TRANSMISSION OF COMPRESSED MOTION VIDEO OVER COMPUTER NETWORKS

Qin Zheng, Kang G. Shin and Emmanuel Abram-Profeta
Real-Time Computing Laboratory
Department of Electrical Engineering and Computer Science
The University of Michigan
Ann Arbor, Michigan 48109–2122
E-mail: {zheng,kgshin,abram}@eecs.umich.edu

Abstract

Transmission of compressed digital motion video represents a challenge to the current computer networks. Due to the highly-varying bit-rate of compressed video signals, transmission over fixed-bandwidth circuits wastes link bandwidths since the circuits have to accommodate the worst-case signal rates. Usage of packet switching, on the other hand, improves link utilization, but could introduce unacceptable transmission delays due to the contention for transmission links. In this paper, we show how this problem can be solved with the concept of real-time channel *which guarantees the timely delivery of video frames without wasting network bandwidth.*

1 Introduction

With dramatic increases in link bandwidth and node processing power at inexpensive costs, real-time transmission of digital motion video over computer networks has now become a reality. Extensive work has been reported on the development of video processor, multimedia workstations, and network support for this new application [1, 2, 3, 4, 5].

Traditionally, video signals are transmitted over dedicated circuits only, e.g., CATV. Clearly, this kind of transmission lacks flexibility (mainly for broadcasting or fixed point–to–point communication) and is not easily adaptable to computer networks for which packet switching is commonly used. To remedy this inflexibility, a hybrid transmission protocol, called FDDI-II, has been proposed to add time-division circuit switching to the existing packet-switched FDDI networks [6, 7]. In an FDDI-II network, users can set up circuits with bandwidth of multiples of 64 Kbits/s up to 98.304 Mbits/s. In this way, continuous media communication like digital voice and video transmissions can be readily accommodated.

However, the problem with the use of fixed-bandwidth circuits is the inefficient use of link bandwidth [1]. The bit-rate of digital video signals, after compressed, is highly time-varying, with the peak rate several times higher than the average rate. To ensure the smooth transmission of video frames, the circuit bandwidth must be set significantly higher than the average signal bandwidth, thus wasting a great deal of circuit bandwidth. Since the bandwidth of video signals is usually very high (from several Mbits/s to several hundred Mbits/s), this kind of waste of circuit bandwidth is highly undesirable, even in high-speed networks.

By statistically multiplexing packets on transmission links, packet switching (e.g., ATM) is believed to be able to support variable bit-rate transmissions and make more efficient use of link bandwidth [1]. However, due to the contention delays at intermediate nodes, packet-switched transmission does not guarantee the timely delivery of packets. This causes a serious problem in transmitting real-time motion video where each video frame is required to be delivered in a timely manner.

A new communication protocol, called *real-time channels*, was recently proposed by Ferrari and Verma [8] which has the advantages of both circuit-switched and packet-switched transmissions. The real-time channel protocol uses two techniques to ensure the ef-

*The work reported in this paper was supported in part by the Office of Naval Research under Grants No. N00014–92–J–1080 and the National Science Foundation under Grant No. MIP–9203895. Any opinions, findings, and recommendations expressed in this publication are those of the authors, and do not necessarily reflect the views of the funding agencies.

ficient use of link bandwidth and the timely delivery of messages.

1. Packet switching with deadline scheduling. Unlike the time-division circuit switching where transmission time slots are reserved exclusively for each established circuit, the real-time channel protocol uses statistical multiplexing of packet transmissions. Thus, no transmission capacity will be wasted due to the idleness of some channels. The real-time channel protocol resembles packet switching in this respect. However, unlike the conventional packet switching where packet transmissions are scheduled on a First-In-First-Out (FIFO) or Round-Robin basis, the real-time channel protocol uses deadline scheduling for packet transmissions. Each packet is assigned a deadline. If two or more packets contend for the same transmission link, the one with the earliest deadline is transmitted first (but an ongoing packet transmission will not be preempted). There are two advantages of using the deadline scheduling policy: (1) urgent messages will be given tight deadlines, so they are more likely to be delivered in time; (2) with a proper deadline assignment scheme, the established channels do not affect each other in meeting their deadlines. This is called protection [9] or firewall s [10] between channels which is very important to prevent some malicious channels from affecting the regular operation of other channels.

2. Channel admission control. The real-time channel protocol requires the source node to establish a channel before sending any real-time message. The channel establishment procedure could succeed or fail, depending on the current network load condition. This is basically the admission control of network traffic to ensure the *timely* delivery of messages over established channels. The real-time channel protocol resembles circuit switching in this respect. However, the real-time channel protocol does not waste link bandwidth.

Research on the real-time channel protocol has mainly focused on two aspects: channel establishment and deadline scheduling of packet transmissions.

