Multiplexing Statistical Real-Time Channels in a Multi-Access Network

Chih-Che Chou

Bell Laboratories
Lucent Technologies, Inc.
Westminster, CO 80234
ccchou@drmail.dr.att.com
ccchou@drmail.dr.lucent.com

Kang G. Shin

Real-Time Computing Laboratory
Dept. of Elect. Eng. and Comp. Science
The University of Michigan
Ann Arbor, MI 48109-2122
kgshin@eecs.umich.edu

Abstract—*Given the client's traffic-generation characteristics and performance requirements, we propose a real-time communication scheme that provides delivery-delay guarantees in a multiaccess local-area network (LAN). This scheme (i) reduces the link capacity that needs to be reserved to an average level as compared to the worst-case level required for deterministic performance guarantees, and (ii) preserves the ability of independent addition and deletion of real-time channels.*

1 Introduction

Several researchers investigated the problem of supporting real-time communication with performance guarantees for given worst-case traffic characteristics in wide-area point-to-point networks [2, 3]. Among these, the concept of "real-time channel" proposed by Ferrari and Verma [3] is the most notable in explicitly addressing the problem of providing delivery-deadline guarantees 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.

In [2], we proposed a scheme for real-time communication on multiaccess networks which can provide performance guarantees according to the user-specified traffic-generation characteristics and performance requirements. However, in order to let the system add/delete real-time channels independently, we used a *channel-based* design in [2] without considering the problem of *multiplexing* real-time channels. As a result, the scheme proposed in [2] under-utilizes the network. In this paper, we significantly improve the scheme in [2] by multiplexing real-time channels originating from the same node in order to achieve higher network utilization without compromising the capability of independent addition/deletion and the performance guarantees of real-time communication.

The paper is organized as follows. In Section 2, we present an overview of the scheme in [2] and the problem statement. Our proposed solution to this problem is detailed in Section 3. In Section 4 we demonstrate via simulation the correctness and effectiveness of the proposed scheme. The paper concludes with Section 5.

2 The Channel-Based Scheme

2.1 Overview of the Channel-Based Scheme

The prime difference of a channel/connection on a multiaccess network from that of a point-to-point network lies in the relationship between nodes and links. In a point-to-point network, a node has complete control of its transmission links and complete knowledge of channels running through the node. Since it is easy to schedule packets of these channels by simply examining them in the queues, one for each channel, one can easily time-multiplex several real-time channels. By contrast, in a multiaccess network, it is difficult to determine which node has the right to transmit a packet at a particular instant. Due to this difference, the schemes proposed in [2, 3] for point-to-point networks are not applicable to multiaccess networks.

In [2], we proposed a scheme which can provide performance guarantees for real-time communication on multiaccess networks under the assumption that the traffic-generation characteristics and performance requirements are known in advance. The performance guarantee is usually given in a statistical form:

$$P(\textit{delay of a packet} \leq D) \geq \textit{a given } Z. \quad (2.1)$$

We also adopted a simple traffic model for real-time packet arrivals, which requires only two parameters for establishing a hard real-time channel: D (seconds) = the user-specified delivery-delay bound for each message, M (packets) = the maximum number of packets that can be generated in an interval of length D.

The work reported in this paper was supported in part by the National Science Foundation under Grant MIP-9203895 and by the 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 view of the funding agencies.

1063-6927/96 $5.00 © 1996 IEEE
Proceedings of the 16th ICDCS

We use a multiaccess *link* (not a ring) as the transmission medium, and a *link control unit* (LCU) on each link for token allocation and resource reservation. The LCU allocates high-priority (i.e., real-time) tokens and normal (i.e., non real-time) tokens to nodes on that link according to their need and fairness. Only the node which currently possesses a real-time (normal) token is eligible to transmit real-time (normal) packets. There is an expiration-time parameter associated with each token, and the node must return the token to the LCU before or at the time the token expires. In order 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. More precisely, the LCU treats each real-time channel as an independent entity and allocates the real-time token to a real-time channel, not to a node. Although the token is eventually issued to the node from which the corresponding channel originates, the node cannot use the real-time token of some channel (say, A) for the transmission of other channel's (say, B's) packets, even when channel A is idle. Thus, the addition/deletion of real-time channels will not compromise the performance guarantees of other real-time channels.

