Real-time Scheduling and On-line Resource Allocation on Scalable Streaming Media Server

Kui Gao[1,2], Wen Gao[1,2], Simin He[1], Yuan Zhang[2,3]
[1]Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100080, China
[2]Graduate School, Chinese Academy of Sciences, Beijing 100039, China
[3]Beijing Broadcasting Institute, Beijing 100024, China
E-mail:{kgao, wgao, smhe, yzhang}@jdl.ac.cn

ABSTRACT

In this paper, we propose a layer-based integrated real-time scheduling algorithm in a single scalable stream and an on-line dynamic resource allocation algorithm among multiple concurrent users for scalable streaming media server over a network with packet loss and variable delay. The layer-based real-time scheduling algorithm efficiently schedules the packets in the buffer of the scalable streaming media server for transmission. The on-line resource allocation algorithm can allocate the server's resource among all the concurrent streams fairly and improve the playback quality in client. Simulation results show that our proposed algorithms outperform the frame-based scheduling algorithm and the off-line resource allocation algorithm in various situations with different round-trip times, channel errors, etc. The low complexity of the proposed algorithms also enables them to be applied in real-time applications.

Keywords: real-time scheduling; scalable streaming; resource allocation; fine granularity scalable (FGS); streaming server

1. INTRODUCTION

Recent advances in multimedia computing and communication technologies have made it feasible to provide real-time media streaming services over the Internet. Due to the wide variation of available bandwidth and transmission errors over the Internet and the variety of end user devices, it is desirable to design a media streaming–oriented coding scheme that can adapt to the channel conditions and the user devices. The server should be able to adapt to the timing constraints of streaming media data and the time-varying network conditions. If the media data does not arrive at user end in time, the playback will be paused, which is annoying to human ears and eyes. Scalable/layered encoding has been believed to be promising to cope with the heterogeneity of user access rates and fluctuation of available bandwidth in video streaming. For instance, the MPEG-4 FGS coding scheme, which has been accepted as a part of MPEG-4 standard [1][2], further provides fine granular scalability. It consists of one non-scalable coded base layer and one or multiple bitplane-encoded scalable enhanced layers. Fine granularity is implemented by decoding the enhancement stream at any point. It differs from all the previous layered video coding schemes where only limited layers are available. Another advantage of FGS is that the bit rate can be adjusted at transmission time with very fine granularity and very little complexity.

In a streaming media system, the server packetizes the coded scalable/layered streams into some packets and then sends them to the client through various networks. Bitstreams at different layers have different contributions to the playback quality obtained in client. Therefore, it is an important problem how we select and schedule packets delivery of a scalable streaming media over a lossy network. The optimized scheduling of layered streaming media delivery was first proposed by Podolsky et al. [4], who adopt the Markov chain to analyze and find the optimal packets transmission and retransmission policies. Chou et al. [5] and Miao et al. [6] also addressed the same problem with a rate-distortion analysis.

The resources in a streaming server include CPUs, memories, and storage devices [7]. Since the server resources, especially the bandwidth or throughput of the server, are limited, only a limited number of concurrent clients with the QoS requested can be served. It is very important to fairly allocate the server resource to multiple concurrent scalable video streams. Floyd et al. [8], Arulambalam et al. [9] and Zhang et al. [10] proposed network bandwidth allocation algorithms, but they did not take into account the limitation of the server resources.

In this paper, we propose a layer-based integrated real-time packet scheduling algorithm for a single scalable stream and an on-line resource allocation algorithm among multiple concurrent streams for the scalable streaming media server over network with packet losses and delay variations. The real-time packet scheduling algorithm, by determining how to transmit/retransmit the packets subjects to a given time, improves the utility of the bandwidth and smoothes the playback quality in client. The on-line resource allocation algorithm allots the server resource to the concurrent streams fairly and dynamically, and improves the total playback quality in client.

The rest of this paper is organized as follows. Section 2 briefly introduces the framework of FGS video coding and the architecture of the scalable streaming system. Section 3 presents the layer-based scheduling algorithm for the scalable layered streaming. Section 4 describes the on-line fair resource allocation algorithm in a streaming server. Section 5 gives some experimental results and comparisons among different algorithms. Section 6 concludes this paper.

