TCP-Friendly Internet Video Streaming Employing Variable Frame-Rate Encoding and Interpolation

JongWon Kim, Young-Gook Kim, HwangJun Song, Tien-Ying Kuo, Yon Jun Chung, and C.-C. Jay Kuo, *Fellow, IEEE*

***Abstract*—A feedback-based Internet video transmission scheme based on the ITU-T H.263+ is presented. The proposed system is capable of continually accommodating its stream size and managing the packet loss recovery in response to network condition changes. It consists of multiple components: TCP-friendly end-to-end congestion control and available bandwidth estimation, encoding frame-rate control and delay-based smoothing at the sender, media- aware packetization and packet loss recovery tied with congestion control, and quality recovery tools such as motion-compensated frame interpolation at the receiver. These components are designed to meet the low computational complexity requirement so that the whole system can operate in real-time. Among these, the video-aware congestion control known as receiver-based congestion control mechanism, the variable frame-rate H.263+ encoding, and fast motion-compensated frame interpolation components are key features. Through a seamless integration, it is demonstrated that network adaptivity is enhanced enough to mitigate the packet loss and bandwidth fluctuation, resulting in a more smooth video experience at the receiver.**

***Index Terms*—End-to-end congestion control, frame interpolation post-processing, internet video streaming, TCP-friendliness, variable frame-rate encoding.**

I. Introduction

TRADITIONALLY, video conferencing systems have been mainly deployed on the transmission media such as integrated service digital network (ISDN) and asynchronous transfer mode (ATM). Under these environments, the compressed video can be delivered without worrying seriously about possible bandwidth fluctuation and stream corruption. However, the explosive growth of the Internet has brought an emerging video streaming/conferencing demand over the best-effort packetized network. This trend has been motivating intense and diverse research activities on transmitting delay-sensitive video through the bandwidth-limited and error-prone Internet. A number of proprietary coding techniques are proposed by the video streaming industry, while the standardized video conferencing system under the ITU-T H.323 umbrella is being actively studied [1]. In practice, H.323 systems are deployed mainly on Intranets or enterprise networks, since today's Internet does not provide the required infrastructure to achieve the quality of service (QoS). One major problem is the end-to-end delay and jitter, which usually breaks the latency requirement of the interactive real-time video. Together with the other packet loss problem, it can be approached in two aspects. One is from the network congestion-control viewpoint, where we should reduce future packets that are likely to be dropped. In addition, transmitted video should behave in a TCP-friendly manner, i.e., using only a fair share of available bandwidth relative to other majority TCP streams [15]. The other is from the video quality control viewpoint, where we should consider the spatio-temporal quality tradeoff to keep the transient quality degradation to minimum. Under low-latency video transmission, lost and late packets are unusable to the decoder, which needs the error resiliency and error concealment techniques in conjunction. Also, adequate quality recovery techniques should be applied to restore the degraded quality due to poor quantization and temporal frame skipping. The adoption of low cost deblocking/deringing filters as a spatial tool and temporal repetition/interpolation as a temporal tool is a general trend.

Internet video applications are more likely to be successful if they are aware of the network characteristics via the use of feedback. Feedback helps to match the transmission to the available bandwidth as well as to increase the robustness of the video transmission. Thus, to overcome the barriers of the best-effort Internet, an integration of video-aware end-to-end congestion control, denoted as receiver-based congestion control mechanism (RCCM), and ITU-T H.263+ video [2] is exploited in this paper. The proposed Internet video streaming system is illustrated in Fig. 1, where the entire Internet is treated as one massive IP cloud with inherent delay and loss, and the transmitter and the receiver are well-defined entry and exit points for the end-to-end video transmission. The real time streaming protocol (RTSP) provides the RTP/UDP/IP data and TCP control connection for a session-level interaction and the proposed congestion control employs feedback modified from RTP/RTCP. Through feedback, the variable bit rate (VBR) video encoder with real-time capability is designed to accommodate a directive usually in form of the bit budget (i.e., the available bandwidth). This bit budget directive is fed back from the receiver to the sender. The purpose of this feedback is to minimize future packet loss. It does so by instructing the encoder the amount of rate reduction required in face of current packet loss. By minimizing packet loss, not only are end users assured uninterrupted

Manuscript received July 1999; revised April 2000. This paper was recommended by Guest Editor Y.-Q. Zhang.

J. Kim, Y.-G. Kim, and C.-C. J. Kuo are with the Integrated Media Systems Center and Department of Electrical Engineering-Systems, University of Southern California, Los Angeles, CA 90089-2564 USA (e-mail: jongwon@sipi.usc.edu; younggoo@sipi.usc.edu; cckuo@sipi.usc.edu).

H. Song was with the Software Engineering Department, Sejong University, Seoul, Korea. He is now with Hongik University, Seoul, Korea (e-mail: hwangjun@kunja.sejong.ac.kr).

T.-Y. Kuo is with Sharp Labs of America, Inc., Huntington Beach, CA 92647 USA (e-mail: tkuo@sharplabs.com).

Y. J. Chung is with the Streaming Group, Akamai Technologies, San Mateo, CA 94404 USA (e-mail: yjchung@akamai.com).

Publisher Item Identifier S 1051-8215(00)08200-8.

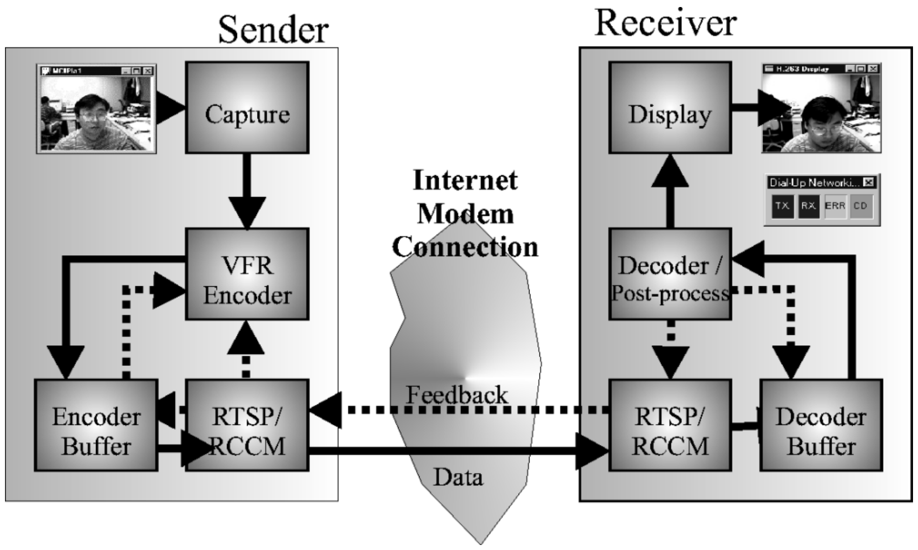


Fig. 1. Illustration of the proposed TCP-friendly Internet video streaming system.

video transmission, albeit temporarily at a lower resolution, but they also behave in a TCP-friendly manner. Another potential of the feedback-driven solution lies in the interactively designed error resiliency, where the encoder and the decoder work in conjunction to recover from packet loss.

To realize the proposed system, the following four key modules have been developed: 1) TCP-friendly end-to-end congestion control module for available bandwidth estimation; 2) rate and delay controlled video compression and transmission module to match the variable frame-rate real-time video encoder to the available bandwidth; 3) media-aware packetization and packet loss recovery module by combining acknowledgment of congestion control and video error recovery; and 4) quality recovery module featuring the fast motion-compensated frame interpolation (FMCI) post-processing. Through a seamless integration, it is demonstrated that network adaptivity is enhanced enough to mitigate the packet loss and bandwidth fluctuation, resulting in a more smooth video experience at the receiver. Several contributions made in this work are as follows. First, a TCP-friendly rate-based end-to-end congestion control is designed with video transmission in mind. The awareness of video transmission in terms of rate and error recovery is intuitively combined in the RCCM. Second, the novel variable frame-rate encoding of H.263+ [23] capable of adapting the inherent motion of video and the bandwidth fluctuation is incorporated into the available bandwidth framework. Also, the FMCI post-processing, the temporal quality recovery tool for the irregularly skipped frames, is another unique feature. Third, the error resiliency options of H.263+ are tied with the application layer framing (ALF) of real-time transport protocol (RTP) [10] to enable efficient media-level error recovery. Finally, provisions for IntServ/DiffServ and multicast video are considered. A three-class H.263+ video based on adaptive temporal layering is devised to leverage the future DiffServ architecture and DS byte [12], [13]. Regarding the issue of QoS, H.323 (version 2) has been designed to encompass the resource reservation protocol (RSVP). This one gives us the opportunity to leverage IETF's IntServ/DiffServ once it is available in the underlying network fabric. For multicast video [9], both RCCM and the three-class layering will provide a good starting point.

This paper is organized as follows. Section II covers features and descriptions of the proposed video-aware RCCM. Following that, the variable frame-rate video encoding is introduced as a key tool to match the available bandwidth in Section III, where the associated video smoothing scheme for end-to-end delay control is also described. Then, the media-aware packetization for error recovery and its interaction with RCCM is explained in Section IV. Section V is allocated for the quality recovery tools featuring the FMCI. In Section VI, each component is verified either through the network simulator (NS) [34] or the real Internet modem experiment. The improved performance of the overall feedback system is demonstrated in a real-world Internet video streaming environment. One can view video streaming as a relaxed form of video conferencing, since it allows more room compared to video conferencing that requires near symmetric encoding/decoding and low latency. Although our demonstration is confined to the video streaming scenario because of the inescapable delay of today's Internet, many tools developed here are also applicable to the low-latency interactive video conferencing. Finally, Section VII gives some concluding remarks.

