Statistical Real-Time Channels on Multiaccess Bus Networks

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

Abstract—Real-time communication with performance guarantees is expected to become an important and necessary feature of future computer networks. In this paper, we present a scheme which can provide real-time communication services with both absolute and statistical performance guarantees on multiaccess bus networks for given input traffic characteristics and performance requirements. The proposed scheme reserves network bandwidth for real-time connections according to their needs. It also allows for independent addition and deletion of real-time connections while preserving existing guarantees. Our extensive simulation results for motion video communication have shown the proposed scheme to outperform the other well-known schemes.

Index Terms—Real-time communication, real-time connection/channel, multimedia systems, FieldBus (SP50), multiaccess networks, digital video transmission.

1 Introduction

Real-time communication with performance guarantees is becoming very important to many applications, like computer-integrated manufacturing, multimedia, and many embedded systems. Performance/deadline guarantees are, by definition, a certain grade of services which are promised by the communication system. Such guarantees may be defined by user-specified parameters at the time of setting up real-time communication services. The maximum message-delivery delay and the maximum (acceptable) message-loss rate are two typical performance parameters. In order to provide performance guarantees, the underlying communication system has to reserve a priori certain resources for "anticipated" real-time traffic in order to meet performance requirements.

Several researchers investigated the problem of supporting real-time communication with performance guarantees for given worst-case input traffic characteristics in wide-area point-to-point networks [9], [12], [13], [14], [23], [27]. Among these, the concept of "real-time channel" proposed by Ferrari and Verma [9], and refined by Kandlur, Shin, and Ferrari [13], is the most notable in explicitly addressing the problem of meeting delivery deadlines in wide-area point-to-point networks. A real-time channel is a unidirectional virtual circuit which, once established, is guaranteed to meet user-specified performance requirements as long as the user does not violate his a priori specified traffic-generation characteristics [9]. Although the main focus of this paper is placed on the issue of providing real-time communication services on *multiaccess bus* networks, we will use the terminology "real-time channel" throughout the paper. However, due to their differences in hardware architecture, those schemes suitable for wide-area point-to-point networks may not be appropriate for multiaccess bus networks. In a point-to-point network, the nodes at the end of a link have complete control of the link, so the end nodes of the link can adopt any scheduling algorithm and utilize the entire link bandwidth, if needed. By contrast, nodes attached to a multiaccess link/bus need to cooperate/coordinate with one another in order to transmit data/control packets, since only one node at a time can transmit packets over the shared medium. Other types of local-area networks will also be discussed, but a multiaccess bus will be used as the interconnecting hardware for our scheme.

In addition to meeting "absolute" performance requirements on point-to-point networks, "statistical" performance requirements have also been investigated by several researchers [18], [20], [30]. However, most of them focused on the problem of defining/computing quality-of-service or grade-of-service and the strategy of multiplexing packets. They do not aggressively reserve resources to guarantee the performance requirements or lack of methods to make sufficient and efficient reservation of resources. Basically, most of them are still best-effort schemes.

Although point-to-point networks can be used in small-area systems (e.g., manufacturing cells and aircraft) to connect a large number of nodes, they are often not cost-effective due to the complexity of connecting hardware and the potentially long delivery delay. Multiaccess bus networks are more suitable than point-to-point networks for many small-area applications, such as computer-integrated manufacturing systems, campus networks, and various general-purpose networks for a small area. They are simple, economical, and their propagation/delivery delays are small. These advantages make multiaccess bus networks a very good candidate as the underlying architecture for small-area real-time applications. Although the coverage of a multiaccess bus network is limited by its bus length, one can cover a larger area by connecting several multiaccess bus networks together, or connecting them to a point-to-

• *The authors are with the Real-Time Computing Laboratory, Department of Electrical Engineering and Computer Science, The University of Michigan, Ann Arbor, MI 48109-2122.*
E-mail: {ccchou, kgshin}@eecs.umich.edu.

Manuscript received 15 Nov. 1994; revised 7 July 1995.
For information on obtaining reprints of this article, please send e-mail to: transpds@computer.org, and reference IEEECS Log Number 100950.0.

point network via routers or bridges. Hence, a multiaccess *bus* will be used in our scheme as the interconnecting hardware.

Several schemes for real-time communication on general multiaccess networks (not limited to bus networks) have been proposed [15], [21], [24], or implemented on the token ring and the token bus, but they generally belong to the "best-effort" category. They do not reserve resources according to the user-specified traffic characteristics and performance requirements, and have no explicit admission control. Thus, even if each node is guaranteed to have access to the network within a certain upper bound of delay, the network still cannot guarantee each node to deliver all real-time messages in time. In this paper, we propose a scheme which can provide performance guarantees for real-time traffic on multiaccess bus networks and, at the same time, improve network utilization.

The paper is organized as follows. In Section 2, we state the problem of providing real-time communication services on multiaccess bus networks. The proposed solution with absolute performance guarantees is presented in Section 3. Section 4 deals with the real-time communication with statistical performance guarantees. Run-time scheduling issues are discussed in Section 5. Section 6 presents comparative simulation results, and the paper concludes with Section 7.

2 Problem Statement

There are two main conceivable approaches to supporting real-time communication. One is the best-effort approach which does not provide any performance guarantees. The other is the "hard" real-time approach which provides "absolute" performance guarantees, but requires a priori reservation of all necessary resources based on the worst-case input traffic-generation behavior. Both of them have drawbacks of their own. The former does not provide any performance guarantees at all, and the latter often underutilizes the reserved resources.

Different applications come with different performance requirements and traffic-generation characteristics. Some applications, such as the conversation between two cooperating robots, may need hard real-time performance guarantees, while many other applications may require less stringent performance guarantees. For example, reading various sensors is a typical task in automated manufacturing systems, and missing some of these readings is tolerable as long as the missing frequency is lower than a prespecified value. For applications of this type, the best-effort approach is not suitable because it offers no performance guarantees at all. The hard real-time approach is not suitable either, as it requires significantly more resources to be reserved than actually needed. We propose to use the concept of *statistical real-time channel* to solve the problem of providing real-time communication with "statistical" performance guarantees for these applications.

A statistical real-time channel is defined as a unidirectional virtual circuit which can provide real-time communication with performance guarantees in statistical terms, e.g., the probability of a packet's delivery before its deadline D is greater than a given number Z. Because of its less stringent performance requirements, a statistical channel is likely to reserve much *less* resources than its hard real-time counterpart, yet provides an acceptable level of real-time performance. Although this concept was proposed in [9], how to realize it has not yet been studied in depth.

Both how user's traffic characteristics and performance requirements are specified and who (which layer) is responsible for this specification have an important bearing on the statistical channels. The current OSI seven-layer model can at best treat real-time communication as best-effort services, since there is no notion of performance guarantees or traffic specification. Moreover, since there are too many layer-to-layer data conversions, the seven-layer model is not adequate for real-time communication. The MINIMAP [11], [26] and the FieldBus [3], [4], [23] protocols have only three layers in order to reduce the time required for layer-to-layer data conversions. Under these protocols, a communication system consists of only three layers: the physical layer, the data link/network layer which is responsible for providing performance guarantees, and the application/user layer which deals with all user interfaces as well as user-specified traffic characteristics and performance requirements.

Obviously, it is the application/user layer's responsibility to derive various user performance requirements or traffic characteristics, since only this layer is aware of user's performance needs and traffic characteristics. The derivation should be independent of the network service provider, i.e., the data link/network layer which is responsible for implementing network communication with performance guarantees. So, the application/user layer is responsible for deriving traffic characteristics and performance requirements, and may choose to regulate user's input traffic [8], [14], [19], [27] in order to preserve existing performance guarantees. Although the exact distribution of incoming traffic may not always be available, an approximate distribution is usually not difficult to obtain off-line for most real-time applications. For example, during interactive playback of stored video, the exact distribution depends on the user's on-line instructions, but the original distribution from a sequential playback is usually a good approximation. For the other applications like video conferencing, we cannot predict in advance the exact distribution of the anticipated traffic, but an approximation is usually not difficult to obtain, e.g., follow industrial standards, or run extensive simulations/experiments and make conservative estimations.

