Delay-Guaranteed Interactive Multiview Video Streaming

Zhibo Chen, Meng Zhang, Lifeng Sun, Shiqiang Yang
Tsinghua National Laboratory for Information Science and Technology
Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China
Email: {chenzb07, zhangmeng00}@mails.tsinghua.edu.cn; {sunlf,yangshq}@mail.tsinghua.edu.cn

Abstract— **Multiview video is well known to involve interactions with audience and offer better view experience than conventional single-view video. The interactive nature of multiview video has made the service very delay-sensitive and thus imposes great challenges for providing multiview streaming service in Internet. Currently, almost none of existing works have addressed the delay issue in multiview streaming. To ensure the quality of service and offer preferable view experience for users, we propose a novel streaming framework to provide delay-guaranteed service for interactive multiview video. The basic tradeoff between consumed server bandwidth and the required delay is carefully studied. We leverage the features of MVC to keep the server bandwidth costs low while guarantee the delay constraint. In addition, we introduce a neighbor-assisted view switching scheme: peer's neighbors are involved in its switching process and their upload bandwidth resources are effectively utilized to reduce the bandwidth costs on server. Simulation results show that our proposed framework meet the delay-guaranteed requirement with restrained server bandwidth costs while maintaining scalability and resilience to users dynamics.**

I. Introduction

As an emerging new media service, multiview video enjoys much credit for its superiority in user interaction: viewers are allowed to change and switch freely between a number of views when watching the same video sequence. The enhanced user interaction enables multiview video service to better meet audience's diversified preferences and provide new view experience for users. Owning to such great advantage over conventional single-view video, multiview video service is promising to be widely deployed in the future and grab a lion's share of the video market.

Recently, with the rapid development of efficient compression method multiview video coding(MVC)[1][2] and peer-to-peer live streaming technology[3][4], providing multiview streaming service in Internet has become a reality. There have been significant studies on multiview streaming recently and many new ideas have been proposed to improve the performance and bring down the server bandwidth costs in multiview streaming[5][6][7][8]. However, almost none of them has paid attention to addressing the delay issue in providing multiview streaming service.

There are two types of delay in multiview streaming: source-to-peer delay and view switching delay. Source-to-peer delay is the major QoS issue in Internet P2P streaming service. Moreover, the interactive nature of multiview video service has made it more delay-sensitive for view switching: users' requests for view switching should be responded without too much delay so that their view experience are not spoiled. Usually, users' delay tolerance varies with the type of their watching program: a conventional film is not so urgent as a live broadcast and does not need strictly real-time delivery, and hence it brings potential possibility for us to provide differentiated services with varied delays, in order to save bandwidth costs on server. Meanwhile, although users can tolerate a certain amount of time, however, the delay should be guaranteed and predictable. Actually, the aim of this paper is to find a practical way to provide efficient low-cost live streaming service with guaranteed delay on Internet for interactive multiview video.

Further, video contents for each view in multiview video are correlated rather than absolutely independent as in conventional single-view video. To provide scalable multiview streaming service, we argue that the features of MVC should be exploited to minimize the server bandwidth consumption with respect to the guaranteed delay constraint for each multiview program. In this paper, we propose a novel streaming framework to provide delay-guaranteed service for interactive multiview video. The basic tradeoff between consumed server bandwidth and the required delay is carefully studied. We leverage the features of MVC to meet the delay constraint while keeping the server bandwidth costs low. Further, we introduce a neighbor-assisted view switching scheme: peer's neighbors are involved in its switching process and their upload bandwidth resources are effectively utilized to reduce the bandwidth costs on server. We conduct comprehensive simulation experiments to evaluate our proposed framework. The results show that it meets the delay-guaranteed requirement with restrained server bandwidth costs while maintaining scalability and resilience to users dynamics.

II. Related Work

Previous research work on multiview streaming focuses on bringing down the server bandwidth costs regardless of the delay issue. [5] employs simulcast coding and streams video contents for each view separately. [6] combines MVC and scalable video coding(SVC) to obtain improved compression efficiency and provide flexible selective streaming service. [9] introduces a standard-based and flexible end-to-end multiview

Supported by the National Natural Science Foundation of China under Grant No.60503063, 863 Program under Grant No.2006AA01Z321

978-1-4244-3828-0/09/$25.00 ©2009 IEEE

streaming architecture. [8] employ NUEPMuT protocol to stream multiview video contents and propose a quick join procedure to reduce the start-up delay. Although [10] addresses the view switching delay issue by managing the application multicast tree(ALM), it requires to measure end-to-end delay among peers, which is not practical in Internet. Further, it does not provide guaranteed delay for the view switching.