II. End-to-End Congestion Control and Available Bandwidth Estimation

A. RCCM: Video-Aware End-to-End Congestion Control

The recent batch of end-to-end congestion control protocols to enforce the TCP-friendly behavior appears to be influenced by existing TCP congestion control mechanisms. As a result, to estimate the round-trip time (RTT) and to determine the congestion status, the sender-based mechanisms are usually employed. Two main approaches are utilized, i.e., window- and rate-based approaches. The window-based approach utilizes the

AIMD (additive increase/multiplicative decrease) rule by adjusting the window size in response to the positive/negative acknowledgment (ACKs). The rate-based approach controls the transmission rate directly based on the AIMD rule or analytical TCP-model equation [17]–[19]. The rate-adaptation protocol (RAP) utilized ACK for every packet to estimate RTT and packet loss to achieve TCP-friendliness [17]. In [19], a modified RTP/RTCP feedback was utilized to estimate RTT, the packet loss rate, and the bottleneck bandwidth. An intensive TCP throughput analysis was performed to derive an elaborate equation to which the sender is adjusting the rate in [18]. One common problem with the referenced works is that they mainly focus on TCP-friendliness and their solutions might result in inefficient usage of resources. When video stream is enforced to exhibit the TCP-friendly behavior, the resulting video may possess less than desirable visual quality. As an alternative, we propose a video-aware congestion control, RCCM, as a successor of our previous model-free least mean square (LMS) bandwidth controller [22]. Features of RCCM include receiver-based control and video-awareness, which allow us to feed back the current network status in an accurate and timely manner. Let us take a close look at the design features of RCCM below.

First, the congestion control module should dictate the dynamically changing bit budget for the video encoder. To achieve this goal, a TCP-friendly available bandwidth dictation is required. We employ the rate-based approach that attempts to control the sending rate explicitly. In RCCM, the receiver will play a key role in estimating the available bandwidth to exploit the advantage over the sender-side estimation as done in [21]. The estimated available bandwidth or relevant measures to help the estimation will be fed back from the receiver to the sender depending on the scheme adopted. The rate will then be adjusted by using the AIMD constraint to impose TCP friendliness. RCCM intends to coordinate the conflicting demand of video and TCP-friendliness. The characteristic of video is best represented by its VBR nature, which requires an overaly smooth bandwidth with some high-peaks. TCP's AIMD behavior is characterized by its not self-limiting and saw-tooth oscillation curve. Thus, whatever scheme claims to achieve TCP-friendliness should mimic a similar behavior in a certain timescale. One may try to deviate from this oscillatory imposition by claiming long-term sense TCP-fairness with the TCP- model equation [18]. However, we believe it may scarce TCP traffic in a short-term sense (since it will be rather reluctant in reducing its sending rate to one half immediately after a packet loss) and possibly cause more trouble as UDP traffic grows in the Internet. Thus, in RCCM, a balancing mechanism is introduced through the AIMD constraint to associate the awareness on video characteristics by relaxing the AIMD rule.

Next, from the error recovery viewpoint, provisions for media- aware packetization, unequal error protection, and timely coordination of automatic repeat request (ARQ) are desired. The stringent delay requirement of video complicates the handling of lost, out-of-order, and duplicated packets. In TCP, fast retransmission occurs whenever there are three-duplicated ACKs because its main purpose is reliability. However, in the case of real-time video, tolerance on problem packets varies according to the packet importance, the existence of alternative or parity packet, and the timing for ARQ request. Thus, we investigate the possibility of intelligent decision at the receiver. A design to let both sender/receiver know the current RTT helps fast retransmission and recovery. With the combined knowledge on the congestion control as well as media, reduction of redundant control and data messages is possible. In addition, though RCCM is mainly tuned for reliable unicast at present, its generalization to the multicast scenario seems promising [9]. It can help the server to accommodate diverse needs of multiple remote nodes. Also, all positive/negative ACKs can be controlled to prevent feedback explosion by restraining from unnecessary ACKs and controlling the ACK frequency in a hierarchical fashion.

B. Receiver-Based Congestion Control and Available Bandwidth Estimation

The rate side design of the RCCM is covered here while the error recovery part will follow later in Section IV. RCCM provides a control interface on the transmission rate while preserving TCP-friendliness. Let us describe RCCM in detail starting from the RTT, which synchronizes the AIMD behavior of RCCM and TCP. In RCCM, RTT is managed similarly in TCP with several exceptions. In server-based schemes such as TCP, only the server is able to track RTT based on ACKs from the receiver, preventing receiver's active role. However, in RCCM, the sender and the receiver share RTT to coordinate the time-sensitive response. More precisely, the sender puts the feedback number and the elapsed time (the time gap between the reception of the feedback and the sendoff of a new packet) in the header. After receiving each packet, the receiver calculates $\mathrm{RTT}_{\mathrm{sample}}$ and updates $\mathrm{RTT}_{\mathrm{estimate}}$ with

$$\mathrm{RTT}_{\mathrm{sample}} = T_{\mathrm{FR}} - T_{\mathrm{FS}} - T_{FE} \tag{1}$$

$$\mathrm{RTT}_{\mathrm{estimate}} = \alpha \times \mathrm{RTT}_{\mathrm{estimate}} + (1-\alpha) \times \mathrm{RTT}_{\mathrm{sample}} \tag{2}$$

where $T_{\mathrm{FS}}, T_{\mathrm{FR}}, T_{\mathrm{FE}}$ are the feedback send time at the receiver, the feedback response receive time at the receiver, and the elapsed time at the sender, respectively. Equation (2) is basically borrowed from TCP and α is often set to 0.9.

In addition to $\mathrm{RTT}_{\mathrm{estimate}}$ and packet loss (to be covered in Section IV), the receiver is responsible in feeding back other indicators on the instantaneous network dynamics. In order to exploit the advantage of estimating network dynamics at the receiver over at the sender [21], the receiver will either estimate the available bandwidth directly by observing the inter-arrival time, the size, and the loss of each packet, or convey more relevant measures like the congestion degree, whose high value indicates a higher probability of imminent packet loss. Our previous approach of the model-free LMS bandwidth estimator [22] belongs to the former approach. However, since the feedback of explicit available bandwidth $Rate_{\mathrm{estimate}}$ might hinder the multicast extension of RCCM, we adopt the latter approach, i.e., the congestion degree approach in this work. In calculating the congestion degree indicator at the receiver, we utilize the bundled feedback number from the sender by combining packets with the same feedback number into a group. Note that this group of packets has been transmitted with the same rate since the sender changes its rate only after receiving the feedback. Thus, the receiver can evaluate the congestion degree employing the sliding-window smoothing approach. For this task, the ratio of receiver's receiving rate over the estimated sender's transmission rate (incorporating the inter-arrival time variation of packets) is averaged and smoothed over the same feedback number group. Then, in the feedback report, the receiver transmits the congestion degree $\text{Cong}_{\text{degree}}$ and the estimated $\text{RTT}_{\text{estimate}}$, along with the feedback number, the feedback type, and other acknowledgments (including the packet loss feedback via NACKs).

The RCCM rate adjustment procedure at the server is described as follows. The sender starts to transmit the packet with an initial rate. A scheme, which maintains the compatibility with TCP's slow start, is required to set the initial sending rate properly. However, at this stage, we have set this to 32 kbits/s, which is within the range of the TCP initial window. Then, in response to subsequent feedback, the sender will first calculate $Rate_{\text{estimate}}$ based on $Cong_{\text{degree}}$. Then, it will go through the AIMD constraint to impose TCP-friendliness and video awareness. By tightening AIMD constraint, we can force RCCM to mimic TCP's AIMD behavior. Note that RCCM manages RTT in a TCP-like way. However, by loosening the AIMD constraint, we can mitigate TCP's inherent oscillation behavior in order to match the input video.

1)

$$\begin{aligned} &Rate_{\text{estimate}} \\ &\quad = Rate_{i-1} \times (Cong_{\text{degree}} ? (1 + C_{\text{CD}}) \,:\, 1 \,:\, (1 - C_{\text{CD}})). \end{aligned} \tag{3}$$

2) If (no packet loss)

$$\begin{aligned} Rate_{\text{AI}} &= Rate_{i-1} + \text{PacketSize}_{\text{average}}/\text{RTT}_{\text{estimate}}. \\ Rate_i &= \max(\min(Rate_{\text{estimate}}, Rate_{\text{AI}} \times (1 + C_{\text{AI}}), \\ &\qquad Rate_{\text{max}}), Rate_{\text{AI}} \times (1 - C_{\text{AI}})). \end{aligned} \tag{4}$$

3) If (packet loss)

$$\begin{aligned} &Rate_{\text{MD}} \\ &\quad = \beta \times Rate_{i-1}. \\ &Rate_i \\ &\quad = \min(\max(Rate_{\text{estimate}}, Rate_{\text{MD}} \times (1 - C_{\text{MD}}), \\ &\qquad Rate_{\text{min}}), Rate_{\text{MD}} \times (1 + C_{\text{MD}})). \end{aligned} \tag{5}$$