On the other hand, the data link/network layer is responsible for testing, checking, accepting/rejecting requests for establishing real-time channels and provides performance guarantees by using both reservation and scheduling schemes which the application/user layer is not usually aware of. Since the derivation of performance requirements and traffic distribution depends greatly on the application under consideration, we will not discuss it any further. We will instead consider the problem of providing real-time performance guarantees at the data link/network layer in multiaccess bus networks under the assumption that the incoming traffic distribution and performance requirements are given. Generally, there are three types of medium access

protocols. The first type is CSMA/CD, such as the Ethernet, which is completely random and cannot provide any guarantee on the maximum access delay and, therefore, is not suitable for real-time applications. The second type is the distributed timed-token protocol, such as the token ring and token bus. In this type of protocol, a token is rotated among all nodes (usually in a fixed order) on the network, and only the node possessing the token is allowed to transmit data/control packets. The token rotation is governed by some rules which guarantee each node to have medium access at least once within some specified period. The third type of protocols uses a control unit for medium access control on each multiaccess link/bus (*not* ring). (Note that we will use "link" and "bus" interchangeably in this paper.) This control unit follows some rules to ensure that user-specified performance requirements will be met. For example, the FieldBus [3], [4], [23] protocol which is a newly-developed industrial standard for process control and manufacturing systems uses a multiaccess bus as the transmission medium and a link active scheduler (LAS) as the control unit of *each* multiaccess bus/link.

We propose a scheme to support real-time communication for type-three multiaccess protocols that include the FieldBus. We will also discuss the differences between type-two and type-three protocols, and identify the advantages of using type-three protocols and the disadvantages of using type-two protocols. Each multiaccess **link** (not ring) under our scheme has a link control unit which is responsible for controlling and coordinating all nodes' accesses to that link. The link control unit is also responsible for allocating tokens and testing whether or not to admit hard/statistical real-time channels (i.e., admission control). Each real-time channel is required to reserve a portion of link bandwidth by sending a channel establishment request to the link control unit before sending its first packet over the link.

3 Hard Real-Time Channels on Multiaccess Links

In order to provide statistical performance guarantees, like

$$P(\textit{delay of a packet} \le D) \ge \textit{a given } Z, \text{ or} \tag{3.1}$$

$$P(\textit{no packet loss in any time interval of a certain length}) \ge Z, \tag{3.2}$$

the communication system needs to know the distribution of packet arrivals in each channel. Those packets missing deadlines are considered "lost." Using the user-specified delay bound D, the maximum packet size S_{max}, the maximum burst size B_{max}, and the maximum packet-arrival rate G_{max}, one can establish a hard real-time channel in a point-to-point network using one of the schemes in [9], [12], [13], [31]. Considering the differences between multiaccess bus networks and point-to-point networks, we will first develop a new scheme for establishing hard real-time channels on multiaccess buses with a more general traffic model than that used for point-to-point networks. We will then identify the additional information needed to establish statistical real-time channels on a multiaccess link.

The primary differences of a channel/connection in a multiaccess bus network from that in a point-to-point network are the relationship between nodes and links, and the bus network's shorter packet-delivery latency. In a point-to-point network, a node has complete control of its transmission links and has complete knowledge on whether or not the packets of a channel running through the node need to be scheduled by simply examining its packet queues, one for each channel; that is, it is easy to multiplex several real-time channels. However, in a multiaccess bus, it is difficult to determine which node has the right to transmit at a particular instant, especially in the presence of both real-time and non-real-time packets.

There exist several medium access control protocols which can achieve fair medium access among all nodes, and some of them can support a limited form of real-time communication. For example, FDDI uses a target token rotation time (TTRT), a high-priority token holding time (for synchronous packets), and a token holding time (THT) to ensure that the maximum time for synchronous packets to wait for medium access will not be greater than $2 \times$ TTRT. Although we can adjust the high-priority token holding time and the algorithm of updating THT for each node using the anticipated load of real-time traffic so as to make a good distribution of transmission capacity, FDDI still has many inherent disadvantages. Two of these disadvantages are most noticeable for real-time traffic. First, TTRTs must be identical for all nodes on the ring. This restriction may seriously limit the use of FDDI when heterogeneous real-time traffic is to be handled. If some node requires a very short TTRT, the network utilization may be significantly reduced, as the token has to be rotated around the ring very fast[1] (thus wasting the bandwidth on token passing). This is also a disadvantage for networks with a ring structure (including a logical ring structure). Second, since FDDI has no efficient way to dynamically change TTRT and the high-priority token holding time for each node, it cannot be used in a network where the real-time traffic load of each node changes due to the addition and/or deletion of real-time channels. In fact, most timed-token protocols (FDDI is one of them) suffer these two problems, and they are not adequate for supporting real-time communication which requires dynamic, independent addition/deletion of real-time channels.

Another typical example of medium access protocol which supports real-time traffic is FDDI II. In addition to being a timed-token protocol, FDDI II adds the circuit switching feature to facilitate the transmission of real-time traffic. Although it provides some degree of dynamic allocation of link bandwidth based on 16 wide band channels (WBC) [1], [2], [22], [24], [25], FDDI II still suffers (not as seriously as FDDI) the aforementioned two problems. Moreover, FDDI II underutilizes the network resources because of the use of WBCs which cannot be shared by other traffic even when they are idle.

The emerging IEEE 802.12 standard [5], [29] is another effort to provide guaranteed bandwidth and bounded access delay for time-critical applications by using existing 10Base-T networks. Since it does not reserve network resources

1. According to the smallest TTRT.

according to the need of each node dynamically, it still suffers the second problem mentioned above.

Before proceeding to describe our scheme, we need to define the *maximum token return time* (MTRT) and *real-time token holding time* (RTHT). The MTRT for a real-time channel is the maximum time interval between two consecutive token allocations to the channel. For example, the MTRT for FDDI is equal to $2 \times$ TTRT. During each of its token possessions, the RTHT of a real-time channel is the maximum time the channel is allowed to transmit its real-time packets.

In order to eliminate the above-mentioned two problems associated with most timed-token protocols, we need an adaptive scheme which allows different real-time channels to have different MTRTs. The ring is *not* a good topology if we want to allow different MTRTs for different channels or nodes; a multiaccess link / bus is a more natural candidate. Note that the ring structure can be used if we have to manipulate RTHT only. On the other hand, the need for dynamically changing the MTRT and RTHT for each node suggests the use of a centralized control unit on *each* multiaccess link which is responsible for token allocation and admission test upon receiving a request for establishing a real-time channel. Note that each centralized control unit controls only one link, not the entire network if it consists of multiple links. The centralized solution is much more efficient and cost-effective than its distributed counterpart for dynamic and independent addition / deletion of real-time channels, because a multiaccess link poses no communication bottleneck, no resource deadlock, no routing problem, and incurs low communication overhead in using a centralized control unit. Besides, the distributed counterpart causes potential incoherence and inefficiency because any change of MTRTs and RTHTs has to be negotiated / accepted by all nodes. Thus, as in the FieldBus [3], [4], [23], we will use a centralized control unit, called the *link control unit* (LCU), on a multiaccess link. The LCU is responsible for token allocation and resource reservation for real-time channels running through that link. The LCU will send high-priority (i.e., real-time) tokens and normal (i.e., non-real-time) tokens to all nodes on the link according to their needs and fairness. Only the node which currently holds the real-time (normal) token is eligible to transmit real-time (normal) packets. There is an expiration-time parameter associated with the token. The node must return the token to the LCU before or at the time the token expires.