When establishing *statistical* real-time channels, in addition to D, M and the performance requirement of requesting real-time channels, the distribution of packet arrivals (see Fig. 1) is needed for the system to efficiently reserve sufficient link bandwidth. We also formally de-

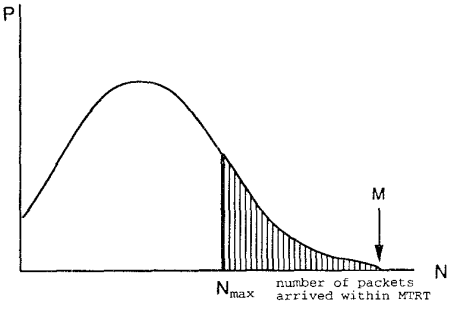

Figure 1: An example distribution of packet arrivals within one MTRT

fine the *Maximum Token Return Time* (MTRT) and the *Real-time Token Holding Time* (RTHT) for each real-time channel [2]. MTRT for a real-time channel is the maximum time between two consecutive token allocations to the same channel. During each token possession, a channel's RTHT is the maximum time the channel is allowed to use for transmitting its real-time packets. If a token of a channel C is returned at time $t = 0$, then the next time channel C will receive the token is $t \leq MTRT - RTHT$. We will in the next section introduce another parameter, $MTRT_N$ for node (as opposed to channel) N, which denotes the maximum time interval between two consecutive token allocations to node N. We would like to stress, to avoid any confusion, that MTRT^C (or MTRT_N) is a parameter of a specific channel C (or node N) and is likely to vary from channel to channel (from node to node). (Since we will discuss only one channel for most of the time, we will drop the superscripts of MTRT and RTHT that designate channels.) From this definition and the other parameters, MTRT can be derived as $MTRT = D$. From a request for establishing a hard real-time channel, we can derive the RTHT as: $RTHT = M \times P_{max}$, where P_{max}, called the *packet time*, is the time needed to transmit a maximum-size packet. If the user-specified performance requirement is not absolute, i.e., a *statistical real-time channel* is requested, MTRT can still be determined by $MTRT = D$. However, RTHT depends on the user's performance requirement. Let N_{max} be the maximum number of packets this real-time channel can transmit during a single token visit. If the requirement is specified as in Eq. (2.1), we can determine N_{max} (in Fig. 1) by: $Z \leq 1 - \frac{\sum_{n=N_{max}}^{M}(n-N_{max})P(n)}{\sum_{n=0}^{M} nP(n)} = 1 - \frac{\sum_{n=N_{max}}^{M}(n-N_{max})P(n)}{G \times MTRT}$, where G is the average packet-generation rate of the real-time channel. Using this equation, we can find the smallest N_{max} satisfying this performance requirement. Then, N_{max} can be used to determine RTHT directly from $RTHT = N_{max} \times P_{max}$. The MTRT and RTHT derived above will be used in the channel establishment process for link capacity reservation and in the run-time scheduling process for token allocation. For other forms of performance requirements, the corresponding N_{max} can be derived similarly.

When a node attempts to establish a real-time channel, it has to provide the LCU with the requested MTRT and RTHT. The LCU will then try to reserve the link capacity for this channel by performing the following admission test: $\sum_i \left(\frac{RTHT^i + overhead^i}{MTRT^i}\right) \leq 1$ where the index i runs over all existing real-time channels and the current request. The main part of overhead is the token passing 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. The deadline-driven scheduling algorithm in [2] can be used by the LCU to allocate the token to each real-time channel with the guaranteed MTRT and RTHT. The LCU will issue the requesting node a token once every MTRT, thus allowing the node to transmit packets of this real-time channel for a period up to RTHT.

2.2 Problem Statement

The method described above is called the "channel-based" scheme in [2], because the link bandwidth is reserved on a per-channel basis. Since more than one real-time channel may originate from a node, multiplexing these channels on a per-node basis may achieve higher

network utilization and induce less overheads. That is, instead of reserving link bandwidth for each individual real-time channel, the system assigns a real-time token to a node at least once in a certain period of time for all real-time channels originating from that node. We may be able to multiplex several real-time channels (especially statistical real-time channels) with much less bandwidth than the sum of their individual bandwidths. However, the ability of independent addition and deletion of real-time channels must not be compromised while multiplexing real-time channels. We will therefore focus on the problem of making correct and efficient link-capacity reservation and run-time scheduling as well as preserving the ability of independent addition and deletion of channels. In the rest of this paper, we will address the problem of multiplexing statistical real-time channels.

3 Node-Based Scheme

In this section, we will focus on the problem of multiplexing *statistical* real-time channels. Note that we cannot multiplex two *hard* real-time channels without compromising their performance guarantees. However, we allow a hard real-time channel to be multiplexed with several statistical channels. In this case, this hard real-time channel will be given the highest priority among all the multiplexed channels. In the rest of this section, the term, "real-time channel" or simply "channel", (unless stated otherwise) will mean a statistical real-time channel.

By assuming that the distribution of incoming traffic for all real-time channels is available at the time of receiving a channel-establishment request, we propose an incremental scheme to add or delete a real-time channel and adjust the reserved link bandwidth when a new real-time channel is accepted or an existing channel is torn down. Real-time channels will be time-multiplexed on a per-node basis. Note that the token-holding time here is referred to the real-time token-holding time for transmitting the packets of real-time channels. The proposed node-based scheme will function as follows. Each node N will compute $MTRT_N$ and $RTHT_N$ for the combined traffic of all real-time channels which originate from node N. As will be seen later, $MTRT_N$ is the smallest MTRT of real-time channels which originate from node N. The LCU then allocates the real-time token to N once every $MTRT_N$, which allows the node to transmit real-time packets up to $RTHT_N$ units of time. As in the channel-based token allocation [2], the node should return the token to the LCU at or before the token holding time expires. If the token for node N is returned at time $t = 0$, then the next time node N will receive the token is $t \leq MTRT_N - RTHT_N$.

