MPEG-4 COMPRESSED VIDEO OVER THE INTERNET

Dapeng Wu[*]   Yiwei Thomas Hou[†]   Wenwu Zhu[‡]   Ya-Qin Zhang[§]   H. Jonathan Chao[*]

[*] Polytechnic University, Brooklyn, NY, USA
[†] Fujitsu Laboratories of America, Sunnyvale, CA, USA
[‡] Bell Laboratories, Holmdel, NJ, USA
[§] Microsoft Research, Beijing, China

ABSTRACT

This paper focuses on transport issues of MPEG-4 video over the Internet. Specifically, we present a packetization algorithm for MPEG-4 bit streams at the sync layer and an end-to-end feedback control mechanism. Our packetization algorithm achieves both efficiency and robustness by exploiting the unique video object plane (VOP) feature in MPEG-4 while our feedback control algorithm is capable of estimating available network bandwidth based on the packet loss information at the receiver. Simulation results demonstrate that once our algorithms are employed by an adaptive MPEG-4 encoder, MPEG-4 video is able to achieve good perceptual picture quality at the receiver under low bit rate and varying network conditions and efficiently utilize network resources.

1. INTRODUCTION

MPEG-4 video is the new international standard designed to establish a flexible content-based visual environment [3]. The design of MPEG-4 video is centered around a basic unit of content called the visual object (VO). It is foreseen that the MPEG-4 VO environment will be capable of addressing multimedia application areas ranging from conventional storage and transmission of video to truly interactive content-based video services.

Fundamental problems still remain on how to successfully support MPEG-4 video over the Internet. For example, to date, the packetization process for MPEG-4 VOP bit stream has not been adequately addressed [5]. Furthermore, it is not clear how MPEG-4 should adapt to varying Internet conditions while still deliver good perceptual quality.

This paper addresses these two issues by presenting a video packetization algorithm for MPEG-4 at the sync layer and an end-to-end feedback control algorithm. Our packetization algorithm achieves both efficiency and robustness by exploiting the unique VOP concept in MPEG-4 video; while our feedback control algorithm is capable of estimating available network bandwidth based on packet loss ratio at the receiver. We show that once employed by an adaptive MPEG-4 encoder, our algorithms are capable of transporting MPEG-4 video over the network with good perceptual quality under varying network conditions and utilizing network resources efficiently.

Prior efforts on video packetization for the Internet include [7] for H.261, [8] for H.263, and [2] for MPEG1/2. But these traditional video coding standards are frame-based and may not be applicable to or optimal for MPEG-4. Recent effort on RTP payload format for MPEG-4 elementary streams was discussed in [5]. But it is not clear in [5] on how to perform packetization for the MPEG-4 VOPs at the sync layer before passing onto the RTP layer.

Previous work on feedback control for Internet video include [6]. Since this algorithm was specifically designed for an H.261-like video, it cannot be applied directly to MPEG-4. This paper extends the feedback control technique in [6] for MPEG-4 video.

The remainder of this paper is organized as follows. In Section 2, we present our packetization algorithm for MPEG-4 VOP bit streams at the sync layer. Section 3 presents our feedback control algorithm. In Section 4, we use simulations results to demonstrate the performance of MPEG-4 video under our algorithms. Section 5 concludes this paper.

2. A PACKETIZATION ALGORITHM

Figure 1 shows the packet format at each layer at the end system. At the sender side, the Compression Layer compresses the visual information and generates Elementary Streams (ESs), which contain the coded representation of the visual objects (VOs). The ESs are packetized at sync layer (SL) with timing and synchronization information, as well as fragmentation and random access information. The SL-packetized streams are multiplexed into a FlexMux stream at the TransMux Layer, which is then passed on to the RTP/UDP/IP protocol stacks before being sent to the Internet. At the receiver side, the video stream is processed in the reversed manner before its presentation.

To date, the packetization process for MPEG-4 video ES at the sync layer has not been adequately addressed [5]. An appropriate packetization algorithm at this layer is essential for the optimal transport of MPEG-4 video over the Internet. In this section, we present a sync layer packetization algorithm that offers both efficiency and robustness for Internet transport.

