FGS-Based Video Streaming Test Bed for MPEG-21 Universal Multimedia Access with Digital Item Adaptation

Chung-Neng Wang[1], Chia-Yang Tsai[2], Han-Chung Lin[1], Hsiao-Chiang Chuang[2], Yao-Chung Lin[2], Jin-He Chen[1], Kin Lam Tong[2], Feng-Chen Chang[2], Chun-Jen Tsai[1], Shuh-Ying Lee[1], Tihao Chiang[2], and Hsueh-Ming Hang[2]

National Chiao-Tung University, Hsinchu, 30050, Taiwan

ABSTRACT

It is a challenging problem to stream real-time video over heterogeneous networks with time-varying channel conditions and devices with different capabilities. In this paper, we present an FGS-based unicast video streaming test bed, which is now being considered by the MPEG-4/21 committee as a reference test bed. The proposed system supports resource adaptation delivery of MPEG-21 DIA scheme which leads to a more strict evaluation methodology according to the MPEG committee specified common test conditions for scalable video coding. It provides easy control of media delivery with duplicable network conditions. To provide the best quality of service for each client, we propose relevant rate control, error protection, and transmission approaches in the content server, network interface, and clients, respectively.

1. INTRODUCTION

It is a challenging problem to stream real-time video over heterogeneous networks for a wide range of consumer electronic applications under universal multimedia access (UMA) [1]. These media suffer from bandwidth fluctuation and several types of channel degradations such as random errors, burst errors and packet losses [2]. Thus, the MPEG-4 committee has adopted various techniques to address the issue of error resilient delivery of video information for point-to-point multimedia communications. The MPEG-4 committee further developed the MPEG-4 Streaming Video Profile (SVP) and Fine Granularity Scalability (FGS) profile [3] that provide a scalable approach specifically for steaming video applications.

In a heterogeneous network, the receiving devices may have limited display, processing power, or may only be able to handle a particular compression format. For devices having FGS decoding feature, the server can truncate the enhancement layer bitstream to fit the variable channel bandwidth. For other devices that are only capable of decoding MPEG-4 Advanced Simple Profile decoder require only the base layer for display. Moreover, the consumers may have different preferences. Thus, MPEG-21 further develops Digital Item Adaptation [1] to interact with the streaming server about the receiver's capabilities and user characteristics. However, it becomes more difficult to develop the minimum set of digital item description schemes since there is no reference software hat supports DIA scheme so far for functionality emulation and advanced analysis. In this paper, we provide such a solution and platform for the MPEG-21 committee to experiment with various user scenarios.

Some schemes [4]-[5] have been proposed to simultaneously stream or multicast video over Internet or wireless channels to a wide variety of devices using MPEG-4 FGS. These schemes show promising results of scalable coding techniques. However, it is difficult to evaluate the results without common test conditions. Thus, the MPEG committee has drafted the testing procedures for evidences on scalable coding and applications and requirements for scalable coding [3] in July 2002.

Moreover, a practical network environment behaves differently for each experiment, which makes it difficult to test the streaming systems with results that can be studied and duplicated. In this paper, we adopt a network simulator, namely NISTnet [6]. The NISTnet provides easy controls to create duplicable network conditions such as packet loss ratio, jitter, bandwidth variation, and delay.

2. FGS-BASED STREAMING TEST BED

The goal is to support MEG-21 DIA scheme with a more strict evaluation methodology according to the specified common conditions for scalable coding. **Figure 1** shows the proposed test bed system architecture, which covers four key modules including the FGS-based Video Content Sever, Video Clients, Network Interface, and Network Simulator.

