LOW-DELAY MESH WITH PEER CHURNS FOR PEER-TO-PEER STREAMING

Y.-T. Hillman Li[†] *Dongni Ren*[†] *S.-H. Gary Chan*[†] *Ali C. Begen*[‡]

[†]Department of Computer Science and Engineering
The Hong Kong University of Science and Technology
Clear Water Bay, Kowloon, Hong Kong, China
[‡]Video & Content Platforms, Research & Advanced Development
Cisco Systems Inc., San Jose, CA 95134, USA

1. INTRODUCTION

Peer-to-peer (P2P) streaming has been shown to be a feasible solution to stream live media to large groups [1]. In order to accommodate heterogeneous uplink bandwidth and achieve robustness against node churns (i.e., departures and failures), a P2P network is often constructed using a mesh. By aggregating the substreams from the parents, a peer can assemble a full stream even in the presence of heterogeneous and fluctuating bandwidth. Because of multiple parents, stream continuity can be better maintained in spite of parent churns.

Traditionally, a peer in a streaming mesh keeps switching and connecting to some nearby parents until a full stream can be assembled. Most of the existing work on mesh is based on random gossip where peers connect to some closest participating peers for data exchange in a rather ad hoc manner [2, 3, 4]. There has not been sufficient work on considering how to reduce source-to-end delay while meeting a certain stream continuity requirement in the presence of bandwidth constraints and peer dynamics (related to node holding time, search time, etc.). We study these issues in this paper. Our previous work [5] has not covered node dynamics and its relationship to the number of parents. In this work, we explicitly consider node dynamics and meeting a certain stream quality requirement.

The video stream is split into k substreams, where $k \geq 1$ (by means of, for examples, multiple description coding, network coding, or simply multiplexing data chunks). Each of these k substreams is pushed into the network spanning trees in such a way that a peer aggregates these substreams from k distinct *streaming* parents to form a full stream. Because all parents are distinct, the interruption due to the churning of any parent is limited. Because packets are lost due to node churns, a peer also connects to an additional t *backup* parents, where $t \geq 0$. The backup parents provide redundancy and robustness to protect peers against disruption caused by the churning of streaming parents. They reserve bandwidth for their children, and retransmit packets (i.e., pulled by children). If a streaming parent leaves the system, a backup replaces it to provide the missing substream. This way, streaming continuity can be maintained without interruption.

Note that all video substreams have to be received at a node before the video can be played back. This means that the (overall) delay of a node is decided by the slowest path among all the spanning trees of the substreams. In other words, the delay of a node is the maximum overlay distance from the source to the node given all its $(k + t)$ parents.

We propose a distributed algorithm that constructs a mesh with a very low delay given a certain number of parents. We study this scheme using simulations and show that our distributed mesh algorithm performs better than the approach adopted by many existing P2P systems.

This work was supported, in part, by the Cisco University Research Program Fund, a corporate advised fund of Silicon Valley Community Foundation (SVCF08/09.EG01), and the Hong Kong Innovation Technology Fund (ITS/013/08).

2. A LOW-DELAY DISTRIBUTED ALGORITHM

When a new peer i arrives, it has to connect to k streaming parents with bandwidth to assemble a full data stream. To achieve that, it first contacts a rendezvous point (RP) that caches a number of peers in the overlay. These nodes are put into the candidate pool. The peer also enlarges the candidate pool by requesting for neighbors from these nodes.

Then, peer i checks the network distance with respect to each candidate j. Peer i also asks j for source-to-end delay in each spanning tree and the amount of available bandwidth. After collecting this information, peer i arbitrarily picks a delivery tree l, selects the peer with minimum path delay among all the candidates in this tree and connects to it to retrieve the substream. The selected parent is removed from the pool and i repeats this process for the remaining delivery trees. This way, it joins all the k delivery trees and fulfills the streaming rate requirement.

After joining all k delivery trees, the new peer i then connects to t backup parents. It searches for backup parents from the candidate pool. The pool consists of peers that possess available bandwidth and are not the streaming parents of i. In addition, peer i prefers those backup parents that do not share a common streaming parent as i. This way, the backup parents are unlikely to be affected by the departure of i's streaming parents. Thus, they provide robust backup service to i.

Upon detecting that a streaming parent of tree l has left, the child contacts one of its backup parents to receive the substream l; by doing so, the backup parent becomes a streaming parent. As a result, the child experiences little interruption. Note that the child now has one less backup parent than before; therefore, it looks for another backup parent by following the search process discussed above.

Peers are required to observe their path delay with respect to the source. This implies that all peers need to have their clocks synchronized. When entering the system, every peer contacts the RP and checks its own time with the clock kept by RP. This way, peers adjust their clocks to the system time.

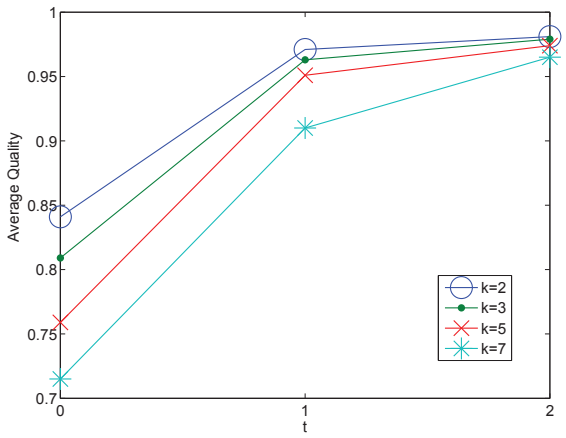

Fig. 1. Average quality under different t and k.