Before proceeding to the description of the channel-multiplexing scheme, we first introduce the distribution of *reserved-but-unused* (RBU) link bandwidth for a node, which is defined as the link bandwidth reserved by the node for real-time channels, but not actually used at run-time. This distribution will be used in the derivation of link bandwidth to be reserved when a new channel is to be added. The probability of no RBU link bandwidth is usually non-zero (i.e., $P(\text{RBU link bandwidth} = 0) > 0$), because the assigned capacity is not usually used up. Basically, we want the new requesting channel to use as much RBU bandwidth as possible before requesting additional link bandwidth from the LCU.

3.1 Channel-Establishment Phase

3.1.1 Establishing a node's the first RT channel

Since there is no real-time channel originating from the node, this request can be handled just as the channel-based scheme [2]. If this request is admissible, the distribution of the RBU link bandwidth must be updated. Before deriving the RBU link bandwidth, we present two intuitive results in theorem form without proofs.

Theorem 1: Suppose $MTRT > 1$ (in packet time). If a real-time performance requirement can be satisfied by a real-time channel with the maximum token return time, $MTRT$, and a real-time token holding time, $RTHT$, then this performance requirement can also be satisfied by a pseudo real-time channel with link bandwidth $RTHT/MTRT$ per packet time. □

Theorem 2: Suppose $MTRT > 1$ (in packet time). If a real-time performance requirement can be satisfied by a pseudo real-time channel with link bandwidth $RTHT/MTRT$ in every packet time, and the delay bound of this channel's packets is at least $MTRT$, then this performance requirement can also be satisfied by any real-time channel with a maximum token return time $MTRT' \leq MTRT$ and a real-time token holding time $RTHT' \geq RTHT \times MTRT'/MTRT$. □

Different real-time channels may have different MTRTs, but we have to use a common MTRT for the distribution of RBU link bandwidth. The smallest possible MTRT is one packet time, and by Theorem 1, we can convert any real-time channel to a pseudo channel with link bandwidth $RTHT/MTRT$ in every packet time. By Theorem 2, we can also convert a pseudo real-time channel back to a real-time channel if the channel's MTRT is given. So, we choose one packet time as the basic time unit for the distribution of RBU bandwidth.

Let X be the random variable representing the number of average packet arrivals for a new real-time channel within one packet time and R be the random variable representing the RBU link bandwidth in a packet time. Let N represent the number of packet arrivals within one MTRT, then $X = \frac{N}{MTRT}$, $R = \frac{RTHT}{MTRT} - X$, and, $p_R(r) = P(\frac{RTHT}{MTRT} - X = r) = P(X = \frac{RTHT}{MTRT} - r)$, for $0 < r < \frac{RTHT}{MTRT}$, $p_R(0) = P(X \geq \frac{RTHT}{MTRT})$,

where $p_R(r) = P(R = r)$ is the probability mass function of the RBU link bandwidth in a packet time. We will use the notation $MTRT_n$ and $RTHT_n$ to denote as the maximum token return time and the real-time token holding time, respectively, for node n, while using $MTRT$ and $RTHT$ for a particular channel. If channels need to be distinguished, we will use $MTRT^C$ and $RTHT^C$ for channel C.

3.1.2 General channel establishment procedure

A real-time channel-establishment request has to be handled differently than the previous case if it is not the first request. Basically, the procedure has three steps.

Step 1: Compute the distribution of $Y = R - X$ first, where R is the current RBU link capacity in a packet time and X the average number of packet arrivals for this new channel within one packet time. Fig. 2 shows an example distribution of Y. Because R and X are

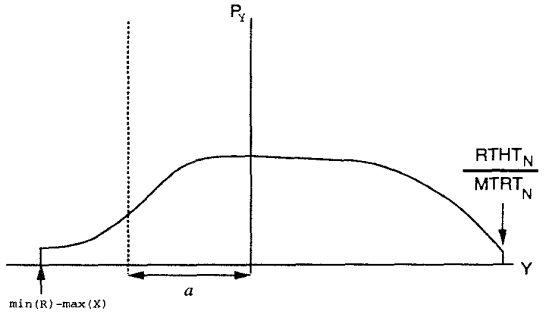

Figure 2: An example distribution of $Y = R - X$

independent, $p_Y(y) = P(R - X = y) = \sum_x P(R = y + x | X = x) \times P(X = x) = \sum_x P(R = y + x) \times P(X = x)$, where $min(R) - max(X) \leq Y \leq \frac{RTHT_N}{MTRT_N}$, and $p_Y(y) = P(Y = y)$ is the probability mass function of Y.