2.1 FGS-Based Video Content Server

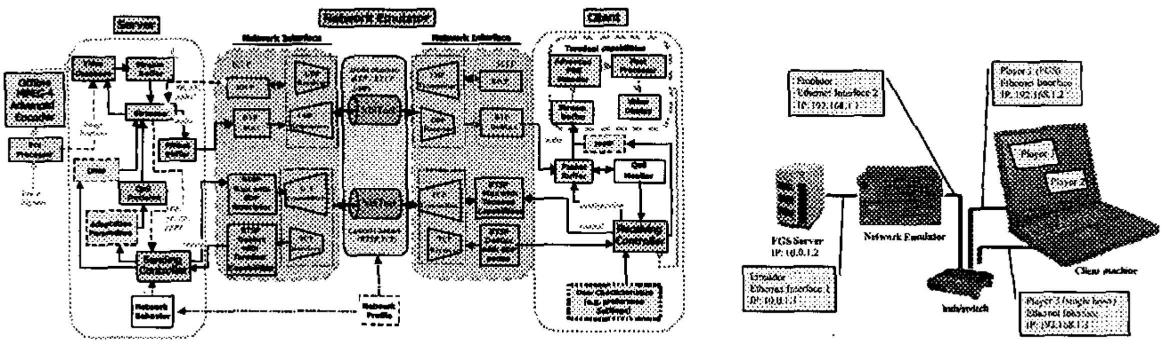
The content server covers seven sub-modules including FGS encoder, video database, stream buffer, streamer, packet buffer, IP protection, and sending controller. The FGS encoder offline compresses each video sequence into a two-layer bitstream with base and enhancement layers. Both bitstreams are stored in the video database and the requested bitstreams are moved to the stream buffer. The streamer, which accepts commands from the sending controller, segments each demanded bitstream into video packets according to MPEG-4 specification. The video packets are put into the packet buffer as the RTP payload. The sending controller interacts with the receiving controller to create a media session for video delivery and a separate RTSP session for accepting the retransmission requests from the individual clients. When initializing a video delivery session, the streamer splits the VO and VOL headers from both bitstreams. In addition, the timestamp (' TP'), SSRC value, and sequence number (' SN') of the first RTP packet are returned from the RTP module in the Network Interface. All these parameters are used to compose SDP (Session Description Protocol) [8]. The SDP message is embedded into the RTSP reply for the DESCRIBE request and sent via the control channel to the receiver.

To provide the best quality of service (QoS), it' s a challenge to adapt the source rate to the current network channel conditions [7]. The network conditions are defined as network profiles in this system. To maintain QoS for each client under the consideration of packet loss ratio, retransmission frequency, and effective bandwidth, we adopt a simple segmentation scheme and a rate control scheme in the streamer. To avoid over- fragmentation of bitstreams, which causes lots of RTP packets with very small payload and increases transmission overhead and the probability of packet loss, we merge small video packets with the preceding packet before storing them to

1. The authors are with the Department and Institute of Computer Science and Information Engineering, NCTU, Taiwan.

2. The authors are with the Department and Institute of Electronics Engineering, NCTU, Taiwan.

0-7803-7761-3/03/$17.00 ©2003 IEEE



(a) System architecture (b) Network connection setup

Figure 1. Architecture and network connection setup of the proposed FGS-based video streaming test bed.

the packet buffer. Based on the bit rates of FGS bistreams and the network profile, the rate control scheme calculates the actual number of delivered packets per second with the following steps.

Step 1: Get the available bandwidth $R(t)$ in bits per second (bps) from the pre-specified network profile.

Step 2: Allocate available bits to each VOP with the underlying weighting function $R_I(t) = \frac{R(t)}{w_I N_I + w_P P_I + w_B B_I}$, where the weights are w_I, w_P, and w_B for I-, P-, and B-VOPs, respectively. The values $w_I = w_P = 1$ and $w_B = 0.6$ are found empirically.

Step 3: With the allocated bits for each type of VOP, send the all of the packets carrying the base layer bitstream.

Step 4: With the remaining bandwidth available in a one-second window, send the maximal number of packets covering the enhancement layer bitstream. If the remaining bandwidth is not sufficient for a full packet, the bitstream will be truncated before a FGS re-synchronization marker or bit-plane start code. The bits actually used are bounded by the allocated budget.

A simple time scheduling approach is used to spread the packets. Since the packet size is specified in the encoding processing, al video packets in the encoded bitstream have almost identical sizes. We use the time interval that is taken to recognize and retrieve every packet data from the bitstream to put each packet into the channels. Thus, we stream out the packets in evenly spaced time intervals to prevent burst packet loss at the receivers.

2.2 Video Clients

As illustrated in **Figure 1** (a), the client has seven sub-modules including the FGS decoder, stream buffer, packet buffer, video display, QoS Monitor, and receiving controller.

In building a video delivery session, the receiving controller parses the SDP information field from RTSP channel. The SDP parser places the TP, SSRC, and SN into the RTP module of the client-side Network Interface module. The VO and VOL header information is copied into the packet buffer before the first packet data for decoding.