The key problem with channel establishment is to verify whether or not the user requested end–to–end delay bound can be satisfied under the current network load condition. Results on this can be found in [8, 11, 9] and we will in Section 2 enhance these for motion video transmissions.

It is also very important to efficiently implement the deadline scheduling of packet transmissions, especially in high-speed networks. To realize deadline scheduling, the waiting packets at each transmission link must be ordered into a priority queue such that the packet with the earliest deadline is always at the head of the queue. The worst-case scheduling time, i.e., the time needed to insert and delete a packet into the priority queue, must be smaller than a packet's transmission time; otherwise, all the advantages of using the deadline scheduling policy will be lost. The readers are referred to [12, 13] for an example hardware implementation of a deadline scheduler.

One special problem arises when one wants to send video signals over real-time channels. A video frame usually consists of several tens to hundreds Kbits, which are too large to fit in a single packet in most packet-switched networks. Thus, the message has to be split into several packets which are then transmitted separately. In this paper, we first generalize the real-time channel establishment procedure discussed in [9, 11] to accommodate large messages. Then, we present simulation results to show the advantages of real-time channels over both circuit and packet switching in transmitting real-time video signals.

This paper is organized as follows. Section 2 presents a modified version of the real-time channel protocol for the transmission of video signals. Section 3 presents the simulation results showing the advantages of real-time channels. The paper concludes with Section 4.

2 Real–time Channels

2.1 Real–time channel protocol

To accommodate large messages, we need to modify the real-time channel protocol described in [9, 11]. The modified protocol is composed of two parts: channel establishment/removal and message transmissions.

Channel establishment/removal:
Channel establishment can be done in a centralized [11] or decentralized manner [8]. We discuss a centralized version here which is easier to manage and implement than the decentralized one.

To establish a real-time channel, the requesting node must first determine three channel parameters, (T, M, D), where T is the minimum message intergeneration time, M is the maximum message size, and D is the requested end–to–end message delay bound.

As discussed in [9], the three–parameter model is a natural way to describe the traffic characteristics for most real-time communication applications. In case of motion video transmissions, each video frame is a message, and $T = 1/30$ second if the frames are transmitted at a rate of 30 frames/second, M is the maximum frame size which depends on the physical size, pixel resolution of the frames and the compression algorithm used, and $D = T$ if one requires that each frame reach the destination before generating the next frame.

The source node then sends a message of requesting a channel to be established that contains the channel parameters together with the addresses of the source and destination nodes to a special node containing the *Network Manager* (NM), which manages all the real-time channels in the network. After receiving a channel-establishment request, the NM first selects a route from the source to the destination over which the real-time channel is to be established. As discussed in [14], the minimum hop routing is usually preferred since it is easier to guarantee an end–to–end delay bound over a shorter route than a longer one. The NM then executes a real-time channel establishment algorithm (to be presented in Section 2.2) to determine whether or not the requested channel can be established over the selected route. Then, the NM sends a reply message back to the source node notifying the acceptance or denial of the channel request. If the channel request is accepted, the reply message also contains the route information and the link delay bound d_i over each link on the route which will be used during the message transmission stage.

After all messages have been transmitted, the source node sends a channel removal message to the NM which deletes the channel from its channel table. A message is also sent to the destination node informing it of the end of transmission.

Message transmission: The source node can start message transmission any time after receiving a positive reply from the NM. A message must be packetized first. In other words, a large message needs to be broken into packets of the size fitting the network specifications, and a header is added to each packet. The packet header contains the information for routing (an outgoing link over which the packet is to be forwarded) and transmission scheduling (the link delay bound) for each intermediate node.

Recall that the deadline scheduling policy is used for transmitting real-time channels' packets. Each packet is assigned a deadline, and when several packets contend for the same transmission link, the one with the earliest deadline is transmitted first. The deadline of a packet is the time by which the last bit of the *message* to which the packet belongs must be transmitted. We define the *arrival time* of a message to be the time when the last bit of the message's first packet arrives. In a store-and-forward packet-switched network, this is the time the first packet of the message becomes eligible for transmission. A straightforward way of deadline assignment would be to set the deadlines of a message's packets over link ℓ_i to be the message's arrival time plus the link delay bound d_i.

However, this simple method causes problems. As will be clear later, in verifying whether a requested real-time channel can be established or not, the worst-case end–to–end delay is calculated from the worst-case link delays, which are in turn calculated based on the assumption that message inter-arrival times will not be smaller than T and message sizes will not be larger than M. This assumption is not always true because: (1) the source node may generate messages faster and larger than the specified values T and M, and (2) even if the source node does not violate its traffic specification, the message inter-arrival times at an intermediate link can still be smaller than T due to the uneven queueing delays at upstream links.