In (3), C_{CD} is the control parameter for congestion degree and it implies the adoption of two separate threshold bounds for $Cong_{\text{degree}}$. That is, based on the two thresholds on $Cong_{\text{degree}}$, $Rate_{\text{estimate}}$ is updated in three ways: 1) being increased by a percentage of C_{CD} (for lower range of $Cong_{\text{degree}}$); 2) remaining the same (for intermediate range of $Cong_{\text{degree}}$); or 3) being decreased by a percentage of C_{CD} (for higher range of $Cong_{\text{degree}}$). Basically it adopts the same adaptation structure of TCP Vegas [32], which utilizes the difference of expected and actual sending rate instead of $Cong_{\text{degree}}$. In (4) and (5), $Rate_{\text{max}}$ and $Rate_{\text{min}}$ are the desired maximum and minimum rates, respectively. $Rate_{\text{AI}}$, $Rate_{\text{MD}}$, C_{AI}, and C_{MD} denote the additive increase (AI) and the multiplicative decrease (MD) rates and the associated control parameters, respectively. Also, β is currently set to 0.5 by following TCP's AIMD behavior. One can set C_{AI} and C_{MD} smaller than C_{CD}, anticipating that smaller C_{AI} and C_{MD} will filter out a rather fast C_{CD} effect. However, C_{MD} can be set more flexibly, since the default MD behavior is dropping the rate to a half. This combination is used to keep the short-term AIMD fairness with the TCP traffic. On the other hand, increasing C_{AI} and C_{MD} can lead to the long-term fairness with TCP. However, as the AIMD constraint get relaxed further, it may resort to the TCP-model equation eventually. Thus, by controlling parameters C_{CD}, C_{AI}, and C_{MD}, RCCM can be sensitive to the TCP-friendliness of preferred timescale. Also, it can adjust the rate in a video friendly fashion by relaxing C_{AI} and C_{MD} temporarily if $Cong_{\text{degree}}$ allows for it. Finally, by using this rate, the packet is sent with the following scheduling:

$$T_j = T_{j-1} + \text{PacketSize}_j / Rate_i. \tag{6}$$

III. Rate- and Delay-Controlled Video Compression and Transmission

A. *Matching Variable Frame-Rate Video Encoder to the Available Bandwidth with Delay Control*

The video encoder needs a rate controller capable of quickly responding to the update of the available bandwidth. However, the complexity and the execution delay of the video encoder make almost impossible adjusting to the available bandwidth in a micro-scale manner. The short-term (instantaneous) fluctuation of the TCP-friendly flow should be smoothed in advance by the encoder/decoder network de-jitter buffers. A longer timescale variation can be successfully modeled into time-varying constant bit rate (CBR) channels (including the feedback VBR and the renegotiated CBR channel) [27]. One may think of a layered or aggregated scalable coding scheme to tackle the bandwidth fluctuation problem in a more systematic manner. However, switching between layers of stream is a difficult task even for aggregated (i.e., simulcast) stream and it is out of our scope. Thus, in our approach, an adaptive temporal layering of video is exploited based on the variable frame-rate encoder for H.263+ [23]. In Section IV, we will extend this variable frame-rate feature to create a multi-class layered video stream primarily along the temporal direction.

While most rate control algorithms examine rate allocation at the macroblock (MB) layer, the variable frame-rate method proposed in [23] handles it from the frame layer. Although technically challenging, it yields a greater degree of control over the temporal quality and the bit budget. Also, the quantization parameter (QP) control, which is the MB-layer rate control in H.263+ test model TMN10 [3], has been incorporated as a component in [23]. This variable frame-rate video allows us to select frames to be encoded and allocate the optimal rate for the selected frames according to a specific cost criterion (in terms of the available bandwidth and the underlying motion of the coded video) with a low computational complexity and a short latency. To achieve this, we consider a frame-layer rate-distortion (R-D) model with respect to the averaged QP of all MBs in each frame. To be more specific, a quadratic rate model and an affine distortion model are employed, and the MB-layer results are used to determine the coefficients of the frame-layer R-D model, keeping the computational complexity negligible. Based on the frame-layer R-D model, we can estimate the anticipated distortion of the current frame. Note that the anticipated distortion increases when fast motion change occurs or when the channel bandwidth decreases suddenly. If the spatial quality goes below a tolerable level due to fast motion change or sudden bandwidth decrease, we gradually tradeoff the temporal quality in preference to the spatial quality by skipping frames in order to reduce the flickering artifact. By adopting this scheme, we can avoid the abrupt change of the encoding frame-rate and improve the overall visual quality accordingly.

B. Video-Transmission Buffer Smoothing for End-to-end Delay Control

The video encoder takes the major role in available bandwidth adaptation to provide graceful quality degradation. However, to help the TCP-friendly rate adaptation, a video smoothing (or rate shaping) technique has to be utilized to control the end-to-end delay [16], [28], [29]. The end-to-end delay consists of the encoder delay for wait and processing, the channel delay for the transmitter/receiver buffer and the packet transmission, and the decoder delay for data-wait, display-wait, and processing [4]. Among these, since the channel transmission delay is not within end- terminal's control and receiver-side buffering is minimally allowed due to low-latency constraint, our focus is on the minimization of the transmitter buffer delay. As a comparison, inside H.263+, a hypothetical reference decoder (HRD) buffer model is used [2]. Note that H.263+ video is originally designed with low-latency CBR video conferencing in mind. It employs a buffer of pre-negotiated size. The buffer is initially empty and examined at every frame-interval for the buffer level control. Assuming a constant delay channel, immediately after removing data the buffer occupancy must be less than a pre-set level. This scenario, however, does not hold for Internet video because of dynamically varying channel.

Hence, in our approach, a video-smoothing scheme is absorbing the fluctuation of the available bandwidth. First, the instantaneous available bandwidth $Rate_i$ is increased or decreased by the amount, which approximates the slope by nth order polynomial. At the same time, it is adjusted according to the transmission buffer occupancy. Note that the buffer control is linked to the transmitter buffer instead of the receiver buffer due to the low- latency assumption. Thus, the available bandwidth $Rate_i$ obtained by RCCM is adjusted to another rate called $TargetRate_i$ as

$$\begin{aligned} &TargetRate_i \\ &\quad = Rate_i + C_{\text{slope}} \times 0.5 \times (Rate_i - Rate_{i-1}) + C_{\text{buffer}} \\ &\quad\quad \times (\text{Buffer}_{\text{TxRef}} - \text{Buffer}_{\text{TxLev}}) \end{aligned} \tag{7}$$

where C_{slope} and C_{buffer} are the control parameters for smoothing, respectively. Currently first-order (linear) slope adjustment is used and is represented by 0.5 (i.e., area of triangle). A conservative approach to bandwidth consumption is taken with a hope to reserve margin for future abrupt fluctuation. Although not explicitly shown in (7), different weights C_{slope} are used for the increasing and the decreasing phases of the available bandwidth curve, respectively. That is, smaller C_{slope} of 0.5 is utilized for the increasing phase and larger C_{slope} of 1.0 is used for decreasing phase. To minimize the end- to-end delay, the transmission buffer level has to be kept as low as possible. However, to prevent underflow, we should set a reasonable desired buffer level $\text{Buffer}_{Tx\text{Ref}}$. The employed packetization strategy and the current transmission rate affect the choice of the desired buffer level. Thus, in (7), a change of $TargetRate_i$ is implemented based on the expected transmitter buffer level $\text{Buffer}_{Tx\text{Lev}}$ after the frame-skip (at the anticipated finish time of frame i). Also C_{buffer} is assigned to 0.7 currently.

Next, it is further adjusted by a weighted temporal smoothing of $TargetRate_i$ as follows:

$$TargetRate_i = \mu \times TargetRate_i + (1-\mu) \times TargetRate_{i-1} \tag{8}$$

where μ represents the weighting control factor for smoothing. The main reason behind this is to help the employed variable frame- rate encoder to coordinate the quality consistency since the encoder's quality regulation can experience difficulty when it has to handle too much fluctuation. Thus, the weight plays a role on the tradeoff between the buffer regulation (smoothing) and the quality adaptation. Keeping it close to one will control the buffer tightly, sacrificing the consistency of video quality. On the contrary, if we lower it, it enables the deployed video encoder to keep the quality more consistent at the cost of buffer regulation.

Finally, when there is packet loss, the sender will initiate a media-level error recovery process. After being informed by the NACK, the sender might initiate among the full intra-frame refresh (FIR), the adaptive intra-frame refresh (AIR), and the standard/enhanced reference picture selection (RPS, a.k.a., NewPred) [2]. For the sender, this action is interpreted as a short-time bandwidth increase. However, it is very challenging to accommodate the bandwidth demand since it is currently undergoing the RCCM rate drop. Thus, for the media-level error recovery in response to NACK, a scheme synchronized with the receiver status (to be described in Section IV) is a candidate solution. By keeping update on the status of the receiver, the sender can possibly reserve bandwidth margin for imminent peak traffic.

IV. Media-Aware Packetization and Its Interation with Congestion Control

A. Media-Aware Packetization and Recovery

Due to the low-latency requirement, the re-transmission is not usually applicable in interactive Internet video. For typical multi-hop Internet connections, an RTT of around 300 ms or more is common, which prevents the use of re-transmission. However, it is important to note that with some provision that relaxes the re-transmission timing (e.g., large decoder buffer), the re-transmission is an essential component in most Internet streaming applications (although not real-time interactive). In a typical environment, packets arrive at the receiver often with a substantial packet loss rate, surmounting up to 20% or more for many inter-continental connections [20]. This justifies the use of a proactive technique like forward error correction (FEC), on which several interesting ideas are recently being reported [5]. However, in our case, the low bandwidth of the Internet modem prevents the full adoption of FEC. The adoption of some error resilient options helps to limit error propagation. Schemes such as intelligent intra-MB refresh, independent segment decoding (ISD), and data partitioning (DP) are applicable regardless of feedback [7]. Of course, feedback-based schemes such as AIR and RPS have more potential, although their applicability is limited. The adoption of enhanced error resiliency options will prevent the catastrophic decoding failure that renders the error concealment almost useless, since a large portion of a video frame is missing due to packet loss. In this work, TCON model adopted by the H.263+ test model [3] is modified and applied. For a missing MB, if the motion vector of the corresponding MB in the previous frame is relatively small, simple replication is applied. If the lost motion is fast, replication can result in jagged and abrupt edge discontinuities. In this case, spatial error concealment based on either simple spatial interpolation or DCT coefficient interpolation [22] is utilized.

In this work, error resiliency option features for loss-resilient H.263+ Internet video are integrated with RFC2429, i.e., the RTP ALF packetization syntax for H.263+ [11]. Although error detection is quite straightforward in packet video, a proper packetization that establishes a correspondence between the packet and the frame structure is key factor in proactive error recovery. To keep a good balance between transmission efficiency and error-resiliency, it is desirable to divide a coded frame into an appropriate number of packets proportional to its importance [6]. However, the combined overhead of RTP/UDP/IP is 40 or more (12+/8/20) bytes per packet, inducing the use of a larger packet—Without the path maximum transfer unit (MTU) discovery mechanism [14], a reasonable limit is 576 bytes—for efficiency. This implies that multiple frames in a packet are preferable in terms of overhead efficiency. In one extreme, the straightforward packetization scheme would be a "one packet, one frame" technique, leading to an overhead of 40 bytes/frame (but probably with fragmentation). In the other extreme, a packet for one GOB or slice can be selected in the presence of high packet loss. Considering this tradeoff, a three-class ALF packetization scheme is chosen in our approach to provide the unequal error protection capability, while keeping the overhead within a reasonable limit. It is also an effort to leverage the future DiffServ architecture [12] by employing the DS byte marking [13].