There are two potential problems with the use of a centralized control unit: fault-tolerance and communication bottlenecking. However, for a multiaccess link, the fault-tolerance problem can be handled easily by duplicating the LCU. As all activities on a multiaccess bus can be seen by all nodes on the bus, we can use a passive LCU to maintain the network status just in case the primary LCU fails. Thanks to the multiaccessibility of the bus, this approach induces little additional communication cost in dealing with an LCU failure.

The communication bottlenecking problem may occur because all control messages have to be routed to the centralized control unit. In point-to-point networks, this could be a serious problem, because control messages may have to travel several hops to reach the centralized control unit, thus incurring high communication overhead for the transmission of control messages. However, in a multiaccess bus network, this communication overhead is small, because messages can be received by all nodes (including the LCU) on the multiaccess bus network, all in one hop. Therefore, the overhead of sending control messages (including tokens) is approximately proportional to the ratio of token-passing time to the lifetime of the token (RTHT). Note that token-passing time includes token transmission time, propagation time, and processing time. The propagation time is usually small on a multiaccess bus. For a bus of typical length 500 meters, the propagation time is about two microseconds. On a 100 Mbps bus, 25 bytes can be transmitted only in two microseconds. Although this ratio depends significantly on the underlying application, it is generally small for typical applications, like voice / video communication, since the traffic volume (per token allocation) of these applications is usually much larger than the token size, and the lifetime of each of these applications is usually much longer than the time needed to set up a channel.

To provide real-time performance guarantees, the LCU reserves link capacity and allocates a real-time token to each node on a *per-channel basis*.[2] The basic idea is to let each real-time channel have its own MTRT and RTHT based on its anticipated real-time traffic load. In our scheme, although the MTRT is defined to be the maximum time interval between two consecutive token allocations, it is approximately the same as the corresponding expected time, as the variance of actual token return time is negligibly small if $MTRT >> RTHT$ (holds in most cases when the network is expected to support many real-time channels).

Among the several models proposed to describe the traffic generated by a real-time channel, the *linear bounded model*—which was originally proposed by Cruz [8] and also adopted by other researchers [13], [14], [27]—is one of the most popular models. The traffic generated by a real-time channel is said to follow the linear bounded model if the number of packets generated in any interval T is bounded by a linear function of the interval length T, $G_{max}T + B_{max}$, where G_{max} is the maximum packet-generation rate of this channel and B_{max} is the maximum burst size. Using the user-specified delivery-delay bound D, one can establish a hard real-time channel in a point-to-point network [13], [14], [27]. Here we adopt an even simpler model which requires only two parameters for establishing hard real-time channels:

- 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.

This model is more general than the linear bounded model, because it can be applied to more cases, and the linear bounded model is only a special case of this model (by letting $M = G_{max}D + B_{max}$). As we shall see, we need additional information for handling statistical real-time channels.

We can derive MTRT and RTHT for each real-time channel by using these two parameters which are provided

2. In [7], we discussed an alternative in which the token is allocated on a *per-node basis*.

by the application layer upon arrival of a request for establishing a real-time channel. Since there is generally an upper bound for the size of a packet in a given protocol, we also assume the transmission time for a maximum-size packet, P_{max}, is available. Since the size of a message may be larger than P_{max}, we need to break this message into several packets for transmission over the network. From these parameters, MTRT and RTHT can be derived as:

$$MTRT := D \tag{3.3}$$

$$RTHT := M \times P_{max}. \tag{3.4}$$

Although there are many other possible solutions with smaller MTRTs, we do not choose a smaller MTRT here, because the smaller the MTRT, the more frequently the system has to allocate the token to this real-time channel and therefore, the less efficient.

When a node attempts to establish a real-time channel, it has to provide the LCU the requested MTRT and RTHT for this channel. The LCU will then try to reserve the link capacity for this channel by performing the admission test:

$$\sum_i \left(\frac{RTHT_i + overhead_i}{MTRT_i} \right) \le 1 \tag{3.5}$$

where the index i runs over all existing real-time channels and the current request. The main part of overhead is determined by the token-passing time which includes token transmission time, propagation time, and processing time. If the admission test can be satisfied after adding this new channel, the LCU will reserve the required link capacity, update the information about the existing real-time channels, and send a confirmation message to the requesting node. Otherwise, the LCU will send a rejection message to the requesting node. Using these parameters, the LCU can use the deadline-driven scheduling algorithm in [17] to allocate the token to each real-time channel with the corresponding guaranteed MTRT and RTHT. Basically, the LCU will issue a token to the requesting node approximately once every MTRT, thus allowing the node to transmit packets of this real-time channel for a period up to RTHT. If the transmission completes before RTHT expires, this node has to return the token to the LCU. The LCU will then issue the next token to this node (for this corresponding channel) no later than ($MTRT - RTHT$) time units after the current token is returned by this node. Thus, if all channels use up all their reserved time, the token will be issued to all nodes exactly once every (their corresponding) MTRT. The run-time scheduling strategy will be discussed later.

4 Statistical Real-Time Channels on Multiaccess Links

If the application can tolerate the loss of a certain percentage of real-time packets, using a hard real-time channel with the resources reserved for loss-free packet transmissions will severely underutilize the network. Thus, we would like to devise a real-time channel which reserves far less resources and meets looser requirements for such applications. In order to achieve this, we need additional information on packet generation, i.e., the distribution of packet interarrival times. Basically, we assume the packet-arrival distribution for a real-time channel (characterized over an interval of length D or MTRT) is given as in Fig. 1, and packet arrivals in an interval of length D are independent and identically distributed (*iid*). Then we can reduce the bandwidth that needs to be reserved for each real-time channel using the packet-arrival distribution and the *iid* assumption. Although this *iid* assumption may seem unreasonable in view of the highly-correlated nature of such real-time traffic as voice and video packets, our simulation results in Section 6 show that the *iid* assumption works well for the transmission of compressed digital motion-video frames. Similar simulation (based on different video sources) results supporting the *iid* assumption can also be found in [7].

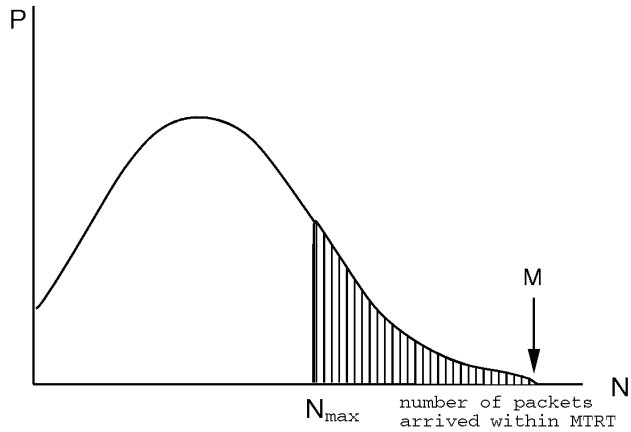


Fig. 1. An example distribution of packet arrivals within one MTRT.

