Design and Evaluation of Real-Time Communication for FieldBus-Based Manufacturing Systems

Kang G. Shin, *Fellow, IEEE,* and Chih-Che Chou, *Member, IEEE*

Abstract— **It is well known that the ability to support predictable interprocess communication is of great significance to computer-integrated manufacturing and process control systems. In this paper, we propose a strategy for an industrial standard, the SP-50 FieldBus, to support both intracell and intercell real-time communications. We first describe our strategy in detail and show that it is compatible with the current FieldBus draft standard. Under our strategy, the capacity of each link is divided into two parts. The first part is managed by the local link active scheduler (LAS) for intracell (intralink) communication. The second part is managed by a proposed global network manager for intercell (interlink) communication. By dividing the link capacity in this way, our strategy allows for fast local intracell connection establishment, while supporting global intercell connections. Using two examples, one for typical manufacturing systems and the other for multimedia networking, we also demonstrate the power and utility of the proposed strategy as compared to token-passing protocols.**

I. Introduction

An automated factory (AF) is usually composed of several workcells (or simply cells), each of which contains robots, sensors, and transport mechanisms. A multiaccess bus connects all devices in a workcell. Bridges then connect multiple workcells to form an AF. Hence, the network of an AF consists of many links, each of which is usually a multiaccess bus. The ability to provide predictable interprocess communication is of great significance to an AF, because unpredictable communication delays may lead to missing the deadline of one or more communicating tasks that collectively monitor and control manufacturing equipment and processes. In this paper, we will address the real-time communication issue of the FieldBus protocol which is designed to support time-critical communication to and from devices in manufacturing and/or process control systems.

Although computer networking has been extensively researched, its specific application to AF's has not yet been addressed thoroughly. Most of analytic research deals with the general area of network communication, focusing on flexible systems design and performance evaluation. However, real-time communication between devices in an AF has seldom been addressed. The manufacturing automation protocol (MAP) which was proposed by GM and other companies for the AF network is based on the OSI seven-layer model and the token bus protocol, IEEE 802.4. The MAP can provide some limited form of real-time communication, but it cannot guarantee the delivery of time-critical messages before their deadlines. In fact, the MAP only provides temporal ordering between devices based on their priorities. The seven-layer MAP is usually too slow to be used for real-time communication, since there are at least 14 layers' delays in a single one-way message communication. Another protocol called MINIMAP employs only the first two layers of MAP and combines the remaining five layers into a single layer. MINIMAP is expected to reduce the communication delay, but it still leaves the real-time issue unaddressed. Token-ring type protocols cannot be used either, for the same reason as the token bus. CSMA/CD type protocols are not applicable to real-time systems because of their unbounded communication delays. As we shall see, the FieldBus has only three layers, reserves the required communication capacity in advance for each *real-time channel* (a point-to-point unidirectional connection which can provide end-to-end delivery delay guarantees), and uses a centralized scheme for scheduling messages in order to guarantee the delivery of real-time messages before their deadlines.

The problem of supporting time-constrained communication has been studied by several researchers, since it plays an important role in many applications, such as exchange of control messages and audio- and video-transmissions over a communication network. Most of these efforts can be classified into two categories. The first category is mainly concerned with the design of medium access protocols for multiaccess networks subject to timing constraints in delivering messages. In this context, most of the proposed schemes can be classified as *best-effort* schemes in which the system tries to ensure that most messages can meet their deadlines, but it cannot give any guarantee of delivery time [13], [17]. However, based on the given information about the message arrival/generation pattern, some of them can make guarantees about their delivery time [18]. The other category deals with the problem of establishing real-time point-to-point channels and providing guarantees of maximum delivery delays [4], [9], [11]. The main issues addressed in these schemes are message scheduling, buffer management, and flow control in the network nodes. However, the type of networks used in these schemes is not suitable for manufacturing automation systems for the

Manuscript received April 23, 1992; revised July 10, 1994. This paper was recommended for publication by Editor A. Desrochers upon evaluation of reviewers' comments. This work was supported in part by General Motors, the National Science Foundation under Grant MIP-9203895, and the Office of Naval Research under Grant N00014-94-1-0229. This paper was partially presented at 1992 IEEE Local Computer Network Conference, Minneapolis, MN, September 1992.

The authors are with the Real-Time Computing Laboratory, Department of Electrical Engineering and Computer Science, University of Michigan, Ann Arbor, MI 48109-2122 USA.

Publisher Item Identifier S 1042-296X(96)03825-6.

following two reasons. First, using a point-to-point network to connect all devices in a workcell is not practical because the network could become too complex to manage if there are tens of devices in a workcell. Second, these schemes may suffer the long delay of channel establishment, even when two communicating nodes are physically close to each other.

Since most time-critical communications are likely to occur between two devices in the same workcell, a fast connection establishment procedure is desirable. Therefore, a multiaccess bus is a natural candidate for connecting devices in a workcell. On the other hand, since the ability to provide predictable communication between any two devices is also essential to the system, we will focus on quick real-time channel establishment within a workcell (i.e., two nodes on the same multiaccess bus), while providing the ability to support real-time communication between *any* two workcells.

The paper is organized as follows. In Section II, we state the problem of supporting time-constrained communication under the FieldBus protocol. The proposed approaches are discussed in Section III. Section IV discusses the compatibility between the proposed protocol and the FieldBus protocol draft. The performance of the proposed protocol is comparatively analyzed in Section V, and the paper concludes with Section VI.

II. Problem Statement

Our main goal is to analyze and/or enhance the data link layer of FieldBus protocol which is designed to support time-critical communication in process control and manufacturing systems. Most of the data link layer protocol of FieldBus has already been well developed. However, the support for time-critical communication is left unspecified. To facilitate our presentation, we need to briefly describe the FieldBus data link layer protocol.

In order to reduce communication latencies, unlike the OSI seven layer model, the FieldBus has only three layers: physical layer, data link layer, and application layer. Since a manufacturing system is composed of many workcells, and each workcell contains a multiaccess bus which connects all the devices in the workcell, the entire network is a collection of multiaccess buses which are further connected via several bridges. A **data link entity** (DLE) of FieldBus is a logically active object, such as a part of the network operating system, which can send/receive packets to and from the interconnection network and acts according to the data link layer protocol of FieldBus. Hence, there could be several DLE's on a node which is physically attached to the interconnection network, such as computers, sensors, or any manufacturing devices. However, it is conceptually simple to treat each DLE as a node. Thus, the terms "DLE" and "node" will be used interchangeably.

The data link layer protocol of FieldBus is a delegated token protocol. There is a central control entity on each multiaccess link which is responsible for scheduling messages on the local link, i.e., this central control entity is responsible for managing the token. The data link layer service of FieldBus provides both connectionless and connection-oriented communication between two peer communicating DLE's. The connection-oriented communication service, which supports both real-time and nonreal-time communication, requires the source DLE to establish a connection, thus allowing the source and intermediate nodes to collect and exchange the information needed for the delivery of packets, e.g., buffer requirements and route information. In general, static routing is used for the delivery of connection-oriented packets, although it is not required by the FieldBus standard [2], [3]. If a real-time connection is requested, the system also has to reserve sufficient bandwidth and perform appropriate admission tests during the connection establishment. The connectionless service, which supports only nonreal-time communication, is similar to traditional multiaccess packet switching networks.