2. SCALABLE VIDEO CODING AND ARCHTECTURE OF STREAMING SYSTEM

2.1. THE FGS VIDEO CODING

In response to the increasing demand on streaming video applications over the best-effort Internet, the coding objective for streaming video is changed to optimize the video quality for a wide range of bit rates. Fine granularity scalable (FGS) video coding [1][2] has been accepted by MPEG-4 as an amendment to the traditional non-scalable MC-DCT approach for streaming video profile. The basic idea of FGS video streaming is to code a raw video sequence into a base layer substream and one or multiple enhancement layer substreams. An FGS encoder, using the motion-compensated DCT coding to be compatible with other standards, such as MPEG-2, MPEG-4, H.263 and H.264, etc., generates a base-layer video to reach the lower bound of the bit-rate range. Then the encoder uses bitplane coding to represent the enhancement streams. The enhancement layer is to code the difference between the original picture and the reconstructed picture using bit-plane coding of the DCT coefficients. The bitstream of the FGS enhancement layers may be truncated into any number of bits per picture/frame after encoding is completed. The decoder should be able to reconstruct an enhancement video from the base layer and the truncated enhancement-layer bitstreams. The enhancement-layer video quality is proportional to the number of bits decoded by the decoder for each picture/frame. Figure 1 shows conceptually such a framework. In an FGS coding scheme, the base layer and all enhancement layers in predicted frame are always predicted from the reconstructed version of the base layer in the reference frame. The fine scalable characteristic of FGS is very important, since the same content can be accessed over heterogeneous network by various receivers with different computing power, memory, display resolutions, etc.

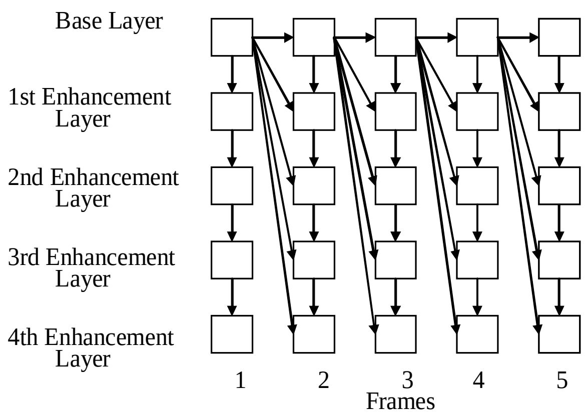

Figure 1. The FGS framework

2.2. THE ARCHITECTURE OF A SCALABLE STREAMING SYSTEM

A typical streaming system consists of clients and servers over a network. Figure 2 shows the architecture of a scalable streaming media system. Each client may make real-time requests for scalable streaming. The client requests are sent to the server via network connections, which also serve for transmission of media data. To satisfy the performance requirements of each client, a scalable streaming server must employ an admission control algorithm to determine whether the server can guarantee the QoS (Quality of Service) requirements of a new client without violating the performance requirements of the clients already being serviced. If a new request is admitted, the server will read the data from the storage devices, packetize them, and feed them into the server's transmission buffers. The server selects one packet at a time from those buffers and sends it over the lossy channel. Some packets may be lost, damaged or delayed (delayed packets are also considered lost if they exceed their playback delay). At the client end, the lost or damaged packets are reported to the server via a feedback channel. For a video streaming session, it is desirable to adjust its sending rate according to the perceived congestion level in the network and the resource available in the server. Through this adjustment, a suitable loss level can be maintained and resources of network and server can be shared fairly among connections. The receiver monitors the network condition and gathers related information; while the sender changes its sending rate according to the available network bandwidth estimated from the packet loss rate, RTT (round-trip-time), and RTO (retransmission timeout) values. A retransmitted packet typically has an extra delay of one or more RTTs, and cannot be guaranteed to arrive at the client on time. In addition, even if there is still time to retransmit at a given time, a decision needs to be made on whether this packet should be retransmitted or not. Therefore, a streaming server needs to simultaneously provide services to multiple concurrent users and guarantee the quality of service for each client. It must efficiently perform two tasks: (1) the scheduler determines the order of serving the set of requests from all the data in the transmission buffers; and (2) the resource allocator allocates the server resource to concurrent users according to the network and server conditions. We adopted a real-time packet scheduling algorithm for the scheduler and an on-line resource allocation algorithm for the resource allocator. The details are described as following.