At the topmost class (Class 1), frames such as I-frame and master P-frames of RESCUE [8] (in a frequency controlled by the packet-loss dynamics) are packetized employing a packet per GOB. The largest overhead can be justified by the importance of these frames in H.263+ video. Of course, the ISD mode of H.263+ will ensure that each packet (GOB or slice) is independently decodable. If the network delay is within a reasonable bound (e.g., in case of expedited forwarding (EF) support of DiffServ [12]), ARQ can be tried with the highest priority to retrieve lost packets. The decoder relies on error concealment temporarily to cover the loss up until packet recovery. Then, multiple GOBs are packed into a single packet at the middle class

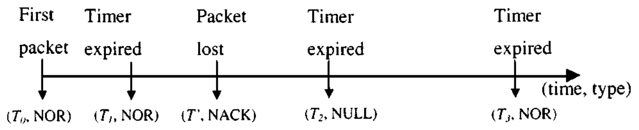


Fig. 2. Feedback mechanism adopted in the receiver.

(Class 2), which corresponds to the normal P-frames of H.263+. For this, one may try a simple interleaving scheme by packing all even and odd GOBs into different packets, making the packetization friendly to error concealment techniques [6]. Finally, for the lowest class (Class 3), the smallest number of packets possible (due to the MTU consideration) for a frame will be used to minimize the overhead. These portions of streams correspond to the least important frames such as B-frame portion of PB-mode, which can be discarded without any error propagation.

The above differentiation rule is applied even to the header loss control. The picture header contains information relevant to the whole frame and appears once per frame. If this picture header is put into only the first packet of frame and gets lost, the decoder may be stuck in the worst case. Fortunately, the picture header does not change much except for several fields like the temporal reference (TR) field, so that it may be recoverable from those of past frames. However, changes in picture sizes, the picture coding mode, the selected reference picture, or optional coding modes are critical. To mitigate this problem, RFC2429 allows the addition of a redundant copy of the picture header in the payload header of each packet. This mechanism can be selectively applied in conjunction with the three-class layering.

B. Media-Aware Packet Recovery Interacting with RCCM

In RCCM, special attention is paid to the reduction of ACK frequency. We want a scheme that is capable of flexibly coordinating data and control packets, achieving efficient utilization of network resource. Thus, multiple types of ACK are hierarchically organized in relation to the three-class layer. As mentioned before, RTCP is utilized for RCCM feedback. Our design for ACK/NACK is extension of the work (i.e., the delayed ACK plus NACK) in [30]. Also, when designing the ACK interval, values such as RTT and the conveyed rate are combined together to achieve an efficient tradeoff [31].

Let us illustrate the feedback mechanism in Fig. 2. Two types of ACK and three types of NACK are considered at this stage. First, the receiver sends a feedback when its timer expires. This timer is based on $\mathrm{RTT}_{\mathrm{estimate}}$ in (2). That is, this ACK_NOR feedback occurs in every interval of K times $\mathrm{RTT}_{\mathrm{estimate}}$ ($K = 1$ for Fig. 2). Initially at $T = T_0$, the receiver receives the first packet and initializes its timer with $\mathrm{RTT}_{\mathrm{sample}}$ ($= \mathrm{RTT}_{\mathrm{estimate}}$). Also, at the same time, the receiver sends the ACK_NOR feedback. After that, the receiver will send out this ACK_NOR feedback unless it encounters a lost packet during an interval of $\mathrm{RTT}_{\mathrm{estimate}}$. When the receiver experiences a missing packet (i.e., the gap in the RTP packet sequence number), it sends out another feedback named NACK (differentiated based on the three-class) after a predefined waiting period T_{wait}. The waiting period should be set adaptively to handle out-of-order packets. Since fairness with TCP is highly dependent on the response to packet loss, special attention is required. Setting T_{wait} to zero makes the proposed scheme to behave more conservatively compared to TCP, because it does not deploy the retransmission timeout module of TCP. So, it is not necessary to decrease the rate for every packet loss. For the case of multiple packet loss within a timeout interval, the sender will restrict the sending rate decrease with another control parameter $N_{\rm decrease}$. The actual implementation requires the modified execution of (3), only if there is NACK of a packet loss and the number of NACKs within timeout interval is not greater than $N_{\rm decrease}$. Thus, by limiting the rate decrease in an interval, the proposed scheme has a comparable module for TCP fast recovery. Finally, if it experiences the packet loss during the timeout interval, the receiver sends the ACK_NULL feedback instead of ACK_NOR at the timeout instance. At $T = T_2$ in Fig. 2, the receiver sends the ACK_NULL feedback because there is a packet loss at $T = T'$. The sender does not increase its rate in response to this feedback but uses its feedback number and RTT$_{\rm estimate}$.

At the receiver, coordinated error recovery based on the three- class media packet can be conducted with intelligent RTP header processing. By differentiating the error recovery response to each layer, more timely and bandwidth-efficient error recovery is possible. Before describing it, let us clarify the issue of "how to detect the class of lost packet." If a regular and fixed layering is employed, the solution may be trivial. However, with the proposed flexible layering, it relies on the assumption that we can infer the lost packet class from the classes of precedent or subsequent packets successfully received. However, if we cannot figure out the class, the receiver may classify the lost packet into Class 2, leaving the responsibility to the sender.

Once the receiver identifies the class of the lost packet, the media-aware packet recovery module will interact with RCCM as follows. For Class 1, unless the FEC technique such as the parity packet is implemented, RCCM will NACK this loss promptly. Then, the sender has the responsibility to check the relevant timing issues and make a decision on retransmission or refreshing. For the retransmission case (within a reasonable network delay or RTT$_{\rm estimate}$), the sender retransmits a stored copy upon the reception of NACK to assure the synchronization between the sender (encoder) and the receiver (decoder). In addition, it has to track the propagation of errors and thus dictate the required reaction of the encoder, which includes refreshing by the I-frame, or initiation of the AIR/RPS mode. Currently a simple error-tracking scheme is incorporated into the proposed system, enabling any kind of media-level error recovery among FIR, AIR, and so on. All the lost packet reports are interpreted at the sender to get synchronized with the decoder. In fact, a GOB-based loss propagation model is adopted in our system to estimate the decoder status from the delayed NACK (one half of RTT at the minimum). For Class 2, everything will be similar except that the sender does not have to retransmit the lost packet. Again the tracking of error propagation will be the major task to be taken care of. Finally, since Class 3 packets are safely ignored, notification will occur for RCCM's response to packet loss, but only after some delay.

V. Quality Recovery from Degraded Video

The main issue in post-processing implementation is a flexible performance at the cost of a reasonable complexity. For spatial recovery, the removal of blocking and ringing artifacts is of

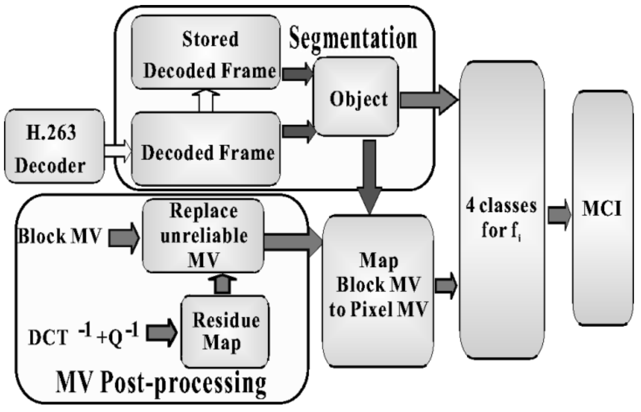


Fig. 3. Block diagram of proposed FMCI implemented as a video post-processing unit.

top importance. While some are designed only for deblocking, other approaches attempt to restore all major artifacts in a unified framework by adopting image restoration schemes. However, restoration approaches generally demand a higher complexity. By formulating the problem with a robust estimation framework, a sub- optimal solution was pursued by using a nonlinear filtering technique of low complexity [24]. For temporal recovery, one may consider the filtering-based motion-compensate interpolation [26] if a fast (more specifically, VLIW) CPU is available. Another interesting solution is the FMCI schemes for H.263+ decoder [25]. The FMCI scheme considers the deformable mapping of the block-based motion field to the pixel-based motion field. The decoder can perform frame interpolation using motion vectors from the encoder, freeing the decoder from additional motion search, which results in a significant complexity reduction in comparison with that in [26]. From experimental results shown in [25], visual quality of the decoded video is improved with the flexibility of inserting as many frames as needed at any time in-between.