One can lower the bandwidth needed for a real-time channel by increasing MTRT or decreasing RTHT. By increasing MTRT or choosing some MTRT > D, we may not be able to satisfy (3.1) or (3.2) without making more assumptions, as a packet could still be lost with this larger MTRT, even when the packet-arrival rate for this channel is far below average. For example, even though the packet-arrival rate is low, the token may not always arrive in time since MTRT > D. Another practical advantage of choosing $MTRT = D$ is the packet-arrival distribution of a real-time channel can be computed off-line, and only one distribution is needed for each application, even if it has several different performance requirements, e.g., different packet-loss rates. Therefore, once D is determined, the distribution of packet arrivals from a source can be characterized and derived from industrial standards and simulations/experiments. We will therefore consider decreasing RTHT and keeping MTRT the same as in (3.3). Fig. 1 shows an example distribution of packet arrivals for a real-time channel within one MTRT (as in (3.3)). The horizontal axis of this figure represents the number, N, of packet arrivals within one MTRT. If N is a continuous (discrete) random variable, the vertical axis represents the probability density (mass) function of N. The shaded area to the right of N_{max} represents the region where some packets of this real-time channel may be lost due to the insufficient bandwidth reserved, where N_{max} is the maximum number of packets this

real-time channel can transmit during each token allocation. We will derive N_{max} using Fig. 1 and the performance requirements such as (3.1) or (3.2). So, we can choose an RTHT large enough to satisfy this condition. That is, within RTHT, the system must be able to transmit at least N_{max} packets of this channel. N_{max} can then be used to determine the RTHT directly by using the relationship $RTHT = N_{max} \times P_{max}$. There are many possible ways to derive the desired RTHT (or N_{max}) when $MTRT = D$, depending on the type of performance requirements. We consider the following three typical performance requirements. Let G denote the *average* packet-arrival rate of a real-time channel.

Case 1: *P(delay of a packet ≤ delay bound D) ≥ a given number Z.*

This inequality is the same as (3.1) and should be satisfied over a sufficiently long period. Considering Fig. 1, if the number of packet arrivals falls in the unshaded region (to the left of N_{max}), all packets can be transmitted before their deadlines, since the node can transmit up to N_{max} packets during each token allocation. Similarly, in the shaded region (to the right of N_{max}), at most $(n - N_{max})$ packets will miss their deadlines. Therefore, (3.1) can be converted to (4.1) by using Fig. 1. Without loss of generality, we can consider N to be a discrete random variable, leading to:

$$Z \le 1 - \frac{\sum_{n=N_{max}}^{M} (n - N_{max}) P(n)}{\sum_{n=0}^{M} nP(n)} \tag{4.1}$$

$$= 1 - \frac{\sum_{n=N_{max}}^{M} (n - N_{max}) P(n)}{G \times MTRT} \tag{4.2}$$

If $\sum_{n=N_{max}}^{M} P(n)$ is small, we can approximate (4.2) with

$$Z \le \frac{\sum_{n=0}^{N_{max}} nP(n)}{G \times MTRT}. \tag{4.3}$$

Using either (4.2) or (4.3), we can find the smallest N_{max} that satisfies this performance requirement. With a minor modification, this case can be applied to many similar performance requirements, e.g., "messages" are considered instead of packets in the performance-requirement formula. For example, if the performance requirement is *P(delay of a message ≤ delay bound D) ≥ a given number Z*, then we can modify (4.2) to:

$$Z \le 1 - \frac{\sum_{n=N_{max}}^{M} f(n - N_{max}) P(n)}{G \times MTRT}, \tag{4.4}$$

where $f(n)$ is the number of messages which will miss deadlines given that n packets will miss their deadlines.

Case 2: *P(no packet loss during any time interval of length ≥ MTRT) ≥ Z.*

This is similar to (3.2). The unshaded area in Fig. 1 should be greater than, or equal to, Z. Therefore, the relation between Z and N_{max} is represented by:

$$Z \le \sum_{n=0}^{N_{max}} P(n). \tag{4.5}$$

Equation (4.5) can be used to find the smallest N_{max} that makes the unshaded area greater than, or equal to, Z.

Case 3: *P(delay of a packet ≤ delay bound D) ≥ a given Z* for all time intervals of length MTRT or larger. This is the strictest requirement among the three cases, because it has to be satisfied during any time interval of length ≥ MTRT. So, we must consider their worst case:

$$N_{max} \ge M \times Z. \tag{4.6}$$

We do not consider this performance requirement over an interval < MTRT, because, in a very short period during which only one packet is lost, it is impossible to satisfy the requirement unless $Z = 0$, i.e., a hard real-time channel.

In Section 6, we will use the first type of performance requirement and derive MTRT and RTHT with the proposed scheme.

5 Run-Time Scheduling

Since each real-time channel can be described by its MTRT (equivalent to "period") and RTHT (equivalent to "execution time"), we can use the deadline-driven scheduling algorithm in [6], [17] for allocating the token to individual real-time channels. Before the LCU grants a real-time channel request, a new schedule for allocating the token (e.g., "s0" in Fig. 2) must be computed in addition to performing an admission test. According to this schedule, the LCU then issues the real-time token to each real-time channel at least once every MTRT, and this token allows a node, upon receiving the token, to transmit packets of the corresponding channel for a period up to RTHT. Note that the LCU may choose to divide one RTHT into several smaller intervals (not necessarily of equal length) and allocate them to the channel within one MTRT as long as the total bandwidth allocated to the channel does not exceed RTHT / MTRT. However, in order to improve network utilization, the LCU tries to allocate a real-time token to each real-time channel exactly once every corresponding MTRT (if all channels use up all their reserved bandwidth) and use the remaining time for transmitting non-real-time traffic. When there are no scheduled activities, the LCU issues the non-real-time token whose expiration time is set to the beginning of the next scheduled activity (including the token passing time), and this token circulates among all nodes on the network.

Let's consider an example to illustrate these scheduling activities. In Fig. 2, suppose "s0" is the original schedule and the start time is 0. The symbol "ns" represents the time that can be used to transmit non-real-time traffic. If the token "A1" is returned at time t, the system will change the original schedule, s0, to a new schedule, s1, by moving the rest of the entire schedule ahead by $(RTHT - t)$ time units. A1 (A2) in Fig. 2 is the first (second) token allocated to channel A, which is determined by MTRT and RTHT.

The run-time packet scheduling performed by individual nodes is simple. The real-time token indicates which real-time channel should be allowed to transmit its real-time packets, and each node discards late packets and

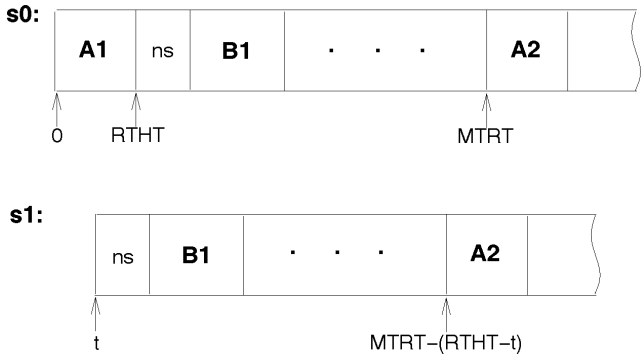


Fig. 2. A run-time scheduling example.

transmits the remaining packets in the queue according to their deadlines. When a node receives the non-real-time token, it transmits non-real-time packets in the queue until the token expires, or continues to circulate the non-real-time token if the node completes the transmission of non-real-time packets before the token expires. Note that, in order to achieve a higher non-real-time traffic throughput, a node does not give real-time packets higher priority when it receives the non-real-time token.

Since no complex scheduling algorithm is required at the node level, the proposed scheme can be implemented on a very simple node with low computing power, e.g., smart sensors in an automated factory. In the next section, we will show via simulation the effectiveness and efficiency of the proposed scheme.

