OVERLAY COLLABORATION TOWARDS REDUCED BANDWIDTH COSTS IN MULTI-VIEW 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

Delivering high-quality multiview video (MV) through Internet is very challenging due to its excessive consumption on server bandwidth resources. Existing solutions encode video contents independently for each view and deliver them separately in isolated view channels, without leveraging the features of MV and taking advantage of multiview video coding(MVC). To minimize the server bandwidth costs, we introduce a novel overlay collaboration framework that unifies all view channels to cooperate in delivering MV: I pictures in MVC are shared among them instead of requesting from server respectively; Surplus resources of hotspot view channels are effectively utilized to help channels with insufficient resources, both of which contribute to remarkable reduction in server bandwidth costs. Simulation experiments show that our method achieves more than 40% reduced bandwidth costs on server while maintaining scalability and resilience to user dynamics.

Index Terms— multiview video, delivery, peer-to-peer, overlay, collaboration

1. INTRODUCTION

Multiview video has emerged as a remedy for conventional single-view video in better meeting audience's diversified preference: it allows user to change the view freely when watching the same video sequence, thus producing enhanced viewing experience for audience. However, the delivery issue has prevented the service from being widely deployed in Internet because of its excessive consumption for bandwidth resource on streaming server. To cope with it, researchers have been working on Multiview Video Coding(MVC)[1][2][3]. However, even with MVC, bit rate remains relatively high: for example, an eight-view MV with resolution at 704x340 in 30fps, consumes 5Mbps to achieve 38 PSNR[4]. Deploying MV service to provide concurrently hundreds of programs, the consumption for bandwidth could be a nightmare. So how to deliver MV with restrained server bandwidth costs while maintaining scalability and guaranteed QoS is a challenging issue for providing multiview streaming service.

Existing simulcast solutions[4][5][6] deliver multiview video in an isolated manner: video contents for each view of the same MV are encoded independently and transmitted separately in their corresponding view channels, which introduces substantial redundancy in delivery as it abandons efficient MVC. Moreover, it suffers from heterogeneity in view channels' bandwidth resource because the surplus resources of hotspot channels can not be utilized to help channels with insufficient resources as they are isolated and unable to cooperate. Therefore, the server bandwidth costs are not minimized. Actually, the fundamental limitation is that it ignores the inherent correlations among views of MV and fails to offer sufficient insight into exploiting the features of MVC. We argue that the unification and collaboration of different view channels are critical to take advantage of MVC, with the aim of achieving reduced server bandwidth costs.

In this paper, based on the peer-to-peer live streaming technology, we present a novel overlay collaboration framework that enables cooperations among different view channels and fully leverage the benefits of MVC to facilitate the delivery of multiview video, in order to reduce bandwidth costs on streaming server. Our contributions are the followings. First, we investigate into current MVC schemes and gain an insight on the potential for collaboration among different view channels. Second, we further present a hierarchical overlay structure unifying these originally isolated view channels and propose an inter overlay collaboration scheme that enables fully cooperation not only within each view channel but also across different view channels. Moreover we propose a useful adaptation strategy for heterogeneous overlay bandwidth resource condition: view channels with abundant resources can contribute to improve the performance of view channels with insufficient resource. Finally, we conduct comprehensive experiments to evaluate our method with existing work. The results show that our approach substantially saves

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

the server bandwidth consumption by more than 40%, while maintaining scalability and resilience to user dynamics.

2. COLLABORATIVE MULTIVIEW STREAMING

2.1. Insight On the Potential for Collaboration

MVC use inter view prediction to exploit the spatial similarities in multiview video: it keeps one view temporal predicted and replaces other views' intra-coded I picture with inter-coded B/P picture. Several prediction structures have been proposed with respect to the tradeoff between coding efficiency and dependency complexity[1][2][3]. However, they have one thing in common that every view of the same MV needs identical I picture for its decoding process. Moreover, intra-coded I picture accounts for a large proportion of server's outband traffic. So, we envision that if all view channels can cooperate to share and exchange I pictures rather than request them from server respectively, the bandwidth costs on server can be greatly reduced.