There are three classes of DLE's in the FieldBus data link layer protocol: Basic, Link Master (LM), and Bridge. The LM class DLE's can also act as another distinct "class" of DLE's: link active scheduler (LAS). Every LM class DLE's have the capability to be a LAS, but each time exactly one LM DLE can act as the LAS of a FieldBus **link.** In order to avoid confusion, the LM class represents those LM DLE's which currently do not act as a LAS in the rest of this paper unless explicitly stated otherwise.

Basic and LM classes are conceptually the same, except that the LM class DLE's are equipped with more functions, while the Basic class DLE's have only those functions which are absolutely necessary for adequate operations on a FieldBus network. In general, these two classes of DLE's are the "user" nodes on the FieldBus networks.

Unlike other popular timed-token protocols (e.g., token rings, token buses, FDDI), FieldBus has a control unit, LAS DLE, for each FieldBus link. (Note that a FieldBus network consists of a set of links.) The LAS DLE is responsible for scheduling messages on the local link. It receives, and responds to, scheduling requests from all DLE's on the same link by giving a token to one of these DLE's which then assumes the exclusive right to use the link over some period of time specified in the token. A bridge DLE which performs a store-and-forward function to connect two or more separate multiaccess links. In the draft standard of the FieldBus data link layer protocol [2], [3], the routing strategy used by bridges is not specified. However, the transparency of bridges to the source node is implicitly assumed in the draft standard.

All of the normal communication requests for use of a link are scheduled by the link's LAS. The LAS generates "polling" tokens for DLE's on the link, and the receiver DLE responds immediately by returning a message which may include requests for future scheduling and the priorities of the current requests. The LAS derives a schedule according to some policy and provides the token to the "winner" DLE.

Based on the above brief description of FieldBus protocol and the nature of workcells in an AF, most of time-critical communication can be handled by the local LAS, since most of real-time communication is likely to take place between two peer DLE's on the same link. As mentioned before, time-constrained communication on a multiaccess bus has been extensively researched. However, most of the proposed schemes cannot give any guarantee on the message delivery delay. Although some of them might be able to support

predictable communication, they require *a priori* information about the message arrival patterns. In order to provide predictable communication services, the authors of [11] proposed a scheme which can give a guarantee on maximum message delivery delays by introducing a centralized network manager, which is responsible for establishing a real-time point-to-point channel between two communicating nodes based on the worst-case resource requirement of the channel. With three parameters—maximum message size, maximum message rate, and maximum burst size—which will be given in the channel establishment phase, the scheme in [11] can give an end-to-end guarantee on the message delivery delay. However, their scheme is built on a point-to-point interconnection network which is seldom used in manufacturing systems due to its inherent complexity. Besides, the centralized connection establishment process could be very slow, since each connection request has to traverse several links to the network manager and a reply must be sent back to both the requesting and destination nodes. Therefore, even if two peer nodes are on the same link, the connection establishment process is time-consuming.

In the draft standard of FieldBus protocol [2], [3], a distributed scheme is implicitly assumed, since the existence of a network manager is not mentioned at all. A distributed scheme can handle nonreal-time communication well, but its ability for handling real-time communication is limited. In order to provide predictable communication for real-time applications, the system has to reserve certain resources (e.g., link bandwidth), verify that certain conditions are met (e.g., the end-to-end delivery delay is small enough), and guarantee that these conditions will always be met during the lifetime of the application. A distributed scheme can handle these operations well if the two peer DLE's are on the same link. However, if the two communicating peer DLE's are located on different links, there are several serious disadvantages associated with a distributed solution. The most significant is the difficulty in coordinating bridges. Without a centralized network manager, each bridge has to be equipped with complete information of the real-time traffic load on each node and each link, and the routes to all the other nodes. Note that since real-time messages have higher priority than nonreal-time messages, we ignore the load of nonreal-time traffic when the system deals with the information for real-time communication. No matter how the information is collected, it is very difficult and expensive to keep the information up-to-date, since frequently broadcasting and processing such information will consume significant link capacity and CPU time. For nonreal-time connectionless traffic, the system can still use conventional routing methods to deliver nonreal-time packets, i.e., each bridge still needs a routing table for nonreal-time messages. This routing table (for nonreal-time connectionless communication) is much easier to construct and maintain than that for both real-time and nonreal-time traffic. Since the goal of this paper is to provide predictable communication for the FieldBus protocol, we will focus on real-time traffic and ignore nonreal-time traffic in the rest of the paper.

For the above reasons and in order to provide predictable communication services with the FieldBus protocol while avoiding the disadvantages in both distributed and centralized schemes, we propose a *hybrid* approach. The real-time communication problem associated with the FieldBus protocol can be decomposed into two related subproblems. The first subproblem is to provide real-time communication between two peer DLE's on the same link. The second subproblem deals with the ability to establish real-time channels between two peer DLE's located on two different links. As mentioned above, establishing a connection between two communicating DLE's and reserving all the required resources are the only way to achieve predictable communication. Since the first subproblem is more likely to happen than the second one, the connection establishment procedure and the resource management for the connections of two peer DLE's on the same link should be fast and efficient. For the second subproblem, in order to improve the utilization of the entire network, and facilitate the routing problem between two peer communicating DLE's, a network manager is used to handle connections between two nonlocal DLE's.

The network manager (NM), which is a centralized service that handles real-time connection establishment and maintenance, is not present on all nodes in the system. Basically, the NM's function includes receiving real-time connection requests, trying to select a route which can provide the requested quality of service, informing all intermediate nodes (LAS and bridges) if such a route is available, and replying to the requesting node. The NM must maintain the information necessary for connection establishment, including the topology of the network and the reservation/utilization status of resources like link capacities and buffer space in all bridges. It also maintains a table containing the resource requirements, the assigned routes and the priorities of all existing connections in the system. In order to ensure the consistency of this data, the NM serializes the connection creation/deletion operations. The detailed NM operations will be described in the next section.

III. Basic Approach

The communication capacity of each link is divided into two parts. One part is concerned with intralink communications which are managed by the local LAS. The other part deals with interlink communications which are managed by a global NM. In other words, all intraworkcell communications between two peer DLE's on a link are scheduled by the LAS on that link. The interworkcell real-time communications between two peer DLE's on two different links are managed by the global NM, and scheduled by the LAS of each link of the path over which the corresponding connection runs. The fraction of the link capacity assigned to each of these two parts depends on the distribution of communication demands on each link. That is, the LAS of a link can reserve a different fraction of link capacity for local (or intra-link) communications based on the characteristics of local communication demands. Since communication traffic may vary, the reserved capacity may also vary. In this paper, we will use a version of the *linear bounded model* to describe the traffic generated by a real-time connection. If D is the maximum delivery-delay bound of a packet, the traffic generated by a real-time connection is said

to follow the linear bounded model if the number of packets generated in any interval $T \geq D$ is bounded by a linear function of the length of the interval T, i.e., $(GT+B)$, where G is the maximum packet-generation rate of this connection and B is the maximum burst size. This model was first proposed by Cruz [8] and also adopted by several researchers [6], [7], [9], [10], [12], [21]. The proposed solutions to the two subproblems are described in detail in the following two subsections.

