STREAMING LAYERED ENCODED VIDEO USING PEERS

Yanming Shen, Zhengye Liu, Shivendra S. Panwar, Keith W. Ross, Yao Wang

Polytechnic University
Brooklyn, NY 11201

ABSTRACT

Peer-to-peer video streaming has emerged as an important means to transport stored video. The peers are less costly and more scalable than an infrastructure-based video streaming network which deploys a dedicated set of servers to store and distribute videos to clients. In this paper, we investigate streaming layered encoded video using peers. Each video is encoded into hierarchical layers which are stored on different peers. The system serves a client request by streaming multiple layers of the requested video from separate peers. The system provides unequal error protection for different layers by varying the number of copies stored for each layer according to its importance. We evaluate the performance of our proposed system with different copy number allocation schemes through extensive simulations. Finally, we compare the performance of layered coding with multiple description coding.

1. OVERVIEW OF PROPOSED PEER-DRIVEN VIDEO STREAMING SYSTEM

In a conventional video on demand system, videos are stored in a dedicated set of servers. When a user requests a video, the network would redirect the client to one or more of its dedicated servers, which would in turn stream the video to the client. But such infrastructure-based architecture would be prohibitively costly, both in terms of server cost and Internet connection cost. On the other hand, in a peer-to-peer (P2P) based video streaming network, users' peers (ordinary computers) store and stream the video to the requesting clients. Some of these peers will be connected to the Internet via residential access networks such as DSL and cable; other peers may be connected via higher speed connections (e.g., DS3 or OC3). Importantly, the cost of these peers and the Internet access would be borne by the users rather than by companies providing the service.

However, for high-quality video streaming applications, the video rate may exceed the uplink bandwidth of most server peers. In order to circumvent the problem of limited uplink bandwidth at network access, we encode each video into multiple substreams and place each substream on a different peer. Each substream has a rate much lower than the total rate of the video, thus reducing the required uplink bandwidth at each peer. In our peer-to-peer video streaming systems, users request to view videos on demand. After a user makes such a request, the system attempts to set up a session, which consists of multiple sub-streams sent from different server nodes to the user. When a server peer disconnects during a streaming session, the system will search for a replacement peer which has the same video substream and sufficient uplink bandwidth. Figure 1 illustrates the proposed system architecture.

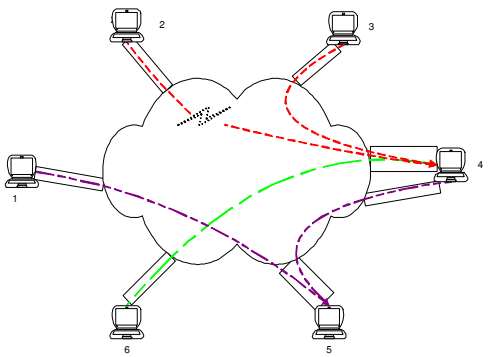

Fig. 1. In this example, nodes 4 and 5 are each receiving a video. Initially, node 4 receives substreams from nodes 2 and 3, and node 5 receives from nodes 1 and 4. Then node 2 disconnects, and the system recovers by assigning node 6 as a replacement. While locating and establishing a replacement, visual quality at node 4 is degraded. We use a fat pipe to indicate the downlink of each node, and a thin pipe to illustrate the uplink of each node. Generally, a node can function as a client only, a server only, or simultaneously as a server and a client (e.g., node 4).

Recently, several video multicast systems making use of peer coordination have been studied [1–4]. But in these systems, the video content is stored in a central server and the peers merely help to relay the video originating from the server by forming one or several multicast trees; thus,

This work is supported in part by the National Science Foundation under Grant CNS-0435228, and also in part by the New York State Center for Advanced Technology in Telecommunications (CATT).

peers only help each other when receiving the same video. Our approach is fundamentally different in that it employs peers as streaming servers directly. In our system, a peer may serve a video even when it is not receiving the video from other peers, or when receiving a different video.