Step 2: The goal of this step is to determine the additional link capacity needed for this new channel in a packet time. As in [2], we use three typical performance requirements to illustrate our approach.

P1: *$P(\text{delay of a packet} \leq D) \geq a$ given Z*: This requirement should be satisfied in an average sense, i.e., over a sufficiently long time period. Using Fig. 2 this performance requirement can be converted to: $Z \leq 1 - \frac{\sum_{n=min(R)-max(X)}^{-a} (-n-a) p_Y(n)}{G \times MTRT}$. We can then find the smallest a for a given Z and the distribution, as in Fig. 2. Since it is desirable that the reserved link bandwidth can be used up in the worst case, i.e., $min(R) = 0$, we get $Z \leq 1 - \frac{\sum_{n=-max(X)}^{-a} (-n-a) p_Y(n)}{G \times MTRT}$. If the requested channel is a hard real-time channel (i.e., $Z = 1$), we can derive $a = max(X)$. With a minor modification, this case can be applied to many similar performance requirements.

P2: *$P(\text{no packet loss in } t) \geq Z$* for any t of length MTRT or greater. The area left to the dotted line in Fig. 2 should be $\leq 1 - Z$. Thus, the relation between Z and a is represented as: $Z \leq 1 - P(Y + a \leq 0)$, and then $Z \leq 1 - \sum_{n=min(R)-max(X)}^{-a} p_Y(n)$. This equation can be used to find the smallest a that makes the area left to the dotted line $\leq 1 - Z$. If $min(R) = 0$ (normally), we can derive $Z \leq 1 - \sum_{n=-max(X)}^{-a} p_Y(n)$, and use it to find the smallest a. Similarly, if a hard real-time channel is requested (i.e., $Z = 1$), $a = max(X)$ can also be derived.

P3: *$P(\text{delay of a packet} \leq \text{bound } D) \geq a$ given Z* for all time intervals of length $\geq$ MTRT. This is a more strict requirement than **P1** and **P2**, because it has to be satisfied during any time interval of length $\geq$ MTRT. So, we must consider its worst case: $Z \leq 1 - \frac{|min(R)-max(X)+a|}{max(X)}$ or $Z \leq 1 - \frac{|-max(X)+a|}{max(X)}$, if $min(R) = 0$. As in **P1** and **P2**, we can find the smallest a satisfying these two equations, and $a = max(X)$ can be derived if the requested channel is a hard real-time channel.

Step 3: The a derived from **P1–P3** represents the additional link capacity in one packet time needed to accommodate this new channel without compromising the performance guarantees of existing channels. Obviously, the case of $a \leq 0$ represents the current RBU link capacity of node N is sufficient to provide the required performance of the new channel, and thus, node N can accept this new channel without asking the LCU for more bandwidth, if $MTRT^{new\ channel} \geq MTRT_N$. After accepting this new channel, the RBU link capacity of node N has to be updated as: $p_{R_{new}}(0) = P(Y \leq 0)$ and $p_{R_{new}}(n) = p_Y(n)$, $\forall n > 0$. If the MTRT needed for this new channel is smaller than the current $MTRT_N$, node N still has to ask the LCU for a smaller $MTRT_N$ with a message containing the new $RTHT_N$ and $MTRT_N$ for this node: $MTRT_{N,new} = MTRT^{new\ channel}$ and $RTHT_{N,new} = RTHT_{N,old} \times \frac{MTRT_{N,new}}{MTRT_{N,old}}$. The probability that the LCU will grant this request (for the new $MTRT_N$) is high, because the request does not increase the bandwidth that needs to be reserved. However, as $MTRT_N$ decreases, the corresponding token passing overhead increases, and thus, this request may not always be accepted.

If the current RBU link capacity of node N is not enough to guarantee the required performance of this new channel (i.e., the a derived in Step 2 is greater than zero), then we first determine the new $MTRT_N$. If the MTRT of the new channel is smaller than the current $MTRT_N$, then let $MTRT_{N,new} = MTRT^{new\ channel}$; otherwise, $MTRT_N$ remains unchanged.

After determining $MTRT_N$, we can compute the new

$RTHT_N$ as: $RTHT_{N,new} = RTHT_{N,old} \times \frac{MTRT_{N,new}}{MTRT_{N,old}} + a \times MTRT_{N,new} \times P_{max}$. The second term in this equation represents the additional link capacity needed in one $MTRT_{N,new}$. Note that if $a = max(X)$ (requesting a hard real-time channel to be established), $max(X) \times MTRT_{N,new} \times P_{max}$ is equal to the link capacity necessary to establish a hard real-time channel in the channel-based scheme, because $max(X) \times MTRT_{N,new} = max(N) \times \frac{MTRT_{N,new}}{MTRT^{new\ channel}}$. Node N sends the LCU a channel-establishment request message which contains the new $MTRT_N$ and $RTHT_N$. The LCU will try to reserve the requested bandwidth and reply with either accept or reject. If the reply from the LCU is reject, the new channel request cannot be accepted, so $MTRT_N$, $RTHT_N$ and the distribution of RBU link capacity remain unchanged. On the other hand, if the reply is accept, the RBU link capacity of node N has to be updated as: $p_{R_{new}}(0) = P(Y + a \leq 0)$ and $p_{R_{new}}(n) = p_Y(n - a)$, $\forall n > 0$.