To solve this problem, we use *logical arrival times* to calculate the message deadlines. If a message arrives (or is generated) too early, its logical arrival time is set to be later than its actual arrival time as if it had arrived on time. If a message of too large size is generated, part of the message will be treated as part of the next message. The message's deadline is then calculated as its logical arrival time plus d_i, and packet transmissions are scheduled according to their deadlines (which are based on logical arrival times). Usage of the logical arrival times has the following advantages.

1. Messages generated faster or larger than the specified values by a malicious channel will not affect the timely transmission of other channels' messages since these violating messages will be assigned deadlines as if they were generated as specified. With the deadlines calculated from the logical arrival times, the deadline scheduling of packet transmissions effectively provides guaranteed protection between channels. This kind of channel protection makes a real-time channel behave like a dedicated circuit and is not achievable with other policies like FIFO, Round-Robin, or priority scheduling.

2. The variation of packet inter-arrival times at an intermediate node due to uneven queueing delays at upstream links is automatically taken care of by logical inter-arrival times. Thus, we can always use the minimum message inter-arrival time to calculate the worst-case delay. Also, using logical inter-arrival times will not cause unnecessary message delays since early messages are eligible for transmission as long as there are no other messages with earlier deadlines.

With the introduction of logical arrival times, we can now present the message packetization and transmission processes.

In addition to the message type, source and destination addresses, and packet sequence number, the header of a packet contains the following fields exclusively for real-time channels: transmission link IDs ℓ_i and link delay bounds d_i, $i = 1, \cdots, k$, where k is the number of links of the channel, logical arrival time t_l, and deadline t_d.

The source node uses variables t_p, t_c to record the times the previous and current messages of the channel are generated, and m_r, m_a to record the size (in bits) of the remaining message (i.e., the part not yet packetized) and the accumulated message size (to detect oversized messages), respectively. Recall that the channel parameters are (T, M, D), and let P be the maximum packet size of the network. Initialize $t_p := -\infty$, $m_r := 0$, $m_a := 0$.

Suppose a message of m bits is generated at time t, then the source node does the following.

Step 1: Set the current message arrival time $t_c := t$, and the remaining message size $m_r := m$.

Step 2: Assemble a packet of size $p = \min\{P, m_r\}$. Fill in the packet header fields, ℓ_i's and d_i's, with the values obtained from the NM. Update the accumulated message size $m_a := m_a + p$. Set the logical arrival time $t_l := \max\{t_c, t_p + T\} + \lfloor m_a/(M+1) \rfloor T$ and the deadline $t_d := t_l + d_1$.

Step 3: Forward the assembled packet to the transmission link identified by the link ID ℓ_1. Update the remaining message size $m_r := m_r - p$. If $m_r > 0$, goto Step 2 to assemble the next packet. Otherwise, update $m_a := \max\{m_a - M, 0\}$ and $t_p := \max\{t_c, t_p + T\}$ (message packetization is done).

The packets forwarded to an outgoing link are scheduled according to their deadlines. The readers are referred to [12, 13] for an example implementation of this scheduling process. If the clocks of the nodes are not synchronized, we assume that the transmitter will put a time-stamp t_t at the header of each packet when it is transmitted.

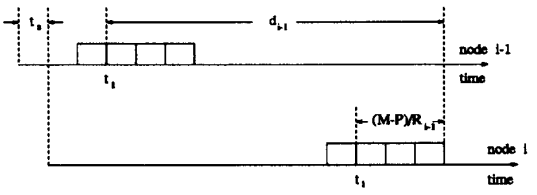

Figure 1: Calculation of a message's logical arrival time at node i.

At an intermediate node, say node i with the outgoing link ℓ_i, a packet arrived at time t is processed as follows.

Step 1': Calculate the clock skew between node $i-1$ and node i: $t_s = t - t_t$. As discussed in [9], the end-to-end propagation delay can be pre-subtracted from the user requested end-to-end delay bound. So t_s also removes the link propagation delay. We ignore the time-stamping time (since hardware can be used for this). Suppose link ℓ_i has a transmission bandwidth R_i, then update the fields of the logical arrival time and the deadline in the packet header as $t_l := t_l + t_s + d_{i-1} - \max\{0, (M-P)/R_i\}$, $t_d = t_l + d_i$.

Step 2': Remove ℓ_{i-1} and d_{i-1} from the packet header and forward the packet to the link ℓ_i.

The calculation of the logical arrival time t_l in Step 1' can be explained with Fig. 1 which shows the logical arrival times of a message at node $i-1$ and i. A message's *link delay* is defined as the time between the logical arrival time of the message and the time when the last bit of the message is transmitted. Then, a message's logical arrival time at node i equals the message's logical arrival time at node $i-1$ plus the worst-case link delay d_{i-1} minus the time needed to transmit $M-P$ bits (i.e., the maximum message minus the first packet). One important fact is that if all channels are established through the NM, which ensures the messages' actual link delays not to exceed the worst-case link delays d_i's, then the actual message arrival times at a node will never be later than the corresponding logical arrival times. If this is violated, then something must have gone wrong: either the NM is not functioning properly or a source node is sending real-time messages without getting an approval from the NM.

