DELAY-BASED OVERLAY CONSTRUCTION IN P2P VIDEO BROADCAST

Jacob Chakareski and Pascal Frossard

Signal Processing Laboratory (LTS4)
Ecole Polytechnique Fédérale de Lausanne (EPFL), Lausanne 1015 - Switzerland

ABSTRACT

We consider streaming video content over an overlay network of peer nodes. Each of the nodes employs a mesh-pull mechanism to organize the download of data units from its neighbours. We propose a novel algorithm for constructing the distribution overlay, where peers are arranged in neighbourhoods that exhibit similar latency values from the origin media server. Such an organization increases data sharing between neighbours in broadcast applications and reduces the play-out latency at a peer. Each of the nodes in the overlay is further equipped with a packet scheduling procedure that requests data units from neighbours in the order of their importance and their popularity within the neighbourhood. Finally, requesting peers share the upload bandwidth of a sending peer in proportion to their transmission rate to that peer in order to discourage free-riding in the system. Our simulation results show that the proposed mesh construction procedure provides improved performance in terms of frame-freeze and playback latency relative to a conventional approach where peer neighbours are selected at random. Corresponding gains in video quality for the media presentation are also registered due to the improved continuity of the playback experience.

Index Terms— peer-to-peer systems, video streaming, distributed computation and control, overlay networks.

1. INTRODUCTION

Delivering live or streaming video content over networks of peers is increasingly becoming common nowadays. Propelled by the steady increase of residential access bandwidth and an audience ever more hungry for a multimedia experience on the Internet, a class of applications have emerged that enable sharing data packets with delivery deadlines over peer-to-peer (P2P) overlay networks. Systems like PPLive [1], PPStream [2], and Coolstreaming [3], have been successfully deployed and tested for broadcasting/multicasting live events and for streaming pre-encoded content to large audiences in the Internet.

Still, the present P2P multimedia experience is marred by uncontrollable start-up delays, frequent freezes of the multimedia playback, and significant fluctuations of audio-video quality. In our view, there are two major reasons for these apparent shortcomings. First, the construction of the overlay network over which the data is delivered and the exchange mechanisms that the peers employ to share their data have been mainly carried over from earlier P2P file/data sharing applications. Hence, as such they are ill equipped to deal with the specificities of multimedia data, such as delivery deadlines and unequal importance for the reconstruction quality. This in turn makes them inefficient in terms of streaming performance for the multimedia presentation that they serve. Second, the presence of free-riding in a system also has a negative effect on the overall performance as it counters the main premise of P2P overlay networks, i.e., that the available system bandwidth increases with the number of peers. Specifically, free-riders are peers that want to obtain content from other peers, but that do not want to serve peers with their own content. Hence, effectively this is manifested as a reduction in serving bandwidth to some peers which in turn causes extended delays and variable audio-video quality of the multimedia presentation at these peers.

To address these deficiencies we design a mesh construction procedure that creates neighbourhoods of peers that exhibit similar delivery delays relative to the broadcast media server. This increases the likelihood of data availability in a neighbourhood thereby reducing the playback latency and the frame freeze frequency of the media presentation at the peers. These improvements also result into a corresponding gain in video quality for the reconstructed presentation. In addition, we design a receiver-driven algorithm for requesting media packets from neighbouring peers that takes into consideration the packets' delivery deadlines and importance for the reconstruction quality of the media presentation. Furthermore, the popularity of a packet within a neighbourhood is also included in the decision mechanism that is part of the algorithm in order to help the dissemination of less frequently encountered data units. Finally, we design a bandwidth sharing procedure that distributes a peer's upload bandwidth among its requesting neighbours in proportion to their own data rate contribution to this peer. Hence, a free-rider is effectively shut down from receiving any useful data from its neighbours, as its rate contribution to them would typically be non-existing.

To our knowledge, the most closely related contemporaneous works are the following. In [4], the authors design a global pattern for content delivery in mesh-based overlays that can utilize the upload bandwidth of most of the peers. In addition, a sweet range for the peers' degree is identified that maximizes the delivered quality to the individual peers in the scenario under consideration. Furthermore, the work in [5] shows how the buffer maps that peers in mesh-based overlays construct in order to facilitate exchange of data with their neighbours can be used to monitor the network-wide quality of the media presentation shared among the peers. Finally, [6] proposes to use layered video for providing incentives in P2P live streaming. In particular, video packets are requested from neighbours in prioritized order based on their layer index and the probability of serving a neighbour is commensurate to the rate contribution received from this neighbour.