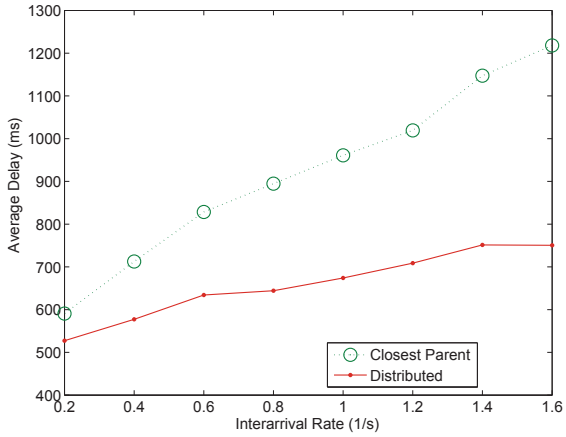

Fig. 2. Source-to-end delay of schemes under study.

3. ILLUSTRATIVE SIMULATION RESULTS

We carry out simulations to evaluate the performance of the proposed algorithm. We also simulate a traditional scheme, the *closest parent*, for comparison purposes. In this scheme, peers look for the closest parents for streaming. Since this scheme captures locality (usually defined by round-trip time) among peers, it is widely adopted by many P2P streaming systems. To be fair in comparison, peers in both schemes look for the same number of parents to achieve the same level of streaming quality.

We use Brite to generate different two levels top-down hierarchical Internet-like topologies. The access link bandwidth of the peer is uniformly distributed between 250 and 2000 Kbps in integral steps. We consider peers arriving in the system according to a Poisson process (with a rate of 1 peer/s). The streaming rate is 500 Kbps. The holding times of peers are exponentially distributed with a mean of 450 seconds.

In order to be fair in comparing the performance, the schemes under study should all provide comparable levels of streaming quality. We require peers to achieve an average video quality of 95% (i.e., packet loss rate is 5%). We show in Figure 1 average quality versus t given k.

Backup parents are always necessary to provide high quality of streaming; otherwise peers experience data interruption from time to time. There should be at least one backup redundancy in order to achieve the target level for small k ($1 \leq k \leq 5$). As k increases (for $k > 5$), two backup parents are needed in order to achieve the same level of quality.

We compare the closest parent scheme with our scheme by plotting the average delay versus interarrival rate in Figure 2. We use $k = 5$ and $t = 1$ for this set of simulation. In general, the delay increases with the arrival rate. When peers arrive more frequently, the system population increases, which leads to a longer overlay diameter, and hence, delay. The results show that our scheme performs significantly better than the closest parents scheme.

The intuition behind closest parents scheme is that peers prefer close parents so that peers can form groups. Data reaching any member of the group will disseminate to other group members quickly. However, it often takes a long trip to transmit data to the groups far away from the source. In our scheme, peers always prefer a parent that is close to the source. That is those parents that receive data earlier. This encourages peers to connect to parents on the upstream and so the bandwidth of upstream peers is better utilized. This way, peers are put as near to the source as possible. Thus, the distributed algorithm results in short source-to-end delays.

4. CONCLUSIONS

In this study, we discuss how to provide low-delay peer-to-peer streaming with high video quality by considering mesh design with backup parents. To achieve robustness against peer churns, each peer has a certain number of streaming parents and backup parents. We have designed a distributed algorithm that constructed a low-delay mesh and at the same time achieved a certain stream continuity for the peers.

We have conducted extensive simulations to study the performance of our algorithms. The results show that our distributed algorithm achieves a lower source-to-peer delay as compared with a traditional scheme. Our results have shown that peer-to-peer live streaming can be delivered in short delays while providing a high level of quality, despite peer churns and the lack of a centralized planner.

5. REFERENCES

[1] Xiaojun Hei, Chao Liang, Jian Liang, Yong Liu, and Keith W. Ross, "A measurement study of a large-scale P2P IPTV system," *IEEE Transactions on Multimedia*, vol. 9, no. 8, 2007.

[2] Xiaofei Liao, Hai Jin, Yunhao Liu, Lionel M. Ni, and Dafu Deng, "Anysee: Peer-to-peer live streaming," in *Proc. IEEE Infocom*, 2006.

[3] X. Zhang, J. Liu, B. Li, and T.-S. P. Yum, "CoolStreaming/DONet: A data-driven overlay network for live media streaming," in *Proceedings of IEEE INFOCOM*, Miami, FL, USA, Mar. 2005, pp. 2102–2111.

[4] V. Pai, K. Kumar, K. Tamilmani, V. Sambamurthy, and A. Mohr, "Chainsaw: Eliminating trees from overlay multicast," in *The Fourth International Workshop on Peer-to-Peer Systems*, feb 2005.

[5] Dongni Ren, Y.-T. Hillman Li, and S.-H. Gary Chan, "On reducing mesh delay for peer-to-peer live streaming," in *IEEE INFOCOM*, Phoenix, Arizona, Apr. 2008, IEEE.