It is clear that the use of large packet size will reduce the the total number of generated packets and overhead.[1]

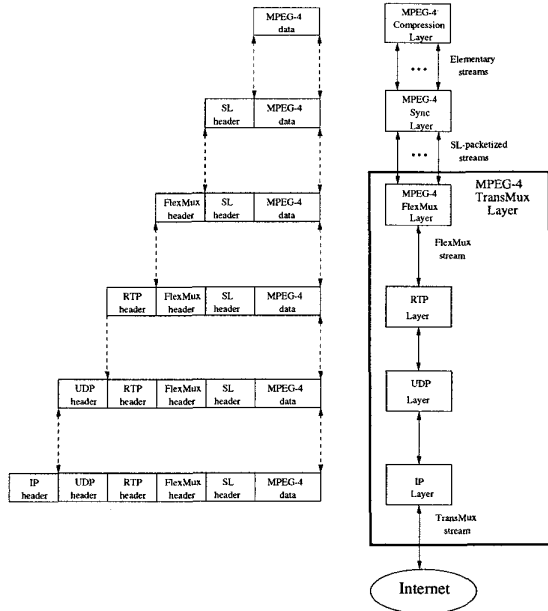[1] The overhead is 50 bytes long, which consists of 3 bytes of

Figure 1: Data format at each processing layer at an end system.

On the other hand, the packet size cannot be larger than the path MTU, which is defined to be the minimum of the MTUs along all the traversing links from the source to the destination. This is because that any packet larger than path MTU will result in IP fragmentation, which bring overhead for each fragmented packet. To make things worse, loss of one fragment packet will corrupt other fragment packets within the original packet. Furthermore, for MPEG-4 video, it is also not advisable to packetize the data that contain information across two VOPs. With these considerations, we choose packet size to be the minimum of the current VOP size and the path MTU.[2]

When a VOP is too large to fit into a single packet, it is necessary to break up the VOP into multiple segments and use multiple packets for these segments. We try to minimize both the number of packets generated for a given MPEG-4 bit stream and the dependency between adjacent packets. The motivation for minimizing the number of packets was the same as above (i.e. minimizing overhead); while the motivation of minimizing the dependency between adjacent packets is to mitigate the dependency of MPEG-4 decoder on any lost packet. In particular, if the MPEG-4 VOP header information is copied into each packet, such dependency can be removed among the packets. Since the size of a macro-block (MB) is always less than the path MTU,[3] a packet should be composed of at least one MB.

Our packetization strategy is the following. If a com-

plete VOP fits into a packet, then packetize such VOP with a single packet. Otherwise, we will try to packetize as many MBs as possible into a packet (with re-synchronization marker copied into each packet for the same VOP) without crossing over into the next VOP even if space is available in the last packet for the current VOP, i.e. MBs from consecutive VOPs are never put into the same packet. Our packetization method achieves both efficiency, which is essential for low bit-rate coding, and robustness to packet loss (due to strict boundary between VOPs among packets and copying of re-synchronization marker into packets for the same VOP).

We first describe the functions and parameters used in our packetization algorithm.
1) BitCount is a counter that registers the number of bits read for current packetization process.
2) MaxPL, or Maximum payload length (in bits), equals to (path MTU − 50 bytes) · 8.
3) VOP_start_code is a predefined code at the beginning of a VOP and is regarded as the boundary between two consecutive VOPs. Our sync layer packetization algorithm is shown as follows.

Algorithm 1   A Packetization Algorithm

while (there is encoded data to be packetized) {
    search for next VOP_start_code and BitCount counts
    the number of bits;
    if ((next VOP_start_code is found) and
    (BitCount − length of VOP_start_code ≤ MaxPL)) {
        /* Packetize by VOP boundary */
        packetize the bits before next VOP_start_code;
        }
    else if (BitCount − length of VOP_start_code
          > MaxPL) {
        /* Packetize by MBs. */
        Packetize as many MBs as possible without
        exceeding MaxPL and without crossing into
        next VOP;
        }
        else { /* Next VOP_start_code is not found,
        i.e. end of video. */
        Packetize the remaining data.
        }
}

3. A FEEDBACK CONTROL ALGORITHM

The current Internet does not widely support any reservation mechanism or QoS. Moreover, the available bandwidth not only is not known a priori but also changes with time. Therefore, a mechanism must be in place for MPEG-4 video source to sense network conditions so that it can encode the video with appropriate output rate.