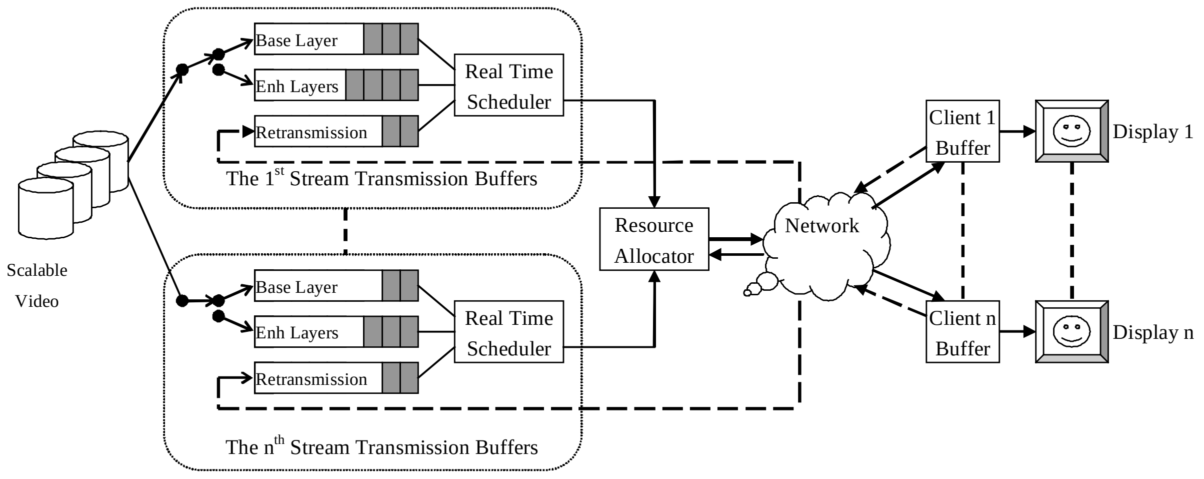

Figure 2. The architecture of FGS video streaming system

3. LAYER-BASED INTEGRATED REAL-TIME SCHEDULING ALGORITHM

Scalable streaming media have timing constraints because of their sensitivity to delay and jitter. Retransmission can be used to recover the lost packets over a best-effort network [7]. We want to find a packet transmission policy to select the packets to be transmitted or retransmitted at any given time during a streaming session, so to improve the playback quality in client. Due to the delivery deadline constraint, not all lost packets can be recovered by retransmission. However, if the server schedules a packet to be sent much earlier than its playback time, this packet will have more chances to be retransmitted before it is too late for display. If a packet is not available at its expected display time at the receiver, it will miss its deadline. In addition, even if there is still time to retransmit a packet at a given time, a decision needs to be made on whether it should be retransmitted or not. Just as the description in Subsection 2.1, the base layer carries the most important information within the FGS coding scheme. The base layer bitstream is very sensitive to the channel error. If the packets of the base layer are lost, the quality of reconstructed video can be degraded severely. As for the enhancement layers, the residue between the original image and the reconstructed image of base layer is compressed with bit plane coding technique to form the enhancement bitstream. Since the bit plane coding produces an embedded bitstream with fine granularity scalability, the enhancement bitstream can be arbitrarily truncated to fit the available channel bandwidth and tolerate the channel errors. Real-time streaming playback requires the server to transmit the data packets prior to their playback instants (i.e., deadlines). Packets scheduling algorithms with real-time deadlines, such as Earliest Deadline First (EDF) [11] can optimize the quality of the video reconstructed in client. EDF scheme transmits the packets in the order of their deadlines but cannot consider the features of the fine scalable video stream. Therefore, we propose a layer-based integrated real-time scheduling algorithm to send the packets of a scalable streaming.

The layer-based integrated real-time scheduling algorithm adopts different scheduling scheme for different layer. The base layer bitstream is the most important information to reconstruct the video, so we adopt hard real-time scheduling [12] scheme to send the packets for the base layer; while the enhancement layer bitstreams give the enhancement quality and packets loss does not give most degraded quality, so we adopt soft real-time scheduling scheme [12].

The loss impact of lower enhancement layer within a frame on video quality is much greater than that of the higher enhancement layers within it. As different layer has a different effect on the playback, we set the higher priority to the lower (more important) layer packets and set the lower priority to the higher (less important) layer. In fact, the priority of the packet is consistent with the distortion. Since it is very complex to compute and compare the distortion of every packet, we set different priority to different layer instead of distortion roughly. It simplifies the determination of how to order the packets in the transmission buffers. The layer-based integrated real-time scheduling algorithm combines considerations of layer distortion and deadlines. Thus, base layer and important enhancement layers data can be transmitted earlier. If it is lost, it can have more chances to be retransmitted. If several packets in the transmission buffer belong to the same layer, packets with earliest deadline are served first. The server transmits the lower (more important) layer packets in the transmit buffer as soon as possible. The algorithm not only improves the utility of the bandwidth but also smoothes the playback quality.