2.2 The real–time channel establishment algorithm

This algorithm is to (i) check whether a requested channel can be established over a given route, and (ii) calculate the delay bound of each link in the route if the channel can be established.

Similar to [9], we first consider the message delay over a single link. For the convenience of presentation, we replace the channel parameter M, i.e., the maximum message size, with a parameter C which equals the transmission time of a maximum-size message, e.g., $C = M/R$ if the link has a transmission rate R. Also, let C_p denote the time needed to transmit a packet.

A set of real-time channels $\tau_i = (T_i, C_i, d_i)$, $i = 1, 2, \cdots, n$ is said to be *schedulable* over a link if for all $1 \leq i \leq n$, the maximum link delay experienced by channel τ_i's messages is not greater than d_i. Then, we have the following schedulability condition.

Theorem 2.1 (Channel Schedulability Condition)

In the presence of non real-time packets, a set of real-time channels $\tau_i = (T_i, C_i, d_i)$, $i = 1, 2, \cdots, n$, are schedulable over a link under the deadline scheduling policy if and only if

$$\sum_{j=1}^{n} \lceil (t - d_j)/T_j \rceil^+ C_j + C_p \leq t, \quad \forall\, t \geq d_{min}$$

where $d_{min} = \min\{d_i : 1 \leq i \leq n\}$.

The proof of the theorem is omitted here because of the space limitation.

For a set of channels $\tau_i = (T_i, C_i, d_i)$, $i = 1, 2, \cdots, n$, define $t_{max} = \max\{d_1, \cdots, d_n, (\sum_{i=1}^{n}(1 - d_i/T_i)C_i)/(1 - \sum_{i=1}^{n} C_i/T_i)\}$. Then, similar to the Theorem 2 of [9], it is easy to prove that one only needs to check the inequality of Theorem 2.1 for a finite set $S = \cup_{i=1}^{n}\{d_i + nT_i : n = 0, 1, \cdots, \lfloor (t_{max} - d_i)/T_i \rfloor\}$. Then, similar to Theorem 3 of [9], the worst-case message link delay can be calculated as stated in the next theorem.

Theorem 2.2 (Worst-Case Link Delay) .

Let $f(t, d_n) = \sum_{j=1}^{n} \lceil (t - d_j)/T_j \rceil^+ C_j + C_p$ and S be the set defined above with $d_n = C_n + C_p$. Then, $d_n = C_n + C_p$ is the worst-case delay if $f(t, C_n) \leq t$, $\forall t \in S_i$. Otherwise, the worst-case delay is $d_n = \max\{d^t : t \in G\}$, where $G = S \cap \{t : f(t, C_n) > t\}$ and d^t is computed as $d^t = C_n + k_f^t T_n + \epsilon_f^t + \epsilon_i^t$, with $k_f^t = \lfloor (f(t, C_n) - t)/C_n \rfloor$, $\epsilon_f^t = f(t, C_n) - t - k_f^t C_n$, $\epsilon_i^t = t - C_n - k_i^t T_n$, $k_i^t = \lfloor (t - C_n)/T_n \rfloor$.

The proof of Theorem 2.2 is similar to that of Theorem 3 in [9] and thus omitted. Now we calculate the end–to–end message delays from link delays. The *end–to–end message delivery delay* is defined as the time period between the generation of the message at the source node and the arrival of the last bit of the message at the destination node. By the definition of link delay, the end–to–end delay no longer equals the summation of link delays. The following theorem gives a formula for the worst-case end–to–end delay from link delays.

Theorem 2.3 (Worst-Case End–to–end Delay)

Suppose a real-time channel with a maximum message transmission time C runs over n links which guarantee the worst-case link delays $d_1, \cdots, d_n$, respectively. Then, ignoring the propagation delay, the worst-case end–to–end message delivery delay can be calculated as

$$D = \sum_{i=1}^{n} d_i - (n-1)\max\{0, (C - C_p)\},$$

where C_p is the packet transmission time.

The proof of the theorem is omitted here because of the space limitation.

In summary, we have the following algorithm for the establishment of a real-time channel in a network.

Algorithm 2.1 (Real-time channel establishment)

Step 1. *Suppose a new channel is to run over the route from the source to destination that contains k links $\ell_1, \cdots, \ell_k$. Using Theorem 2.2, calculate the worst-case message link delay d_{max}^j over link ℓ_j, $j = 1, \cdots, k$.*

Step 2. *Calculate the worst-case end–to–end message delay D_{max} using Theorem 2.3. If $D_{max} \leq D$, the requested real-time channel can be established. Assign the delay bound of ℓ_j to be $d^j = d_{max}^j + (D - D_{max})/k$. Otherwise, the channel establishment request is rejected.*

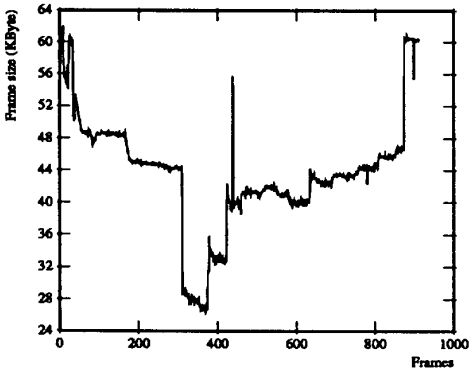

Figure 2: Video frame sizes.

