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\(^2\), Shuh-Ying Lee\(^2\), 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 a wide range of consumer electronic applications through heterogeneous networks with time-varying channel conditions and devices with varying capabilities. In this paper, we present an FGS-based unicast video streaming test bed system, which is now being considered by the MPEG-4/21 committee. The proposed test bed supports MPEG-21 DIA scheme which leads to a more strict evaluation methodology according to the MPEG committee specified common conditions for scalable video coding. It provides easy control of the 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 Streaming Video Profile (SVP) and Fine Granularity Scalability (FGS) profile [3] that provide a scalable approach specifically for streaming video applications.

For the heterogeneous network, the receiving devices may have limited display, processing power, or may only be able to handle a particular compression format. For some devices having FGS compliant decoding feature, the server can truncate the enhancement layer bitstream to fit the variable transmission rate. Other devices that employ only 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 that 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 on scalable coding techniques show promising results of scalable coding. However, it is difficult to evaluate the results without common test conditions. Thus, the MPEG committee has drafted the testing procedures for evidence on scalable coding [6] and applications and requirements for scalable coding [7] 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 [8]. The NISTnet provides easy control instructions to create duplicable network conditions such as packet ratio, jitter, bandwidth variation, and delay.

2. FGS-BASED STREAMING TEST BED

The goal is to support MPEG-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 the base and enhancement layer bitstreams. 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.

To provide the best quality of service (QoS), it’s a challenge to adapt the source rate to the current network channel conditions [9]-[10]. 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 much fragmentation of bitstreams, which causes lots of RTP packets with very small payload and increases the probability of packet loss, we merge

---

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.
small packets with the preceding packet before storing to the packet buffer. Based on the bit rates of FGS bitstreams 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 bit rate to each VOP with the underlying weighting function \( R_i(t) = w_i N_i + w_P P_i + w_B B_i \), 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 at one second, 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.

We put the packets in even-spaced time intervals into the channels 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.

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.

---

**Figure 1.** Architecture and network connection setup of the proposed test bed system.

<table>
<thead>
<tr>
<th>Application Control</th>
<th>Layered Video data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Commands</td>
<td>Base layer</td>
</tr>
<tr>
<td></td>
<td>Enhancement layer</td>
</tr>
<tr>
<td>RTP</td>
<td>RTP</td>
</tr>
<tr>
<td>TCP</td>
<td>UDP with retransmission</td>
</tr>
<tr>
<td></td>
<td>UDP</td>
</tr>
<tr>
<td>IP</td>
<td>IP</td>
</tr>
</tbody>
</table>

**Figure 2.** Network protocol stack

**Figure 3.** Illustration of (nonstandard) RTP header.

Step 1: Packet collection:
Each packet is identified with its “sequence number” from the RTP header. Since the arrival time of each packet may not be in the order of sequence number, we use the first second of the 3 seconds to wait for the packet arrival.

Step 2: Packet loss detection:
Within the packet buffer, we have flags to register the successful arrival of every packet. The loss detection procedure is activated periodically each second to balance the overhead and QoS status. As the flag of a packet is triggered as “OFF” in this period, the packet loss is detected.

Step 3: Retransmission:
When packet loss is detected, the receiving controller sends a retransmission request with the sequence number of the packet. We use the remaining 2 seconds to wait for the requested packet to arrive at the packet buffer. As the 2 seconds nearly expire, the retransmitted packets that have not arrived at the packet buffer in time are declared as permanent loss. To avoid the retransmitted packets to occupy the effective bandwidth of the normal media delivery, any lost packet may have \( N \) times chances for retransmission, where \( N \) is set as 3 empirically.

To decode and display the bitstream corrupted by packet loss, 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 [11]. 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 following 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 [12].

2.3 Network Interface

To link the server, client and transmission channel, the 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 [9]. 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) [13] are used to provide an end-to-end network transport for video streaming. The session control protocol use the real-time streaming protocol (RTSP) [14] that specifies the messages and procedures to control the media delivery during an established session. Since there is no standard that specifies ways to deliver FGS bitstream via RTP and to support MPEG-21 DIA via RTSP over IP, we use nonstandard RTP and RTSP headers for FGS-based video streaming. Figure 3 shows the (nonstandard) RTP-header used in this system.