3.2 Channel-Deletion Phase

Deletion of an existing real-time channel must also be done independently of other existing channels. After closing an existing real-time channel, we must update the RBU link capacity using the performance requirement and the distribution of the traffic arrivals of the deleted channel. Basically, the capacity reserved for the deleted channel is added back to the current RBU link capacity of the node from which the deleted channel originates. If there is an excessive RBU link capacity, we may try to decrease the link capacity to be reserved for the node and/or return the excessive capacity back to the LCU.

An intuitive way to achieve the above goal is to develop a procedure which can "add" the bandwidth used by the deleted channel back to the current RBU link capacity of this node without compromising the performance guarantees of other real-time channels. Let X and R be defined the same as in Section 3.1.1, except that X now represents the packet arrival rate of the deleted channel. Therefore, the new RBU link capacity can be expressed as $R_{new} = R_{old} + X$. However, since X and R_{old} are not independent, we cannot simply compute the distribution of R_{new} from the distributions of R_{old} and X.

In order to derive the distribution of R_{new}, we need to re-compute MTRT and RTHT for the remaining channels as if we were re-establishing them according to the original order of their arrival. If there is an excessive RBU link capacity, the node returns it to the LCU.

This method will reserve only the necessary link capacity, as it is basically the re-computation of link bandwidths for adding channels. If the number of remaining real-time channels is very large, its computation may become time-consuming. Fortunately, the efficiency of handling a channel-closing request is not as important as that of handling a new channel-establishment request, because the system can always close the channel first and compute the RBU link bandwidth later. Due to space limitation, we will omit the introduction of an alternative algorithm which uses the Fourier Transform to handle channel deletion.

3.3 Run-time Scheduling

As in the channel-based scheme [2], the deadline-driven (normal Earliest Due-Date first) scheduling algorithm can be used by the LCU to schedule the token for real-time channels. When the total utilization (including overheads) is less than 1, the deadline-driven scheduling algorithm can guarantee all deadlines as long as the input traffic follows the pre-specified traffic-generation characteristics.

For individual nodes, the scheduling algorithm has to consider the existence of message (frame) inter-dependency. If all messages (frames) are independent of each other in a data stream, i.e., the performance requirement of a real-time channel can be characterized directly by the delivery rate of messages (frames) of the channel, then the deadline-driven scheduling algorithm can also be used as the primary scheduling discipline by each individual node. However, since only N_{max} packet times are reserved in one MTRT for a channel, the system has to give lower priority to the channel which had transmitted at least N_{max} packets during the previous MTRT. This strategy can prevent the burstiness of some channels from degrading other channels' performance. Special customized scheduling policies can also be easily added at the channel-level, i.e., a node can give a certain channel higher priority according to the requirements of the application at hand.

If there exists inter-dependency among the messages (frames) of a data stream, i.e., the "effective" delivery of some frame depends on the delivery of some other frame(s), the normal EDD scheduling algorithm alone is not sufficient to provide adequate performance guarantees. In Section 4 , we introduce briefly a multiple-due-date (MDD) scheduling policy [1] to solve the frame-dependency problem.

4 Simulations

There are two goals of our simulation. First, we want to demonstrate that the channel multiplexing can reduce the link capacity that needs to be reserved and thus, improve the network utilization. The second goal is to show that the channel-multiplexing scheme can preserve the performance guarantees.

4.1 Simulation Models

We use a 100 Mbps multiaccess link/bus as the physical medium for transmitting digital video frames. The video data used are obtained from a 5376-frame (about 3 minutes at the rate of 30 frames per second) sequence

of the movie "Star Wars" [1]. The maximum one-way transmission delay of a frame is assumed to be less than 100 ms in order to achieve the quality of live video. Note that the transmission delay (100 ms) includes only the queuing delay and the actual transmission time, i.e., encoding, decoding and other processing times are not included. At the transmission rate of 30 frames per second, 3 frames will be transmitted in each 100 ms. We also allow random jitters which are assumed to be uniformly distributed between $[-416, 417]$ (packet times). Note that 416.7 is the frame inter-arrival time. The maximum packet size of the network is assumed to be 1 Kbytes, so 100 ms is equal to 1250 packet times or $D = 1250$. The packet time will be used as the basic time unit in the rest of this section. The performance requirement for these video frames is assumed to be P(*a frame can be reconstructed by its deadline*) $\geq$ *a given* Z. In our simulation, the MPEG video compression algorithm generates one I-frame and seven P-frames for every eight frames. Since the I-frames are coded independently, they can be used to reconstruct a picture independently. The P-frames are coded with reference to the previous I-frame, so the previous I-frame is necessary for the reconstruction of P-frames. Thus, the delivery rate of frames does not directly imply the same frame reconstruction rate.