III. Proposed Framework

A. Architecture

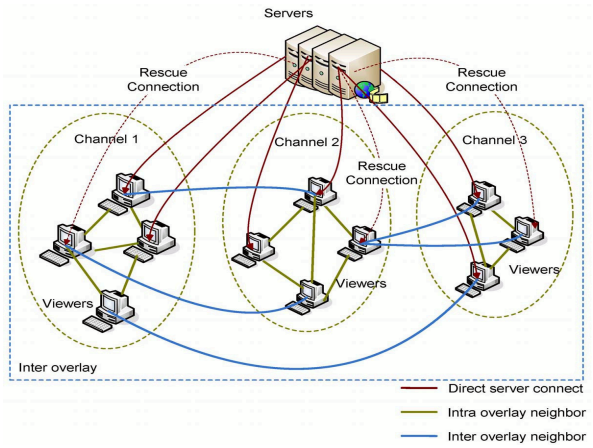

Fig. 1. Overlay Architecture

We target to provide delay-guaranteed multiview streaming service with low bandwidth costs on server. To support such service, we first address the issue of overlay construction. For a multiview program, we construct n mesh-based overlays for each view channels of the program, where n equals the number of view. We call them intra overlay, as illustrated in Fig. 1. Each viewer chooses the view to watch and joins its corresponding intra overlay. To minimize the bandwidth costs, we aim to leverage the features of MVC: I pictures are shared and exchanged among these view channels instead of requesting from the server respectively. To enable communication and data exchange among different view channels, we further connect viewers belonging to different intra overlays. All the viewers, despite their varied view channel belongings, form a large overlay. We call it inter overlay. As shown in Fig. 1, each viewer maintains a rescue connection with the server. We assume the default transmit protocol is UDP. Once an absent streaming packet is about to pass its deadline, it will be requested through the rescue connection from the server. Then we briefly introduce the peer(viewer) joining and neighbor finding process: to join a steaming session, a peer first contacts a rendezvous point(RP) which is a server maintains a list of currently online nodes in every view channel. Then according to its selection for view channel, the peer randomly chooses 15 nodes from the corresponding intra overlay as its intra neighbors. Then, the peer chooses two random nodes from each view channel except its own as its inter neighbors. The hybrid neighborhood selection mechanism ensures rich connection both in intra overlay and inter overlay and contributes to efficient content delivery.

B. Content Delivery

After constructing the overlays, the next question is what strategy do we employ to deliver the streaming contents so that viewers across different view channels can receive pictures needed to render their watched video with low delay while keep server bandwidth costs low. For the streaming delivery, the video streaming is packetized into fixed-length packets called streaming packets marked by sequence numbers. We categorize all streaming packets into *I packet*, *Reference P packet* and *None-reference B/P packet* according to their loading contents and employ different strategies for their delivery. To keep the server bandwidth costs low, we let intra overlays to share and exchange the I packets rather than request from the server respectively. To balance the overload on each intra overlay and achieve fast I packet exchanging, we introduce a slicing scheme which makes full use of viewers' upload bandwidth resources. We evenly slice I pictures into N parts denoted by $I_1, I_2....I_n$, where n equals the number of view channels of the program. Server's direct neighbor peers in intra overlay O_i will request slicing Ii from the server and then disperse it in O_i. Then through the inter overlay, all I picture slicings are exchanged between inter overlay neighbors. Reference P packets and none-reference B/P packets of each view are first delivered to server's direct neighbors in each intra overlay and then dispersed within these intra overlays. We adopt pull-push as the basic protocol for packet scheduling and its details can be found in [4].