With the FGS coding scheme, the video content can be compressed over any desired the base layer bitrate range R_{BL} and the enhancement layer bitrate R_{EL}. The total bitrate is $R_{total} = R_{BL} + R_{EL}$. When the i^{th} streaming is admitted, the server will set its sending rate $R_{s,i}(t)$ to be no less than $R_{BL,i}(t)$ (i.e. $R_{BL,i}(t) \leqslant R_{s,i}(t)$) in the interval $[t, t+T)$, where T is the scheduling period. Subsequently, the server transmits packets of the enhancement layer and retransmits the lost packets using a bit-rate $R_{soft,i}(t) = R_{s,i}(t) - R_{BL,i}(t)$. So the scheduling task can be divided into two parts. The server adopts hard real time scheduling for base layer at a sending rate of $R_{hard,i}(t) = R_{BL,i}(t)$ in the interval $[t, t+T)$ and soft real-time scheduling for packets of enhancement layers and retransmitted packets at a sending rate $R_{soft,i}(t)$.

Let $p_{m,n}$ denote the packet of the n^{th} layer in frame m. The packets are put into the transmission buffers according to the decoding order. The release-time $a_{m,n}$ is the earliest time at which the packet $p_{m,n}$ becomes ready for scheduling in the transmission buffer. But the packets of different layers in the same frame have different release-time. The release-time $a_{m,0}$ of the base layer packet $p_{m,0}$ is earlier than other packets of enhancement layers and the packet $p_{m,0}$ will have more chances to be retransmitted before it is too late for display. As for the enhancement layer, the packets of more important layers have earlier release-time than those of less important layers (i.e. $a_{m,k} \leqslant a_{m,l}$, $(k \leqslant l)$). Deadline $d_{m,n}$ is the latest time at which the packet $p_{m,n}$ should be sent to the client, otherwise it is too late for playback. We assume that different layers in a frame have the same deadline d_m. The schedule-time $s_{m,n}$ is the time at which the scheduler sends packet $p_{m,n}$ to client. The RTT (round-trip-time) is defined as the interval from the time a packet is sent from the server to the time the server gets feedback of this packet from the client. The packet loss probability over the lossy channel is ε. Information related to RTT and channel error ε can be received by server via client's feedback. The size of the packet $p_{m,n}$ is $b_{m,n}$. The processing time of the packet $p_{m,n}$ is

$$c_{m,n} = \begin{cases} b_{m,n} / R_{hard,i}(t) & if \quad n = 0 \quad and \quad it \quad is \quad scheduled \quad at \quad first \quad time \\ b_{m,n} / R_{soft,i}(t) & others \end{cases}.$$

The fulfill-time of a packet $p_{m,n}$ is $f_{m,n} = s_{m,n} + c_{m,n}$. The decoding time is dt_m. A packet $p_{m,n}$ is ready for scheduling if the following conditions are satisfied: the current time t_{cur} ($t_{cur} \in [t,t+T)$)is later than its release-time $a_{m,n}$, and its fulfill-time $f_{m,n}$ is earlier than its deadline, *i.e.*, $a_{m,n} \leqslant t_{cur}$ and $t_{cur} + c_{m,n} \leqslant d_m$.

The precise description of layer-based real-time integrated scheduling algorithm for delivery of scalable streaming media over a lossy network is given below.

Layer-based real-time integrated scheduling algorithm:

Step 1: Get the current sending rate $R_{s,i}(t)$ from the resource allocator; compute the sending rate of the base layer; Let $R_{hard,i}(t) = R_{BL,i}(t)$, then $R_{soft,i}(t) = R_{s,i}(t) - R_{hard,i}(t)$;

Step 2: Let $P_{BL}(t)$,$P_{EL}(t)$ and $P_{RE}(t)$ be the sets of ready packets with earliest deadline in the buffers of base layer, enhancement layer and retransmission packets at the current time t_{cur} ($t \le t_{cur} < t+T$) of the i^{th} streaming, respectively. Compare t_{cur} with the deadline d_m of the packet $P_{m,n}$ in the server transmission buffer. If the packet deadline $d_m > t_{cur}$, remove the it from the buffers;

Step 3: Schedule and send the packets from $P_{BL}(t)$ at the sending rate $R_{hard,i}(t)$. And set the timeout of this packet $t_{m,n} = t_{cur} + RTO$;

Step 4: Select the packets from $P_{EL}(t)$ and $P_{RE}(t)$ with the lowest (most important) layer and send them to the client via the network at the sending rate $R_{soft,i}(t)$. And set the timeout $t_{m,n} = t_{cur} + RTO$;

Step 5: If the server gets acknowledgement (ACK) of a packet from the client, remove the packet from the buffer; else if a packet reaches its timeout $t_{m,n}$, move it from transmission buffer into retransmission buffer, and set its release time as the current time t_{cur}; Go to Step 1.