The proposed system can use either multiple description coding (MDC) or layered coding to generate multiple substreams. With the MD system, all substreams can be treated equally. Also, the disconnect of a server peer will only interrupt the delivery of a description temporarily, which will not cause severe degradation in the received video. The layered system generates layers with varying importance to visual quality, and the system must be designed so that lower layers, which are more important, are delivered with higher probability. The MD system design is simpler, but the source coder requires a higher bit rate to reach the same target decoding quality as a layered codec, in the absence of transmission loss. Thus the layered approach is more efficient in utilizing the peer resources (that is, can serve more requests for the same target decoding quality). But this higher efficiency would be obtained at higher design and operational cost.

Our work on the MD-based system was reported in [5]. For the layered system, one main challenge lies in how to provide unequal protection to different layers commensurate with their importance, so that more important layers can be delivered with a higher probability. One way to realize this unequal error protection is by storing the important layers at many peers. In this work, we examine the impact of the copy number allocation schemes on the overall system performance. We also compare the performance of the MD and layered system.

2. SYSTEM MODELING AND PERFORMANCE ANALYSIS

2.1. Layered video

We encode each video into multiple layers using a MPEG-4 Fine-Grained Scalable (FGS) Coder [6]. The FGS coding technique encodes a video into two layers, the base layer and the enhancement layer. The base layer contains the most essential quality information. The enhancement layers provide quality enhancements. FGS allows the user to adjust the relative sizes of the base layer and enhancement layer and further allows the enhancement layer to be broken up into an arbitrary number of hierarchical layers. Using the FGS coding techniques, the video can be encoded into M identically-sized hierarchical layers, with the layer m encoded at R_m bits. The layer m can only be decoded if all lower layers from layer 1 to layer m are available. Therefore, a higher layer is useless for the client if any of the lower layers is not available. When decoding up to m layers, there is an associated distortion $D_m(R_1 + R_2 + \ldots + R_m)$,where $D(R)$ represents the rate-distortion function of the underlying scalable coder. Let $P(m)$ denote the probability of receiving layer m. Then the expected distortion D of the received video is

$$D = \sum_{m=0}^{M} D_m \prod_{i=0}^{m} P(i)(1 - P(i+1)). \quad (1)$$

2.2. The number of copies for each layer

Since the layers have varying importance, the system should provide unequal treatment to the layers, to increase the likelihood of delivering the most important lower layers. Intuitively, we should have more copies of the lower layers than the upper layers. This way, when a peer serving one important layer disconnects from the network, it is likely to find another peer having the same layer that is connected to the network and have surplus uplink bandwidth to serve this layer. Consider a particular video with total rate R_t, duration T, and it is encoded into M layers with each layer having a size R_tT/M. Let the number of copies of layer m be $C_m, m = 1, 2, \ldots, M$. Assume the total storage allocated to this video is S. Then we have the storage capacity constraint:

$$\sum_{m=1}^{M} C_m = \frac{SM}{R_tT}. \quad (2)$$

Ideally, we would like to assign C_m so that the expected distortion in (1) is minimized, while satisfying the constraint in (2). Because the probability $P(m)$ depends not only on C_m, but also the network statistics and the number of ongoing streaming sessions, it is not easy to solve for the optimal number of copies for each layer analytically. Here, we consider some general allocation schemes to shed some insights on this complex problem.

Assume $C_m = a(M - m + 1)^b$, where a is the normalization constant. We consider three kind of general allocations:

1. Concave: $b < 1$
2. Linear: $b = 1$
3. Convex: $b > 1$

When b becomes larger, it means that we make more copies for the lower layers given the same storage capacity constraint.

2.3. Simulation settings

2.3.1. Video data

We coded the "Foreman" video sequence in CIF (352x288) resolution into a scalable bit stream using the MPEG-4 FGS codec [7], at a base layer rate of 150 kbps. Each Group of

Frames(GOF) has the duration of $T = 1$ second and comprises 15 frames. The bit stream is further partitioned into M layers, where M is varied from 4 to 32. The total rate of a video (R_t) is set to be 512 kbps.

