layer; consequently, the OPC packets carry the maximum packet load of 512 data bytes. For comparison purposes, the frame times (including the overheads) are computed for the different packets.

Table 4. Computed frame times and experimentally measured delays on wireless networks; all times in ms.

Network type	802.11a	802.11g	802.11b
Frame time (UDP), computed	0.011	0.011	0.055
Median delay (UDP), measured	0.346	0.452	1.733
Frame time (OPC), computed	0.080	0.080	0.391
Median delay (OPC), measured	2.335	2.425	3.692

6 Conclusions and Future Work

The features of three candidate control networks — Ethernet (CSMA/CD), ControlNet (Token Bus), and DeviceNet (CAN) — were discussed in detail. With respect to Ethernet, which is becoming more and more prevalent in control network applications, we described and contrasted the three main implementation types: hub-based, switched, and wireless. For all protocols we first described the MAC mechanisms, 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. We then focused on exploring timing parameters related to end-to-end delivery of information over the networks. These timing parameters, which will ultimately influence control applications, are affected by the network data rate, the periods 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 the magnitude and characteristics of the expected and measured time delays. Simulation results were presented for several different scenarios. The timing analyses and comparisons of message time delay given in this chapter should be useful for designers of NCSs. For example, the basic differentiation of the network approaches will help the designer to match



(a) UDP delays, 3Mbit/sec cross-traffic (b) UDP delays, 22Mbit/sec cross-traffic

(c) OPC delays, 3Mbit/sec cross-traffic (d) OPC delays, 22Mbit/sec cross-traffic

Fig. 11. Distributions of packet delays for different values of cross-traffic throughput on a 802.11a network

basic requirement rankings against network approaches. Also, analyses such as device delay experiments reveal to the designer the importance of device delay and device delay variability in NCS design.

Control systems typically send small amounts of data periodically, but require guaranteed transmission and bounded time delay for the messages. The suitability of network protocols for use in control systems is greatly influenced by these two criteria. Although Ethernet (including hub-based and wireless) 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 machine-level communication with controllers, sensors, and actuators, deterministic networks are generally more suitable for meeting the characteristics and requirements of control systems. For control systems with short and/or prioritized messages, CAN-based protocols such as DeviceNet demonstrate 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.

Future NCS research efforts are expected to focus on controller design for NCSs, which can differ significantly from the design of traditional centralized control systems. For example, controller design optimized to the delay expected in an NCS is explored in [12], and balancing quality of service and quality of performance (QoP) in control networks can be effected using techniques such as deadbanding [15].

Another body of future NCS research will focus on the utilization of Ethernet for control [16], with a special emphasis on wireless Ethernet. While wireless Ethernet is beginning to proliferate in manufacturing diagnostics, its acceptance as an NCS enabler has been very slow to occur due to issues of reliability, performance, and security [13]. However, the enormous flexibility, cost savings, and reliability benefits that could potentially be achieved with wireless systems will continue to drive wireless NCS research, with a focus not only on control system design, but also on higher performing, more reliable, and more secure networks for control. Its is easily conceivable that, within 10 years, wireless will be the preferred medium for NCSs.

Acknowledgement. Many of the results described in this chapter are described in more detail in [10] and [11]. The authors would like to thank Alexander Duschau-Wicke, a student at the University of Kaiserslautern in Germany, who visited the University of Michigan in 2004 and performed the wireless Ethernet experiments described in Section 5.3.