3 Simulation Results

As discussed earlier, the transmission of compressed digital motion video requires both *variable* link bandwidths and *timely* delivery of video frames. Using fixed-bandwidth circuits wastes link bandwidth and using packet switching cannot guarantee the timely delivery of video frames. In this section, we verify these claims with simulations and show that real-time channels outperform both circuit and packet switching with respect to the utilization of link bandwidth and the timeliness of frame delivery.

3.1 Simulation Models

The network used for simulation is a 100 Mbits/s ring with 20 nodes. When video transmission is introduced, the 100 Mbits/s ring could easily get congested and thus become the bottleneck of the system.

Our simulation goal is to evaluate and compare the transmission delays of video frames using circuit switching, packet switching, and real-time channels. The video data used here are obtained from a sequence of CNN headline news, stored on a laser disk. The size of each frame is 512 × 512 black and white pixels. The number of bits in each frame, after compressed with JPEG [15], is plotted in Fig. 2. At a 30 frames/second transmission rate, the video sequence needs an average 10.5 Mbits/s, and a peak 15.3 Mbits/s transmission bandwidth.

The frame-transmission delays using a fixed-bandwidth circuit are easy to calculate. Let τ_i be the time when the ith frame is generated at the source node, and t_i be the time when the last bit of the ith

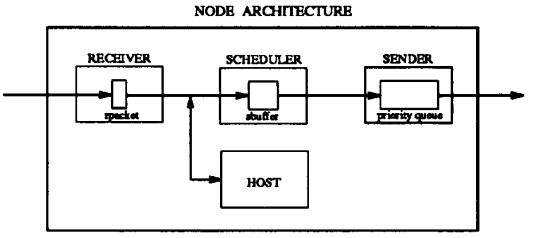

Figure 3: The simulation model of node architecture for real-time channels.

frame has been transmitted. Ignoring the propagation delays, the delay experienced by the ith frame is $d_i = t_i - \tau_i$. Starting from time 0 and assuming a 30 frames/second transmission rate, τ_i's and t_i's can be recursively calculated as follows:

$$\tau_i = i/30, \quad i = 1, 2, ...$$

$$t_i = \max\{t_{i-1}, \tau_i\} + S_i/B, \quad i = 1, 2, ...$$

$$t_0 = 0$$

where B is the bandwidth of the circuit, and S_i is the number of bits in the ith frame.

Unfortunately, the frame-transmission delays using packet switching or real-time channels cannot be derived analytically. We wrote a detailed simulator (an 1100-line C program) to emulate the real transmission process and observe the delays each individual frame experienced. The simulation model of the node architecture which supports real-time channels is shown in Fig. 3.

In our simulation, each video frame is divided into packets of 10 Kbits each. The packet deadlines are calculated as described in Section 2. The packets received from the up-ring or generated at the local host are first put in the scheduler buffer *sbuffer*, and then are inserted in the *priority queue* according to their deadlines. When the sender completes the transmission of a packet, it fetches and transmits the next packet at the head of the priority queue. See [12] for a detailed account of the node architecture.

The same simulator is used for packet switching with one change: packets from *sbuffer* are queued at the tail of the priority queue. This reflects the fact that packet transmission is usually scheduled on a First-In-First-Out (FIFO) basis in an ordinary packet switching node.

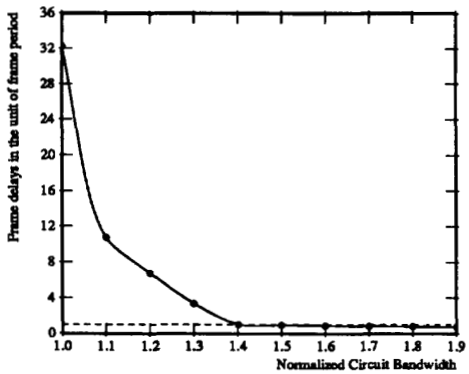

Figure 4: Maximum video frame delays with circuit switching.