The FMCI scheme is implemented in the decoder as a video post-processing unit as shown in Fig. 3, which is cascaded with the standard H.263+ decoder without changing the syntax because the TR required by FMCI is already defined in the header. FMCI consists of three main units: motion preprocessing, segmentation and MCI (motion-compensated frame interpolation) prediction. The motion-preprocessing unit is used to modify the block-based motion field to achieve a better frame interpolation result. Once the post- processed motion field is obtained, we map it to the pixel-based motion field for MCI prediction. We adopt the deformable (i.e., affine) block transform to map the block-based motion field to the pixel-based motion field. The second unit of FMCI performs object segmentation of decoded frames, which is useful in providing the moving object location to MCI. We do not use any complicated segmentation procedure, partly because we do not want to increase the computational load in the decoder and partly because the segmentation result is rough due to the use of the block-based motion field only. For the third unit, classification of regions into stationary (*SB*), covered (*CB*), and uncovered backgrounds (*UB*), and the moving object (*MO*), commonly used in standard MCI, is adopted here. Two extensions of FMCI can further enhance the overall performance. One is the adaptive frame skip (AFS)

TABLE I
THROUGHPUT RATIO COMPARISON OF RCCM AND VARIOUS TCP SCHEMES ACCORDING TO N_{decrease} IN THE MEAN AND THE STANDARD DEVIATION FROM 24 SIMULATIONS

C_{CD} / C_{AI} / C_{MD} (%)	$N_{decrease}$	Tahoe		Reno		NewReno		SACK	
		Avg	SD	Avg	SD	Avg	SD	Avg	SD
* / 0 / 0	1	0.91	0.30	2.03	1.16	0.90	0.31	0.74	0.24
	2	0.67	0.24	1.15	0.59	0.68	0.23	0.58	0.20
	3	0.64	0.21	1.07	0.53	0.66	0.22	0.56	0.19
	4	0.65	0.21	1.03	0.52	0.65	0.21	0.73	0.22

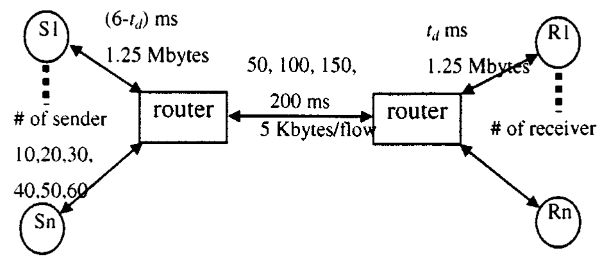


Fig. 4. NS simulation topology for RCCM versus TCP.

scheme applied at the encoder. The other is the hybrid frame interpolation (HFI) scheme that incorporates both frame repetition and FMCI at the decoder [25]. By adopting AFS, the frame-interval can be chosen according to the performance of FMCI. By using HFI, a simple rule can help to select either FMCI or the frame repetition dynamically. For a more detailed description of FMCI, we refer to [25]. Finally, it is worth mentioning that a variation of FMCI is applicable to error concealment. While FMCI interpolates frames bi-directionally from previous and current frames, error concealment predicts the missing GOB from previous frames. With the GOB interleaving packetized scheme [6], we can use motion fields of above and below the missing GOB as the vertex vectors, and conceal missing GOB from mapped patches of the previous frame. This approach performs better than TMN10 TCON, since more spatial GOB correlation is utilized and the affine FMCI mapping results in less blocky artifacts.

VI. System Integration and Experimental Results

A. NS Simulation of RCCM for TCP-Friendliness

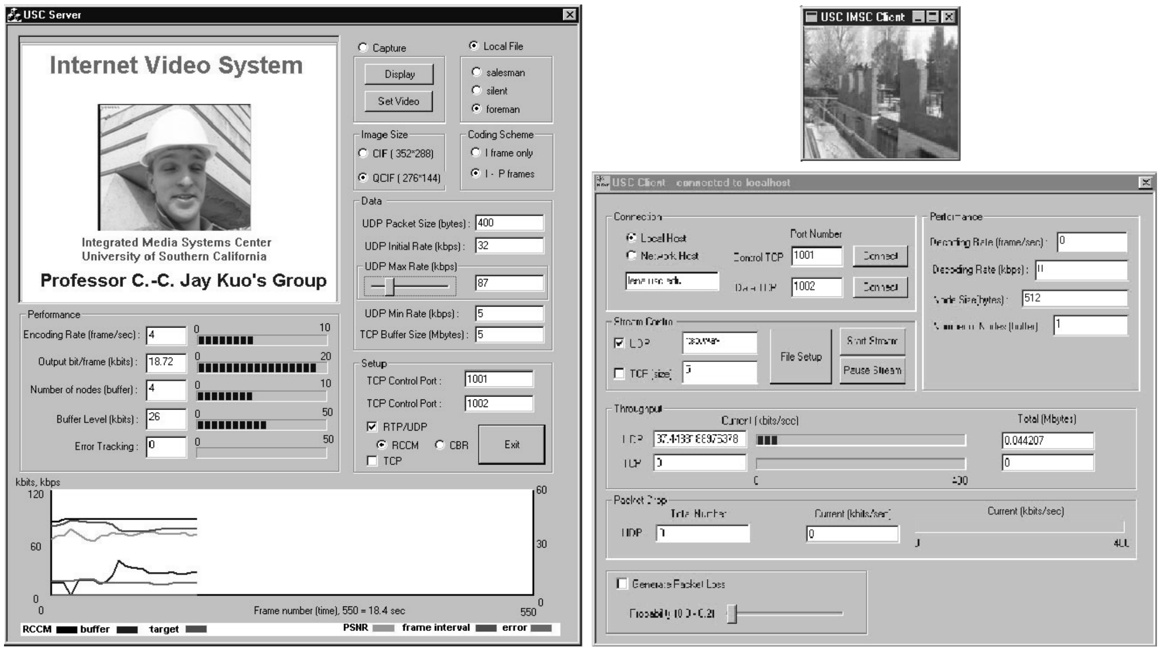
The TCP-friendliness of the proposed RCCM is verified by using the NS [34]. The simulation topology and parameters are given in Fig. 4. We vary the total number of flows between 10 $\sim$ 60 with 10 flows gap, resulting in 6 sets of different flow numbers. Each flow takes a 5 kbytes of bandwidth with a packet size of 100 bytes, which moves from each sender to the receiver of the same index. The buffer size at the bottleneck link is set to four times of the bandwidth-delay product, since it allows NS simulation to exhibit the full range behavior of TCP (i.e., slow start, congestion avoidance, fast retransmission, and fast recovery). Also, to validate the wide range RTT support, four delay situations in the bottleneck link (i.e., 50, 100, 150 and 200 ms) are evaluated. The sum of either-end side delay is 6 ms, while the delay from each sender to the first router is a uniformly distributed random value t_d. The link bandwidth between the router and the end terminals (either the sender or the receiver) is set to 1.25 Mbytes, respectively.

Four existing TCP algorithms (i.e., Tahoe, Reno, NewReno, and SACK TCP) are compared with the proposed RCCM. Table I gives the throughput ratio comparison of RCCM over TCP according to N_{decrease}. The extreme case of sticking to the AIMD short-term friendliness is evaluated in this case by setting C_{AI} and C_{MD} to zero. Each item represents the average and the standard deviation (SD) of the throughput ratio, which is calculated over 24 simulations (four different delays multiplied by six different flow numbers). Each simulation length is set to 150 s. Except for rather outdated Reno TCP, RCCM takes less bandwidth than that of TCP regardless of N_{decrease}. This confirms that the proposed algorithm behaves more conservatively than TCP in general. Even for Reno TCP, the throughput ratio is around 1 for $N_{\text{decrease}} > 1$.

In Table II, the throughput ratios between RCCM and other TCPs are provided for several combinations of control parameters C_{CD}, C_{AI}, and C_{MD}. Four different configurations are attempted to evaluate the relation among these parameters. Two sets (2% and 5%) are used for both C_{AI} and C_{MD}, while again two different values (5% and 10%) are tried for C_{CD}. First, when the AIMD behavior is tightly kept by small C_{AI} and C_{MD} of 2%, the throughputs of RCCM are increased compared to those of Table I. Also, the choice of C_{CD} plays an effective role in controlling the RCCM traffic as shown in Table II. These results imply that, by keeping a tighter AIMD constraint, we can easily control the RCCM traffic without hurting the TCP friendliness. Next, let us interpret the results when the AIMD constraint is further relaxed to 5% (i.e., C_{AI} and C_{MD} to 5%). We would like to verify that setting C_{AI} and C_{MD} high leads to the long-term fairness with TCP. Results for 5% C_{CD} in Table II shows that RCCM throughput is decreased compared to those of Table I. On the contrary, results for 10% C_{CD} shows that RCCM throughput is the highest among all configurations. This means that when we relax the AIMD constraint, it becomes difficult to regulate the RCCM traffic in the sense of short-term TCP-friendliness. Also more precise estimation and control on $Cong_{\text{degree}}$ and C_{CD} is highly demanded and eventually RCCM have to resort to the TCP model equation. However, based on the above results, we can verify that, by controlling parameters C_{CD}, C_{AI}, and C_{MD}, RCCM can maintain TCP-friendliness in any preferred timescale.

TABLE II
THROUHPUT RATIO COMPARISON OF RCCM AND VARIOUS TCP SCHEMES ACCORDING TO CONTROL PARAMETERS IN THE MEAN AND THE STANDARD DEVIATION FROM 24 SIMULATIONS

C_{CD} / C_{AI} / C_{MD} (%)	$N_{decrease}$	Tahoe		Reno		NewReno		SACK	
		Avg	SD	Avg	SD	Avg	SD	Avg	SD
5 / 2 / 2	1	0.99	0.17	2.94	0.76	1.03	0.22	0.75	0.14
	2	0.66	0.12	1.82	0.49	0.68	0.12	0.54	0.11
10 / 2 / 2	1	1.20	0.18	3.11	0.60	1.27	0.27	0.88	0.15
	2	0.76	0.12	1.63	0.39	0.75	0.13	0.62	0.11
5 /5 /5	1	0.79	0.16	2.03	0.75	0.85	0.17	0.66	0.14
	2	0.62	0.13	1.15	0.46	0.67	0.13	0.53	0.11
10 / 5 / 5	1	1.41	0.23	3.53	0.53	1.75	0.38	1.03	0.17
	2	0.72	0.12	1.70	0.39	0.81	0.15	0.57	0.08



Capture [✓ Local File]
salesman / silent / [✓ foreman]
Image Size: CIF (352*288) / [✓ QCIF (276*144)]
Coding Scheme: I frame only / [✓ I - P frames]
Data
UDP Packet Size (bytes) : 400
UDP Initial Rate (kbps) : 32
UDP Max Rate (kbps): 87
UDP Min Rate (kbps) : 5
TCP Buffer Size (Mbytes) : 5
Setup
TCP Control Port : 1001
TCP Control Port : 1002
[✓ RTP/UDP] [✓ RCCM] CBR
TCP
Performance
Encoding Rate (frame/sec) : 4
Output bit/frame (kbits) : 18.72
Number of nodes (buffer) : 4
Buffer Level (kbits) : 26
Error Tracking : 0

Connection: [✓ Local Host] / Network Host
Port Number: Control TCP 1001; Data TCP 1002
Stream Control: [✓ UDP]; TCP (size) 5
Performance: Decoding Rate (frame/sec) : 0; Decoding Rate (kbps) : 0; Node Size(bytes) : 512; Number of Nodes (buffer) : 1
Throughput: Current (kbits/sec) UDP 37.4133166975378, TCP 0; Total (Mbytes) 0.044207, 0
Packet Drop: Total Number UDP 0; Current (kbits/sec) 0
Generate Packet Loss
Probability (0.0 - 0.2)

Fig. 5. Snapshot of the proposed integration system and its interfaces.