Under our architecture, we let the MPEG-4 video source gradually increase its transmission rate to probe available network bandwidth. Such rate increase will first have the source's rate reach the available network bandwidth. Then the source rate will overshoot the available network bandwidth and fall into the congestion region. Congestion is detected by the receiver through packet loss in the received packets. The receiver sends feedback RTCP packets to the
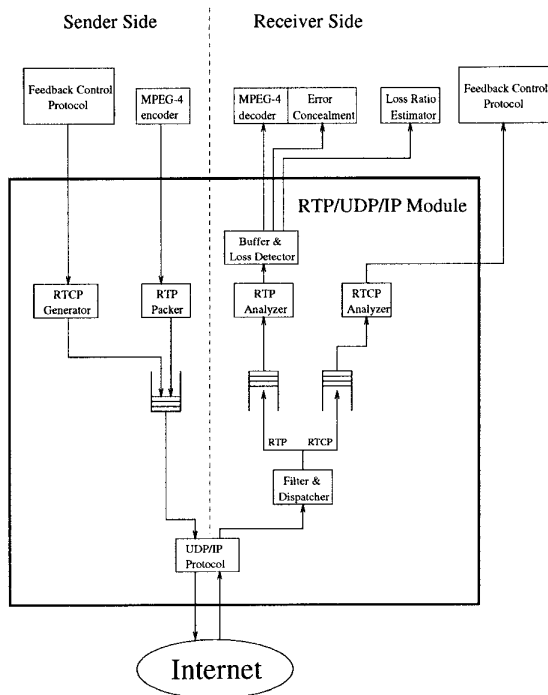
---

sync layer header, 3 bytes of FlexMux header, 16 bytes of RTP header, 8 bytes of UDP header, and 20 bytes of IP header.

[2] If path MTU information is not available, the default MTU, i.e. 576 bytes, will be used.

[3] The maximum size of a MB is 90 bytes and the default path MTU is 576 bytes.

Figure 2: Architecture of RTP/UDP/IP Module.

Figure 3: A peer-to-peer network.

source to indicate congestion status. Once source receives such feedback, it decreases its transmission rate.

Figure 2 shows the architecture of our RTP/UDP/IP module. This module is the key component to realize our rate-based feedback control. At the sender side, an MPEG-4 encoder generates a packetized stream (FlexMux stream), which is turned into RTP packets by RTP packer. Meanwhile, the control information is transferred to the RTCP generator. The resulting RTCP and RTP packets go down to UDP/IP layer for transport over the Internet. At the receiver side, arriving IP packets are first un-packed at UDP/IP layer, then dispatched by Filter & Dispatcher to RTP and RTCP analyzers. RTP analyzer first un-packed RTP packets and then put them into a buffer to examine packet loss information (through RTP packet sequence number). On the other hand, the RTCP analyzer un-packs the RTCP packets and sends the feedback information to the Feedback Control Protocol component.

In consistent with the RTP/RTCP standard [4], we let the source periodically send one RTCP control packet for every N_s MPEG-4 video packets. The receiver sends a feedback RTCP control packet back to the source upon receiving N_r packets (or at least once every 5 seconds). The returning RTCP packet contains the packet loss ratio P_{loss} the receiver observed during the N_r packet time interval since the previous feedback RTCP packet. Rate control actions are taken by the encoder upon receiving a backward RTCP packet.
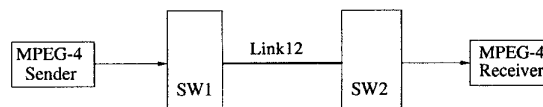
Algorithm 2   A Feedback Control Algorithm

Sender Behavior

- The sender starts to transmit at an initial rate of IR, i.e., $r := $ IR, which is greater than or equal to its minimum rate MR;

- For every N_s transmitted RTP data packets, the sender sends a forward RTCP control packet;

- Upon the receipt of a backward RTCP packet with the packet loss ratio P_{loss} from the receiver, the output rate r at the source is adjusted according to the following rule:

$$\text{if } (P_{loss} \leq P_{threshold})$$
$$\qquad r := \min\{(r + \text{AIR}), \text{PR}\};$$
$$\text{else}$$
$$\qquad r := \max\{(\alpha \cdot r), \text{MR}\}$$

Receiver Behavior

- The receiver keeps track of the sequence number in the RTP header of the arriving packets;

- Upon receiving N_r packets (or at most 5 seconds), the receiver sends a feedback RTCP packet to the source containing packet loss rate P_{loss} it observes since sending last RTCP feedback packet.

During a control action, the feedback control algorithm (Algorithm 2) adjusts the output rate r of the MPEG-4 encoder so that the packet loss ratio P_{loss} stays below a predetermined value $P_{threshold}$. Unlike [6], where a multiplicative increase rate adjustment is employed when an feedback RTCP packet indicates that there is no congestion, we employ additive increase in Algorithm 2, which is a conservative rate increase approach to adapt to available network bandwidth. Our experience shows that a multiplicative increase usually brings much larger source rate oscillation and more packet loss in a large network than a conservative rate adjustment such as additive increase. On the other hand, we employ multiplicative decrease in Algorithm 2 should the source find that the P_{loss} is larger than threshold in the returning RTCP packet. We find that such swift rate reduction at the source is necessary to shorten congestion period and reduce packet loss.