A. Intralink Communication

Intralink (intracell) communications occur between two peer DLE's on the same multiaccess link. This type of communication can be either connection-oriented or connectionless. Intralink real-time communications are handled by the LAS DLE on the corresponding multiaccess link, because the LAS is designed to be the centralized control entity for scheduling messages on that local link under the FieldBus protocol [2], [3]. Since real-time communication requires a bounded delay in message delivery, we need to reserve all the required resources in advance in order to guarantee all real-time messages to be delivered before their deadlines. This implies that real-time communication be connection-oriented.

In the proposed connection establishment procedure, the operations of the peer DLE's are the same as those in the description of the draft standard [2], [3]. That is, the source DLE will make a connection request which contains necessary information for establishing the connection according to the FieldBus protocol, such as a frame control field, the destination address, and the quality of service. In order to support real-time communication, the traffic generation characteristics and the delivery delay requirement also need to be specified in a connection request message. Specifically, in addition to the required information in the draft standard, the following two parameters must be included in the connection request message:

- D (seconds): the user-specified delivery-delay bound for a message;
- M (packets): the maximum number of packets that can be generated in an interval of length D.

These two parameters describe a more general model than the linear bounded model, because the linear bounded model is only a special case of this model (by letting $M = GD+B$). Note that the maximum packet size is not included here because in the FieldBus protocol [2], [3], the maximum size of a high priority packet is fixed to be 64 bytes.

Based on these two parameters, the LAS can make necessary and sufficient reservation to guarantee all real-time messages to be delivered by their deadlines [7]. After receiving a connection request, the LAS will try to reserve the link capacity for this connection by performing the following admission test:

$$\sum_i \left(\frac{M_i P_{\max} + \text{overhead}_i}{D_i} \right) \leq 1 \qquad (1)$$

where $P_{\max}$ denotes the time needed to transmit a maximum-size packet and the index i runs over all existing real-time connections and the currently requested connection. The main part of overhead is determined by the token passing time.

If the admission test can be satisfied after adding this new connection, the LAS will reserve the required link capacity, update the information about the existing real-time connections, and send a confirmation message to the requesting node. Otherwise, the LAS will send a rejection message to the requesting node.

Using these parameters, the LAS can use the deadline-driven scheduling algorithm described in [14] to allocate tokens to each real-time connection. Basically, after an intralink real-time connection i with parameter D_i and M_i is established, the LAS will allocate a token to connection i at least once every D_i units of time and during the lifetime of each token, at least M_i maximum-size packets of connection i can be transmitted.

In addition to the operations specified in the draft standard, the LAS has to respond to all connection requests for local real-time connections since all real-time connections running through this local link are subject to its approval. Note that the LAS's approval is not required for nonreal-time connections since no resource and link capacity has to be reserved in advance for them. A connection request message includes a *reservation* (0 or 1) field, representing whether the local LAS has reserved the requested capacity or not. When the LAS receives a real-time connection request from a local DLE with a destination DLE located on the same link, the LAS will try to reserve the requested link capacity and respond to this connection request. Consequently, a real-time connection request is handled in three steps (see Fig. 1), whereas a nonreal-time connection request requires only two steps. First, the requesting DLE sends a connection request to the corresponding local LAS with reservation $= 0$ which represents the connection request has not yet been approved by the local LAS. Both the LAS and the designated receiver DLE receive this request, but the designated receiver DLE will ignore all real-time connection requests with reservation $= 0$. The LAS will respond to this request by sending out the modified connection request message with the requested quality of service and reservation $= 1$, if the LAS can reserve a sufficient capacity and accept this connection request. Otherwise, the LAS will reject this request by sending a rejection message to the requesting DLE. After receiving the positive response from the LAS, the receiver DLE will respond as described in the draft standard, i.e., report the connection request to the destination user, and the user will decide whether to accept this request or not. If the connection cannot be accepted, the receiver DLE will respond with a disconnection message, which will be received (almost) simultaneously by both the LAS and the requesting DLE since the link is a multiaccess bus. The LAS will then release all the resources reserved for this connection.

If there are already too many real-time connections established nearly exhausting all the link capacity under the control of a LAS, the LAS can make a request to the global NM for more link capacity for local usage. However, this is subject to the availability of the remaining link capacity at the NM level.

In order to be compatible with the current draft standard, if the source DLE requires an immediate reply for a real-

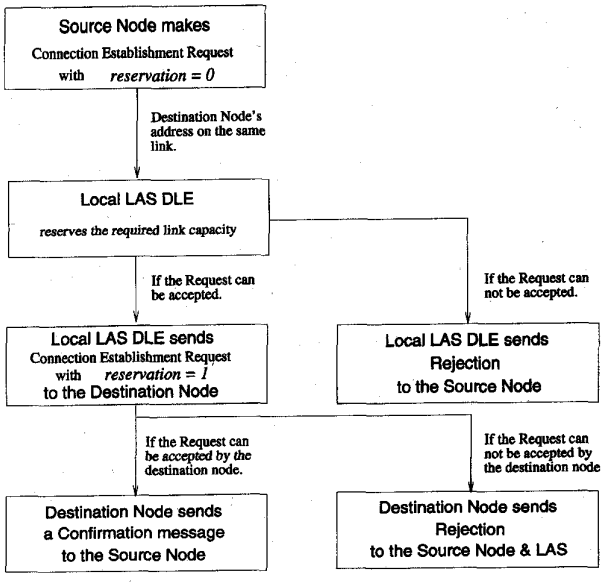


Fig. 1. Procedure for handling an intralink connection request.

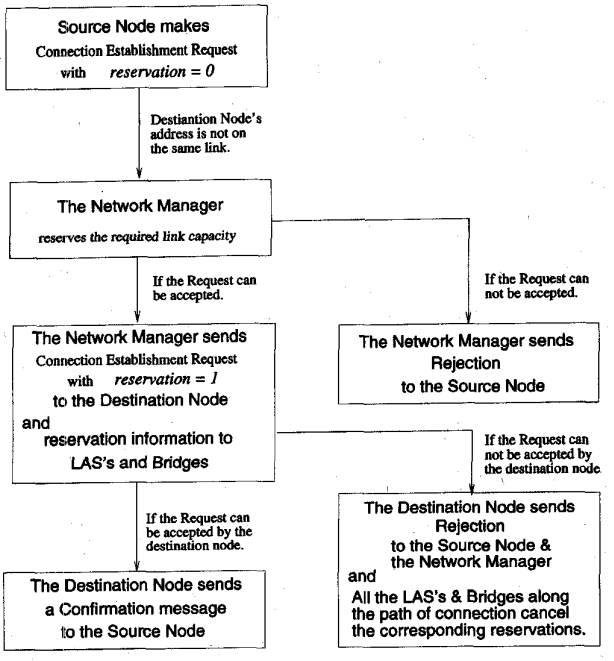


Fig. 2. Procedure for handling an interlink connection request.

time connection request, the receiver DLE will respond with an acknowledgment immediately. Otherwise, the source node will send the connection request again as specified in the draft standard. If the addressed destination still does not respond, the source DLE will report the failure of the destination DLE to the user, and the LAS will stop the reservation process and free all the resources reserved thus far for this particular request. When an immediate reply is required, the LAS will respond to the requesting DLE in the very next time slot after detecting the required immediate reply.