In order to translate the performance requirement correctly, we use a multiple-due-date (MDD) scheduling policy [1] to ensure that a P-frame will not be transmitted until the previous I-frame is transmitted. Basically, we associate a frame with two due dates: the first (*scheduling-due-date*) is used in the EDD algorithm for normal EDD scheduling and the second is called the *drop-due-date* which indicates the time a frame is no longer useful. If a frame misses its scheduling-due-date, the system will reset the scheduling-due-date to some time later when this frame will be used again (before its drop-due-date). Thus, if an I-frame arrives at time t, its scheduling-due-date will be set to $t + 1250$ (in packet time) and its drop-due-date will be set to $t + 1250 \times 8/3$. Every time an I-frame misses its scheduling-due-date, the scheduling-due-date will be extended by $1250 \times 1/3$ packet times until its drop-due-date is reached. Similarly, if a P-frame arrives at time t, both of its due-dates will be set to $t+1250$, since a P-frame has no value after its scheduling-due-date. Since the goal of the simulation is to demonstrate the effectiveness of the proposed channel multiplexing scheme, we will not discuss the MDD strategy any further. See [1] for details.

Since the maximum one-way transmission delay is 100 ms and the performance requirement is given in statistical form, $MTRT = 1250$. We need the distribution of traffic arrivals within one MTRT to derive RTHT, the link capacity to be reserved. By adding three consecutive frame sizes, we can derive the distribution of traffic arrivals (in Kbytes) within one MTRT. Fig. 3 shows the distribution of traffic arrivals within one MPEG channel. The performance requirement can then be expressed as

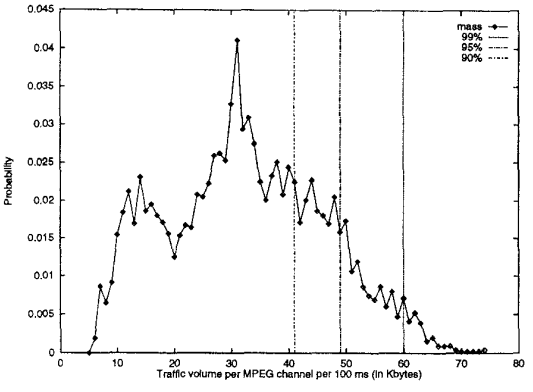

Figure 3: Distribution of MPEG frame arrivals

$Z \leq 1 - \frac{\sum_{n=N_{max}}^{74} P(n)}{\sum_{n=6}^{74} 3P(n)} = 1 - \sum_{n=N_{max}}^{74} \frac{1}{3} P(n)$, since if N_{max} is sufficiently large, each point in Fig. 3 will result in loss of at most one frame. By adding 0.5 packet time per frame as the operation overhead, we get: Z=99%: N_{max}=60, Z=95%: N_{max}=49, and Z=90%: N_{max}=41.

However, since I-frames will not be discarded until their drop-due-dates, and we always try to send I-frames before their associated P-frames, the amount of data that needs to be delivered for the reconstruction of the P-frames associated with the missed I-frames will be affected. Let P be the probability that an I-frame will miss its scheduling due-date. The amount of data "expected" to be scheduled for the transmission and reconstruction of the next P-frame before the next scheduling-due-date (i.e., 33.3 ms later after the I-frame's scheduling-due-date) is the size of the P-frame plus $P \times S_I$, where S_I is the size of an I-frame. Similarly, the amount of data expected to be scheduled for the reconstruction of the n-th next P-frame before the n-th future scheduling-due-date (i.e., $n \times 33.3$ ms later) is the size of the P-frame plus $P^n \times S_I$, where $n \leq 7$ in our simulation. Since P is usually small (e.g., less than 0.1), this effect diminishes rapidly.

Fig. 4 shows the distribution of the "expected" traffic arrivals of one (90%) MPEG channel, i.e., $P = 0.1$ under the MDD scheduling policy. The performance requirement can then be expressed as: $Z \leq 1 - \frac{\sum_{n=N_{max}}^{77} P(n)}{\sum_{n=6}^{77} 3P(n)} = 1 - \sum_{n=N_{max}}^{77} \frac{1}{3} P(n)$. Again, if N_{max} is sufficiently large, each point right to the dotted line with label 90% in Fig. 4 will result in loss of at most one frame. By adding 0.5 packet time per frame as the operation overhead, we get: $Z = 90\%$, (i.e.,$P = 0.1$): $N_{max} = 42$.