B. Integrated Internet Video Streaming System and Its Interface

To evaluate the performance of the proposed integrated system, a real-world video streaming application is built by the software implementation (C++) on the Windows NT box. Modules shown in Fig. 1 have been developed and combined through the multitasking threads. First, a real-time H.263+ encoder with a variable frame-rate controller is implemented by extending the UBC H.263+ (version 3.2.0). For the RCCM module, an application level implementation is developed on top of the Windows sockets to realize the TCP-friendly transmission and error recovery. A server module encompassing the RCCM sender and the video encoder is constructed in a powerful dual NT machine. Then, a separate client module, consisting of the RCCM receiver, the real-time H.263+ decoding and the limited post-processing enhancement unit, is implemented as shown in Fig. 5. Due to the limited power of the software implementation, some modules such as the H.263+ encoder and the motion-compensated frame interpolator do not operate in 30 fps real-time. For example, the H.263+ encoder can encode up to 5 fps with fully motion-compensated P-frames. Fortunately, the proposed system is evaluated over 56 Kbps Internet modem connection between a streaming server attached to USC LAN and an Internet modem client. Under such an Internet modem connection, where the video traffic can take up to 40 kbits/s (minus the bandwidth for audio and modem loss), the frame-rate limit for quality QCIF (176×144) video is around $6 \sim 9$ fps. Thus, to evaluate the proposed integration, a frame rate of 5 fps is an acceptable choice.

The real-time encoding and streaming of H.263+ video is performed under the following network restrictions and other limitations, which is reflected in the end-to-end delay analysis shown in Fig. 6. The timing including encoding delay (excluding the encoding wait time), transmission waiting delay,

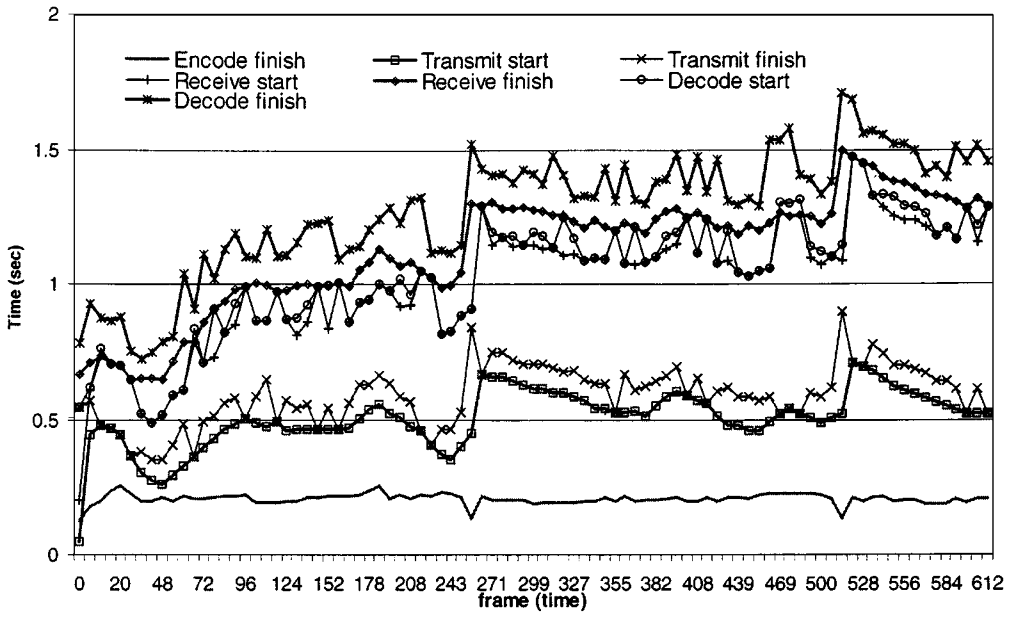


Fig. 6. Typical end-to-end timing from 40 kbits/s Internet modem scenario (20 s for 30 fps source).

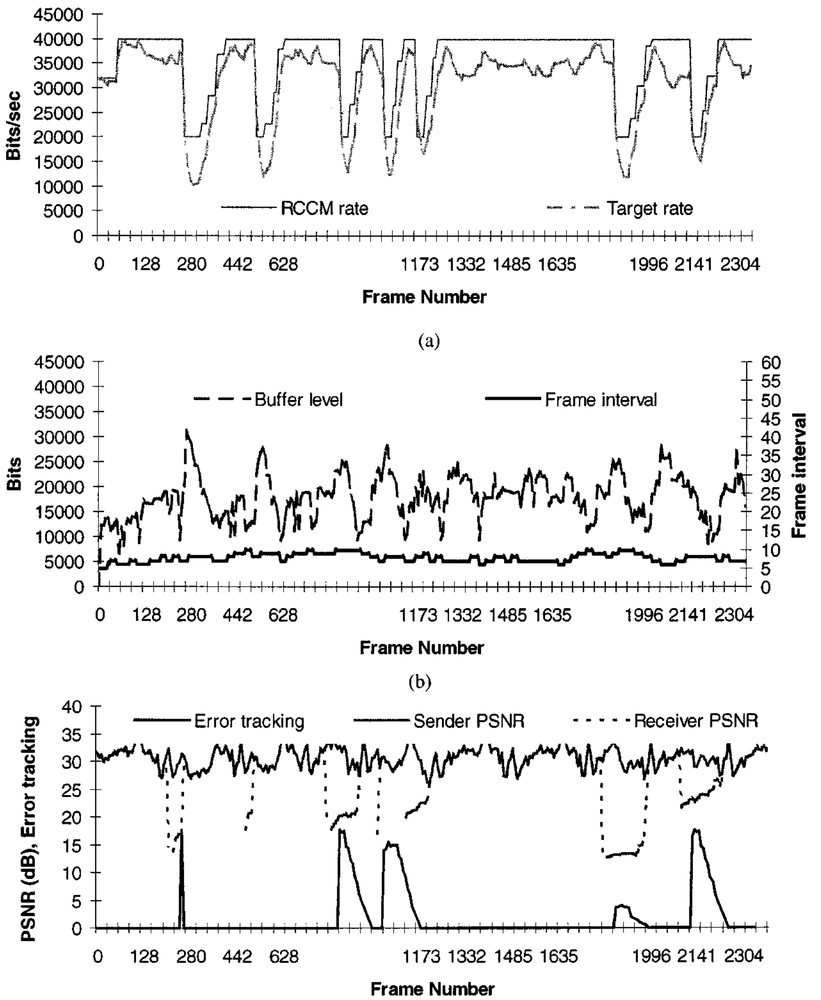


Fig. 7. Real-time performances with a variable frame-rate encoding and error tracking (256 frames of "Foreman" sequence is circulated with forced I-frame insertion). (a) RCCM rate and its adjusted target rate. (b) Buffer level at the start of encoding and the chosen frame-interval. (c) PSNR values and the error-tracking index.

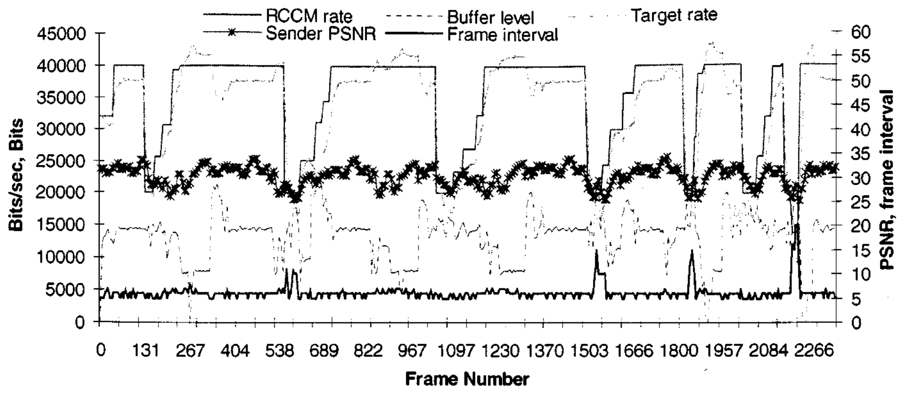


Fig. 8. Real-time performances in a constant frame-interval case for comparison.

transmission delay, and decoding delay is depicted. From Fig. 6, we can verify that the integrated system can maintain the end-to-end delay within 2 s over tens of minutes. Note that only partial results are shown in Fig. 6 for the presentation purpose. Among delays, transmission delay takes up the highest portion of up to 1 s (i.e., the gap between the transmit start/finish and receive start/finish), which is beyond our control. Also, the video encoder takes up around 200 ms delay and decoding takes on the average around 250 ms reflecting the asymmetric processing power of the server and client. Lastly, the buffering and transmission at the server causes varying delay between 300–700 ms.

C. Matching Video Encoder to Available Bandwidth (Internet Modem)

To evaluate the matching capability of the variable frame-rate encoder to the available bandwidth, we have fed the standard video in YUV format repetitively. Among several sequences tested, results for "Foreman" sequence are presented because it contains both talking head and fast motion together. In order to mimic the real-time capturing situation, the feeding has followed the original capture timing. The simulation is started by using the RTSP connection setup from the client module with an initial rate of 32 kbits/s. In Fig. 7, the performances in terms of the RCCM rate, the adjusted target rate, the buffer level at the start of encoding, the PSNR values of the sender and the receiver, the frame interval, and the error-tracking index are plotted, respectively. From RCCM rate curve of Fig. 7(a), one can easily notices that matching Internet's available bandwidth is a challenging task due to the fluctuation after packet loss. In simulation, oscillation behavior of RCCM rate is upper limited by the preset $Rate_{\max}$ (40 kbits/in this case). Also, we can temporarily observe the stepwise granularity in the increasing phase of the RCCM rate, since the RCCM rate is changed by the interval of $\mathrm{RTT}_{\mathrm{estimate}}$. Typical values for $\mathrm{RTT}_{\mathrm{estimate}}$ are in the range among $150 \sim 1000$ ms, which is inferred as twice transmission delay of Fig. 6. By adjusting the RCCM rate to target rate, the proposed scheme manages the buffer level (i.e., eventually the end-to-end delay). The frame-interval variations depicted at Fig. 7(b) are always initiated at the instances of packet loss. Peaks in the buffer level is somewhat correlated with the periodic insertion of I-frames (in the cycle of 256 frames, but not at fixed frames).