The communication overhead for the proposed scheme is low, since the following two properties are generally true for typical (soft) real-time audio / video communication. First, the time needed to establish a channel is usually much shorter than the lifetime of an application. Hence, this overhead does not have significant effects on the overall communication cost, as this is only a one-time cost for each channel. Second, the total volume of successfully-delivered traffic (per token allocation) of these applications is usually much larger than the *equivalent network size of a token*. The equivalent network size of a token is defined to be the amount of traffic which can be transmitted on a bus network during one token-passing time on this particular bus network. Recall that one token-passing time includes token-transmission time, propagation time, and processing time. The ratio of the average successfully-delivered traffic volume per token allocation to the equivalent network size of two tokens (one to issue and the other to return) can be treated as the actual utilization of this real-time channel, since the token-passing time is the primary source of communication overhead in our scheme. Although this ratio depends greatly on the underlying application, it is usually small. In the simulation example presented in the next section, when the equivalent network size of each token is 500 bytes long, the ratio is shown to be less than one percent for the proposed scheme.

6 Simulation

In this section, we present a numerical example to demonstrate the effectiveness of the proposed scheme. We will use both the FDDI and the proposed scheme on a single link / bus to transmit compressed digital motion-video frames and compare their performances. This example shows that the proposed scheme reserves the bandwidth required for real-time traffic and also utilizes the network efficiently in the absence of real-time traffic.

6.1 Simulation Model

Environment for the proposed scheme and incoming traffic: In order to make a fair comparison with FDDI, we simulated a 100 Mbps 2,000-meter long multiaccess link / bus with 20 or 50 nodes. Thus, the upper bound of the propagation delay on this bus is about 10 microseconds. The video data are sampled from a sequence of CNN headline news (911 frames) stored on a laser disk [32]. The size of each frame, after JPEG compression [28], [32] is plotted in Fig. 3. The quality of video can be characterized by the rate of "successfully-delivered" frames, where a successfully-delivered frame is defined as one which is 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. If we use the transmission rate of 30 frames per second, three frames will be transmitted in each 100 ms. Assume the maximum packet size of the network is one Kbyte, and define the time to transmit a maximum-size packet as the *packet time*. Therefore, 100 ms is equal to 1,250 packet times (1 packet time = 80 microseconds), or D = 1,250 (in packet time). We will use a packet time as the basic time unit in the rest of this section. Using (3.3), we get $MTRT = D$ = 1,250 packet times. The performance requirement can then be specified as:

$$P(\textit{delay of a frame} \le 1{,}250) \ge \textit{a given } Z. \qquad (6.1)$$

Discussed below are three cases: Z = 99%, 95%, and 90%. (All the other cases can be handled similarly.)

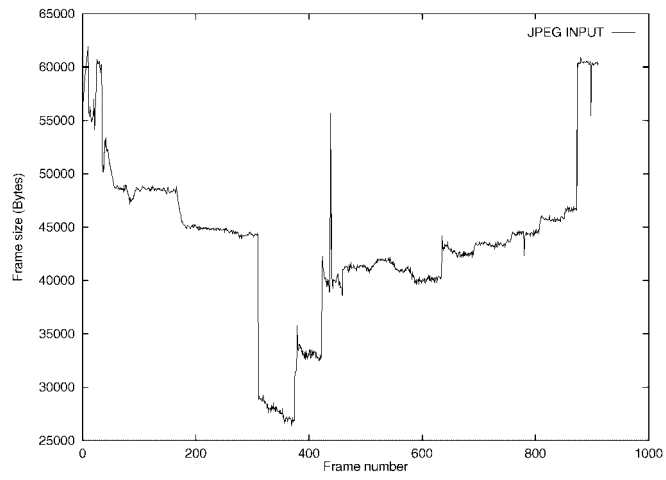


Fig. 3. An example of frame sizes (JPEG compression).

We need the distribution of traffic arrivals within one MTRT (= 1,250) to derive RTHT. By adding the sizes of three consecutive frames (because there are three frame arrivals in 100 ms), we can derive the distribution of traffic volume (in Kbytes) within D =1,250. Fig. 4 shows the distribution

of traffic volume (three frame arrivals) within one MTRT. For the scheme proposed in Section 4, we use this distribution to compute the bandwidths necessary for various frame-delivery rate requirements. Let T_{max} be the time needed to transmit one maximum-size message. We get T_{max} = 62 from Fig. 3. The performance requirement can then be expressed as:

$$Z \le 1 - \frac{\sum_{n=N_{max}}^{186} P(n)}{\sum_{n=80}^{186} 3P(n)} \tag{6.2}$$

$$= 1 - \sum_{n=N_{max}}^{186} \frac{1}{3} P(n), \tag{6.3}$$

since, if N_{max} is sufficiently large ($> M - T_{max} = 185 - 62 = 123$), each point in Fig. 4 will result in loss of at most one frame. This corresponds to $f(n) = 1$ in (4.4). Note that $\sum_{n=80}^{186} 3P(n)$ is the expected number of frames which will arrive within one MTRT (100 ms).

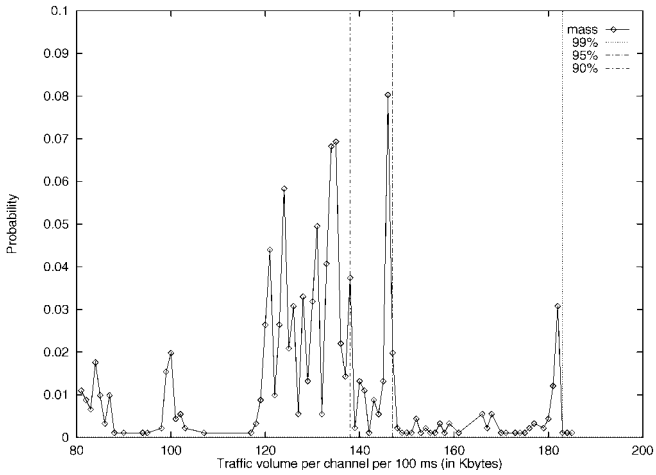


Fig. 4. An example distribution of traffic volume in an interval of length 100 ms.

Token passing overhead for the proposed scheme: The overhead to pass the token is the sum of token transmission time, propagation time, and processing time. If we assume the size of the token is 100 bytes, its transmission time is eight microseconds. The upper bound of the token propagation time is 10 microseconds, given the length of the bus is 2,000 meters. Thus, it is safe to assume the processing time to be less than 22 microseconds (2,200 instructions on a 100 MHz RISC CPU) and the upper bound of the total overhead to pass the token is 40 microseconds, i.e., 0.5 packet time. The total token-passing overhead per token allocation to a real-time channel involves passing the token twice (one to issue and the other to return), and, hence, we use one packet time for this overhead.

By adding one packet time as the token passing overhead, we get:

- Z = 99%: N_{max} = 183.
- Z = 95%: N_{max} = 147.
- Z = 90%: N_{max} = 138.

According to (3.5) the network is expected to support six 99 percent video channels, eight 95 percent video channels, or nine 90 percent video channels. Certain uniformly-distributed non-real-time traffic that requires from zero percent to 90 percent of the total link capacity is added to each node during the simulation. However, the source nodes of real-time channels are randomly chosen. (It is possible that one node may become the source node of all real-time channels.) The traffic data of real-time channels are taken from Fig. 3 and each channel has its own starting frame (also randomly chosen).

Environment for the FDDI network: In both 20-node and 50-node FDDI rings, we use the same input traffic as described in the proposed scheme's environment, except we assume the source nodes of established real-time channels are distributed evenly. For example, in a 20-node FDDI network, when we want to simulate four established channels and number the nodes from 1 to 20 in the clockwise order, the four source nodes of the four real-time channels will be randomly chosen among the following sets of nodes: (1, 6, 11, 16) or (2, 7, 12, 17) or (3, 8, 13, 18) or (4, 9, 14, 19) or (5, 10, 15, 20). Since the set of source nodes is randomly chosen, all nodes have an equal probability to be the source node of a real-time channel. Therefore, the high-priority token holding time is also distributed evenly among all nodes on the FDDI ring. For instance, in the 20-node environment, the high-priority token holding time of each node will be $TTRT / 20$.