2. MINIMUM DELAY MESH CONSTRUCTION

The key idea of the technique described here is to organize the peers in neighbourhoods featuring similar delays from the media server for all their members. Specifically, we desire to construct a neighbourhood such that every peer in it will exhibit a roughly equal latency in receiving the media packets sent by the server originally. This will increase the likelihood of having the peers in a neighbourhood being

interested in obtaining content from one another. That is because the sliding windows used to exchange content with other peers (see Section 3.1) will be positioned more or less at the same location relative to the time line of the media presentation. At the same time, this feature will contribute to a lower likelihood of a frame freeze during playback. This is the situation where a video frame is not received/decoded by a peer by the time it is due to be displayed. Hence, the peer freezes the last displayed frame on the screen in order to conceal the loss of this (present) frame. As the peers have bigger prospects for querying data from their neighbours now, the chance of encountering frame freeze during playback decreases. Finally, another advantage of such a neighbourhood is shorter playback start of the media presentation at a peer. In particular, as data sharing among the members of a neighbourhood is increased, the time that it takes for a peer to acquire the necessary amount of data in order for playback to start is reduced. In the remainder of this section, we describe in detail the proposed mesh construction approach.

There is a registry server that keeps track of the peers already in the network. For each peer the server maintains an entry comprising the peer's IP address and the minimum delay that this peer measures with respect to the media server. The nature of this second quantity will become clear as we proceed with the discussion here. The registry server maintains a sorted list of the registered peers in increasing order based on their minimum delays.

A connect procedure for a peer joining the network comprises the following steps. The peer contacts the registry server providing it its IP address. In return, the server provides the sorted list of peers that the connecting peer can use to establish its own neighbourhood in the network. Specifically, the peer begins to contact the nodes in the network starting from the head of the list, checking for the following two quantities. The peer wants to know if the contacted node can accept a new neighbour[1]. At the same time, the peer measures the network delay from the node, based on the data rate that the node is willing to spend on sending packets to the peer. Once the peer has a sufficient number of nodes willing to accept it as their neighbour, the peer creates its own neighbourhood with these nodes. At the same time, the peer provides to the registry server its own minimum delay entry which the peer computes as follows. The peers adds the delay it has measured from a neighbour to the minimum delay value for that neighbour from the sorted list provided by the registry server. The smallest of these values for the neighbourhood of the peer is returned to the registry server[2]. This concludes the connect procedure for a peer.

In Figure 1, we show an illustration of a mesh constructed using our procedure. Going top to bottom, we can see that the delay that the nodes experience from the media server increases. Still, nodes within a neighbourhood, denoted in dashed ellipses in Figure 1, exhibit similar latencies in receiving media packets issued by the server earlier, which in turn increases their collaboration in delivering these packets to one another, as argued previously. Please note that for clarity we do not include in Figure 1 every connection between the nodes in the network.

Periodically, the nodes can check if their minimum delays have changed. This can happen as the data rates at which they receive data from their neighbours can change over time. Hence, a node can update then its minimum delay value stored with the registry server.

[1]Recall here that the nodes usually maintain a bound on the number of neighbours that they are willing to accept in order not to overwhelm their resources by the demanding/requesting neighbours.

[2]We assume that all the latency between two nodes in the network occurs at their access points. Hence, a joining node always measures a smaller overall delay to the media server through a node higher in the sorted list.

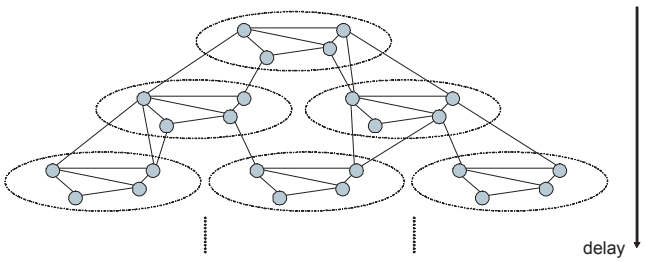


Fig. 1: An example of a minimum delay mesh construction. Peers are represented as small circles and neighbour nodes featuring similar delivery delays are grouped in sets (ellipsis). The later are organized further along increasing latency.