C. Server Scheduling

The packet scheduling on server is different from the peers. The bandwidth consumption on server can be divided into two parts: in normal streaming process, deliver packets to server's direct neighbors in each intra overlays by active push or passive pull; respond to users' rescue requests and send late packets to users. We call the former *normal traffic* and the latter *rescue traffic*. Server's bandwidth consumptions on the two kinds of traffic are closely correlated. If the server delivers identical contents to more peers in each intra overlay, the consumption on normal traffic will naturally increases. However, the more wide distribution of the identical contents will facilitate and accelerate its delivery in each intra overlay, and in turn, reduce peers' chances of requesting for late packets and consequently the bandwidth consumption on rescue traffic. On the other hand, although the server can reduce its bandwidth consumption on normal traffic by limiting the number of direct delivery, the possibility of late packets will increase and hence, incur more bandwidth costs on rescue traffic. Here we propose a simple and useful adaptation method to address the problem. For any packet, if the server directly pushes only one copy to any of the peers, we call it a **1-time-server-push**. We say that the ***server push times*** is 2 or it is **2-times-server-push**, if the server pushes two copies of

each packet to any of the peers, and so on. Assume the current server outgoing traffic rate is r_s and streaming rate is r, and the current server push times is γ. If $r_s/r > 2\gamma$, we let $\gamma \leftarrow 2\gamma$; while if $r_s/r < 1.2\gamma$, we let $\gamma \leftarrow \gamma/2$. This means once the server outgoing traffic rate exceeds 2 times pushed packet bit rate, we double the server push times; while if the server outgoing traffic rate goes down to 1.5 times pushed packet bit rate, we reduce the server push times to half.

To get the packet precise playback deadline, all nodes should perform clock synchronization with the server when they join in a streaming session, using network time protocol (NTP) or just a simple "ping-pong" synchronizing message if the precision requirement is not high. We use t_{gen} to denote the generated clock time of packet i at the server. We let t_{rtt} represent the round-trip time between the viewer and the server. If the required guaranteed delay is T, the viewer will send a packet request to the server, once the packet does not arrive until clock time $t_{gen} + T - t_{rtt}$. The server will send the packet to the viewer as soon as it receives the request.

D. Neighbor-Assisted View Switching

The interactive nature of mutlivew video has decided that view switching is an important aspect in providing multiview streaming service: the QoS issue of multiview streaming service hinges largely on how fast view switch can be done, because user's view experience can be easily spoiled by slow and unpredictable switch delay. On the other hand, achieving fast view switching requires excessive bandwidth costs on streaming server because existing solutions employ a straightforward strategy: video contents are directly streamed from server to the peer engaging in view switching, in order to reduce the switch delay. However, the high-cost switching method downgrades the system scalability and has a risk of compromising the overall performance, if there are a large number of peers performing switching operation concurrently. To address the problem, we propose a neighbor-assisted, delay-guaranteed view switching scheme with reduced bandwidth costs on streaming server. Recall that in our overlay architecture, each peer has two inter overlay neighbors in each view channels except its own. We take advantage of these peers' upload bandwidth resources to facilitate the switching process and alleviate the bandwidth demand on the server. In our scheme, when peer P want to switch to view channel VC_i, it will send switch packet to its inter overlay neighbors in VC_i and the server. Assuming T_s is the guaranteed time for view switching, the server will push streaming packets of the first half of T_s to P through the rescue connection; On the other hand, P's inter overlay neighbors in VC_i will temporarily stops responding to other peers' requests and devote all their bandwidth resources to stream the packets of the second half of T_s to P. Meanwhile, P will engage in a normal joining and neighbor finding process. During the switch process, once an absent streaming packet is about to pass its deadline, it will be requested from the server.

IV. Performance Evaluation

We evaluate the performance of our proposed framework with existing non-delay-guaranteed solutions, namely simulcast and MVC using simulation experiments. In simulcast solution, video contents for each view are encoded and transmitted separately. MVC solution encodes all the views as a whole but delivery contents for each view separately. We make a little changes on them to let them meet the delay-guaranteed requirement: once a packet is about to pass its deadline, it will be requested from the server. We adopt *KS_PIP* as the prediction structure for MVC in our experiments. [11]'s experiment indicates that *KS_PIP* maintains MVC's advantage in coding gains(average 1.4db versus full MVC's 1.6db) while has less complex structure. We implement an event-driven packet-level simulator coded in C++ to conduct the experiments in this section[1]. In our simulation, all streaming and control packets and node buffers are carefully simulated. For the end-to-end latency setup, we employ real-world node-to-node latency matrix(2500X2500) measured on Internet[12]. The default MVC streaming rate is set to equivalent to 300kbps per view channel in simulcast coding. As for the MVC parameters, the number of views of multiview video is 8 and Group of Picture(GOP)size is 8. B/P picture size are regarded as the same and I frame size is 15 times of B/P picture. To simulate the bandwidth heterogeneity of the peers, we use four different typical DSL with capacities of 3Mbps, 1Mbps, 384kbps and 128kbps respectively. we adjust their fraction to obtain different peer resource index (PRI). PRI is defined as the ratio of the total peer upload capacity $\sum_{i=1}^{n} u_i$ to the minimum bandwidth resource demand (i.e, streaming rate r times the viewer number n), that is., $PRI = \sum u_i/nr$.