For simplicity, basic client-server RTSP message exchange and RTSP retransmission request are employed for control the delivery. There are four basic client-server RTSP messages including DESCRIBE, SETUP, PLAY, and TEARDOWN. For an instance in Figure 4, 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. Based on 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 requests GET_PARAMETER 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 with the time stamp, sequence number, and payload length in the RTP Mux module prior to the transport. The RTP packets are then transmitted using UDP over IP, and received and de-multiplexed by RTSP DeMux into the packet buffer at the clients for playback. The RTSP packets of the control messages are transported via TCP over IP. Additionally, for the DIA descriptions, an XML parser and an XML compositor are included under RTSP DeMux and Mux modules, respectively.

For simplicity, basic client-server RTSP message exchange and RTSP retransmission request are employed for control the delivery. There are four basic client-server RTSP messages including DESCRIBE, SETUP, PLAY, and TEARDOWN. For an instance in Figure 4, 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. Based on 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 requests GET_PARAMETER 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 with the time stamp, sequence number, and payload length in the RTP Mux module prior to the transport. The RTP packets are then transmitted using UDP over IP, and received and de-multiplexed by RTSP DeMux into the packet buffer at the clients for playback. The RTSP packets of the control messages are transported via TCP over IP. Additionally, for the DIA descriptions, an XML parser and an XML compositor are included under RTSP DeMux and Mux modules, respectively.

To emulate a realistic network conditions for FGS-based scalable streaming scheme, we adopt the NISTnet network emulator [8], which is a shareware and can provide the easy control methods to generate required and identical network conditions such as packet ratio, jitter, bandwidth variation, and delay. The time-varying network conditions are pre-specified as a network profile as illustrated in Figure 5. With the network profiles defined by users, the same test conditions can be used for advanced comparisons.

3. Experimental Results

According to the requirements from the MPEG committee [6], [7], the test bed is set up as in Figure 1 (b). Based on this test bed, the scalable and single layer bitstreams can be transported 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 QCIF and YUV format. Each sequence has 300 frames and the

```
<xml encoding="UTF-8">
  <DIDL xmlns="urn:mpeg:mpeg21:2002:01-DIDL-NS">
    <UserCharacteristics>
      <User>
        <Account>johndoe</Account>
        <Password>@%FHG%^&SS</Password>
      </User>
    </UserCharacteristics>
    <TerminalCapabilities>
      <Decoding>
        <Format>FGS</Format>
      </Decoding>
      <Hardware>
        <ScreenSize>
          <Height>288</Height>
          <Width>352</Width>
        </ScreenSize>
      </Hardware>
    </TerminalCapabilities>
    <DIDL/>
  </DIDL>
</xml>
```

Figure 4. Illustration of the client-server RTSP DESCRIBE message with MPEG-21 DIA description.

# A Simple Network Profile

<table>
<thead>
<tr>
<th>#</th>
<th>Time (sec)</th>
<th>PLR (%)</th>
<th>Bandwidth (bps)</th>
<th>Mean Delay (ms)</th>
<th>SDD</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>100000</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>20</td>
<td>0</td>
<td>50000</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>500</td>
<td>0</td>
<td>200000</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Figure 5. Illustration of network profile used to control NISTNet. PLR indicates the packet loss ratio and SDD means the standard deviation of the delay.

2.4 Network Simulator

To emulate a realistic network conditions for FGS-based scalable streaming scheme, we adopt the NISTnet network emulator [8], which is a shareware and can provide the easy control methods to generate required and identical network conditions such as packet ratio, jitter, bandwidth variation, and delay. The time-varying network conditions are pre-specified as a network profile as illustrated in Figure 5. With the network profiles defined by users, the same test conditions can be used for advanced comparisons.

3. Experimental Results

According to the requirements from the MPEG committee [6], [7], the test bed is set up as in Figure 1 (b). Based on this test bed, the scalable and single layer bitstreams can be transported 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 QCIF and YUV format. Each sequence has 300 frames and the
Table 1. PSNR of Y component for the both sequences under various target bit rates.

<table>
<thead>
<tr>
<th>Seq.</th>
<th>Total bit rate (kbps)</th>
<th>PSNR (dB)</th>
<th>Total bit rate (kbps)</th>
<th>PSNR (dB)</th>
<th>Total bit rate (kbps)</th>
<th>PSNR (dB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Foreman</td>
<td>64</td>
<td>27.9</td>
<td>128</td>
<td>26.8</td>
<td>256</td>
<td>25.7</td>
</tr>
<tr>
<td>News</td>
<td>64</td>
<td>30.0</td>
<td>128</td>
<td>32.5</td>
<td>256</td>
<td>33.7</td>
</tr>
</tbody>
</table>

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

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 Server, Video Clients, Network Interface, and Network Simulator. Even the functionalities of the proposed system are limited, this test bed is currently considered for gathering the evidence on scalable coding 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.

5. REFERENCES