In the layer-based real-time integrated scheduling algorithm, the step 3 and step 4 are performed in parallel. Since $R_{hard,i}(t) = R_{BL,i}(t)$, all the packets in the base layer buffer can be sent and no packet misses the deadline. But some packets of enhancement layer and retransmission buffers are discarded if they miss their deadline. The algorithm transmits the most important packets to reconstruct the playback quality as soon as possible, so it not only improves the utility of the bandwidth but also smoothes the playback quality.

4. FAIR RESOURCE ALLOCATION ALGORITHM

Resources in streaming server include CPUs, memories, buses and storage devices. Since resources are limited, the streaming server can only support a limited throughput (i.e. server bandwidth). It is very important to fairly allocate the server resource, especially the server bandwidth, to multiple concurrent scalable video streaming. Different user has different variation of network bandwidth and bit rate of playback stream. Therefore, traditional off-line algorithms such as round robin, priority-based scheduling cannot guarantee a large number of concurrent accesses fairly. To compensate for the unpredictability and variability in particular streaming session, we adopt an on-line server resource allocation algorithm for scalable video streaming server to allocate server bandwidth. The resource allocation algorithm is responsible for allocating the fair share of the server resource among all concurrent connections simultaneously.

For the i^{th} streaming session, an estimate for the available network bandwidth $R_{net,i}(t)$ can be generated in the interval $[t,t+T)$ by the TFRC protocol [8] in the streaming system. The total bitrate (including base layer and enhancement layers) of the streaming is $R_{total,i}(t)$. $R_{loss,i}(t)$ is the rate for sending all the retransmission packets before their deadline in the interval $[t, t+T)$. The i^{th} streaming maximal available sending rate $R_{avi,i}(t) = \min(R_{total,i}(t) + R_{loss,i}(t), R_{net,i}(t))$ without limitation of the server resource. If the total available sending rate of all streaming is less than the server bandwidth U_s, every streaming can sent the data at it's sending rate $R_{avi,i}(t)$. If the total sending rate is larger than the server bandwidth U_s, we adopt a dynamic allocation scheme. The server find the minimum sending rate $R_{min\_avi}(t)$ from all the streaming sessions and allocate it to all the streaming. If the total allocation sending rate is larger than the server bandwidth U_s, the server will allocate equal bandwidth to each streaming. Otherwise, the server only allocates the minimum bandwidth $R_{min\_avi}(t)$ to the streaming of the minimum sending rate. Then the server allocates the residual server bandwidth to the others iteratively. The on-line resource allocation algorithm is given as following.

On-line resource allocation algorithm:

Step 1: Let the set of the clients in a streaming server be $CL(t)=\{C_1, C_2, \ldots, C_i, \ldots, C_k\}$. Let the set of the estimated available network bandwidth of all clients in $CL(t)$ be $N(t)=\{R_{net,1}(t), R_{net,2}(t), \ldots, R_{net,k}(t)\}$ in the interval $[t, t+T)$. $R_{net,i}(t)$ ($1 \leqslant i \leqslant k$) can be obtained by TFRC. The set of the total bitrate (including substreams of base layer and enhancement layers) of all the streaming is $B(t)=\{R_{totat,1}(t), R_{total,2}(t), \ldots, R_{total,k}(t)\}$. The set of rate for sending all the retransmission packets before their deadlines in the interval $[t, t+T)$ is $L(t)=\{R_{loss,1}(t), R_{loss,2}(t), \ldots, R_{loss,k}(t)\}$.

Step 2: Let the set of the maximal available sending rates is $AVI(t)=\{R_{avi,1}(t), R_{avi,2}(t), \ldots, R_{avi,n}(t)\}$.
$R_{avi,i}(t) = \min(R_{total,i}(t) + R_{loss,i}(t), R_{net,i}(t))$ ($1 \leqslant i \leqslant k$) .