In our simulation, we assume that the bottleneck is always at the last hop and the server bandwidth is large enough, hence the end viewer can always get sufficient streaming packets to play back due to the rescue connection with the server, that is, all the viewers can watch a full quality of video within the required. Therefore, in this section, we mainly study the consumed server bandwidth with respect to different guaranteed delay requirement under different user behaviors and network conditions. Initial server push times is set to 1.

Fig. 2 shows the server bandwidth consumptions with respect to different delay constraints in static environment. We set peer resource index(PRI) to 1.55, and each view channel has 100 users. As shown that when the delay constraint is very urgent, i.e., 2sec, our I picture sharing and server adaptive strategy has prevented the excessive bandwidth consumption on server. Under such urgent delay constraint, the server will quickly increase the server push times as to stream more copies of contents to peers, accelerate their dispersing rate and thus reduce the possibility of late packet requests. On the other hand, simulcast and MVC can only rely on the server's rescue packets to meet the delay constraint, and as a result, consume tremendous bandwidth resources on server.

[1]The simulator is available online for free downloading at http://media.cs.tsinghua.edu.cn/∼zhangm

Fig. 3 shows the bandwidth costs on server with respect to varied PRI. The delay constraint is set to 5 sec, and peer number is 100 in each view channel. As shown that our method can effectively utilize peers' upload bandwidth resources to alleviate the bandwidth demand on the streaming server.

Fig. 4 shows the bandwidth costs on server when the peer number scales. We set PRI to 1.55 and delay constraint to 5 sec. As shown that our method has a much slower increase in server bandwidth consumption when the peer number scales. It can be attributed to our adaptation strategy: the server can push more copies to peers rather than waiting for the late packet requests.

We investigate in high peer churn environment in Fig. 5. We use Weibull(λ, k) distribution with a CDF $f(x) = 1-e^{-(x/\lambda)^k}$ to randomly generate the lifetime of the viewers. And we assume the peer joining process is a Poisson Process with rate of 20 per sec and the maximum online user number is 800. The results indicates that even in high churn environment, our framework still achieves substantially less bandwidth consumption on server comparing with the other methods.

V. Conclusion

In this paper, we propose a novel streaming framework for delay-guaranteed interactive multiview video service. We study the tradeoff between server bandwidth consumption and the delay constraint and leverage the features of MVC to reduce the bandwidth consumption. We propose a neighbor-assisted method to achieve fast low-cost view switching. We conduct simulation experiments to evaluate our framework. The results show that it achieves substantially less server bandwidth consumption than existing multiview streaming solutions and maintains scalability and resilience to user dynamics.

References

[1] K. Mueller, P. Merkle and T. Wiegand, "Multi-view video coding based on h.264/avc using hierarchical b-frames," in *Picture Coding Symposium 2006*.

[2] A. Vetro, P. Pandit and A. Smolic, "Joint draft 3.0 on multiview video coding," in *Joint Video Team, Doc. JVT-W209, 2007*.

[3] X. Zhang, J. Liu, B. Li, and Y.-S. Yum, "Coolstreaming/donet: a data-driven overlay network for peer-to-peer live media streaming," *INFOCOM 2005*.

[4] M. Zhang, Q. Zhang, L. Sun, and S. Yang, "Understanding the power of pull-based streaming protocol: Can we do better?" *Selected Areas in Communications, IEEE Journal on*, Dec. 2007.