The token passing overhead is ignored in the FDDI case, because we want to demonstrate the superiority of our scheme by showing that our scheme (with token passing overhead considered) is still significantly better (in terms of supporting real-time communication) than FDDI, even if we ignore FDDI's token passing overhead (which includes token-propagation time). So, the TTRT is set to 50 ms, i.e., a half of the maximum frame delivery delay required.

From one to 10 established channels, we use 10 sets of real-time channel source nodes sequentially for each case, i.e., 100 sets are used through the simulation for FDDI. During the course of simulation for each case (determined by the number of established channels), each of the 10 sets (of real-time channel source nodes) was randomly chosen as the example mentioned above. The traffic data of real-time channels will be taken from Fig. 3 and each channel has its own starting frame (also randomly chosen).

6.2 Simulation Results

The main goal of our simulation is to evaluate and compare the maximum and average *frame-miss rates* of both the FDDI and the proposed scheme. (The frame-miss rate is defined as the percentage of frames missing their deadlines.) We will also evaluate the improvement of network utilization by using statistical channels and examine the bandwidth available for the transmission of non-real-time traffic in the presence of real-time traffic. Our scheme is shown to always outperform FDDI and, at the same time, have the ability to provide performance guarantees.

Note that the one percent, five percent, and 10 percent lines in Figs. 5, 6, 7, and 8 are the lines which mark the corresponding frame miss rates in these figures for the purpose of easy comparison.

Figs. 5 and 6 show the maximum frame-miss rate of our scheme (90 percent, 95 percent, and 99 percent lines), as well as the maximum (FDDI line) and average (FDDI avg line) frame-miss rates of 20-node and 50-node FDDI networks under the condition that the non-real-time traffic load is equal to 50 percent of the total link capacity, i.e., 50Mbps. The frame-miss rate is defined based on a channel, i.e., the ratio of the number of frame misses for a channel 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 channel frame-miss rates. Each point in the figure represents 1,000 cycles of the sequence for each channel, i.e., about 911,000 frames or 8.4 hours at the rate of 30 frames per second for each channel. Note that each time the video clip repeated for a real-time channel, a starting frame was randomly chosen again so that each cycle of the simulation would appear different overall. As predicted by the analytic model, both figures show that six 99 percent, eight 95 percent, and nine 90 percent channels can be supported under the proposed scheme. For those points where the link does not have sufficient bandwidth for both real-time and non-real-time traffic, e.g., 7-10th at 99 percent channels, 9-10th at 95 percent channels, and 10th at 90 percent channel, we reserved the entire link capacity for real-time channels, and non-real-time traffic is transmitted only when the bandwidth reserved for real-time traffic is not used up completely. Since we reserved the link bandwidth using the worst-case values, this is very likely to happen.

For example, we reserved $\lceil 1{,}250/7 \rceil$ packet times for each channel when there are seven 99 percent real-time channels. (Note, however, that our scheme does not allow a seventh 99 percent real-time channel, because the system cannot guarantee the required performance.) When a seventh 99 percent channel is added, about two percent of real-time packets of this channel will miss deadlines. The FDDI can provide four to five real-time channels at a 50 percent non-real-time load. As we will see later, the FDDI's ability to support real-time communication is highly sensitive to the non-real-time traffic load. The average frame-miss rate for our scheme is very close to the maximum miss rate, so we only plot the maximum frame-miss rate in Figs. 5 and 6. By contrast, the average and maximum frame-miss rates of FDDI are significantly different when the FDDI ring 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 channels regardless of the number of nodes on the multiaccess link. This observation implies that the variance of frame-miss rate of our scheme is very small, whereas the FDDI suffers a large variation of frame-miss rate.

Figs. 7 and 8 show the maximum frame-miss rate of our scheme (90 percent, 95 percent, and 99 percent lines), as well as the maximum (FDDI line) and average (FDDI avg line) frame-miss rates of 20-node and 50-node FDDI networks under the condition that the non-real-time traffic load is equal to 90 percent of the total link capacity, i.e., 90Mbps. At a high non-real-time traffic load like this, the FDDI becomes nearly incapable of supporting real-time communication. It can only support/handle two channels (< 10 percent frame-miss rate) in the 20-node ring and one

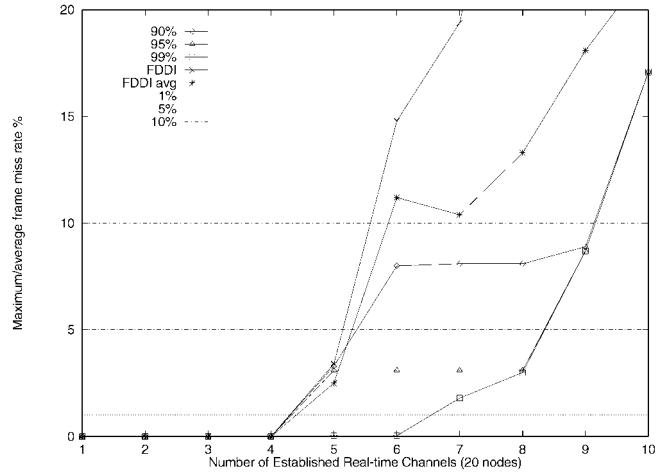


Fig. 5. Maximum frame-miss rate at a 50 percent non-real-time traffic load (20 nodes).

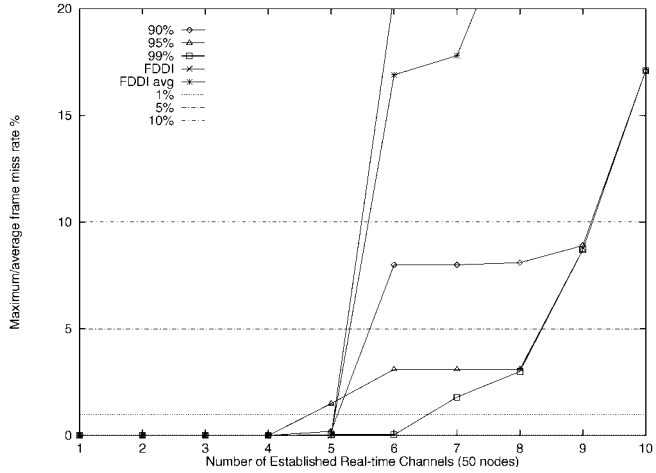


Fig. 6. Maximum frame-miss rate at a 50 percent non-real-time traffic load (50 nodes).

channel in the 50-node ring. By contrast, our scheme is insensitive to the non-real-time traffic load. The system can still provide six 99 percent, eight 95 percent, or nine 90 percent real-time channels at a high non-real-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. (Similar to the previous case, the FDDI's average frame-miss rate significantly differs from its maximum frame-miss rate.) As far as the ability to support real-time communication is concerned, our scheme is shown to be far better than the FDDI.

Our simulation also shows that network utilization can be improved significantly by using statistical channels. According to the simulation model, the network can support nine 90 percent channels, while only six channels can be established if no more than one percent of frames are allowed to miss their deadlines.