Step 3: Let U_s is the available bandwidth of the server.
If $\sum_{i=1}^{k} R_{avi,i}(t) \le U_s$, $R_{s,i}(t)=R_{avi,i}(t)$ ($1 \leqslant i \leqslant k$), go to step1 waiting for next internal $[t+T, t+2T)$.

Step 4: Select the minimal available sending rate $R_{min\_avi}(t) = R_{avi,i}(t)$ ($1 \leqslant i \leqslant k$) from set $AVI(t)$.

If $R_{min\_avi}(t)*k \geqslant U_s$, set $R_{s,1}(t) = R_{s,2}(t) = \ldots = R_{s,k}(t) = U_s/k$. Go to step1, waiting for next internal $[t+T, t+2T)$.

Else, $R_{s,i}(t) = R_{min\_avi}(t)$, remove C_i from $CL(t)$. Go to step1.

The on-line resource allocation algorithm not only adapts the bandwidth fluctuations dynamically and utilizes the resource for the streaming fairly, but also improves the total playback quality in client.

5. SIMULATION RESULTS

A two-state Markov model proposed by Gibert [13] is used to simulate packet losses in Internet channel. This model can characterize the error sequences generated by data transmission channels. In good state (G) errors occur with low probability while in bad state (B), they occur with high probability. The errors occur in cluster or bursts with relatively long error free intervals (gaps) between them. The state transitions are shown in Figure 3 and summarized by the following transition probability matrix:

$$P = \begin{bmatrix} 1-\alpha & \alpha \\ \beta & 1-\beta \end{bmatrix}.$$

The average packet loss rate is:

$$\varepsilon = \frac{\alpha}{\alpha+\beta}.$$

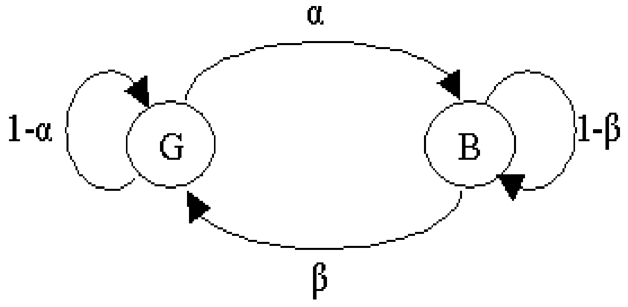

Figure 3. Two-state Markov model for the network simulation.

Details of the model can be found in [13]. It is assumed that the sending rate can be decided by TCP-friendly Rate Control (TFRC) protocol. The receiver monitors the network condition and gathers related information, while the sender changes its sending rate according to the available network bandwidth estimated from the packet loss rate, round trip time, and retransmission timeout values. The protocol uses an equation-based way to estimate available bandwidth [8]:

$$R_{net} = \frac{s}{R\sqrt{\frac{2p}{3}} + t_{RTO}(3\sqrt{\frac{3p}{8}})p(1+32p^2)},$$

where s is the packet size, R is the round trip time, t_{RTO} is the retransmission timeout value, and p is the packet loss ratio. The sending rates in different RTTs and loss rates simulated by ns-2 [15] are shown in Table1 and Figure 4.

Table1. The average available network bandwidth with different RTTs and loss rates (Kbytes/sec)

Loss rate	0.5%	1%	2%	5%	10%
RTT=20ms	1010.359	662.8832	397.6093	172.0611	70.53102
RTT=40ms	538.0144	368.3213	238.6904	120.793	51.11024
RTT=80ms	283.7569	197.9084	133.8039	70.25274	34.90544
RTT=120ms	194.0224	135.8944	92.69293	50.98491	27.69494
RTT=160ms	146.6914	102.1699	69.22119	38.96971	21.51618

The MPEG-4 FGS-MoMuSys encoder/decoder [14] is used in the simulation. The base layer is encoded with MPEG-4, and the enhancement layer is encoded with FGS coding. Extensive simulations have been performed to test the performance of the proposed algorithms. The sequences Foreman, Coastguard and Akiyo in CIF format are used in the simulation. They are encoded with 30 frames per second and 300 frames are encoded and transmitted. For example, the maximum level of bitplane is 7 in the sequence Foreman, so there are 7 Enhancement layer. Different bitplane has different size. The enhancement layer 0 (EL0) has the smallest size, yet it is most significant. The enhancement layer 6 (EL6) is the largest in size, yet it is the least significant. The average rate of base layer is 173.54 Kbps and average rate of all enhancement layers is 18,035.59 Kbps. The packet size is 1024 bytes. The playback frame rate is 30 Hz.