B. Interlink Communication

The entire operation for establishing a connection between two DLE's on different links is conceptually similar to the case when they are on the same link. The only significant difference is that the real-time connection requests are granted by the NM, rather than a local LAS. The real-time connection establishment procedure still consists of three steps (see Fig. 2). First, the requesting DLE makes a real-time connection request with reservation $= 0$ which represents the connection request has not yet been approved by the NM. Since the destination DLE is not on the same local link as the requesting DLE, the local LAS will ignore this connection request, and the bridges will forward this request to the NM by using datagram services. (Note that in the nonreal-time connection request case, the bridge will forward the request to the addressed destination DLE, instead of the NM.) Then the NM will try to find a path from the source DLE to the destination DLE with the required quality of service. If there are many paths available, it will choose a path by balancing the network traffic. If such a path is found, the NM will reserve the required resources and send a connection request to the destination DLE with reservation $= 1$ and the source address is the requesting DLE. At the same time, the NM also informs all the LAS's and bridges along the path of this new connection. Each intermediate bridge in this path will have the information on the size of reserved buffer space and how to forward messages of the newly-established connection. Each LAS also has the information on how to assign the required link capacity to some intermediate bridges. On the other hand, if no path can be found with the requested quality of service, the NM will send a rejection message to the requesting DLE.

Finally, after the destination DLE receives the connection request from the NM, the receiver DLE will report this request to the destination user who will then decide its acceptance/rejection. In case of acceptance, the destination DLE will send a confirmation message back to the requesting DLE. Otherwise, a disconnection message will be sent along the established path to the requesting node, thus making all intermediate LAS's and bridges aware of this disconnection. A disconnection message is also sent to the NM by the destination DLE, and the NM will update its information and release all the associated resources. In this case, the NM does not have to inform the intermediate nodes of the cancellation of the connection, since all intermediate LAS's and bridges have already been informed of this by the destination DLE.

If the destination DLE does not exist or does not respond within a timeout period to the connection request sent by the NM, the failure of the destination DLE can be detected by the bridge on the same link where the destination DLE resides and will be reported to the NM by returning the connection request to the NM. The NM may choose to retry or inform the requesting DLE and all intermediate LAS's and bridges about the rejection of this connection.

IV. Compatibility with the Draft FieldBus Protocol

Since most part of the FieldBus protocol has been well developed and gained general acceptance from the manufacturing and process control communities, the proposed scheme for

real-time communication must be compatible with the draft FieldBus protocol. The most notable aspects of the proposed scheme are the introduction of the network manager, the division of link capacity, and the difference in establishing real-time and nonreal-time connections.

Since the NM is responsible for all interlink (intercell) real-time connections, it has to be reachable from all entities in the entire network. There are two ways to achieve this goal: 1) all entities that may make an interlink real-time connection request maintain the NM's address individually; and 2) all bridge DLE's are responsible for recognizing interlink real-time connection request packets, forwarding interlink real-time connection requests to the NM, and thus, only bridge DLE's have to know the NM's address. Both 1) and 2) are compatible with the current draft FieldBus protocol, and easy to implement.

The second difference introduced by the proposed scheme is the division of the link capacity into two parts which are controlled by either the corresponding local LAS or the NM. This link capacity division can be easily accommodated into the current protocol. In the current FieldBus protocol standard, the entire link communication capacity is controlled by the local LAS, regardless whether the communication is intralink or interlink and whether it is connectionless or connection-oriented. In order to improve the utilization and traffic balancing of the network, we introduced the NM for interlink real-time communication, and as mentioned before, a portion of link capacity is controlled by the NM. Each LAS can still function as described in the draft FieldBus protocol except some portion of link capacity is assumed to have already been reserved for global usage. The LAS just follows the NM's instruction (in the request form) when assigning the requested link capacity to some designated bridge DLE(s) and/or the application DLE(s). Since the NM is only allowed to allocate a prenegotiated portion of the link capacity, and never exceeds it, the LAS must follow the NM's instruction. At the same time, the LAS may also grant its local requests without any further checking with the NM. The portion of link capacity which is controlled by the NM can be negotiated, i.e., when a LM DLE becomes the active LAS on a link, it first informs the NM of the portion of link capacity of this local link that the NM can use. From a scheduling perspective, the local LAS schedules the tokens according to the requests granted by both itself and the NM. Since both the LAS and the NM cannot exceed their prenegotiated limits, the messages for real-time connections can always be delivered in time once the connection has been established.

The real-time connection establishment procedure is also different from the nonreal-time counterpart. Establishing a real-time connection requires three steps, while establishing a nonreal-time connection requires only two steps. This difference comes from the fact that a real-time connection establishment has to be granted by either the local LAS or the NM. The real-time connection request is considered valid by the node where the destination DLE resides even before the destination DLE accepts the request. In such a case, the destination DLE may receive not-yet-accepted (reservation = 0) requests, but it will ignore them. So, the real-time connection request can be processed by the DLE's in the same way as a nonreal-time connection request. The draft FieldBus protocol uses a state-driven procedure to manage a nonreal-time connection, which is briefly described here. As we shall see later, a real-time connection can also be achieved by this state-driven procedure. The source (requesting) node makes a real-time or nonreal-time connection request, initiates a state machine ($V_c(ST)$ in the standard) for the connection, and enters "Outgoing Connection Pending" state from "Idle" state after the request is sent [2], [3]. A node in "Outgoing Connection Pending" state that has received a connection establishment confirmation enters "Data Transfer Ready" state and begins transmission. On the other hand, a node receives a nonreal-time or a valid real-time connection request will enter "Incoming Connection Pending" state. After receiving the user's positive response to this request, the DLE in "Incoming Connection Pending" state sends out a connection establishment confirmation to the requesting node, enters "Data Transfer Ready" state, and begins transmission. A user's negative response (reject or close) to a state machine will force the node to send a disconnection message and the associated state machine will enter "Idle" state. Therefore, real-time connection requests can be processed by the same state-driven model with minor modification (adding a reservation bit) as the nonreal-time connection case. However, the NM and the LAS have to be equipped with the ability to handle real-time connection requests, make appropriate reservations, and respond to such requests adequately. The bridge DLE's also require additional functions to make correct run-time scheduling and flow control in order to provide predictable communication [10], [11]. Although these changes to the LAS and bridge are nontrivial, they are compatible with the draft FieldBus protocol, and are necessary to support real-time communication.

V. Performance Evaluation

In this section, using two protypical example systems we demonstrate the superior performance of the proposed modification of FieldBus. The first example deals with a typical manufacturing system network and the second example is concerned with multimedia networking; both the examples require real-time communication services. These examples are used to compare the proposed scheme and the token-passing protocols (e.g., token buses, token rings, and FDDI) in terms of the ability of supporting real-time communication.