TABLE III
PSNR COMPARISON OF VARIABLE AND CONSTANT FRAME-RATE ENCODING SCHEMES BASED ON 10 MINUTES OF REAL-TIME VIDEO TRANSMISSION OVER INTERNET MODEM

	RCCM (Kbps)		PSNR (dB)	
	Average	Standard Deviation	Average	Standard Deviation
Foreman Variable Frame-rate	36.12	6.96	30.92	2.01
Foreman Constant Frame-rate	35.79	7.70	30.20	2.04

Then, to verify the gain of variable frame-rate encoding, the performance is compared to that of constant frame-interval case. Results provided in Fig. 8 are obtained utilizing the same system with the variable frame-rate result of Fig. 7 except for the frame-rate control scheme and associated fine-tunings. By comparing performance curves, we can observe that a very large frame skip is occurring in the case of constant frame-rate encoding. This large temporal gap is always noticeable to the viewers and perceived as a distinct temporal distortion. Also, the variable frame-rate scheme increases the average PSNR (comparison is based on the PSNR value at the sender to exclude the packet loss effect) by 0.7 dB as provided in Table III. Note that, to make the quantitative comparison fair, results of enough length of time (10 min) are compared with similar network fluctuation.

D. Media-Aware Packetization and Error Recovery (Internet Modem)

Among the three-class layered packetization described in Section IV-A, only two-layer packetization is deployed in Internet modem evaluation. That is, a layered packetization of the I/P-frames is implemented with some modification. At least one packet is generated for every GOB of the I-frame (Class 1) to simplify its error concealment and fast recovery. In fact, QCIF video around 40 kbps leads to one packet per GOB (9 packets for a frame in total) for almost every case. For intermediate P frames (Class 2), a varying number of packets are generated based on the adopted MTU. However, fragmentation is allowed only on GOB boundaries. As shown above in Fig. 6, the end-to-end delay in Internet modem situation is up to 1 s. This large delay implies that no retransmission is possible,

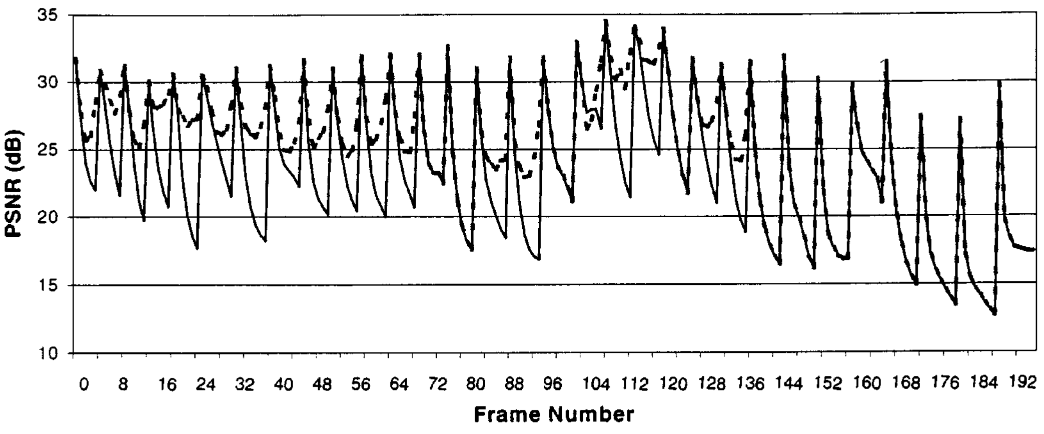


Fig. 9. PSNR comparison curve of FR and FMCI-HFI (dotted curve) for the "Foreman" sequence.

TABLE IV
PSNR PERFORMANCE OF PROPOSED FMCI-HFI VERSUS FR IN AVERAGED PSNR OVER 256 FRAMES

Averaged PSNR (dB)	FMCI-HFI (all frames)	FR (all frames)	Gain of FMCI-HFI (all frames)	FMCI-HFI (interpolated frames only)	FR (interpolated frames only)	Gain of FMCI-HFI (interpolated frames only)
Suzie	30.39	27.69	+2.70	29.44	26.27	+3.17
Foreman	24.26	22.84	+1.42	23.04	21.37	+1.67

and one has to devise sophisticated FEC scheme to protect the prioritized classes.

Leaving this as future task, our evaluation has focused on the validation of the designed ACK scheme. We depict, in Fig. 7, the error tracking managed at the sender based on NACKs. The sender keeps its record (including the packet sequence number, the associated frame-number and GOB number) till its reception is acknowledged or a pre-defined timeout is past. In response to each NACK, it tracks the lost portion (in GOB units) due to the packet loss in a frame. By assuming a simple error propagation pattern into subsequent frames, the impact of NACK to the latest to-be-encoded frame is calculated at the server. Note that a lot of packets corresponding to subsequent frames are either waiting in the transmitter buffer, being sent to the decoder, or decoded by the decoder. These on-the-fly packets are also subject to error propagation, unless refreshed by either I-frame or I-MBs. This eventually contributes to the PSNR gap of the sender and the receiver as shown in Fig. 7(c). For each packet loss, the PSNR gap appears first at the receiver/decoder right away and then, only after transmission delay, the sender is notified about the loss through RCCM feedback. It then starts to react in both rate and error aspects (i.e., reducing the rate and figuring out the best recovery scheme). This happens only after a large feedback delay of up to 1 s (corresponding to 30 original frames) as shown in Fig. 7(c). However, with our simple GOB-based error tracking, we can track the progress of error propagation at the sender. Take a careful look at the shape of the PSNR gap and that of error tracking index. If these two curves show similar shape, the error tracking can serve as a criterion by which the server decides the proper timing for refreshing (FIR at this stage). From Fig. 7(c), the curve for error tracking shows that a simple GOB-based loss propagation model tracks the error propagation effectively despite of the large feedback delay.

E. Evaluation of Temporal Quality Recovery Tools

The proposed FMCI enhances the motion smoothness obviously with an improved PSNR as shown in Fig. 9. It shows the post-processing gain in terms of PSNR for the "Foreman" sequence under the Internet modem evaluation. The PSNR of FMCI-HFI is indicated by the dotted line while that of FR is shown with a solid line. The periodical PSNR peaks observed are for the transmitted (noninterpolated) frames. Obviously the interpolated frame gives lower PSNR than the transmitted frame. But, the proposed FMCI-HFI provides a significant improvement over FR. In Fig. 9, overlapping period of two curves means that FMCI-HFI falls back to FR. This fallback usually happens when there is simple scene change or equivalent fast motion. For frames between the 150–200th of "Foreman" sequence, where the fast camera panning leads to a scene change, FMCI falls back to FR. As a quantitative comparison, we also summarize the PSNRs of FMCI-HFI versus FR in Table IV. The results are shown for two sequences "Foreman" and "Suzie," respectively. We can see the higher gain for relatively easy-to-interpolate "Suzie" sequence as expected. In addition to PSNR, we can observe a significantly visual improvement with less jitter in the playback. The pros of proposed FMCI is the low complexity since no motion search is required in the decoder unlike the traditional MCI. The disadvantage is that it has to buffer the frames to perform interpolation.

VII. Conclusion and Future Work

The integrated system and its associated components presented in this paper can be conceived as a total solution to various problems encountered in the context of VBR video transmission over the noncentralized error-prone best-effort Internet. The RCCM module with the available bandwidth estimator provides the bandwidth consumption level to minimize the packet loss in a TCP-friendly manner. The real-time variable frame-rate controller at the encoder responds by tailoring the stream to fit the available bandwidth and meet the end-to-end delay requirement. The media-aware packetization and recovery is achieved as a joint effort of the RCCM module and the media-recovery module. Last, the fast motion-compensated frame interpolation restores good visual quality by interpolating frames. The proposed system and its associated tools have been proven to provide much-enhanced video at the receiver via real-world Internet modem experiments.

Even though we covered as many aspects of feedback-based Internet H.263+ video streaming as possible, our work can only provide preliminary insights into diverse building blocks and solutions of Internet video. Some interesting future works are listed below. More quantitative study on the interaction of RCCM and video rate control is required. Switching of video application from the on-line video encoder to the off-line encoder followed by rate and/or error transcoding would be interesting. To explore the advantage of fully blown media-level error recovery, one can try the packet corruption model reported in [33] for enhanced error tracking while at the same time incorporate the additional bandwidth demand for error recovery into the video smoothing constraint. Finally, an evaluation of the proposed three-layer video over the DiffServ network environment may be conducted in the end-to-end QoS video context.

References

[1] *Packet-Based Multimedia Communication Systems,* ITU-T Recommendation H.323 Version 2, Jan. 1998.

[2] *Video Coding for Low Bit Rate Communication*, ITU-T Recommendation H.263 Version 2, Jan. 1998.

[3] *Video Codec Test Model Near-Term Version 10 (TMN10),* Q.15/SG16, Document Q15-D-65, Apr. 1998.

[4] R. Fryer and R. Lambert, "*Draft Delay Model for H.26L Proposed Codec Evaluation*,", ITU-T Q.15/SG16, Document Q15-C-32, Dec. 1997.

[5] B. Girod, K. Stuhlmuller, M. Link, and U. Horn, "Packet loss resilient Internet video streaming," *Proc. SPIE VCIP '99*, vol. 3653, Jan. 1999.

[6] S. Wenger and G. Cote, "Using RFC2429 and H.263+ at low to medium bit-rates for low-latency applications," in *Proc. Packet Video Workshop '99*, Apr. 1999.

[7] G. Cote, S. Shirani, and F. Kossentini, "Optimal mode selection and synchronization for robust video communications over error prone networks," IEEE J. Select. Areas Commun., submitted for publication.