In our simulations, we assumed that the server maximal throughput (i.e. the server bandwidth) is 100Mbps and the server support 80 concurrent clients simultaneously. The channel packet loss rate varies from 0.5% to 10% and the RTT varies from 20ms to 160ms. The utility of the server bandwidth is 100% by on-line resource allocation algorithm, while the utility of the server bandwidth is 66.51% by round-robin resource allocation algorithm. The playback quality is measured by PSNR of the video frames reconstructed in client based on all available packets.

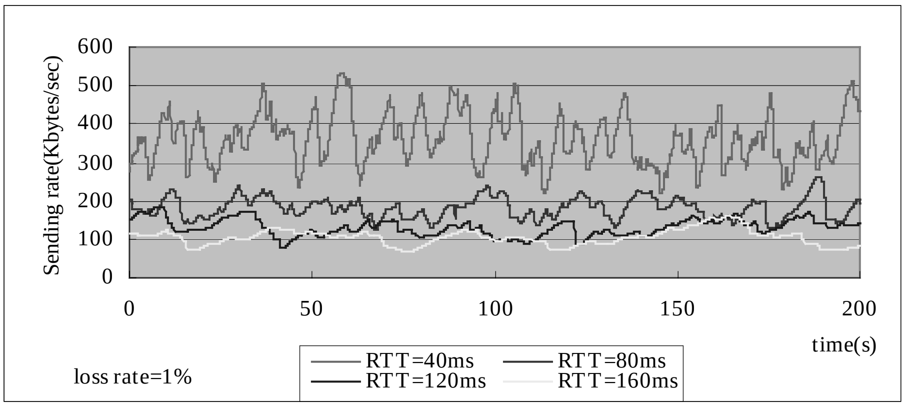

Figure 4.Comparisons of available bandwidth for different RTT connections

Table 2 shows the average PSNR for different sequences using different resource allocation algorithms under various packet loss ratios and different sequences, where each result is obtained by averaging about 300 frames. The layer-based integrated real-time scheduling algorithm (LBRT) is adopted to schedule the packets. It can be seen that, overall, on-line resource allocation algorithm outperforms off-line round-robin algorithm. Figure 5 shows comparisons of PSNR with different scheduling algorithms and resource allocation algorithms. Obviously, the layer-based integrated real-time scheduling algorithm improves the utility of the bandwidth and smoothes the playback quality.

Table 2. Comparison performance (average PSNR in dB, RTT=20ms) with different loss rates and different sequences (OLRA: on-line resource allocation algorithm. RR: round-robin resource allocation algorithm)

		Foreman (CIF)		Coastguard (CIF)		Akiyo (CIF)	
		OLRA	RR	OLRA	RR	OLRA	RR
Loss rate	0.5%	41.619	36.993	41.518	37.77	43.881	41.461
	1%	41.565	36.971	41.454	37.754	43.836	41.457
	2%	39.734	36.889	40.007	37.733	42.529	41.47
	5%	36.846	36.263	37.691	37.327	41.577	41.388
	10%	33.748	33.748	35.457	35.457	40.792	40.792
AVG		38.7024	36.1728	39.2254	37.2082	42.523	41.3136

6. CONCLUSION

In this paper, we propose a layer-based integrated real-time scheduling algorithm and an on-line resource allocation algorithm for scalable media streaming server. The layer-based real-time scheduling algorithm is efficient and simple for delivery of scalable streaming media over a lossy network. The on-line resource allocation algorithm can adjust the sending rate dynamically and fairly according to the network and server status and improves the playback quality in client. The simulation results show that the layer-based real-time scheduling algorithm and the on-line resource allocation algorithm outperform the frame-based scheduling algorithm and the off-line (round-robin) resource allocation algorithm in various situations with different RTTs, channel errors, etc. The studies of admission control, buffer management, data storage and retrieval for scalable streaming server over the Internet are interesting topics for future work.