Example 1: We simulated a typical manufacturing system (Fig. 3) to determine the percentage of accepting real-time connection requests and the average maximum nonreal-time message throughput achievable under different link load conditions for both intracell and intercell communications [2], [3], [10], [14]. A cell in Fig. 3 represents a workcell.

Real-time communications in manufacturing systems are usually periodic in nature, and the deadline of a message is related to its period. For example, the controller of a workcell usually must read sensors periodically, and the sensed data should be processed before the next period begins. Based on this observation, we assume that each real-time connection has its own period, and in each period, a fixed amount of

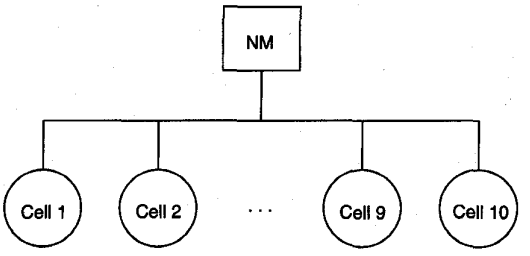


Fig. 3. Two-level hierarchical network for an AF.

time is assigned to the source node of the connection if the connection can be established, because the size of each time-critical message is limited in the FieldBus. This fixed amount of time is assumed to be used for handling a message of 256 bytes—which includes a maximum of 128 bytes user data for time-critical messages—and all overheads such as token passing time, transmission delay, and framing delay. The reason why we do not use any specific message packetization is that the ISA (Instrument Society of America) has not yet finalized the formats of the packet header and various tokens [2], [3]. Those connections needing to send more than 256 bytes in a period are simulated by making multiple connection requests.

Arrivals of real-time connection requests are assumed to be exponentially distributed with a fixed rate. The real-time connection periods are assumed to be uniformly distributed within the range from 20–500 ms for the manufacturing system under consideration. Since all operations in a manufacturing system are usually carefully planned in advance, the lifetime of each activity will be approximately the same for all of its invocations. Therefore, in addition to the arrival rate and period, the lifetime of each connection is assumed to be normally distributed with a fixed mean ranging from one to tens of seconds (depending on the type of the corresponding activity) and a small variance. For example, a robot may need a connection with a short-life device for 10 s when it operates on an assembly line, but it may only need a connection with a long-life device for the next several seconds when the transport belt is moving.

In this example, we use several different lifetime distributions, but, as long as the total requesting load remains the same, the percentage of accepting real-time connection requests and the average maximum achievable nonreal-time message throughput do not make any significant difference.

Figs. 4 and 6 show the probability of accepting an intracell/intercell real-time connection request under a wide range of requested real-time connection load. When the requested real-time connection load is under 100% of the link capacity, the speed of the link is important to the acceptance probability. The reason for this tendency can be given as follows. The requested connection's load on a link is the sum of the loads of all requested connections regardless whether they are accepted or not. Insofar as a single connection request is concerned, the load of a single real-time connection request is measured by the number of *slots* (each of which is the time required to handle a 256-byte packet) the connection needs per second or its period in terms of slots, and this load occupies a higher

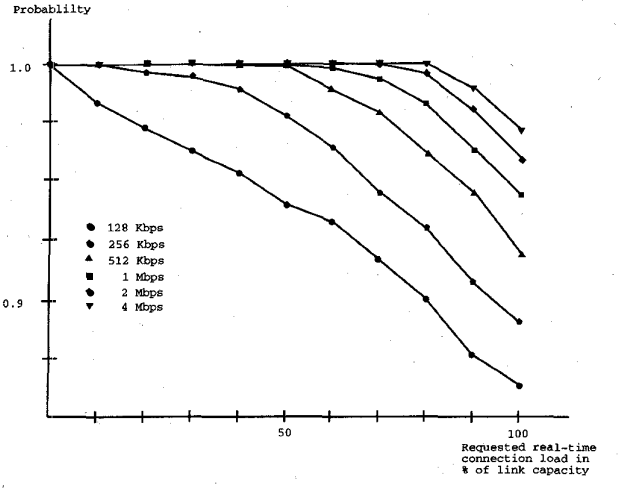


Fig. 4. Probability for honoring intracell real-time connection requests.

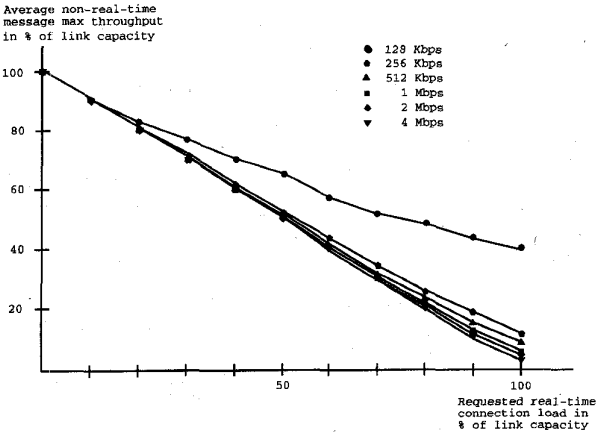


Fig. 5. Average maximum throughput for intercell nonreal-time messages.

percentage of the capacity of a low-speed link than it does on a high-speed link. Therefore, when the requested real-time connection load is less than 100% of the link capacity (i.e., the link has some unused capacity), the chance that a new requested connection cannot be accommodated in the remaining capacity of a low-speed link is much higher than a high-speed link.

Figs. 4 and 5 show, respectively, the percentage of accepting real-time connection requests and the average maximum achievable nonreal-time message throughput for intracell communication. As can be seen in Fig. 4, the higher link speed, the higher acceptance probability. The nonreal-time message throughput has an opposite trend. In Fig. 5, the low-speed link has a higher nonreal-time message throughput in terms of percentage of the link capacity, since a lower percentage of link capacity is reserved for real-time communications.

Figs. 6 and 7 show, respectively, the percentage of granting real-time connection requests and the average maximum achievable nonreal-time message throughput for intercell communication. For these plots we used a 4 Mbps backbone to connect 10 workcells, and three different link capacities for connections within a workcell, i.e., 128 Kbps, 256 Kbps, and

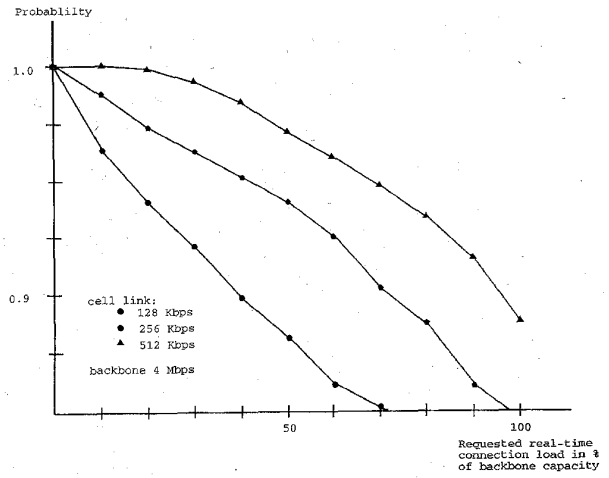


Fig. 6. Average maximum throughput for intracell nonreal-time messages.