The subsequent link capacity to be reserved for new channels is also based on this modified distribution of traffic arrivals when the MDD scheduling policy is used. Using the same method, we can also obtain new modified distributions of frame arrivals for 95% and 99%

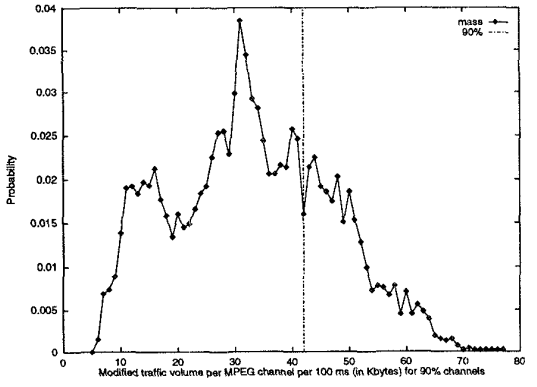

Figure 4: Distribution of modified MPEG frame arrivals

channels. However, since P is small in these two cases, N_{max} does not change for both 95% and 99% channels. Therefore, without channel multiplexing, the network is expected to support twenty-one 99% MPEG channels, twenty-six 95% MPEG channels, or thirty 90% MPEG channels, according to the admission test in Section 2.1. As we shall see later, the node-based multiplexing scheme reduces significantly the total link capacity that needs to be reserved, and our simulation results confirm its effectiveness.

4.2 Simulation Results

Fig. 5 shows the normalized link capacity needed for adding a new (node-based) channel with respect to the average traffic arrival rate of the channel. The horizontal lines "99% channel based", "95% channel based" and "90% channel based" correspond to the link capacity needed for a new 99%, 95% and 90% channel-based channel, respectively. The capacity needed for adding a new 90% node-based channel rapidly converges to the dotted line "$y = 0.9$" representing the average real-time traffic which needs to be delivered timely for achieving the required 90% frame reconstruction rate. Thus, the system only needs to reserve a link capacity according to about 90% of the *average* real-time traffic rate of a new channel when there are sufficiently many channels originating from a node (about 5 in this example). The 95% (and 99%) line also demonstrates the same trend as the 90% line, i.e., converges to a constant which is close to 95% (and 99%) of the average real-time traffic arrival rate of a channel. Thus, the network is expected to support about thirty-four 99%, thirty-seven 95% and forty 90% channels, as compared to 21, 26 and 30 without channel multiplexing. Therefore, the network utilization and real-time channel admissibility are improved significantly with channel multiplexing. In other words, if there are sufficiently many channels originating from a node, we can provide performance guarantees for statistical real-time channels by reserving the link capacity based on the *average* case rather than the worst case.

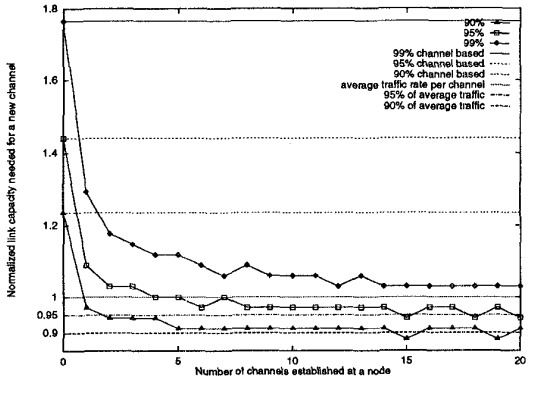

Figure 5: Normalized link capacity needed for adding a new channel with respect to the average traffic rate per channel

The second goal of our simulation is to verify that the channel-multiplexing scheme can also preserve performance guarantees. Although we have results for 99%, 95% and 90% channels, we present only the results for 90% channels due to space limitation. 95% and 99% channels follow the similar trend as the 90% channels.

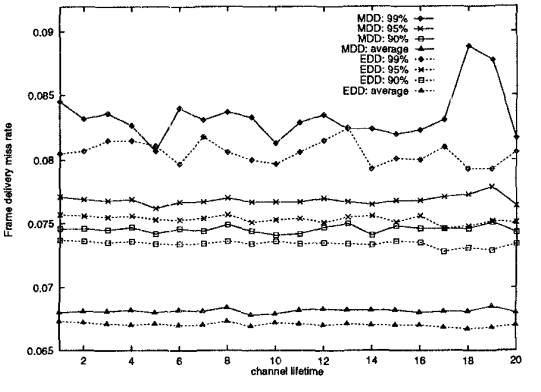

Figure 6: Frame-delivery miss rate for multiplexing 90% channels

We will first present the simulation result of channels with a short lifetime (3 minutes to 1 hour). Although the statistical guarantees are defined based on the assumption of infinitely long arrival data streams, as can be seen later, most (more than 99%) of channels with a short lifetime can still provide the required performance. Note that we use 99-percentile instead of the maximum miss rate in the short-lifetime channel experiments, since the maximum is too sensitive to a single random sample and does not reflect the real distribution of the data.