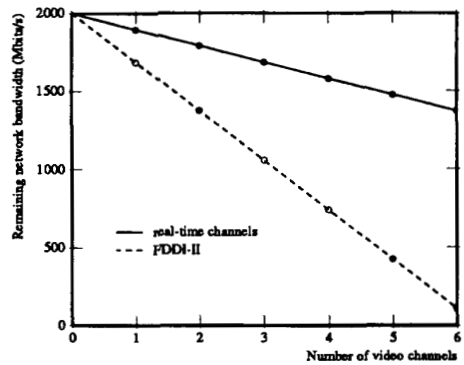

Figure 5: The remaining network bandwidth after establishing video channels.

3.2 Circuit Switching vs. Real-time Channels

Link efficiency: As stated earlier, the main problem with circuit switching is the waste of circuit bandwidth. There is a tradeoff between circuit bandwidth and video-frame transmission delays. Reserving too little circuit bandwidth introduces large frame delays, while reserving too much bandwidth wastes network capacity since no one else is allowed to use the reserved bandwidth. Our first experiment is thus to investigate the relation between the circuit bandwidth and the maximum frame delay.

We use FDDI-II in our simulation since it is a well-defined protocol and supports circuit-switched transmissions in packet-switched ring networks. The simulation results are plotted in Fig. 4, where the x-axis represents the circuit bandwidth normalized by the average signal bandwidth, i.e., 10.5 Mbits/s, and the y-axis represents the maximum frame delays normalized by the frame period, i.e., 33.3 ms. From Fig. 4 one can see that the maximum frame delays are very sensitive to the circuit bandwidth allocated. For the video sequence used in our simulation (Fig. 2), the maximum frame delay is as large as a 32-frame period, or approximately 1 second, if the circuit bandwidth is set to be the average signal bandwidth. To make the maximum frame delay smaller than one frame period, the circuit bandwidth has to be at least 1.5 times the signal bandwidth, wasting 50% of the network capacity.

With real-time channels, on the other hand, no network capacity will be wasted. The average bandwidth used by a real-time channel is always equal to the average bandwidth of the signals transmitted over the channel. When a real-time channel is idle, the full link bandwidth can be used to handle other traffic.

Another disadvantage of FDDI-II is that an established circuit always runs through every node in the network regardless of the distance between the source and destination. This aggravates the problem of wasting circuit bandwidth since it requires the reservation of some links' bandwidth even if they are not used for communicating packets between the source and destination. If we define the network bandwidth as the total number of links in the network times the bandwidth of each link, then the network bandwidth available for other traffic after establishing N FDDI-II circuits with bandwidth B in our 100 Mbits/s 20-node ring would example be $2000 - 20NB$ Mbits/s which is plotted in Fig. 5 with $B = 15.75$ Mbits/s. With real-time channels, on the other hand, only those links which connect the source and destination nodes are used. In a 20-node ring network, the average number of links needed to connect a pair of source and destination nodes is 10. Thus, the network bandwidth available after establishing N real-time channels is $2000 - 10NB_s$ Mbits/s (as plotted in Fig. 5) on the average, where $B_s = 10.5$ Mbits/s is the video signal bandwidth. From Fig. 5, we see that real-time channels use significantly less network bandwidth than FDDI-II.

Ability to accommodate video channels: We now compare the number of video channels that each protocol can support. As stated earlier, FDDI-II

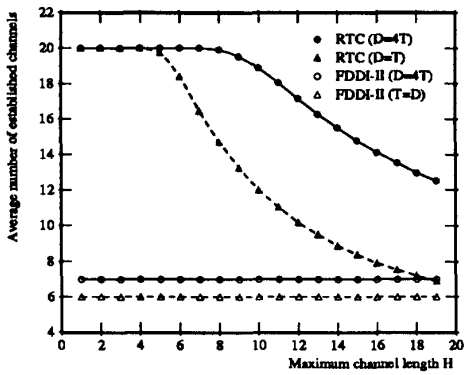

Figure 6: Average number of successfully established video channels with real-time channels and FDDI-II. The lengths of real–time channels are uniformly distributed in $[1, H]$.

allows users to set up circuits with bandwidth of multiples of 8 Kbits/s up to 98.034 Mbits/s in a single-ring network. From Fig. 4 we see that to make video frame-transmission delay smaller than one frame period, the circuit bandwidth should be at least 1.5×10.5 Mbits/s = 15.75 Mbits/s. With the total available bandwidth of 98.034 Mbits/s, at most 98.034/15.75 = 6 video channels can be established.

The number of video channels that can be established with real-time channels depends on the locations of the source and destination nodes. We assume that each host node wants to set up a video channel to a destination node which is h links away, where h is a random integer variable uniformly distributed over $[1, H]$, and $1 \leq H \leq 19$ is an integer parameter (H cannot be larger than 19 in a 20-node ring). For each H, we use Algorithm 2.1 to establish real-time channels for each node. The parameters of the real-time channels are set to: minimum message inter-arrival time $T = 33$ ms (30 frames/second), maximum message transmission time $C = 5$ ms (500 Kbits maximum message size), requested end–to–end delay bound $D = 33$ ms (one frame period), and the packet transmission time $C_p = 0.1$ ms (10 Kbits packet size). Each experiment was repeated 10000 times. Out of 20 requested channels, the average numbers of establishable channels for different values of H are plotted in Fig. 6 (the dashed curve).