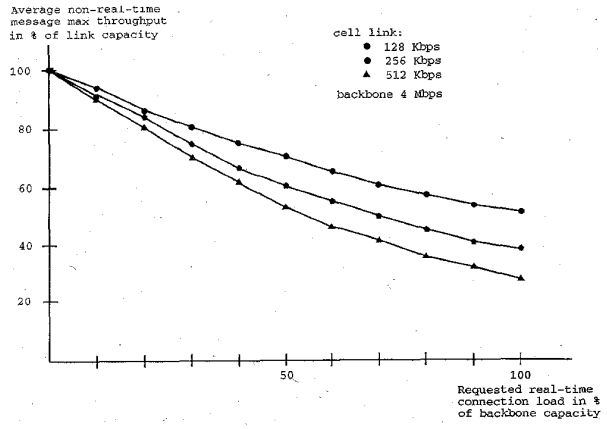


Fig. 7. Average maximum throughput for intercell nonreal-time messages.

512 Kbps. In our strategy, intercell and intracell communications do not interfere with each other, since they use different portions of the link capacity. These two figures follow the same trend as Figs. 4 and 5. However, the percentage of accepting intercell real-time connection requests drops much faster, since an intercell connection can be established only if all links on the connection have sufficient capacity to support this connection's requirements, while an intracell connection only needs one link which has sufficient capacity to establish. Because a smaller percentage of link capacity is reserved for real-time connections, the average maximum achievable nonreal-time message throughput is higher in the intercell communication case (as can be seen from Figs. 5 and 7).

Example 2: Since most real-time communication takes place between two DLE's on the same link, this example focuses on the ability of supporting intralink real-time communication. Both the normal token-passing protocol and the proposed scheme are used to send compressed digital motion-video frames and their performances are compared. This example shows that the proposed scheme reserves a bandwidth required for real-time traffic and utilizes the

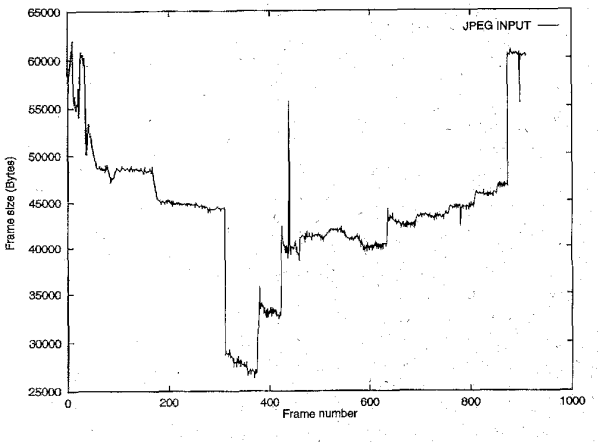


Fig. 8. An example of frame sizes (compressed with JPEG).

network resources efficiently in the absence of real-time traffic, thus, can provide adequate real-time performance and outperform conventional token passing protocols.

The underlying interconnection hardware is a 100 Mbps multiaccess link/bus with 20 or 50 nodes. The video data were obtained by sampling a sequence of CNN headline news, stored on a laser disk [24]. The size of each frame, after compressed with JPEG [22], [24], is plotted in Fig. 8. The video quality can be characterized by the rate of "successfully-delivered" frames, where a "successfully-delivered" frame is defined as the one delivered to its destination correctly before the corresponding deadline. The maximum one-way transmission delay of each frame must be less than 100 ms in order to achieve the quality of live-video performance, i.e., $D = 100$ ms. If we use the transmission rate of 30 frames per second, 3 frames will be transmitted during each 100 ms period. Assume the maximum packet size supported by the network is 1 Kbytes, then by adding the sizes of three consecutive frames (because there are three frame arrivals in 100 ms), we can derive $M = 185$ packets. By adding the token passing overhead of 1 Kbytes per 100 ms (i.e., 500 bytes each way), the network is expected to support $\lfloor 1250/186 \rfloor = 6$ real-time connections under the proposed scheme because the 100 Mbps link can be used to transmit 1250 maximum-size packets during each 100 ms period.

Certain uniformly-distributed nonreal-time traffic that requires from 0–90% of the total link capacity is added to each node during the simulation. However, the source nodes of real-time connections are randomly chosen. (It is thus possible that one node may become the source node of all real-time connections.) The traffic data of real-time connections were taken from Fig. 8 and each connection has its own starting frame (also randomly chosen).

On the same multiaccess link (with 20 or 50 nodes), we will also use the above real-time traffic to assess the ability of supporting real-time communication under the normal token-passing protocol. However, we assume the token-passing overhead is negligible and the source nodes of existing real-time connections are distributed evenly among all nodes. Although these assumptions will make the token-passing protocol work better than actually is, this example will show

that the proposed scheme still performs much better than the token-passing protocol in supporting real-time communication.

Though the underlying network is a multiaccess bus, the token will be passed among all nodes on the bus in a manner which is exactly the same as FDDI [1], [5], [15], [16], [19], [20], [23]. Since the packet delivery deadline is 100 ms, the target token rotation time (TTRT) of the token-passing protocol is set to 50 ms. Depending on the number of nodes (20 or 50 in this example), the high-priority token holding time of each node is set to be $50/20 = 2.5$ ms or $50/50 = 1$ ms. Basically, each node is guaranteed to receive a token at least once every $2 \times$ TTRT (100 ms) and upon capturing the token, the node is allowed to transmit real-time (high-priority) traffic for up to 2.5 ms on a 20-node network or 1 ms on a 50-node network. Nonreal-time traffic can be transmitted only when a node holds the token and the elapsed time since this node's last release of the token is less than TTRT. After receiving the token, a node may transmit real-time traffic for up to 2.5 ms (or 1 ms) first, and then if the above condition is met, more real-time traffic or nonreal-time traffic can be transmitted until the above condition becomes no longer true or all packets are transmitted, whichever occurs first. The token will then be passed to the (logically) next node.

The goal of this example is to evaluate the capability of supporting real-time communication by comparing the maximum and average *frame-miss rates* of both the token-passing protocol and the proposed scheme. (The frame-miss rate is defined as the percentage of frames missing their deadlines.) Our scheme is shown to always outperform the token-passing protocol in supporting real-time communication.

Figs. 9 and 10 show the maximum frame-miss rate of our scheme, and the maximum and average frame-miss rates of 20-node and 50-node token-passing networks at a 50% nonreal-time traffic load. The frame-miss rate is defined for each connection, i.e., the ratio of the number of frame misses for a connection C to the total number of frames of C. The maximum (average) frame-miss rate is defined to be the largest (average) value of individual connection frame-miss rates. A point in the figure represents 30 000 cycles of the sequence for *each* connection, i.e., about 911 000 frames or 8.4 h at the rate of 30 frames/s for each connection. As one can derive from (1), both figures show that 6 connections can be supported under the proposed scheme. For those points where the link does not have sufficient bandwidth for both real-time and nonreal-time traffic (e.g., 7–10th connections), we reserved the entire link capacity for real-time connections, and nonreal-time traffic is transmitted only when the bandwidth reserved for real-time traffic is not fully used. Since we reserved the link bandwidth using the worst-case values, this is very likely to happen.