At the client side, we use the packet buffer and QoS monitor to check the packet reception status and request packet retransmission when any packet loss occurs. To prevent the retransmitted packets from occupying a large percentage of the effective bandwidth, the QoS monitor adopts a simple approach in managing the occurrence of retransmission requests. When the streaming session between the server and client is established, the client will fill the packet buffer with 3 seconds of packets and then move the media data into the stream buffer. The buffer fullness is monitored by the QoS monitor with the following steps for error protection of FGS base-layer bitstream using retransmission.

Step 1: Packet loss detection:

Each packet has its own "sequence number", so the consecutive packets have consecutive sequence numbers. Hence, the QoS monitor will check if the currently accessed sequence number in packet buffer is identical to that of the required packet.If it is, wait for one second and redo Step 1. Otherwise, go to Step 2.

Step 3: Retransmission:

If packet loss is detected in Step 1, the QoS monitor will inform the controller that packet loss happens. The controller will signal server through RTSP channel as feedback. Because we set a lower bound for preventing the packet buffer from underflow, the decoding process will not be often interrupted with the empty buffer. If the round trip time of the re-transmitted packet is less than the packet buffer delay, which equals to the number of unprocessed packets, the caused damage is handled by the crash-proof video decoder and the picture quality is improved with error resilience and concealment methods. Similarly, re-transmission mechanism is to deal with packet loss situation under the constraints of the effective transmission bandwidth.

To decode and display the bitstream corrupted by lost packets, a crash proof decoder with error resilience and concealment is implemented in our proposed system. At the enhancement layer, we utilized the error resilient tools to verify the robustness of decoding process [9]. At the base layer, the bitstream errors are detected with a prior knowledge of the bitstream syntax and its semantics. For the syntactic errors we will detect errors that are caused by invalid codeword or stuffing bits for the decoding frame or structure. In particular, we will check the cases such as more than 64 coefficients in a block; MB number exceeds the VOP's MB number, or codeword not in the VLC table [10].

2.3 Network Interface

To link the server, client and transmission channel, a standard RTSP/RTP-based network interface is adopted. As shown in **Figure**

2, the network interface adopts three categories of network protocols covering the network-layer protocol, transport protocol, and session control protocol. Similar protocol stack can be found in [7]. The network-layer protocol using IP networks serves the basic network support such as the network addressing. The transport protocols including UDP, TCP, and real-time transport protocol (RTP) [11] are used to provide an end-to-end network transport for video streaming. The session control protocol use the real-time streaming protocol (RTSP) [12] that specifies the messages and procedures to control the media delivery during an established session. Since there is no standard RFCs or Internet drafts that specify ways to map FGS bitstream to RTP payload and to support MPEG-21 DIA via RTSP, we create our own RTP payload mapping and RTSP content messages for streaming.

For simplicity, basic RTSP methods and a non-standard RTSP retransmission request are employed for session control. There are four basic client-server RTSP messages including DESCRIBE, SETUP, PLAY, and TEARDOWN. For an instance in **Figure 3**, the message DESCRIBE is used to describe the terminal capabilities and user characteristics. Within the DESCRIBE message, a new content type as application/mpeg21_dia is declared to support MPEG-21 DIA scheme. The MPEG-21 DIA descriptions are transmitted through a RTSP packet when a client wants to subscribe to a server. After successfully subscribing to the server, the client uses the SETUP message to create the media delivery session with specified terminal capabilities, transport protocols, and port numbers. The PLAY starts to transport the media under the built session and TEARDOWN ends the transport and the underlying session. Due to the tradeoff between the best QoS and usage of the effective network bandwidth, the retransmission is employed only for the base layer bitstream. Whenever there are packet losses, the client can send (non-standard) RTSP GET_PARAMETER requests to the server for retransmission of the missing packets from the base layer bitstream.

To support both the network protocol stack and MPEG-21 DIA scheme, the proposed network interface has six sub-modules. The network interface connecting the server and network includes RTP Mux, UDP transmitter, RTSP Mux with DIA descriptor, RTSP DeMux with DIA parser, TCP transmitter, and TCP receiver. The network interface connecting the client and network has the same sub-modules except for RTP DeMux and UDP receiver.

The media data is packetized into RTP packets in the RTP Mux module prior to the transport. The RTP packets are then transmitted using UDP, and received and de-multiplexed by RTP DeMux into the packet buffer at the clients for playback. The RTSP stream of the control messages is transported via TCP. In addition, for the DIA descriptions, an XML parser and an XML compositor are included under RTSP DeMux and Mux modules, respectively.