[5] Kurutepe, E. and Civanlar, M.R. and Tekalp, A.M, "Interactive transport of multiview videos for 3dtv applications," in *Packet Video Workshop 2006*.

[6] Kurutepe, E. and Civanlar, M.R. and Tekalp, A.M, "Client-driven selective streaming of multiview video for interactive 3dtv," *Circuits and Systems for Video Technology, IEEE Transactions on*, Nov. 2007.

[7] E. Kurutepe, T. Sikora, "Feasibility of multi-view video streaming over p2p networks," in *3DTV-CON 2008*.

[8] E. Kurutepe, T. Sikora, "Multi-view video streaming over p2p networks with low start-up delay," in *ICIP 2008*.

[9] E. Kurutepe, A. Aksay, C. Bilen, C. G. Gurler, T. Sikora, G. Bozdagi Akar, A. M. Tekalp, "a standards-based,flexible, end-to-end multi-view video streaming architecture," in *Packet Video Workshop 2007*.

[10] H. L. J. Kim, K. Choi and J. W. Kim, "Multi-view 3d video transport using application layer multicast with view switching delay constraints," in *3DTV Conference, 2007*.

[11] P. Merkle, A. Smolic, K. Muller, and T. Wiegand, "Coding efficiency and complexity analysis of mvc prediction structures," in *EURASIP 2007*.

[12] "Meridian node to node latency matrix (2500×2500)," 2005, meridian project. [Online]. Available: http://www.cs.cornell.edu/People/egs/meridian/data.php

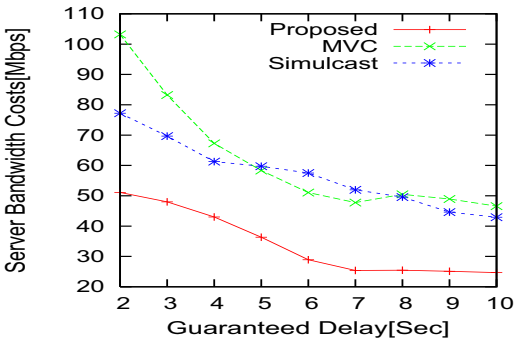

Fig. 2. Server bandwidth costs with respect to guaranteed delay.

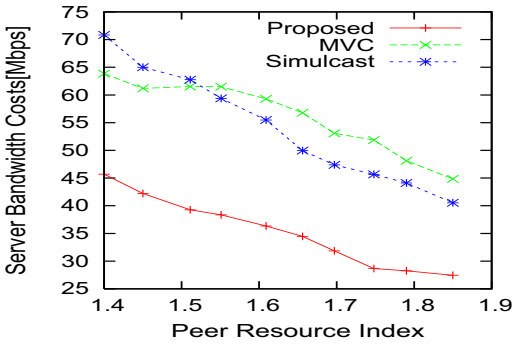

Fig. 3. Server bandwidth costs with respect to PRI.

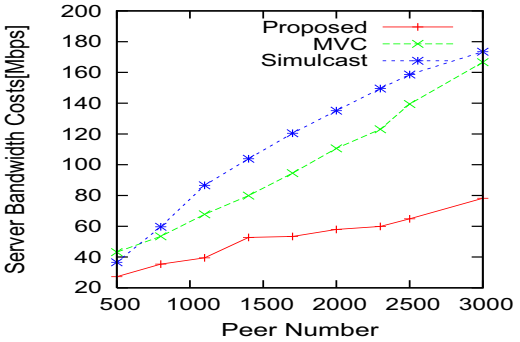

Fig. 4. Server bandwidth costs with respect to peer number.

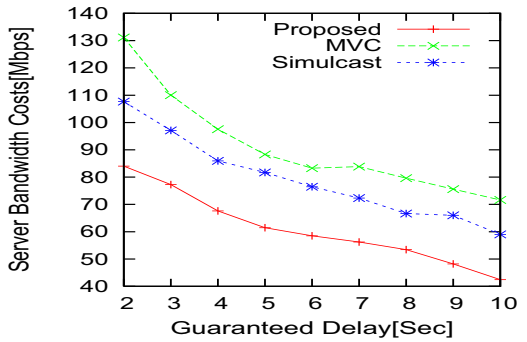

Fig. 5. Server bandwidth costs with respect to guaranteed delay in dynamic environment(Weibull(500,2)).