For example, we reserved $\lceil 1250/7 \rceil$ packet times for each connection when there are 7 real-time connections. (Note, however, that our scheme does not allow a seventh real-time connection, because the system cannot guarantee the required performance.) When a seventh connection is added, only about 2% of real-time packets of this connection will miss deadlines.

The token-passing protocol on a multiaccess bus can provide 4–5 real-time connections at a 50% nonreal-time load. As we shall see, the token-passing protocol's ability to support real-time communication is highly sensitive to the nonreal-time traffic load. The average frame-miss rate for our scheme is very close to the maximum miss rate, so we only present the maximum frame-miss rate in Figs. 9 and 10. By contrast, the average and maximum frame-miss rates of the token-passing protocol are significantly different when the multiaccess bus cannot transmit all the real-time messages before their deadlines. The FDDI's frame-miss rate is also sensitive to the number of nodes on the ring, but our scheme can provide the *same* number of real-time connections regardless of the number of nodes on the multiaccess link. This observation implies that the variance of frame-miss rates of our scheme is very small, whereas the token-passing protocol suffers a large variation of frame-miss rates.

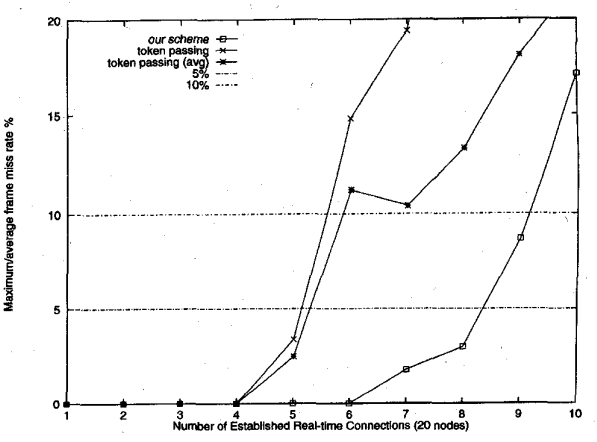


Fig. 9. Maximum frame-miss rate at 50% nonreal-time traffic load (20 nodes).

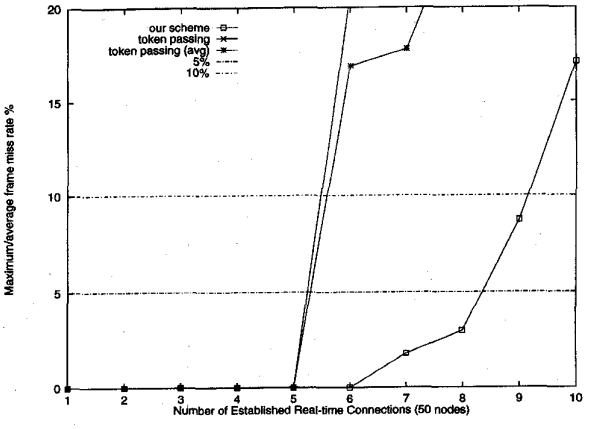


Fig. 10. Maximum frame-miss rate at a 50% nonreal-time traffic load (50 nodes).

Figs. 11 and 12 show the maximum frame-miss rate of our scheme as well as the maximum and average frame-miss rates of 20-node and 50-node token-passing networks at a 90% nonreal-time traffic load. At a high nonreal-time traffic load like this, the token-passing protocol almost becomes incapable of supporting real-time communication. It can only allow for one connection (without loss) on both 20-node and 50-node networks.

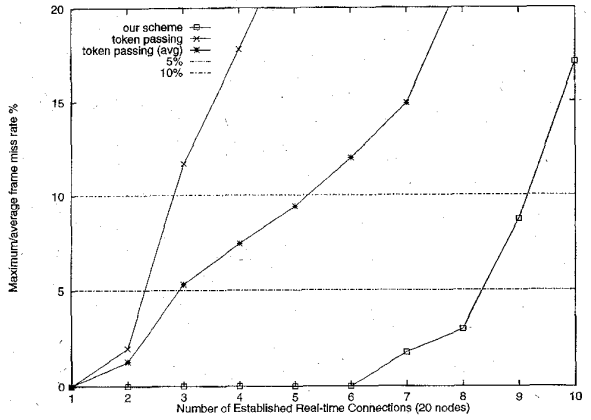


Fig. 11. Maximum frame-miss rate at a 90% background load (20 nodes).

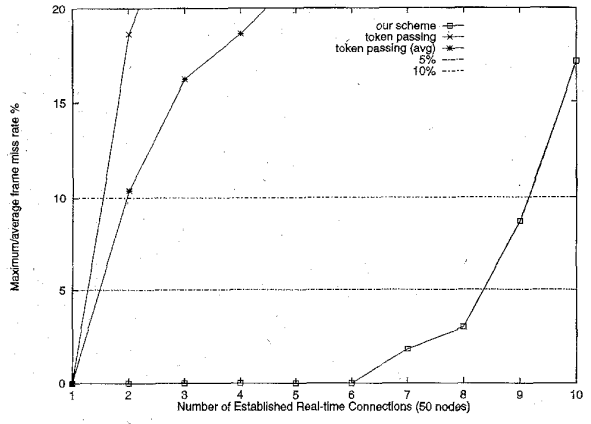


Fig. 12. Maximum frame-miss rate at a 90% background load (50 nodes).

By contrast, our scheme is insensitive to the nonreal-time traffic load. The system can still provide 6 real-time connections at a high nonreal-time traffic load. Again, since the average frame-miss rate of our scheme is very close to the maximum frame-miss rate, we plot only the maximum frame-miss rate. (Similarly to the previous case, the token-passing protocol's average frame-miss rate significantly differs from its maximum frame-miss rate.) As far as the ability of supporting real-time communication is concerned, our scheme is shown to be far better than the token-passing protocols, such as token buses, token rings, or FDDI.

We also calculated the token-passing overhead of the proposed scheme under the assumption that the size of a token is 500 bytes. Since two tokens are needed for each token allocation (one to issue and the other to return), and the average traffic arrival rate of a real-time connection is approximately 132 Kbytes per 100 ms, the token-passing overhead is approximately 0.76%.

VI. Conclusion

In this paper, we proposed a strategy to support real-time communication under the FieldBus protocol which provides end-to-end delivery delay guarantees for time-critical messages. This strategy provides a fast local mechanism for establishing intraworkcell real-time connections, while supporting global interworkcell real-time connections. The proposed strategy is fully compatible with the current draft standard of FieldBus protocol, and also provides flexibility for the choice of scheduling algorithms, adaptability for different traffic loads. Numerical examples are also given based on typical manufacturing systems as well multimedia networking.

Our future work will focus on the scheduling algorithms for the LAS and the route selection algorithm (to be used by the network manager) which are also very important to the success of FieldBus.

Acknowledgment

The authors would like to thank G. Workman and B. Gross of General Motors for their technical support for the work described in this paper.

References

[1] "FDDI hybrid ring control, draft proposal to American National Standard," 1989.

[2] *Fieldbus Standard for Use in Industrial Control Systems—Part 3: Data Link Service Definitions,* Draft Standard, ISA-dS50.02 ISA/SP50-1993-359L, Instrument Society of America, 1993.