From this figure, we see that real-time channels outperform FDDI-II by far when the channel length is short. The number of establishable channels decreases as the channel length increases.

Another interesting feature is that one can significantly increase real-time channels' ability to accommodate video channels by relaxing end–to–end delay bounds. For example, if each receiving node has a buffer capacity to store 4 video frames, then the end–to–end frame delivery delay bound can be relaxed to be four frame periods, i.e., $D = 4T = 120$ ms. From Fig. 4, we see that with circuit switching, the circuit bandwidth can be reduced to be 1.3 times the average signal bandwidth, i.e., 1.3×10.5 Mbits/s = 13.65 Mbits/s. Thus, 98.034/13.65 = 7 video channels can be established in an FDDI-II network. The number of establishable video channels with real-time channels for $D = 4T$ is also plotted in Fig. 6 (the solid curve), which shows a significant improvement over FDDI-II.

3.3 Packet Switching vs. Real-time Channels

Ordinary packet switching exhibits two problems when used for real-time communication. First, its First-In-First-Out (FIFO) or Round-Robin scheduling policy treats all packets equally. Urgent messages do not receive the requisite transmission priority, and thus could easily miss their deadlines. Second, no efficient traffic control scheme is used to prevent network from congestion. When the network gets congested, real-time messages, just like non real–time messages, will get delayed or lost.

With the real-time channel protocol, urgency of a message is represented by the deadlines of its packets. The deadline scheduling of packet transmissions gives priority to the most urgent packets. The amount of real-time traffic over each link is controlled by the channel establishment algorithm that guarantees the timely delivery of all real-time messages of any established channel. The existence of non real-time traffic in the network has virtually no effect on real-time channels since non real-time packets always have lower transmission priority than real-time packets. When the network is congested with excessive non real-time traffic, only non real-time messages will get delayed or lost.

We verified the above claims with simulations. 19 video channels were established in the 20-node ring network described in Section 3.1. Channel 0 covers the whole ring, i.e., with the source node 0 and the destination node 19. Channel i, $i = 1, \cdots, 18$, has the source node i and the destination node $i + (3 - (i - 1) \bmod 3)$. All 19 channels transmit the video frames

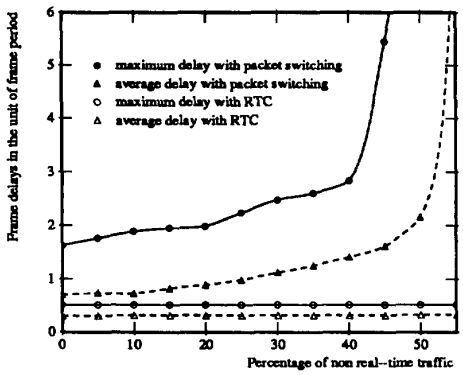

Figure 7: Maximum and average video frame delays with Real-time Channels (RTC) and ordinary packet switching.

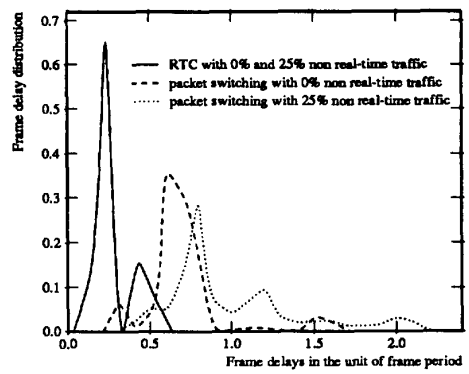

Figure 8: Frame delay distributions. The figure shows the number of frames, normalized by the total number of frames transmitted, whose transmission delays lie in each interval equal to a 1/10 of the frame period.

displayed in Fig. 2. The maximum frame transmission delay is required to be smaller than one frame period.

A certain amount of non real-time traffic is generated with random source and destination nodes. The amount of traffic is measured by the percentage of the network bandwidth needed to transmit them. For example, 10% of non real-time traffic in a 100 Mbits/s ring means that each link will carry an average of 10 Mbits/s of non real-time packets. Recall that each video channel carries signals of an average bandwidth of 10.5 Mbits/s, and up to 4 video channels are established over some links. Thus the network will get congested if the non real-time traffic over one link exceeds 100 Mbits/s -4×10.5 Mbits/s $= 58$ Mbits/s, or the total non real-time traffic exceeds 58% of the network bandwidth.

The maximum and average frame delays of channel 0 are plotted in Fig. 7. Fig. 8 plots the distribution of frame delays. One can see that real-time channels are virtually independent of the amount of non real-time traffic. The maximum and average delays keep well below the requested one frame period, even when the network is nearly congested.