4. SIMULATION RESULTS

In this section, we implement our proposed packetization algorithm and feedback control for the MPEG-4 video coder in [1] on our network simulator. We perform a simulation study to examine the performance of MPEG-4 video over the network.

We employ the standard peer-to-peer benchmark network configuration shown in Fig. 3 for the Internet environment. We emphasize that such simple network configuration captures the fundamental property of a transport

Table 1: Simulation parameters.

End System	MaxPL		4208 bits
	IR		10 Kbps
	AIR		0.5 Kbps
	PR		200 Kbps
	MR		5 Kbps
	α		0.95
	N_s		79
	N_r		25
	$P_{threshold}$		5%
	Buffer Size		1 Mbytes
Switch	Buffer Size		10 Kbytes
	Packet Processing Delay		4 μs
	Buffer Management		Tail Dropping
Link	End System to Switch	Speed	10 Mbps
		Distance	1 km
	Inter-Switch	Speed	15/50/25 Kbps
		Distance	1000 km

Figure 4: Source output rate and available link bandwidth.

Figure 5: Link utilization and packet loss ratio.

path within the Internet cloud since there is only one bottleneck link (i.e., the one with minimum bandwidth among all the traversing links) between the sender and the receiver. Furthermore, we stress that despite the multi-path and thus arriving packets out of sequence problem in the Internet, the validity and generality of our findings will not be compromised by the simple peer-to-peer network configuration since our architecture and algorithms are designed and implemented entirely on end systems (sender and receiver). Therefore, a packet arriving after the threshold due to multi-path routing can simply be treated as a lost packet at the destination and our architecture and algorithms remain intact under such scenario.

At the source side, we use the standard raw video sequence "Miss America" in QCIF format for the MPEG-4 video encoder. The encoder performs MPEG-4 coding as specified in [1] and adaptively adjusts its rate under our feedback control algorithm (Algorithm 2). The encoded bit stream is packetized with RTP/UDP/IP protocol overhead and sent to the network. Packets may be dropped due to congestion in the network. For arriving packets, the receiver extract the packet content to form the bit stream for the MPEG-4 decoder. For a lost packet, the Video Object Plane (VOP) associated with the lost packet will be discarded and a previous VOP will be copied over.

Table 1 lists the parameters used in our simulation. We use 576 bytes for the path MTU. Therefore, the maximum payload length, $MaxPL$, is 526 bytes (576 bytes minus 50 bytes of overhead) [5].

We run our simulation for 450 seconds. Since the there is only 300 continuous frames in "Miss America" sequence, we repeat the video sequence cyclically during the simulation run. The link capacity is varying from 15 Kbps during $[0, 150)$ seconds to 50 Kbps during $[150, 300)$ seconds to 25 Kbps after 300 seconds (see Fig. 4).

Figure 4 shows the network link bandwidth and source rate behavior during the 450 second simulation run. We find that the source is able to adju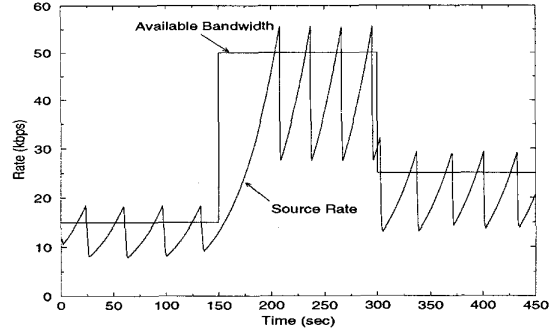st its rate to keep track of the varying network available bandwidth. Figure 5 shows the li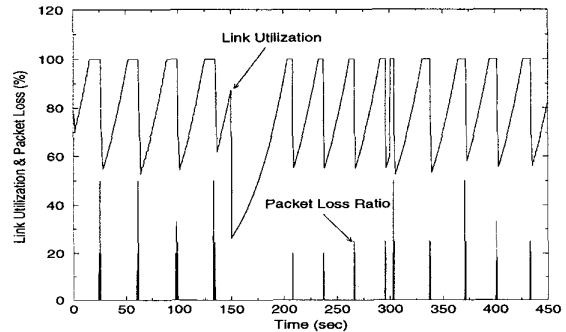nk utilization and packet loss ratio during the same simulation run. We find that the results in Fig. 5 is consistent with that shown in Fig. 4. The oscillation in source rate (Fig. 4) and network utilization (Fig. 5) are due to the propagation delay of the links and the binary nature of our feedback control algorithm. The source performs additive rate increase until it reaches the available link bandwidth. After that it overshoots it and results in congestion and packet loss. The packet loss is detected at the receiver and such information is conveyed to the source. Upon receiving such feedback, the source decreases its output rate. Despite the oscillations, the average utilization of the bottleneck link is over 80%, which is a reasonably good result for feedback control in a wide area Internet. Furthermore, we find that the average packet loss ratio is only 0.35%, which demonstrates the effectiveness of our feedback control algorithm.