[3] *Fieldbus Standard for Use in Industrial Control Systems—Part 4: Data Link Protocol Specification,* Draft Standard, ISA-dS50.02 ISA/SP50-1993-360L, Instrument Society of America, 1993.

[4] D. P. Anderson, S. Y. Tzou, R. Wahbe, R. Govindan, and M. Andrews, "Support for continuous media in the dash system," in *Proc. 10th Int. Conf. Distributed Computing Systems,* May 1990, pp. 54–61.

[5] D. Bertsekas and R. Gallager, *Data Networks.* Englewood Cliffs, NJ: Prentice-Hall, 1987.

[6] C.-C. Chou and K. G. Shin, *Multiplexing Statistical Real-Time Channels on a Multiaccess Network,* to be published.

[7] ——, "Statistical real-time channels on multiaccess networks," in *Proc. Int. Conf. Computer Communication,* Aug. 1995, pp. 29–34.

[8] R. L. Cruz, "A calculus for network delay and a note on topologies of interconnection networks," Ph.D. dissertation, Univ. Illinois at Urbana-Champaign, July 1987.

[9] D. Ferrari and D. C. Verma, "A scheme for real-time channel establishment in wide-area networks," *IEEE J. Select. Areas Commun.,* vol. 8, pp. 368–379, Apr. 1990.

[10] D. D. Kandlur and K. G. Shin, "Design of a communication subsystem for HARTS," CSE Div., Dep. Elect. Eng. Comput. Sci., Univ. Michigan, Ann Arbor, Tech. Rep. CSE-TR-109-91, 1991.

[11] D. D. Kandlur, K. G. Shin, and D. Ferrari, "Real-time communication in multi-hop networks," in *Proc. 11th Int. Conf. Distributed Computing Systems,* 1991, pp. 300–307.

[12] D. D. Kandlur, "Networking in distributed real-time systems," Ph.D. dissertation, Univ. Michigan, Ann Arbor, 1991.

[13] J. F. Kurose, M. Schwartz, and Y. Yemini, "Multiple-access protocols and time-constrained communication," *ACM Comput. Surveys,* vol. 16, pp. 43–70, 1984.

[14] C. L. Liu and J. W. Layland, "Scheduling algorithm for multiprogramming in a hard-real-time environment," *J. ACM,* vol. 20, no. 1, pp. 46–61, Jan. 1973.

[15] F. E. Ross, "FDDI—A tutorial," *IEEE Commun. Mag.,* vol. 24, pp. 10–17, May 1986.

[16] ——, "An overview of FDDI: The fiber distributed data interface," *IEEE J. Select. Areas Commun.,* vol. 7, no. 7, pp. 1043–1051, Sept. 1989.

[17] K. G. Shin, "Real-time communications in a computer-controlled workcell," *IEEE Trans. Robot. Automat.,* vol. 7, pp. 105–113, Feb. 1991.

[18] J. K. Strosnider and T. E. Marchok, "Responsive, deterministic IEEE 802.5 token ring scheduling," *J. Real-Time Syst.,* vol. 1, pp. 133–158, Sept. 1989.

[19] A. S. Tanenbaum, *Computer Networks,* 2nd ed. Englewood Cliffs, NJ: Prentice-Hall, 1989.

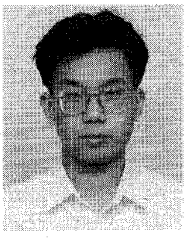
[20] M. Tangemann and K. Sauer, "Performance analysis of the timed token protocol of FDDI and FDDI-II," *IEEE J. Select. Areas Commun.,* vol. 9, no. 2, pp. 271–278, Feb. 1991.

[21] D. C. Verma, "Guaranteed performance communication in high speed networks," Ph.D. dissertation, Univ. California, Berkeley, 1991.
[22] G. K. Wallace, "The JPEG still picture compression standard," *Commun. ACM,* vol. 34, no. 4, pp. 30–43, Apr. 1991.
[23] J. Walrand, *Communication Networks: A First Course.* New York: Irwin and Aksen Assoc., 1991.
[24] Q. Zheng, K. G. Shin, and E. Abram-Profeta, "Transmission of compressed digital motion video over computer networks," in *COMPCON Spring'93,* 1993, pp. 37–46.

Kang G. Shin (S'75–M'78–SM'83–F'92) received the B.S. degree in electronics engineering from Seoul National University, Seoul, Korea, in 1970, and both the M.S. and Ph.D. degrees in electrical engineering from Cornell University, Ithaca, NY, in 1976, and 1978, respectively.

He is currently Professor and Director of the Real-Time Computing Laboratory, Department of Electrical Engineering and Computer Science, University of Michigan, Ann Arbor. He has authored/coauthored more than 350 technical papers (more than 150 of these in archival journals) and numerous book chapters in the areas of distributed real-time computing and control, fault-tolerant computing, computer architecture, robotics and automation, and intelligent manufacturing. He is currently writing (jointly with C. M. Krishna) a textbook entitled *Real-Time Systems*, which is scheduled to be published by McGraw-Hill in 1996. In 1985, he founded the Real-Time Computing Laboratory, where he and his colleagues are currently building a 19-node hexagonal mesh multicomputer, called **HARTS,** and middleware services for distributed real-time fault-tolerant applications. He has also been applying the basic research results of real-time computing to multimedia systems, intelligent transportation systems, and manufacturing applications ranging from the control of robots and machine tools to the development of open architectures for manufacturing equipment and processes. From 1978 to 1982, he was on the faculty of Rensselaer Polytechnic Institute, Troy, NY. He has held visiting positions at: the U.S. Air Force Flight Dynamics Laboratory; AT&T Bell Laboratories; Computer Science Division within the Department of Electrical Engineering and Computer Science at UC Berkeley; International Computer Science Institute, Berkeley, CA, IBM T. J. Watson Research Center; and Software Engineering Institute, Carnegie Mellon University. He also chaired the Computer Science and Engineering Division, EECS Department, University of Michigan, for three years beginning January 1991.

Dr. Shin received the Outstanding IEEE Transactions on Automatic Control Paper Award for a paper on robot trajectory planning in 1987. In 1989, he received the Research Excellence Award from The University of Michigan. He was the Program Chairman of the 1986 IEEE Real-Time Systems Symposium (RTSS), the General Chairman of the 1987 RTSS, the Guest Editor of the 1987 August special issue of IEEE TRANSACTIONS ON COMPUTERS on Real-Time Systems, a Program Co-Chair for the 1992 International Conference on Parallel Processing, and served on numerous technical program committees. He also chaired the IEEE Technical Committee on Real-Time Systems during 1991–1993, was a Distinguished Visitor of the Computer Society of the IEEE, an Editor of IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED COMPUTING, and an Area Editor of *International Journal of Time-Critical Computing Systems.*



Chih-Che Chou (S'91–M'93) received the B.S.E.E. degree in electrical engineering from National Taiwan University, Taipei, Taiwan, in 1988, and both the M.S. and Ph.D. degrees in computer science and engineering from the University of Michigan, Ann Arbor, in 1992 and 1994, respectively.

He joined AT&T Bell Laboratories in 1994 and is currently a Member of Technical Staff. His research interests include real-time communications, real-time operating systems, and communication networks.