With the ordinary packet switching, on the other hand, frame delays increase as non real-time traffic increases. This makes the transmission of real-time video signals very unreliable. Notice that several large file transmissions could easily get the network congested. During such a transmission period, the frame delays could exceed 10 frame periods, interrupting the receiving side even if a few frame buffers are used.

Adding a priority mechanism to the ordinary packet switching, i.e., giving real-time packets a higher transmission priority than non real-time packets, could alleviate the problem. However, the performance still will not match that of real-time channels for the following reasons.

1. There exist differences in urgency among real-time messages. For example, the messages of channel 0 (which need to travel 19 links) are more urgent than those of other channels (which only travel at most 3 links). Also, some nodes may have larger frame buffers than others, and thus the frames sent to these nodes can tolerate larger transmission delays. A priority mechanism usually does not distinguish these differences, the effect of which can be seen in Figs. 7 and 8. In the absence of non real-time messages, the frame delays of channel 0 with packet switching are still larger than that with real-time channels. The reason for this phenomenon is that when real-time channels are established, Algorithm 2.1 gives channel 0 smaller link delays than that of the other channels. Thus, channel 0's packets will be assigned tighter deadlines and are thus more likely to be transmitted before the packets of other channels. So, the real-time channel protocol is more flexible than others in accommodating heterogeneous real-time traffic (i.e., with different transmission delay and bandwidth requirements)

which is unachievable with priority mechanisms.

2. No efficient traffic control schemes are known for (priority) packet switching. Thus it is still possible that real-time traffic could temporarily congest the network. The real-time channel protocol, on the other hand, will avoid congestion by rejecting requests for establishing new channels at the channel establishment stage.

3. Even if a traffic control scheme is employed for (priority) packet switching, the established channels could still affect each other. A video channel, for example, could create more traffic than specified from time to time (say, when the scene moves very fast). This extra traffic could affect the timely transmission of other channels' messages. As discussed in Section 2.1, this will not happen to the real-time channel protocol due to the deadline scheduling of packet transmissions.

4 Conclusions

In this paper, we have presented an enhanced version of the real-time channel protocol for the transmission of compressed digital motion video over computer networks. This protocol can guarantee the timely delivery of video frames without wasting network bandwidth. Extensive simulation results have also shown the protocol's superiority over the ordinary circuit/packet switching switching protocols.

References

[1] M. Liebhold and E. M. Hoffert, "Toward an open enviroment for digital video," *Communication of ACM*, vol. 34, no. 4, pp. 104 - 112, April 1991.

[2] E. A. Fox, "Advances in interactive digital multimedia systems," *Computer*, pp. 9 - 21, October 1991.

[3] B. I. Szabo and G. K. Wallace, "Design considerations for JPEG video and synchronized audio in a Unix workstation enviroment," in *proceedings of summer 1991 Usenix Conference*, pp. 353 - 368, 1991.

[4] T. D. C. Little and A. Ghafoor, "Network considerations for distributed multimedia object composition and communication," *IEEE network magazine*, pp. 32 - 49, November 1990.

[5] M.-S. Chen, Z.-Y. Shae, D. D. Kandlur, T. P. Barzrilai, and H. M. Vin, "A multimedia desktop collaboration system," Research Report, IBM Research Division, T. J. Watson Research Center, 1992.

[6] F. E. Ross, "An overview of FDDI: The fiber distributed data interface," *IEEE Journal on Selected Areas in Communications*, vol. 7, no. 7, pp. 1043 - 1051, September 1989.

[7] M. Teener and R. Gvozdanovic, "FDDI-II operation and architectures," in *proceedings of the 14th conference on local computer networks*, pp. 49–61, 1989.

[8] 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, no. 3, pp. 368–379, April 1990.

[9] Q. Zheng and K. G. Shin, "On the ability of establishing real-time channels in point–to–point packet–switched networks," *IEEE Transactions on Communication* (in press), 1992.

[10] L. Zhang, "Virtual Clock: A new traffic control algorithm for packet-switched networks," *ACM Trans. Computer Systems*, vol. 9, no. 2, pp. 101–124, May 1991.

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

[12] Q. Zheng and K. G. Shin, "Real–time communication in local area ring networks," to appear in *Proc. 17th Conference on Local Computer Networks*, 1992.

[13] A. Indiresan and Q. Zheng, "Design and evaluation of a fast deadline scheduling switch for multicomputers," RTCL working document, December 1991.

[14] Q. Zheng and K. G. Shin, "Fault–tolerant real–time communication in distributed computing systems," in *Proc. 22nd Annual International Symposium on Fault–tolerant Computing*, 1992.

[15] G. K. Wallace, "The JPEG still picture compression standard," *Communication of ACM*, vol. 34, no. 4, pp. 30 - 43, April 1991.