2.3.2. Network model

In our simulations, we assume a homogeneous system in which each peer has the same uplink bandwidth (256 kbps) and storage capacity (230 MBytes). There are 300 nodes in the network. Each node in the network alternates between "connect" and "disconnect" status. We model the connected time as an exponentially distributed random variable with mean α. Similarly, the disconnected time is another exponentially distributed random variable with mean β. Then the probability that a node is in "connect" status is $\frac{\alpha}{\alpha+\beta}$

We have $J = 30$ videos. Each video has the same size but not the same popularity, with the popularity of these videos following the Zipf distribution with parameter $a = 1.1$. The number of new requests for all videos is modeled as a Poisson process with a rate of $\lambda = 1.5$ requests/min. The length of each video is 2 hours. The storage capacity allocated to a particular video is proportional to its request rate. Then for each video, we apply the three allocation schemes to decide the number of copies for its layers. Each node stores at most one layer for a particular video.

When a node requests a video, a central manager will try to find M serving nodes that have the M substreams of the video, with each node also having sufficient surplus uplink bandwidth to serve one additional substream. Given a choice, for each substream, we choose the server that is the least loaded over all servers that have this substream. If during service a node disconnects, the central server will look for a replacement node that has the same substream and sufficient uplink bandwidth.

We set a loading parameter Q_{max}. If the total number of sessions in the network is greater than Q_{max}, then the new requests are blocked. This is to preserve enough serving capacity in the network to serve ongoing sessions, given that nodes may disconnect in the middle of a session. In our simulations, we choose eight values for Q_{max}: 20, 40, 60, 80, 100, 120, 140, 160.

2.4. Simulation results

Figure 2 shows the simulation results of the three allocation schemes with $M = 32$ (For other values of M, we get similar results) and $\alpha = 4$ hours, $\beta = 1$ hour (node "connect" probability is 0.8). From the figure, we can see that when the network is lightly loaded (Q_{max} is less than about 50), the concave allocation has the best video quality among all schemes. But when the network is heavily loaded (Q_{max} is greater than 100), the convex allocation is the best.

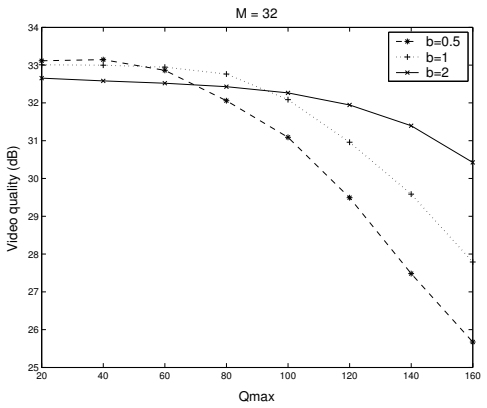

Fig. 2. Video quality under different allocation schemes

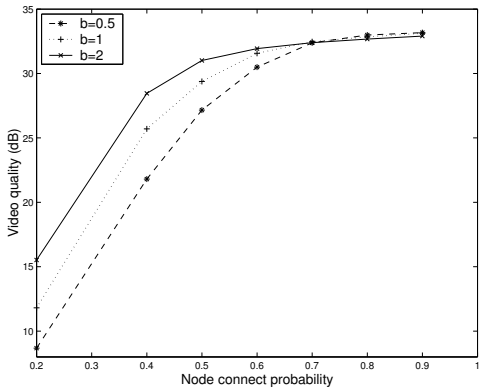

Fig. 3. Video quality under different allocation schemes

When the network is lightly loaded, if a server peer disconnects during a streaming session, it is more likely to find a replacement peer that is connected and has surplus uplink bandwidth. The concave allocation makes more copies for the higher layer as compared to the convex allocation, thus a streaming session receives more layers on the average. But when the network is heavily loaded, it becomes more difficult to find a replacement peer if a server peer disconnects. The convex allocation makes more copies for the lower layers, thus nodes have a higher probability of receiving the more important lower layers as compared to the concave allocation.

Figure 3 shows the three allocation schemes with varying node "connect" probability, $M = 16$ and $Q_{max} = 60$. We can see that the convex allocation is the best over a large range of the "connect" probability. When the "connect" probability increases beyond a certain point, the concave allocation becomes slightly better, which means that we should make relatively more copies for the higher layers as nodes become more stable.