Fig. 9 shows the actual non-real-time traffic throughput (at a 90 percent non-real-time traffic load) and the capacity which is not reserved for real-time traffic under our scheme. As can be seen from this figure, the actual throughput is higher than the capacity which is not reserved for real-time

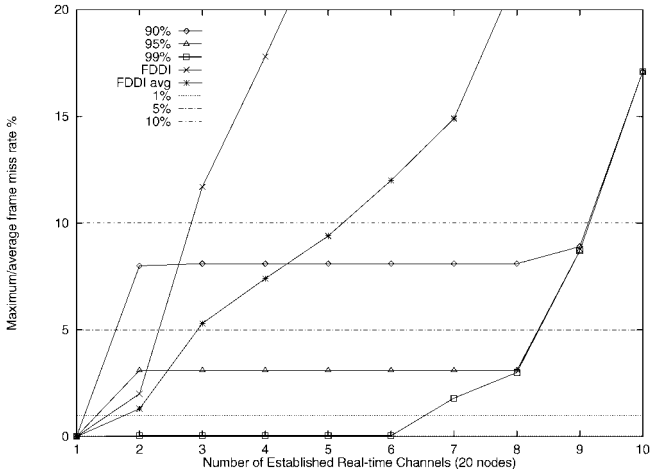


Fig. 7. Maximum frame-miss rate at a 90 percent non-real-time traffic load (20 nodes).

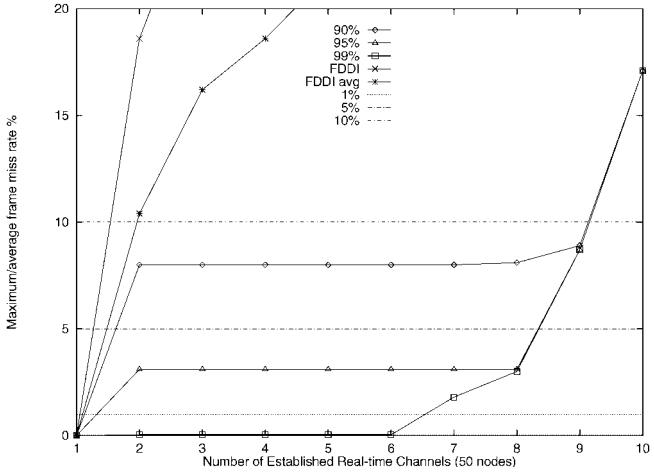


Fig. 8. Maximum frame-miss rate at a 90 percent non-real-time traffic load (50 nodes).

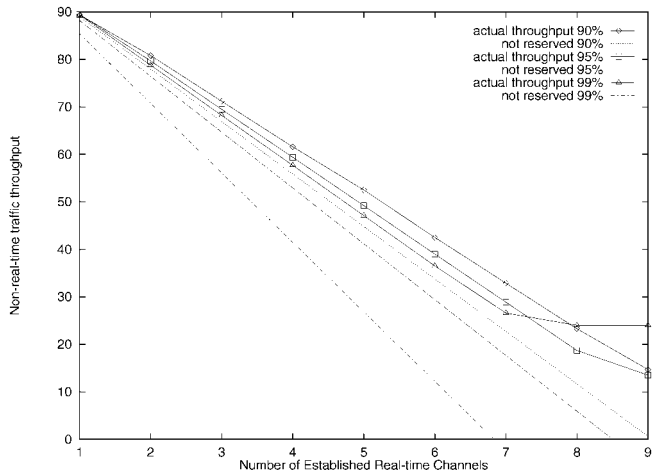


Fig. 9. Non-real-time traffic throughput as percent of total link capacity.

traffic, because our scheme requires the token to be returned to the LCU if a node has no packets of the corresponding real-time channel to transmit. Thus, the unused portion of the reserved capacity can be used to transmit non-real-time packets, thus improving network utilization. This improvement is significant, especially when the reserved capacity is much higher than the average need. For example, when there are six 99 percent channels, only 12.2 percent of the link capacity is left for non-real-time traffic if we use circuit switching. In our scheme, however, the actual throughput is 36.5 percent, a 24.3 percent improvement over the circuit-switching case (in terms of total link capacity).

We also calculated the token passing overhead of the proposed scheme under the assumption that the equivalent network size of a token is 500 bytes. Recall that the equivalent network size of a token is defined to be the amount of traffic which can be transmitted on the bus network during the time period of length one token passing time on this bus network. One token passing time includes token transmission time, propagation time, and processing time. Since two tokens are needed for each token allocation (one to issue and the other to return), the token passing overheads are approximately 0.84 percent (90 percent channels), 0.79 percent (95 percent channels), and 0.75 percent (99 percent channels), since the average successfully-delivered traffic volume per 100 ms are 119 Kbytes (90 percent channels), 127 Kbytes (95 percent channels), and 132 Kbytes (99 percent channels).

Since the average frame-miss rate and the maximum frame-miss rate of the proposed scheme are very close to each other, we can assume that whether a frame will miss its deadline or not (under the proposed scheme) follows a Bernoulli distribution and the number of missed/lost frames of each channel follows a Binomial distribution. Let Y be a random variable denoting the number of lost frames of a channel. For a particular point in Figs. 5, 6, 7, and 8, let n be the number of samples (frames) and p be the frame-miss rate of a channel in the corresponding environment. Therefore, n = 911,000 and p is the mean of Y/n. By applying the central limit theorem [10], [16],

$$\frac{Y - np}{\sqrt{n(Y/n)(1 - Y/n)}}$$

has a limiting distribution that is normal with mean 0 and variance 1. We can then find an approximate 99% confidence interval for the frame-miss rate, p, of our simulation. We can thus derive

$$P\left(-2.60 < \frac{Y - np}{\sqrt{n(Y/n)(1 - Y/n)}} < 2.60\right) = 0.99.$$

As a result, for a large n, if the experimentally-determined value of Y is y, then the interval

$$\left[\frac{y}{n} - 2.60\sqrt{\frac{(y/n)(1 - y/n)}{n}}, \frac{y}{n} + 2.60\sqrt{\frac{(y/n)(1 - y/n)}{n}}\right]$$

provides an approximate 99 percent confidence interval of p. In our simulation (n = 911,000), for any experimental value y

$$2.60\sqrt{\frac{(y/n)(1 - y/n)}{n}} < 0.003.$$

The 99 percent confidence interval is even smaller in cases where k (> 1) channels exist, because $n = k \times 911{,}000$ in such cases, i.e., the sample size increases.

Our scheme is shown to combine the advantages of circuit switching and packet switching. It provides performance guarantees for real-time channels and can also transmit non-real-time packets during the idle period reserved for real-time channels.

7 Conclusion

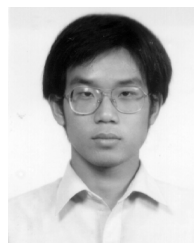
In this paper, we presented a new scheme for providing real-time performance guarantees given traffic-generation characteristics and performance requirements. In addition to its ability to provide performance guarantees, the proposed scheme can also improve network utilization by using statistical (as opposed to hard) real-time channels for the performance requirements specified in statistical terms. Since the traffic-generation model used in the proposed scheme is very general and the information needed for this scheme is easy to obtain, the scheme is easy to implement and is useful for many applications. Our simulation results have shown that this scheme is effective and efficient in supporting both real-time and non-real-time communication.

Acknowledgments

The work reported in this paper was supported in part by the U.S. National Science Foundation under Grant MIP-9203895 and by the U.S. Office of Naval Research under Grant N00014-94-1-0229. Any opinions, findings, and conclusions or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of the funding agencies.