2.2. Topology Basis for Collaboration

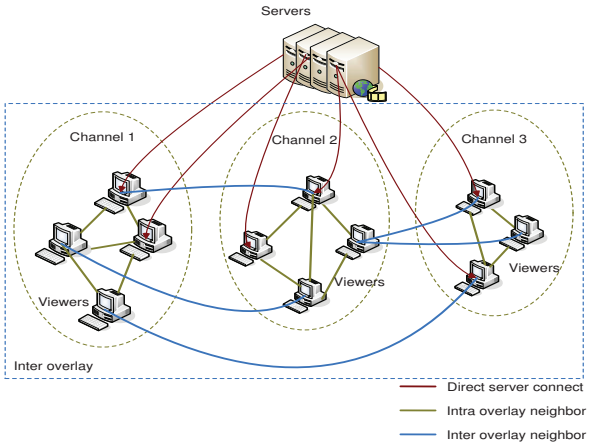

Fig. 1. Overlay Architecture

Conventional single-view streaming service organizes view channels on isolated overlays, so they can not communicate and exchange data with each other. To address the problem, we introduce a novel hierarchical overlay structure serving as topology basis for our collaboration. As illustrated in Fig. 1, our architecture consists of two type of overlays: *intra overlay(AO)* and *inter overlay(RO)*. Viewers of each view channel are organized into several mesh-based AOs just as in conventional single-view streaming service. On top of these AOs, We further connect viewers belonging to different AOs, forming the RO which unifies these isolated AOs and enables communications and data exchanges between viewers among them. When joining a streaming session, each peer engages in a two stage of neighboring finding process: it first randomly chooses a number of nodes in the same AO as AO neighbors; Then it chooses a random node from each AO except its own and N random nodes across all AOs as its RO neighbors, where N is the number of views.

2.3. Collaborative 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 cooperate to receive pictures needed to render their watched video with low delay while keep server bandwidth costs low. 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. We propose a slicing scheme to achieve fast I packet dispersing in order to realize efficient cooperation among all view channels. 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 I_i 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. We set the request interval τ_{inter} in inter overlay a little longer than the packet arrival time $rtt + \tau_{intra}$ after requesting in each intra overlay, where rtt is the round trip time to neighbor and τ_{intra} is the request interval for intra overlay, so that the exchanging of I picture slicings among inter overlay neighbors can be efficient as each I picture slicing is first dispersed widely in its corresponding intra overlay and more peers can participate in the collaboration later. Considering that reference P packet only accounts for a small proportion of server outband traffic and the needed reference P pictures differ with view channels, the delivery of reference P packets are treated the same as none-reference B/P packet: they are sent and dispersed only in each intra overlay in order to ease the process of neighbor finding. We adopt pull-push as the basic protocol for packet scheduling and its details can be found in [7].

3. ADAPTION FOR HETEROGENEOUS OVERLAY RESOURCES

Heterogeneity in view channel resources is a common scenario in multiview streaming: hotspot view will attract numerous audiences and have abundant bandwidth resources for supporting streaming service, while cold spots of the same program could have very few users and presumably results in insufficient resources for sustaining streaming service. (Here we call the former *rich overlay* and the latter *lean overlay*.) In this circumstance, existing delivery method has nothing to do but allocate additional server bandwidth resources for lean overlay. We envision that the surplus resources of rich overlays should be utilized to help lean overlays so that the additional bandwidth costs for server can be minimized.

Note that it is not practical to measure user's bandwidth resource in Internet. So the immediate problem here is how to identify lean overlay and the degree of its resource shortage.

Recall that in our proposed architecture, an overlay's bandwidth resource consumption can be divided into two parts: delivering packets in its intra overlay and exchanging packets with other intra overlays through the inter overlay. Lean overlay O_i's lacking in bandwidth resources will have a twofold impact: the delivery of packets in O_i is hindered which causes viewers of O_i to request some absent packets from server; The dispersing of I_i in inter overlay will be slowed and viewers in other intra overlays have to resort to server for absent packets when the deadline is due. We call the former as *Intra Rescue Traffic* and the latter as *Inter Rescue Traffic*. Further we use CA_i(bytes) to denote the bandwidth costs for intra overlay O_i's intra rescue traffic and CR_i(bytes) to represent the inter rescue traffic caused by I_i. Both CA and CR can be computed easily at the streaming server. CA_i directly reflects the degree of bandwidth shortage in O_i and CR_i indicates whether I_i has been effectively dispersed to other intra overlays which sheds some light on O_i's bandwidth resource condition.

Here, we define ***Overlay Bandwidth Resource Gap***(OBRG) in each intra overlay. It is defined as the average bit rate the server need to allocate for each peer to support their streaming service. In practice, we use CA_i to get a rough estimation that $OBRG_i = 8CA_i/tn_i$, assuming that t is the streaming time and n_i is the number of peers in O_i. Though may not be precise, OBRG reflects the general resource condition of each intra overlay. Our experiment results indicate that intra overlay with $OBRG > r/10$ should be considered as lean one and the performance after cooperation is distinctive; Intra overlay with $OBRG < r/100$ and hundreds of peer numbers can be considered as rich one, where r is the streaming rate.