At the same time, the node informs its neighbours that its own minimum delay has changed so that they can revaluate in turn and if needed the minimum delay values for their respective neighbourhoods. Note that the case of departure of one or multiple neighbours is covered with the above consideration as then the data rate(s) of the departed neighbour(s) to a node would also change (become zero). Lastly, a departing node can inform the registry server of its decision so that its registry entry can be removed.

3. SCHEDULING FRAMEWORK

3.1. Receiver-Driven Packet Requests

Each peer maintains a sliding window of data units that periodically advances. A peer buffers the already received data units from this window, while it seeks to request the rest of them from its neighbours. Peers periodically exchange maps describing the presence/absence of data units from their windows. In this way, a peer can discover the presence of missing data units in the neighbourhood.

More formally, let W denote the set of data units in the current sliding window at peer n and let $M \in W$ denote the subset of missing data units from W that the peer can request from its neighbours. For each data unit $l \in M$, the peer computes its augmented importance as $I_l \cdot P_l(k, N) \cdot U(t, t_{d,l})$, where k is the number of peers in the neighbourhood of n from which this data unit is available and N is the size of the neighbourhood of n in number of peers. Furthermore, I_l is the importance of the data unit for the reconstruction quality of the media presentation [7], while $P_l(k, N)$ is the popularity factor of data unit l in the neighbourhood of peer n. In essence, $P_l(k, N)$ is a monotonically decreasing function of the ratio k/N and is used to place greater emphasis on data units less frequently encountered among the peers in order to alleviate their dissemination within the network. Similarly, $U(t, t_{d,l})$ denotes the urgency of receiving this data unit by peer n, where t is the current time, and $t_{d,l}$ is the delivery deadline for l, as introduced earlier. $U(t, t_{d,l})$ is a monotonically increasing function of the ratio $t/t_{d,l}$ and is used to place priority on data units due to be received and decoded sooner in the future.

The data units from M are sorted in decreasing order based on their augmented importance values. Peer n then requests the data units from M in that order starting from the head of the sorted list. For each data unit to be requested, n considers selecting the neighbour from which this data unit would be delivered fastest such that its delivery deadline would not be exceeded. In particular, let l be the next data unit to be requested from the sorted list. Node n computes

the probability of receiving l from each of the neighbours that has it, and ranks them in decreasing order based on it. If the first entry in the ranked list has a non-zero probability of delivering the data unit to n, the node then selects the corresponding peer neighbour and sends it a request for l. Otherwise, n does not request this data unit and proceeds to the next entry in the sorted list of data units to be considered for requesting. It should be noted that computing the probability of receiving a data unit from a peer involves not only the statistical properties of the communication channels (forward/backward) between n and that peer, but also the number of pending requests from n to which the peer has not responded yet.

3.2. Download Rate Estimation and Peer Replacement

A peer periodically estimates the respective download rates from its neighbours. This is done by computing the total amount of data received from each neighbour since the last time the download rate was computed. In this way, a peer can sort its neighbours based on their send rate contributions to this peer. Then, the peer can periodically replace the least contributing neighbour with a new peer selected using the procedure described in Section 2. Furthermore, if the peer experiences multiple neighbour nodes with no rate contribution, it will simultaneously replace all of them with newly selected neighbours. The replaced nodes in this latter case will typically represent free-riders that are not interested in sharing their resources with other nodes in the network.

3.3. Sender Upload Bandwidth Sharing

The algorithm for sharing the upload bandwidth of a peer among its requesting neighbours operates as follows. Let UB_n be the upload bandwidth of node n, and let PR_n denote the set of neighbours from which n has pending requests at present. Then, to every node $k \in PR_n$, n allocates the share of its upload bandwidth $r_{n,k} = (\tilde{r}_{k,n} / \sum_{k \in PR_n} \tilde{r}_{k,n}) UB_n$, where $\tilde{r}_{k,n}$ denotes the present estimate of the sending rate from k to n. Hence, nodes that contribute more of their sending rate to n will receive in return a larger share of its own upload bandwidth.