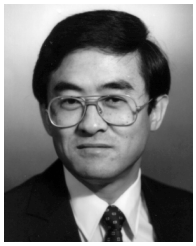
References

[1] *FDDI Token Ring Media Access Control*. ANSI Standard, 1988.
[2] *FDDI Hybrid Ring Control*. Draft Proposal to Am. Nat'l Standard, 1989.
[3] *Fieldbus Standard for Use in Industrial Control Systems—Part 3: Data Link Service Definitions (Draft Standard) (ISA-dS50.02 ISA/SP50-1993-359L)*. Instrument Soc. of Am., 1993.
[4] *Fieldbus Standard for Use in Industrial Control Systems—Part 4: Data Link Protocol Specification (Draft Standard) (ISA-dS50.02 ISA/SP50-1993-360L)*. Instrument Soc. of Am., 1993.
[5] A.R. Alberecht and P.A. Thaler, "Introduction to 100vg-anylan and the IEEE 802.12 Local Area Network Standard," *Hewlett-Packard J.*, vol. 46, no. 4, pp. 6-12, Aug. 1995.
[6] K.R. Baker, *Introduction to Sequencing and Scheduling*. New York: Wiley, 1974.
[7] C.-C. Chou and K.G. Shin, "Multiplexing Statistical Real-Time Channels on a Multiaccess Network," *Proc. 16th Int'l Conf. Distributed Computing Systems*, pp. 133-140, May 1996.
[8] R.L. Cruz, "A Calculus for Network Delay and a Note on Topologies of Interconnection Networks," PhD thesis, Univ. of 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. Selected Areas in Comm.*, vol. 8, pp. 368-379, Apr. 1990.
[10] R.V. Hogg and A.T. Craig, *Introduction to Mathematical Statistics*, fourth edition. Macmillan, 1978.
[11] V.C. Jones, *MAP/TOP Networking*. McGraw-Hill, 1987.
[12] D.D. Kandlur and K.G. Shin, "Design of a Communication Subsystem for HARTS," Technical Report CSE-TR-109-91, CSE Division, Dept. of Electrical Eng. and Computer Science, The Univ of Michigan, 1991.
[13] D.D. Kandlur, K.G. Shin, and D. Ferrari, "Real-Time Communication in Multi-Hop Networks," *Proc. 11th Int'l. Conf. Distributed Computing Systems*, pp. 300-307, 1991. (An improved version appeared in the Oct. 1994 issue of *IEEE Trans. Parallel and Distributed Systems*.)
[14] D.D. Kandlur, "Networking in Distributed Real-Time Systems," PhD thesis, Univ. of Michigan, 1991.
[15] J.F. Kurose, M. Schwartz, and Y. Yemini, "Multiple-Access Protocols and Time-Constrained Communication," *ACM Computing Surveys*, vol. 16, pp. 43-70, 1984.
[16] A. Leon-Garcia, *Probability and Random Processes for Electrical Engineering*, first edition. Addison-Wesley, 1989.
[17] 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.
[18] R. Nagarjan and J.F. Kurose, "On Defining, Computing and Guaranteeing Quality-of-Service in High-Speed Networks," *INFOCOM*, vol. 3, pp. 2,016-2,025, 1992.
[19] A.K.J. Parekh, "A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks," PhD thesis, Dept. of Electrical Eng. and Computer Science, The Massachusetts Inst. of Technology, Feb. 1992.
[20] V. Ramaswamy and W. Willinger, "Efficient Traffic Performance Strategies for Packet Multiplexers," *Computer Networks and ISDN Systems*, vol. 20, pp. 401-407, 1990.
[21] F.E. Ross, "FDDI—A Tutorial," *IEEE Comm. Magazine*, vol. 24, pp. 10-17, May 1986.
[22] F.E. Ross, "An Overview of FDDI: The Fiber Distributed Data Interface," *IEEE J. Selected Areas in Comm.*, vol. 7, no. 7, pp. 1,043-1,051, Sept. 1989.
[23] K.G. Shin and C.-C. Chou, "Design and Evaluation of Real-Time Communication for FieldBus Based Manufacturing Systems," *IEEE Trans. Robotics and Automation*, vol. 12, no. 3, pp. 357-367, June 1996.
[24] A.S. Tanenbaum, *Computer Networks*, second edition. Prentice Hall Int'l, 1989.
[25] M. Tangemann and K. Sauer, "Performance Analysis of the Timed Token Protocol of FDDI and FDDI-II," *IEEE J. Selected Areas in Comm.*, vol. 9, no. 2, pp. 271-278, Feb. 1991.
[26] A. Valenzano, C. Demartini, and L. Ciminiera, *MAP and TOP Communications, Standards and Applications*. Addison-Wesley, 1992.
[27] D.C. Verma, "Guaranteed Performance Communication in High Speed Networks," PhD thesis, Univ. of California at Berkeley, 1991.
[28] G. K. Wallace, "The JPEG Still Picture Compression Standard," *Comm. ACM*, vol. 34, no. 4, pp. 30-43, Apr. 1991.
[29] G. Watson, A. Alberecht, J. Curcio, D. Dove, S. Goody, J. Grinham, M. Spratt, and P. Thaler, "The Demand Priority Mac Protocol," *IEEE Network*, vol. 9, no. 1, pp. 28-34, Jan. 1995.
[30] O. Yaron and M. Sidi, "Calculating Performance Bounds in Communication Networks," *INFOCOM*, vol. 2, pp. 539-546, 1993.
[31] H. Zhang and D. Ferrari, "Rate-Controlled Static-Priority Queueing," Technical Report TR-92-003, Computer Science Division, Univ. of California at Berkeley, Jan. 1992.
[32] Q. Zheng, K.G. Shin, and E. Abram-Profeta, "Transmission of Compressed Digital Motion Video Over Computer Networks," *Proc. COMPCON Spring '93*, pp. 37-46, 1993.



Chih-Che Chou is also known as Chris C. Chou. He received the BSEE degree in electrical engineering from National Taiwan University, Taipei, Taiwan, in 1988, and both the MS and PhD degrees in computer science and engineering from the University of Michigan, Ann Arbor, Michigan, in 1992 and 1994, respectively.

He joined AT&T Bell Laboratories (now Lucent Technologies) in 1994 and he is currently a member of the technical staff. His research interests include real-time operating systems, real-time communications, multimedia applications, and communication networks. He is also applying the basic research results of real-time computing to commercial telecommunication systems. He is a member of the IEEE and the IEEE Computer Society.



Kang G. Shin received the BS degree in electronics engineering from Seoul National University, Seoul, Korea, in 1970, and both the MS and PhD degrees in electrical engineering from Cornell University, Ithaca, New York, in 1976 and 1978, respectively. He is professor and director of the Real-Time Computing Laboratory, Department of Electrical Engineering and Computer Science, The University of Michigan, Ann Arbor, Michigan.

He has authored/coauthored almost 400 technical papers (about 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 has coauthored (jointly with C.M. Krishna) a textbook, *Real-Time Systems*, McGraw-Hill, 1997. In 1987, he received the Outstanding IEEE Transactions on Automatic Control Paper Award for a paper on robot trajectory planning. In 1989, he also received the Research Excellence Award from The University of Michigan. In 1985, he founded the Real-Time Computing Laboratory, where he and his colleagues are investigating various issues related to real-time and fault-tolerant computing.

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. (The latter is being pursued as a key thrust area of the newly-established National Science Foundation Engineering Research Center on Reconfigurable Machining Systems.)

From 1978 to 1982, he was on the faculty of Rensselaer Polytechnic Institute, Troy, New York. He has held visiting positions at the U.S. Airforce Flight Dynamics Laboratory, AT&T Bell Laboratories, Computer Science Division within the Department of Electrical Engineering and Computer Science at the University of California at Berkeley, and International Computer Science Institute, Berkeley, California, IBM T.J. Watson Research Center, and Software Engineering Institute at Carnegie Mellon University. He also chaired the Computer Science and Engineering Division, EECS Department, The University of Michigan, for three years beginning in January 1991.

He is an IEEE fellow, 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 August 1987 special issue of *IEEE Transactions on Computers* on real-time systems, a program cochair 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 1991-93, was a distinguished visitor of the Computer Society of the IEEE, an editor of *IEEE Transactions on Parallel and Distributed Systems*, and an area editor of the *International Journal of Time-Critical Computing Systems*.