After identifying the lean overlay and rich overlay, we let rich overlays to partially or wholly substitute lean overlays in dispersing their corresponding I picture slicings in order to alleviate their bandwidth resource shortage. To be specific, we first estimate the roughly average bit rate for dispersing I picture slicing $r_{IS} = (2n-1)r\rho/n$, which means the bandwidth resource gain lean overlay O_i can have, if it stops dispersing I_i to other intra overlays, assuming r is the streaming rate, ρ is the proportion of I picture in MVC and n is the number of intra overlay. Then the ratio $\lambda = OBRG_i/r_{IS}$ indicates what proportion of r_{IS} should O_i have, to offset its deficit in bandwidth resource, which also determines how much rich overlay should substitute O_i for dispersing I_i. If $\lambda > 1$ O_i's peers will stop responding to inter overlay requests. To realize the substitution, the server sends a packet informing lean overlay O_i to ban inter overlay requests for the λ proportion of I_i and on the other hand, informs rich overlay O_j to request the banned part from the server and disperse it instead.

4. PERFORMANCE EVALUATION

We evaluate the performance of our scheme Collaborative MVC, with others, namely simulcast and MVC using simulation experiments. We adopt MVC with *KS_PIP* prediction structure in our experiments. [1]'s experiment indicates that *KS_PIP* maintains MVC's advantage in coding gains(average 1.4db versus full MVC's 1.6db) with 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[8]. The streaming rate for simulcast solution is set to 300kbps and the equivalent rate for MVC. For MVC, we set Group of Picture(GOP) size at 8. B/P picture has equal size and I picture is 15 times the size of them. To simulate the bandwidth heterogeneity of peers, we use four typical DSL nodes with upload capacities of 3Mbps, 1Mbps, 384kbps and 128kbps respectively. we adjust their fraction to obtain varied peer resource index (PRI). PRI is defined as the ratio of the total peer upload capacity to the minimum bandwidth resource demand. We assume that the bottleneck is always at the last hop and the server bandwidth is large enough. End users will request packets from the server if the packets do not arrive before deadline, hence all end users can watch a full quality of video. So in this section, we mainly study the server bandwidth consumption with respect to varied PRI conditions and view channels under different user behaviors and network conditions.

Fig. 2 shows the server bandwidth costs with respect to different PRI in static environment. As shown that, when PRI increases, our method achieves dramatic reduced bandwidth costs compare with the others. When PRI is above 1.5, collaborative MVC consumes nearly 40% less bandwidth than others. Fig. 3 shows the peer's upload bandwidth utilization and explains the previous results: the utilization of Collaborative MVC is always kept above 0.9 which means that peer's upload resource is effectively utilized to deliver contents and cooperate in exchanging I pictures which contributes to reduced server costs. Utilization of Simulcast is about 0.7 in average, indicating that simulcast can not make full use of peer upload resources. Although MVC achieves high utilization, I pictures are still requested from server by each view channel respectively. The reduction in server bandwidth costs is limited by this redundant delivery.

Fig. 4 suggests the server bandwidth costs with increased view number of the same multiview video. We can see that owing to the sharing of I picture and collaboration among different view channels, introducing additional view channel will have restrained impact on server bandwidth costs as new channel requests much video contents from other channels instead of the server. Fig. 5 shows the performance when the peer number scales. We can see that Collaborative MVC's advantage in reduced server bandwidth costs is not affected by the increasing number of peers. We investigate in high peer churn environment in Fig. 6. We use Weibull(λ, k) distribu-

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

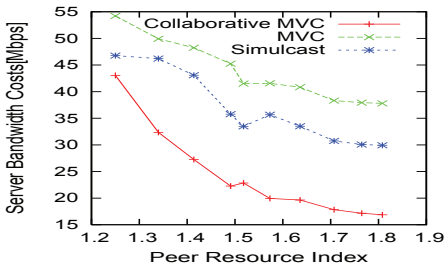

Fig. 2. Bandwidth costs on server with respect to varied PRI in static environment. 8 Views, 100 Users per channel.

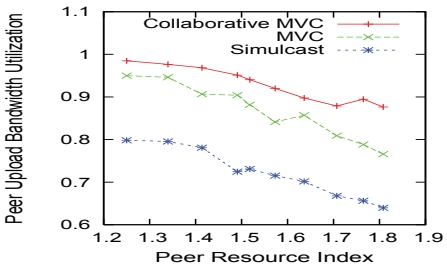

Fig. 3. Peer upload bandwidth utilization with varied PRI in static environment. 8 Views, 100 Users per channel.