References

1. D. Bertsekas and R. Gallager. *Data Networks*. Prentice-Hall, Englewood Cliffs, NJ, second edition, 1992.
2. B. J. Casey. Implementing Ethernet in the industrial environment. In *Proceedings of IEEE Industry Applications Society Annual Meeting*, volume 2, pages 1469–1477, Seattle, WA, October 1990.
3. ControlNet specifications, 1998.
4. DeviceNet specifications, 1997.
5. A. Duschau-Wicke. Wireless monitoring and integration of control networks using OPC. Technical report, NSF Engineering Research Center for Reconfigurable Manufacturing Systems, University of Michigan, 2004. Studienarbeit report for Technische Universit at Kaiserslautern.
6. J. Eidson and W. Cole. Ethernet rules closed-loop system. *InTech*, pages 39–42, June 1998.
7. Y. Koren, Z. J. Pasek, A. G. Ulsoy, and U. Benchetrit. Real-time open control architectures and system performance. *CIRP Annals—Manufacturing Technology*, 45(1):377–380, 1996.
8. S. A. Koubias and G. D. Papadopoulos. Modern fieldbus communication architectures for real-time industrial applications. *Computers in Industry*, 26:243–252, August 1995.
9. K. C. Lee and S. Lee. Performance evaluation of switched Ethernet for networked control systems. In *Proceedings of IEEE Conference of the Industrial Electronics Society*, volume 4, pages 3170–3175, November 2002.

10. F.-L. Lian, J. R. Moyne, and D. M. Tilbury. Performance evaluation of control networks: Ethernet, ControlNet, and DeviceNet. *IEEE Control Systems Magazine*, 21(1):66–83, February 2001.
11. F.-L. Lian, J. R. Moyne, and D. M. Tilbury. Network design consideration for distributed control systems. *IEEE Transactions on Control Systems Technology*, 10(2):297–307, March 2002.
12. F.-L. Lian, J. R. Moyne, and D. M. Tilbury. Time-delay modeling and optimal controller design for networked control systems. *International Journal of Control*, 76(6):591–606, April 2003.
13. J. Moyne, J. Korsakas, and D. M. Tilbury. Reconfigurable factory testbed (RFT): A distributed testbed for reconfigurable manufacturing systems. In *Proceedings of the Japan-USA Symposium on Flexible Automation*, Denver, CO, July 2004.
14. J. Moyne, N. Najafi, D. Judd, and A. Stock. Analysis of sensor/actuator bus interoperability standard alternatives for semiconductor manufacturing. In *Sensors Expo Conference Proceedings*, Cleveland, OH, September 1994.
15. P. G. Otanez, J. R. Moyne, and D. M. Tilbury. Using deadbands to reduce communication in networked control systems. In *Proceedings of the American Control Conference*, pages 3015–3020, Anchorage, AK, May 2002.
16. P. G. Otanez, J. T. Parrott, J. R. Moyne, and D. M. Tilbury. The implications of Ethernet as a control network. In *Proceedings of the Global Powertrain Congress*, Ann Arbor, MI, September 2002.
17. G. Paula. Building a better fieldbus. *Mechanical Engineering*, pages 90–92, June 1997.
18. J. Pinto. Fieldbus—conflicting "standards" emerge, but interoperability is still elusive. *Design Engineering, UK*, October 1999. Available at http://www.jimpinto.com/writings/fieldbus99.html.
19. R. S. Raji. Smart networks for control. *IEEE Spectrum*, 31(6):49–55, June 1994.
20. K. K. Ramakrishnan and H. Yang. The Ethernet capture effect: Analysis and solution. In *Proceedings of the 19th Conference on Local Computer Networks*, pages 228–240, Minneapolis, MN, October 1994.
21. A. S. Tanenbaum. *Computer Networks*. Prentice-Hall, Upper Saddle River, NJ, third edition, 1996.
22. K. Tindell, A. Burns, and A. J. Wellings. Calculating controller area network (CAN) message response times. *Control Engineering Practice*, 3(8):1163–1169, August 1995.
23. E. Vonnahme, S. Ruping, and U. Ruckert. Measurements in switched Ethernet networks used for automation systems. In *Proceedings of IEEE International Workshop on Factory Communication Systems*, pages 231–238, September 2000.
24. J. D. Wheelis. Process control communications: Token bus, CSMA/CD, or token ring? *ISA Transactions*, 32(2):193–198, July 1993.