Fig. 6 shows the frame-delivery miss rates among all established channels under both the MDD and the

EDD scheduling with the proposed channel multiplexing scheme, i.e., the link-bandwidth reservation is made based on the data in Fig. 5. Each unit of lifetime represents 5374 frames, i.e., about 3 minutes. The "20" represents all channels with lifetime 20 or more. Each line in the figure represents a certain percentile (or average) among all samples with the same lifetime. For example, in Fig. 6, the line "MDD:90%" shows the 90-percentile frame-delivery miss rate of all 90% real-time channels with the same lifetime under the MDD scheduling. In this figure, the 99-percentile lines of both EDD and MDD are below the corresponding performance requirement line (0.1). That is, both of them are adequate for this case.

Fig. 7 shows the frame-reconstruction miss rates among all established channels under both the MDD and the normal EDD scheduling with the proposed channel multiplexing scheme. The MDD scheduling per-

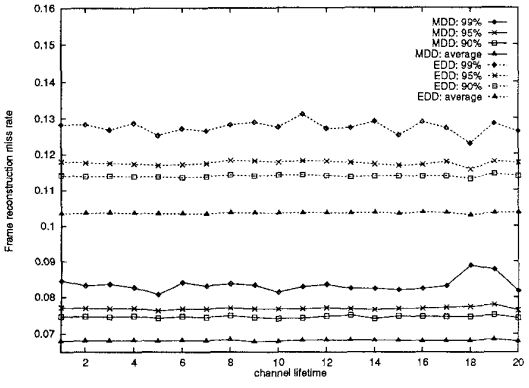

Figure 7: Frame-reconstruction miss rate for multiplexing 90% channels (short lifetime)

forms significantly better than the normal EDD in terms of frame-reconstruction rate. As can be seen in Fig. 7, the 99-percentile frame-reconstruction miss line of the MDD scheduling is below the corresponding required miss rate line, and thus, can provide the required performance guarantees. However, even the average miss rate line of EDD lies above "$y = 0.1$" in Fig. 7. Thus, the EDD scheduling is not appropriate for the short lifetime streams of non-independent frames.

The capability of providing performance guarantees can be shown in the long lifetime channel simulations. Fig. 8 shows the frame-reconstruction miss rate for multiplexing long lifetime channels. As can be seen from the figure, the node-based scheme (with MDD) works very well, i.e., the maximum frame-reconstruction miss rate among all channels is always kept under the corresponding required upper bound before the network is saturated. Each point in Fig. 8 represents the transmission of about $24,000,000$ frames per channel or 222 hours at the rate of 30 frames per second. From Fig. 8, the network with the node-based scheme can provide

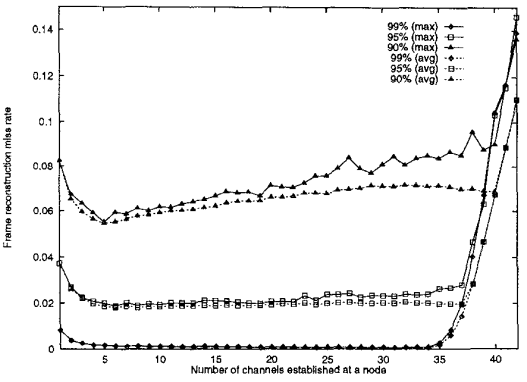

Figure 8: Reconstruction miss rate for long lifetime channels

performance guarantees for up to thirty-four 99% channels, thirty-seven 95% channels, or forty 90% channels as expected. In Fig. 8, the frame-miss rate starts high for the first channel and drops to the lowest point when there are about five channels, then rises very slowly until the network is saturated. This trend can be explained by the reserved link capacity for adding a new channel. For the first 5 channels, we reserve more than the average need for each channel, so the frame-miss rate keeps dropping. After that, the capacity we reserve for a new channel is approximately equal to the average need of each channel. Thus, the (approximately) same amount of the RBU capacity is shared by more and more channels so that the frame-miss rate rises slowly.

5 Conclusion

We presented a scheme which can provide real-time performance guarantees for given traffic-generation characteristics and performance requirements. When there are already a sufficient number of channels established, the proposed scheme is shown to be very effective in reducing the link capacity that needs to be reserved to the level of average real-time traffic from the original worst-case level of traffic, while preserving the capability of independent addition/deletion real-time channels.

References

[1] C.-C. Chou and K. G. Shin, "Statistical real-time video channels over a multiaccess network," *Proc. High-Speed Networking and Multimedia Computing Symposium,IS&T/SPIE Symposium on Electronic Imaging Science and Technology*, pp. 86–96, February 1994.

[2] C.-C. Chou and K. G. Shin, "Statistical real-time channels on multiaccess networks," *Proc. The International Conference on Computer Communication*, 1995.

[3] D. Ferrari and D. C. Verma, "A scheme for real-time channel establishment in wide-area networks," *IEEE Journal on Selected Areas in Communications*, vol. SAC-8, pp. 368–379, April 1990.