4. EXPERIMENTS

In this section, we examine the performance of the proposed framework for streaming actual video content. In the simulations, we employ the common test video sequence *Foreman* in CIF image size encoded at 30 frps using a codec based on the scalable extension (SVC) of the H.264 standard [8]. The content is encoded into four SNR-scalable layers, with data rates of 455 kbps, 640 kbps, 877 kbps, and 1212 kbps, respectively. The corresponding video quality of the layers is 36.5 dB, 37.8 dB, 39.1 dB, and 40.5 dB, respectively, measured as the average luminance (Y) PSNR of the encoded video frames. The group of pictures (GOP) size of the compressed content is 30 frames, comprising the following frame type pattern IBBPBBP..., i.e., there are two B-frames between every two P frames or P and I frames. The 300 frames of the encoded sequence are concatenated multiple times in order to create a 900 second long video clip that is used afterwards in our simulations.

The P2P network in the experiments comprises 1000 peers, out of which 5% are free-riders, while we distribute the rest in two categories: cable/dsl peers and ethernet peers, in the ratio 7:2.5. The upload bandwidth for ethernet and cable/dsl peers is 1000 and 300 kbps, respectively, while the corresponding download bandwidth values for these two peer type categories are 1500 kbps and 750 kbps. The downlink data rate for free-riders is set to 1000 kbps. The content is originally stored at a media server with an upload bandwidth of 6 Mbps. The play-out delay for the presentation is set by the peers to 15 seconds. This is the initial amount of data that each peer needs to accumulate in its buffer before starting the playback of the presentation. The size of the sliding window introduced in Section 3.1 for keeping track of data units at each peer is 30 seconds of data. Sending requests to its neighbours is considered by a peer at intervals of 1 sec. The contribution of each sending peer in terms of data rate is measured by the receiving peer every 30 seconds of time. The exclusion of the least contributing peer in a neighbourhood and the consecutive selection of a new replacement neighbour is done by a peer every 30 sec.

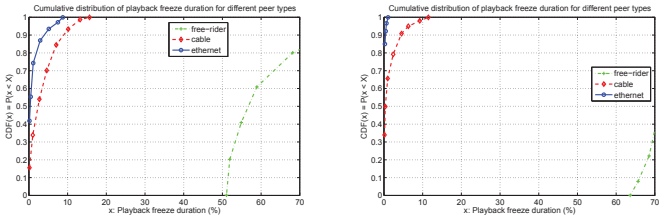
The performance of the packet requesting algorithm from Section 3.1 relative to existing schemes and the resilience of our framework to the influence of free-riders were examined elsewhere [7]. Therefore, in this paper we focus on examining the additional improvement in performance that the mesh construction procedure from Section 2 provides. In particular, we present the relative performance gains of the Min Delay algorithm from Section 2 over a standard Random mesh construction technique where each peer selects its neighbours at random.

The metrics over which we measure performance are (i) frame freeze duration and (ii) normalized play-out time. We describe them in the following. Frame freeze duration is the percentage of time relative to the duration of the whole presentation during which a peer experiences frozen video content on its display. Remember that this happens whenever a video frame is not received and decoded by the peer by its decoding/delivery deadline. In order to compensate for this, the peer conceals this frame with the last decodable frame that it has in its buffer. The content of this latter frame is kept on the screen (hence the name freeze frame) until a subsequent frame is decodable and therefore ready to be displayed next.

Next, recall from earlier that a peer has a parameter denoted play-out delay that is set ahead of time. As described previously, this parameter corresponds to the amount of data that the peer needs to buffer from the initial part of the presentation before the playback actually starts at the peer. Typically, it would take a peer a longer period of time than the actual value of its play-out delay parameter to gather the necessary amount of data for the playback to start. Furthermore, one can compute the absolute minimum of this quantity based on the hop distance of a peer from the media server and the data rate at which the server is streaming the presentation (typically the encoding rate of the presentation). Hence, we define normalized play-out time as the ratio of the actual time that a peer requires to fill up its play-out buffer initially and the minimum value of this quantity, as described above.