3. COMPARISON WITH THE MD SYSTEM

In this section, we compare the performance of the MD system using the MD-FEC codec [5] and the layered system with the same simulation settings as described in Section 2.3

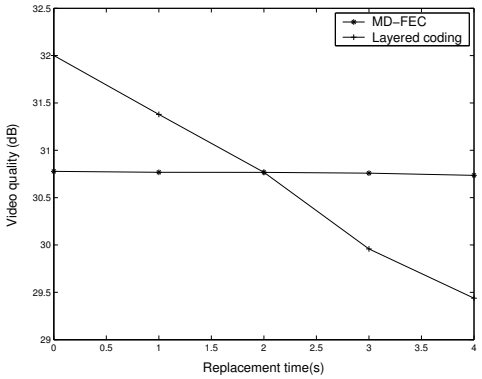

Fig. 4. Comparison between MD-FEC and layered system with varying replacement time

Figure 4 shows the comparison between MD-FEC and layered system with $Q_{max} = 120$ and varying replacement time. The video quality is affected during the time needed to find a replacement node. In MD-FEC system, we choose the optimal M which achieves the best video quality. In the layered system, the number of layers for a video is varied from 4 to 32, and the three allocation schemes are considered. Then we choose the allocation scheme and M that have the best video quality. As seen from Figure 4, when the replacement time is small, the layered coding approach performs better. As we know, layered coding requires a lower bit rate to reach the same video quality as a MD-FEC codec. Therefore, when the network is more reliable (a substream is less likely to be lost), layered coding is more efficient than MD-FEC. When the replacement time increases, MD-FEC has a better performance than layered coding. The reason is that MD-FEC has an inherent protection against substream loss. When a single substream is lost, for MD-FEC, the video quality is only slightly affected. But for layered coding, all layers higher than this substream can not be decoded at the receiver.

4. CONCLUSION

In this paper, we presented a video streaming scheme based on a peer-to-peer architecture, where each peer stores and streams videos to the requesting client peers. We encode each video into multiple layers and place each layer on a different node. This circumvents the uplink bandwidth constraint of most server peers. We have investigated how to allocate the limited storage capacity to layers with varying importance. Given the storage capacity constraint, the number of copies for each layer has a dramatic impact on the average distortion. We also studied the comparison between layered coding and MD-FEC. Our results showed that layered coding is a better choice when the system can find and switch over to another server peer quickly, while MD-FEC performs better if the replacement time is non-negligible. So far, we have not considered the effect of pre-fetching on either the MD or the layered system. By pre-fetching, the system can avoid the loss of a substream upon a node disconnect even when the replacement time is non-negligible. But prefetching will increase the play-out delay, increase the memory requirement at the client, and complicating the overall system design to some extent. As an extension of the current work, we plan to investigate the design and performance evaluation of the system with pre-fetching.

5. REFERENCES

[1] H. Deshpande, M. Bawa, and H. Garcia-Molina, "Streaming live media over a peer-to-peer network," Tech. Rep., Stanford University, 2001.

[2] V. N. Padmanabhan, H. J. Wang, P. A. Chou, and K. Sripanidkulchai, "Distributing streaming media content using cooperative networking," in *NOSSDAV*, 2002.

[3] A. Rowstron and A. Singh, "Splitstream: High-bandwidth content distribution in a cooperative environment," in *Proc. of 2nd International Workshop on Peer-to-Peer Systems (IPTPS '03)*, Berkeley, CA, USA, February 2003.

[4] Y. Guo, K. Suh, J. Kurose, and D. Towsley, "P2Cast: peer-to-peer patching scheme for VoD service," in *Proceedings of the 12th World Wide Web Conference (WWW-03)*, Budapest, Hungary, May 2003.

[5] X. Xu, Y. Wang, S.S. Panwar, and K.W. Ross, "A peer-to-peer video-on-demand system using multiple description coding and server diversity," in *IEEE International Conference on Image Processing (ICIP)*, Oct. 2004.

[6] H. Radha, M. van der Schaar, and Y. Chen, "The MPEG-4 fine-grained scalable video coding method for multimedia streaming over IP," *IEEE Trans. on Multimedia*, pp. 53–68, March 2001.

[7] MoMusys code, "MPEG-4 verfication model version 18.0," in *ISO/ IEC JTC1/SC29/WG11 Coding of Moving Pictures and Audio*, January 2001.