A measure of the difference between the original video sequence and the received video sequence is the peak signal-to-noise (PSNR). Figure 6 shows the PSNR of Y component of the MPEG-4 video at the receiver for the same simulation run in Figs. 4 and 5.

To examine the perceptual quality of the MPEG-4 video, we play out the decoded video sequence at the receiver. Figures 7, 8, and 9 show sample video frames at the receiver during $[0, 150)$, $[150, 300)$, and $[300, 450]$ second time interval, respectively. Recall that we repeat the "Miss America" video sequence cyclically during the simulation run since we only have 300 frames available. The sample frames shown
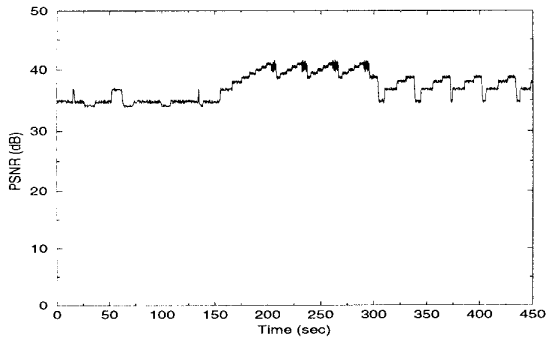
Figure 6: PSNR of MPEG-4 video at the receiver.

Figure 7: Sample frame of MPEG-4 video at the receiver during [0, 150) second time interval.

in Figs. 7, 8, and 9 all show the same scene but at different time interval and all have good perceptual quality.

Figure 8: Sample frame of MPEG-4 video at the receiver during [150, 300) second time interval.

In summary, based on the simulation results in this section, we conclude that our algorithms can 1) transport MPEG-4 video streams over the network with good perceptual picture quality under both low bit rate and varying network conditions; and 2) adapt to available network bandwidth and utilize it efficiently.

5. CONCLUDING REMARKS

The contributions of this paper are two-folded. First, we presented, for the first time, a packetization algorithm for MPEG-4 video bit streams at the sync layer for Internet transport. Our packetization algorithm achieved both efficiency and robustness by considering the VOP concept

Figure 9: Sample frame of MPEG-4 video at the receiver during [300, 450] second time interval.

in MPEG-4 video. Second, we designed an end-to-end feedback control algorithm for adaptive MPEG-4 video encoder. Simulation results demonstrated that our algorithms can support MPEG-4 video over the Internet with satisfactory performance.

6. ACKNOWLEDGMENTS

The authors would like to thank Hung-Ju Lee and Tihao Chiang of Sarnoff Corporation for their assistance related to this work.

7. REFERENCES

[1] T. Chiang and Y.-Q. Zhang, "A new rate control scheme using quadratic rate distortion model," IEEE Trans. on Circuits and Systems for Video Technology, vol. 7, no. 1, pp. 246–250, Feb. 1997.

[2] D. Hoffman, G. Fernando, and V. Goyal, "RTP payload format for MPEG1/MPEG2 video," Internet Engineering Task Force, RFC 2038, Oct. 1996.

[3] ISO/IEC JTC 1/SC 29/WG 11, "Information technology – coding of audio-visual objects, part 1: systems, part 2: visual, part 3: audio," FCD 14496, Dec. 1998.

[4] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, "RTP: a transport protocol for real-time applications," Internet Engineering Task Force, RFC 1889, Jan. 1996.

[5] H. Schulzrinne, D. Hoffman, M. Speer, R. Civanlar, A. Basso, V. Balabanian, and C. Herpel, "RTP payload format for MPEG-4 elementary streams," Internet Draft, Internet Engineering Task Force, Mar. 1998.

[6] T. Turletti and C. Huitema, "Videoconferencing on the Internet," IEEE/ACM Trans. on Networking, vol. 4, no. 3, pp. 340–351, June 1996.

[7] T. Turletti and C. Huitema, "RTP payload format for H.261 video streams," Internet Engineering Task Force, RFC 2032, Oct. 1996.

[8] C. Zhu, "RTP payload format for H.263 video streams," Internet Engineering Task Force, RFC 2190, Sept. 1997.