2.4 Network Simulator

In the test bed, an IP network emulator, NISTnet [6], is used to provide repeatable network environments that are close to practical wide-area heterogeneous network environments. NISTnet is a public domain Linux-based IP network emulator developed by the National Institute of Standard Technology. It provides a simulated IP environment with many controllable channel parameters such as packet loss ratio, jitter, bandwidth variation, and delay. A GUI module that interacts with the NISTnet kernel module is developed for this test bed project. The time-varying network conditions are recorded in a network profile text file as illustrated in **Figure 4**. In addition to controlling the NISTnet module to simulate the pre-specified network profile, the GUI module also displays real time channel usage plots and statistics of both the uplink and downlink connections.

<table>
<tr><td rowspan="2">Application Control Commands</td><td colspan="2">Layered Video data</td></tr>
<tr><td>Base layer</td><td>Enhancement layer</td></tr>
<tr><td>RTSP</td><td colspan="2">RTP</td></tr>
<tr><td>TCP</td><td>UDP with retransmission</td><td>UDP</td></tr>
<tr><td colspan="3">IP</td></tr>
<tr><td colspan="3">Data Link</td></tr>
<tr><td colspan="3">Physical layer</td></tr>
</table>

Figure 2. Network protocol stack

```
DESCRIBE rtsp://140.113.211.184/foreman.m4v RTSP/1.0
CSeq: 0
User-Agent: NCTU FGS Player: 176x144, 16-bit color, FGS, 10
Accept: application/sdp

RTSP/1.0 200 OK
 CSeq: 0
 Content-Type: application/sdp
 Content-Length: 522
 v=0
 o=StreamServer   1060873957046701727   1016147297000   IN   IP4
140.113.211.184
 s=NCTU mpeg4 stream
 e=server@nctu.edu.tw
 c=IN IP4 140.113.211.184
 t=0 0
 a=control:*
 a=range:ntp=0-
 m=video 0 RTP/AVP 96
 a=rtpmap:96 MP4V-ES/90000
 a=control:trackID=0
 a=fmtp:96          profile-level-id=17;          config=
000001010000012002044007a82c2090a21f
 m=video 0 RTP/AVP 97

 a=rtpmap:97 MP4V-ES/90000
 a=control:trackID=1
 a=fmtp:97          profile -level-id=18;          config=
00000101000001218928500lec705841217ffb6db6b6db6db6c924920f
 a=depends_on:trackID=0
```

Figure 3. Illustration of the client-server RTSP DESCRIBE message with SDP description.

```
# A Simple Network Profile
# Time(sec)  PLR(%)  Bandwidth(bps)  Mean Delay(ms)  SDD
   0          0        100000          0              0
   20         0        50000           0              0
                         :
   500        0        200000          0              0
```

Figure 4. Illustration of the network profile used to control NISTnet. PLR indicates the packet loss ratio and SDD means the standard deviation of the delay.

3. Experimental Results

According to the requirements from the MPEG committee [3], the test bed is set up as in **Figure 1** (b). Based on this setup, scalable and single layer bistreams can be transported simultaneously to the clients via identical network environment, which is controlled with the NISTnet, to achieve a fair comparison of their performance and error robustness.

To demonstrate the streaming performance of this test bed, we adopt 2 test sequences including foreman and news, which are in YCbCr 4:2:0 QCIF format. Each sequence has 300 frames and the source frame rate is 30 frames per second (fps). For each sequence, the first frame is encoded as I-VOP and remainders are encoded as P-VOPs. The output frame rate is 10 fps. The base layer bitstream is encoded at 32kbps. The network profiles cover three average network bandwidths, 64kbps, 128kbps, and 256kbps. In addition, to show the quality variation over time, Novel sequence in CIF and with 2000 frames is used. The sequence is encoded in 30fps into a structure of IBBP... The base layr is encoded at 512kbps. In the network profile, the time-varying bandwidth in a range of [512kbps, 1024kbps] and three packet loss ratios, 0%, 5%, and 10% are adopted for simulations. In all simulations, the TM5 rate control scheme is used to fit with the target bit rate.