[8] I. Rhee, "Error control techniques for interactive low-bit rate video transmission over the Internet," in *Proc. ACM SIGCOMM '98*, Sept. 1998.

[9] X. Li, M. H. Ammar, and S. Paul, "Video multicast over the Internet," *IEEE Network Mag.*, pp. 46–60, Mar./Apr. 1999.

[10] H. Schulzrinne, S. Casner, R. Fredrick, and V. Jacobson, "RTP: A transport protocol for real-time applications,", RFC 1889, Jan. 1996.

[11] C. Bormann *et al.*, "*RTP payload format for the 1998 version of ITU-T rec. H.263 video (H.263+)*," RFC2429, May 1998.

[12] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss, "An architecture for differentiated services,", IETF RFC 2475, Dec. 1998.

[13] K. Nichols *et al.*, "Definition of the differentiated services field (DS field) in the IPv4 and IPv6 headers,", RFC 2474, Dec. 1998.

[14] J. Mogul and S. Deering, "Path MTU discovery RFC 1191," DECWRL and Stanford Univ., Stanford, CA, Nov. 1990.

[15] S. Floyd and K. Fall, "Promoting the use of end-to-end congestion control in the internet," IEEE Trans. Networking, vol. 6, pp. 458–472, 1998, submitted for publication.

[16] S. Jacobs and A. Eleftheriadis, "Streaming video using TCP flow control and dynamic rate shaping," *J. Vis. Commun. Image Rep.*, vol. 9, pp. 211–222, Sept. 1998.

[17] R. Rejaie, M. Handley, and D. Estrin, "RAP: An end-to-end rate-based congestion control mechanism for realtime streams in the internet," in *Proc. IEEE INFOCOMM'99*, Mar. 1999.

[18] J. Padhye, J. Kurose, and D. Towsley, "A TCP-friendly rate adjustment protocol for continuous media flows over best effort networks," Univ. Massachusetts, Boston, CMPSCI Tech. Rep.TR 98-04, Oct. 1998.

[19] D. Sisalem and H. Schulzrinne, "The loss-delay based adjustment algorithm: A TCP-friendly adaptation scheme," in *Proc. NOSSDAV'98*, July 1998.

[20] V. Paxson, "End-to-end routing behavior in the internet," *IEEE Trans. Networking*, vol. 5, pp. 601–615, Oct. 1997.

[21] M. Allman and V. Paxson, "On estimating end-to-end network path properties," in *Proc. ACM SIGCOMM '99*.

[22] Y. J. Chung, J. Kim, and C.-C. Jay Kuo, "Real-time streaming video with adaptive bandwidth control and DCT-based error concealment," *IEEE Trans. Circuits Syst. II*, vol. 46, pp. 951–956, July 1999.

[23] H. Song, J. Kim, and C.-C. Jay Kuo, "Real-time encoding frame rate control for H.263+ video over the internet," *Signal Processing: Image Commun.*, vol. 15, Sept. 1999.

[24] M. Shen, J. Kim, and C.-C. Jay Kuo, "Fast compression artifact reduction technique based on nonlinear filtering," in *Proc. ISCAS '99*.

[25] T.-Y. Kuo, J. Kim, and C.-C. Jay Kuo, "Motion- compensated frame interpolation scheme for H.263 codec," in *Proc. ISCAS '99*, May 1999.

[26] D. Bagni, V. Riva, and L. Albani, "An innovative temporal post-processing to increase the frame rate in H.263 encoding systems," in *Proc. PCS '99*, Apr. 1999.

[27] T. V. Lakshman, A. Ortega, and A. R. Reibman, "VBR video: Tradeoffs and potentials," *Proc. IEEE*, vol. 86, pp. 952–973, May 1998.

[28] N. G. Duffield, K. K. Ramakrishnan, and A. R. Reibman, "SAVE: An algorithm for smoothed adaptive video over explicit rate networks," *IEEE Trans. Networking*, vol. 6, pp. 717–728, Dec. 1998.

[29] J. Rexford and D. Towsley, "Smoothing variable-bit-rate video in an inter-network," *IEEE Trans. Networking*, vol. 7, pp. 202–215, Apr. 1999.

[30] J. Waldby, U. Madhow, and T. V. Lakshman, "Total acknowledgment: A robust feedback mechanism for end-to-end congestion control," in *Proc. ACM SIGMETRICS '98*.

[31] M. Allman, "On the generation and use of TCP acknowledgments," *ACM Comput. Commun. Rev.*, vol. 28, Oct. 1998.

[32] L. Brakmo, S. O'Malley, and L. Peterson, "TCP Vegas: New techniques for congestion detection and avoidance," in *Proc. ACM SIGCOMM '94*, May 1994.

[33] G. de los Reyes, A. Reibman, and S. F. Chang, "A corruption model for motion compensated video subject to bit errors," in *Proc. Packet Video Workshop'99*.

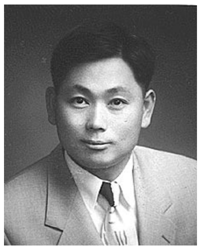
[34] UCB/LBNL/VINT. (1998) Network Simulator—NS (version 2). [Online] Available: http://www-mash.cs.berkeley.edu/ns

JongWon Kim was born in SangJu, Korea, in 1964. He received the B.S., M.S., and Ph.D. degrees from Seoul National University, Seoul, Korea, in 1987, 1989, and 1994, respectively, all in control and instrumentation engineering.

During 1994–1999, he was with the Department of Electronics Engineering, KongJu National University, KongJu, Korea, as an Assistant Professor. Since July 1997, he has been with the Signal and Image Processing Institute of Electrical Engineering Systems Department, University of Southern California, Los Angeles, where he is now a Research Assistant Professor. His current research interests are in the areas of multimedia signal processing and communications, focusing on the reliable and flexible delivery system for integrated multimedia over wired/wireless IP networks, for which he has authored around 60 technical publications in international conferences and journals.

Young-Gook Kim received the B.S. degree from the Physics Education Department, Seoul National University, Seoul, Korea, in 1993, and the M.S. degree in electrical engineering from the University of Southern California (USC), Los Angeles, in 1997, where he is currently working toward the Ph.D degree in the Integrated Media Systems Center and Signal and Image Processing Institute.

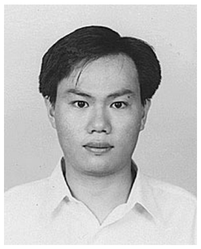
From 1996 to 1998, he was with Advanced Network Technology, USC, as a Research Assistant. His research interests are in the areas of real-time streaming via Internet system & Broadband networking system including wireless media communications.



HwangJun Song received the B.S. and M.S. degrees in 1990 and 1992 from the Department of Control and Instrumentation (Electrical Engineering), Seoul National University, Seoul, Korea. In 1999, he received the Ph.D. degree in electrical engineering from the University of Southern California, Los Angeles.

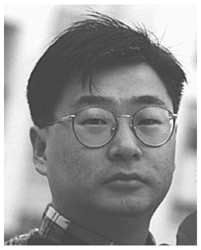
From 1990 to 1992, he was a Research Assistant with Seoul National University. In 1992, he was a Research Engineer at LG Industrial Labs, Korea. From 1992 to 1994, he served full-time in Korea Army. In 1995, he was a part-time Lecturer at various colleges. From 1995 to 1999, he was a Research Assistant at the Signal and Image Processing Institute and Integrated Media Systems Center, University of Southern California, Los Angeles, sponsored by the National Science Foundation. He was with Sejong University, Seoul, Korea, during the spring 2000 semester. Since the fall semester 2000, he has been with Hongik University, Seoul, Korea. His research interests include multimedia signal processing and communication, image/video compression, digital signal processing, network protocols necessary to implement a functional image/video applications, control systems, and fuzzy-neural systems.

In 1999, Dr. Song received the Sarnoff Technical Achievement Award.



Tien-Ying Kuo received the B.S. degree from the National Taiwan University, Taipei, Taiwan, in 1990, and the M.S. and Ph.D. degrees from the University of Southern California, Los Angeles, 1994 and 1998, respectively, all in electrical engineering.

Presently, he is with the Department of Digital Video, Sharp Laboratories of America, Huntington Beach, CA. His current research interests are in the areas of digital signal and image processing, video coding, multimedia communications, and HDTV technologies.



Yon Jun Chung received the B.S.E.E. degree from the Massachusetts Institute of Technology, Cambridge, in 1989, and the M.S. and Ph.D. degrees in electrical engineering in 1994 and 1998 from the University of Southern California, Los Angeles.

He is currently with the Streaming Group, Akamai Technologies, San Mateo, CA, after previously being with Ford Aerospace. His research interests include protocols DiffServ, RTP, RTSP, RSVP, and related video-compression algorithms H.263 & MPEG-4 necessary for real-time network video.

C.-C. Jay Kuo (F'99) received the B.S. degree from the National Taiwan University, Taipei, Taiwan, in 1980, and the M.S. and Ph.D. degrees from the Massachusetts Institute of Technology, Cambridge, in 1985 and 1987, respectively, all in electrical engineering.

He was Computational and Applied Mathematics Research Assistant Professor in the Department of Mathematics, University of California, Los Angeles, from October 1987 to December 1988. Since January 1989, he has been with the Department of Electrical Engineering Systems and the Signal and Image Processing Institute, University of Southern California, Los Angeles, where he currently has a joint appointment as Professor of Electrical Engineering and Mathematics. His research interests are in the areas of digital signal and image processing, audio and video coding, wavelet theory and applications, multimedia technologies, and large-scale scientific computing. He has authored around 400 technical publications in international conferences and journals.

Dr. Kuo is a member of SIAM, ACM, and a Fellow of SPIE. He is the Editor-in-Chief for the *Journal of Visual Communication and Image Representation*, and has served as Associate Editor for IEEE TRANSACTIONS ON IMAGE PROCESSING duriing 1995–98 and IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY during 1995–1997. He received the National Science Foundation Young Investigator Award (NYI) and Presidential Faculty Fellow (PFF) Award in 1992 and 1993, respectively.