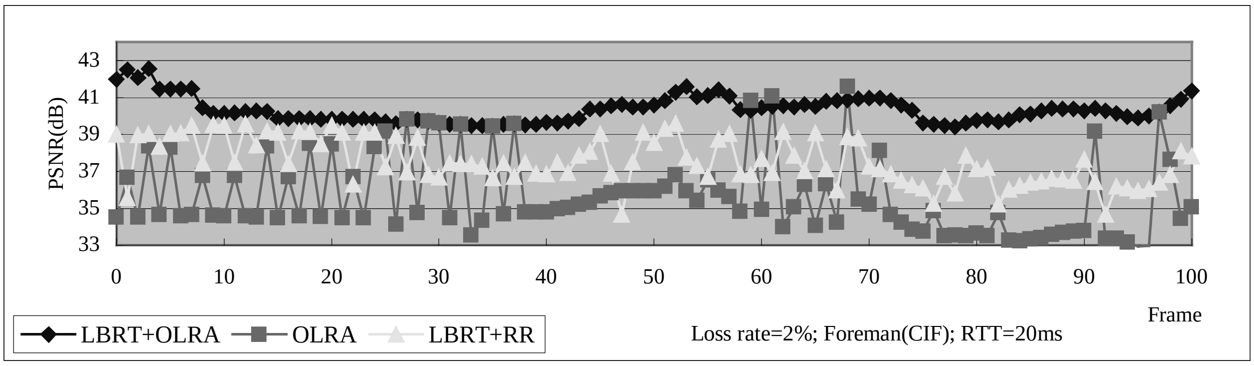

Figure 5. The PSNR comparisons of different algorithms

7. ACKNOWLEDGMENT

This work was supported by the National Fundamental Research and Development 973 Program of China under grant No. 2001CCA03300; the National High Technology Development 863 Program of China under grants No. 2001AA112100 and No. 2002AA119010; the National Natural Science Foundation of China under grant No. 69983008; the Knowledge Innovation Program of Chinese Academy of Sciences under grant No. KGCXZ-103.

8. REFERENCES

1. W. Li, “Overview of fine granularity scalability in MPEG-4 video standard”, *IEEE Transactions on Circuit and System for Video Technology*, vol. 11, no. 3, pp. 301-317, March 2001.
2. W. Li, “Fine granularity scalability in MPEG-4 for streaming Video”, *IEEE international symposium on Circuit and System (ISCAS 2000)*, pp. 299-302, Geneva, May 2000.
3. F. Wu, et al, “A framework for efficient progressive fine granularity scalable video coding”, *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 11, no. 3, pp. 332-344, March 2001.
4. M. Podolsky, S. McCanne and M, Vetterli. “Soft ARQ for layered streaming media”, *Journal of VLSI Signal Processing Systems, Special issue on multimedia signal processing*, vol. 27, no. 1-2, pp. 81-97, February 2001.
5. P. A. Chou and Z. Miao, "Rate-distortion optimized streaming of packetized media," *IEEE Transactions on Multimedia*, February 2001 (submitted), online available at: http://research.microsoft.com/~pachou/.
6. Z. Miao and A. Ortega. “Expected run-time distortion based scheduling for delivery of scalable media”, *The 12th International Packet Video Workshop (PVW 2002)*, Pittsburgh, PA, April 2002.
7. D. Wu, et al, “Streaming video over the Internet: approaches and directions”, *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 11, no. 3, pp. 282-300, March 2001.
8. S. Floyd, et al, “Equation-based congestion control for unicast applications”, *SIGCOMM 2000*, February 2000.
9. A. Arulambalam, X. Chen and N. Ansari, “Allocating fair rates for available bit rate service in ATM networks”, *IEEE Communications Magazine*, vol. 34, no. 11, pp. 92-100, November 1996.
10. Q. Zhang, W. Zhu, and Y-Q. Zhang, ”Resource allocation for multimedia streaming over the Internet”, *special issue on Multimedia over IP in IEEE Translations on Multimedia*, vol. 3, no. 3, pp. 339-355, September 2001.
11. C. Liu and J. Layland, “Scheduling algorithms for multiprogramming in hard-real-time environment,” *Journal of the Association for Computing Machinery*, vol. 20, no. 1, pp. 46-61, January 1973.
12. G.C. Buttazzo, “Hard real-time computing systems: predictable scheduling algorithms and applications”, Kluwer Academic Publishers, 1997.
13. J.R. Yee and E. J. Weldon. “Evaluation of the performance of the error-correcting codes on a Gilbert channel”, *IEEE Transactions on Communications*, vol. 43, no. 8, pp. 2316-2323, August 1995.
14. “ISO/IEC 14496-5:2001/FDAM1”, ISO/IEC JTC1/SC29/WG11/N4711, Jeju Island, March 2002.
15. ns-2 network simulator. http://www.isi.edu/nsnam/ns, 2002.