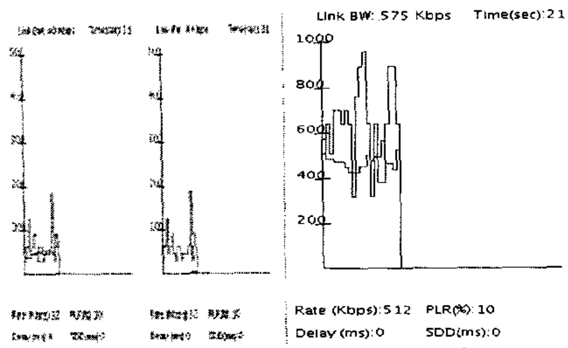
Table 1 illustrates the PSNR of the Y components of both sequences for streaming without retransmission and **Figure 6** shows two bandwidth curves of network conditions. It's interesting that the PSNR of Foreman sequence is decreased as the average bit rate is increased. The PSNR degradation may be caused by the following two reasons. Firstly, since the increased bit rate may increase packet loss rate when the channel bandwidth is insufficient. Secondly, the retransmission is not enabled. In addition, **Figure 5** gives the PSNR of the Y component of Novel sequence under the specified network conditions. The results show that the proposed test bed system can provide better QoS for the client under large network bandwidth variation.

4. Concluding Remarks

In this paper, we proposed an FGS-based unicast streaming system with MPEG-21 DIA as a test bed of scalability over the Internet. In the system architecture, there are four major modules covering the Video Content Sever, Video Clients, Network Interface, and Network Simulator. Even the functionalities of the proposed system are limited for the moment, this test bed is a good starting point for a complete environment for gathering evidences on scalable coding performances and for exploiting applications and requirements for scalable coding. To emulate the real transmission networks, statistics of real networks will be added into the proposed test bed system in the future.

Table 1. PSNR of Y component for the both sequences under various target bit rates.

Seq.	Bit rate (kbps)	PSNR (dB)	Bit rate (kbps)	PSNR (dB)	Bit rate (kbps)	PSNR (dB)
Foreman	64	27.9	128	26.8	256	25.7
News	64	30.0	128	32.5	256	33.7



(a) News_QCIF (b)Foreman_QCIF

Figure 6. Illustration of NISTnet network profiles and real channel conditions over time for streaming the Foreman sequence. The blue lines are the specified bandwidth in the network profiles and the red lines are the real bandwidth.

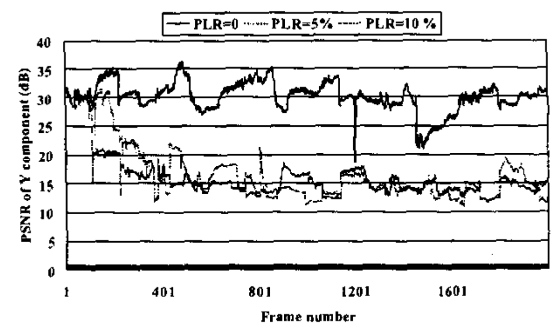


Figure 5. PSNR of the Y component of Novel sequence under the specified network conditions.

5. REFERENCES

[1] A. Vetro, A. Perkis, and S. Devillers, "MPEG-21 Digital Item Adaptation WD (v2.0)," *ISO/IEC JTC1/SC29/WG11 N4944*, Klagenfurt, July 2002.

[2] W. Li, "Overview of fine granularity scalability in MPEG-4 video standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 11, pp. 301–317, Mar. 2001.

[3] MPEG documents: www.cselt.it/mpeg.

[4] H. M. Radha, M. van der Schhar, and Y. Chen, "The MPEG-4 Fine Grained Scalable video coding method for multimedia streaming over IP," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 11, pp. 53–67, Mar. 2001.

[5] W. Zhu, Q. Zhang, and Y.-Q. Zhang, "Network-adaptive rate control with unequal loss protection for scalable video over internet," *in Proc. of ISCAS2001*, pp. V.109–V.112, 2001.

[6] NISTnet: http://snad.ncsl.nist.gov/itg/nistnet/

[7] D. Wu, Y. T. Hou, W. Zhu, Y.-Q. Zhang, and J. M. Peha, "Streaming video over the Internet: approaches and directions," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 11, pp. 282–300, Mar. 2001.

[8] M. Handley and V. Jacobson, "SDP: Session Description Protocol," *Internet Engineering Task Force, RFC 2327*, April 1998.

[9] H.-C. Huang, C.-N. Wang, and Tihao Chiang, "Streaming video in error-prone environments", *in Proc. of ISCOM2001*.

[10] R. Talluri, "Error resilient video coding in the ISO MPEG-4 standard," *IEEE Communications magazine*, June 1998.

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

[12] H. Schulzrinne, A. Rao, And R, Lanphier, "Real time streaming protocol (RTSP), " *Internet Engineering Task Force, RFC 2326*, Apr. 1998.