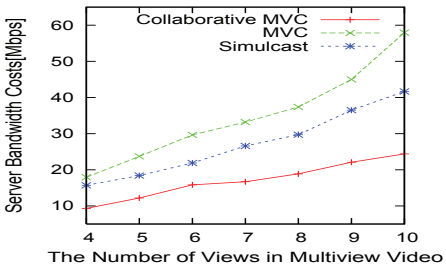

Fig. 4. Bandwidth costs on server with respect to the number of views of MV. PRI = 1.65, 100 Users per channel.

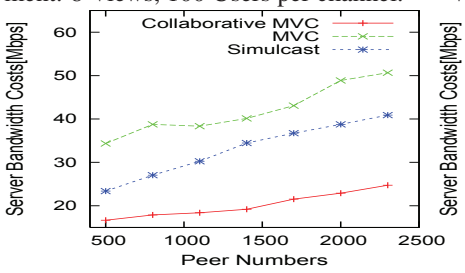

Fig. 5. Bandwidth costs on server with respect to peer numbers. 8 Views . PRI = 1.75.

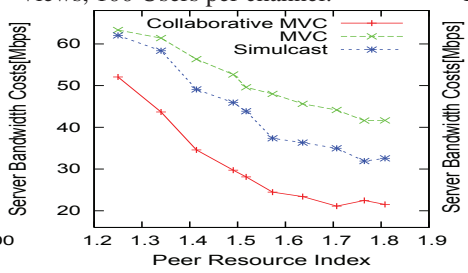

Fig. 6. Bandwidth costs on server with respect to varied PRI in dynamic environment(Weibull(500,2)). 8 Views

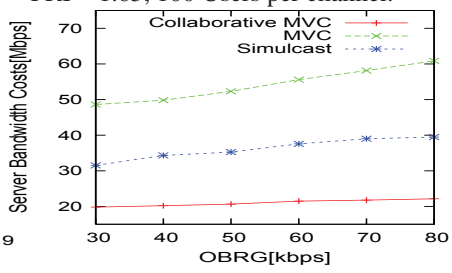

Fig. 7. Bandwidth costs on server with respect to OBRG. 8 Views

tion 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 in this figure. The results indicates that even in high churn environment, our method still achieved remarkable reduction in server bandwidth costs.

To simulate the scenario of heterogeneous view channels, we let 4 channels have 40 viewers respectively and identical OBRG as the lean overlays, and the other 4 channels have 200 viewers with abundant upload resources as the rich overlays. We vary OBRG to study the performance of our adaptation under different lean overlay resource conditions. As shown in Fig. 7, when OBRG increases, our method has an negligible growth in server bandwidth costs comparing with the steady increase in Simulcast and MVC. This can be attributed to the cooperation nature of our method and the adaptation: the surplus bandwidth resources of rich overlays are effectively utilized to disperse lean overlays' I picture slicings and let them devote their limited resources to deliver packets within their intra overlays, saving the server from allocating additional resources to sustain streaming service for lean overlays.

5. CONCLUSION

In this paper, we propose a novel collaboration framework in the delivery of multiview video, which enables effective cooperations among different view channels and utilize surplus bandwidth resources of rich overlays to help lean overlays in order to reduce bandwidth costs on streaming server. Simulation results have shown that our method achieves remarkable reduction in server bandwidth costs than existing solutions while maintaining scalability and resilience to user dynamics.

6. REFERENCES

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

[2] P. Merkle and et al, "Efficient compression of multiview video exploiting inter-view dependencies based on h.264/mpeg4-avc," in *IEEE ICME 2006*.

[3] Joint Video Team of ITU-T VCEG and ISO/IEC MPEG, "Comparative study of mvc prediction structures," *DOC-JVT*, vol. Jan, 2007.

[4] E. Kurutepe, M.R. Civanlar, and A.M. Tekalp, "Client-driven selective streaming of multiview video for interactive 3dtv," *IEEE CSVT*, Nov. 2007.

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

[6] Li Zuo, Jian Guang Lou, Hua Cai, and Jiang Li, "Multicast of real-time multi-view video," in *IEEE ICME 2006*.

[7] M. Zhang, Q. Zhang, L. Sun, and S. Yang, "Understanding the power of pull-based streaming protocol: Can we do better?," *IEEE JSAC*, Dec. 2007.

[8] "Meridian node to node latency matrix (2500×2500)," meridian project, 2005.