First, in Figure 2 we compare the difference in frame freeze duration experienced by the peers when each of the two mesh construction algorithms is used. Specifically, in Figure 2a we show the cumulative distribution functions (CDF) of the frame freeze duration for the three peer types in the case of random mesh construction, while in Figure 2b we show the corresponding results for the case of minimum delay mesh construction. It can be seen that by employing the latter algorithm both ethernet peers and cable peers experience a significant reduction in frame freeze duration. For example, now 90% of the cable peers experience frame freeze for not more than 5% of the time while the media presentation is playing at their ends, relative to around 10% of the time for the case of random mesh construction, as observed from the corresponding graphs in Figures 2b and 2a, respectively. Similarly, when the minimum delay algorithm is used none of the ethernet peers experience frame freeze longer

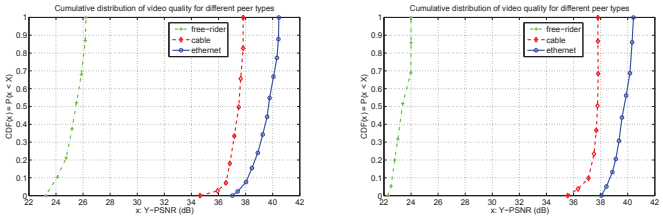
than 2 - 3 % of the time, compared to the case of random mesh construction where 10% of these peers experience frame freeze in the range 5 - 9 % of the time. Note that in both cases (a) and (b), the performance of free-riders is quite degraded, which is in fact desirable and is due to the send rate proportional uplink bandwidth sharing scheme from Section 3.3.



(a) Random (b) Min Delay

Fig. 2: Frame freeze duration (%) for different peer types and mesh construction algorithms.

The reduction in frame freeze duration when the Min Delay algorithm is used should result into a corresponding improvement in average video quality observed by the peers. This is confirmed with the graphs shown in Figure 3 that represent the distribution of video quality for the different peer types in the case of (a) Random and (b) Min Delay. Here, video quality is measured as the average Y-PSNR (dB) of the reconstructed video frames at each peer. It can be seen that the performances of ethernet peers and cable peers have improved as their CDF graphs are somewhat shifted to the right in Figure 3a relative to those in Figure 3a for Random mesh construction. Furthermore, the performance of free-riders in both cases is quite degraded and is due to the same reason explained earlier.



(a) Random (b) Min Delay

Fig. 3: CDF of average video quality (Y-PSNR) for different peer types and mesh construction algorithms.

Next, we study the differences in normalized play-out time for a peer between random and minimum delay mesh constructions. In Figure 4, we show the cumulative distribution functions of the normalized play-out time for a peer for each of the two mesh construction algorithms. As expected, we can see from Figure 4 that when our algorithm is employed the peers observe much shorter play-out times which in turn improves their audio-visual experience of the media presentation. For example, for 90% of the peers the playback of the presentation can start no longer than twice the preset play-out delay in the case of minimum delay mesh construction, compared to about six times the preset play-out delay for the same percentage of peers for random mesh construction.

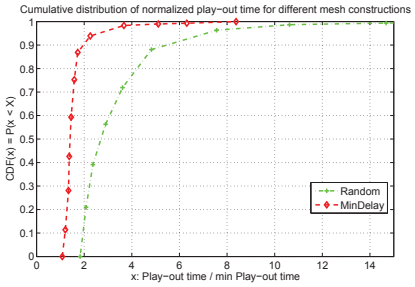


Fig. 4: CDF of normalized play-out time of a peer.

5. CONCLUSIONS

We have proposed a mesh-pull based P2P streaming framework. The framework comprises three major building blocks. (i) An overlay construction algorithm that creates peer neighbourhoods that exhibit similar latencies from the media server across their member nodes. (ii) An algorithm for requesting data from neighbours that maximizes the video quality at the peer while taking into account the popularity of the data units within the neighbourhood. (iii) A technique for sharing the upload bandwidth of a sending peer that effectively marginalizes the influence of free-riding in the system. Through experiments we examined the additional performance improvement that the mesh construction procedure provides relative to a frequently employed scheme of selecting peer neighbours at random. It is shown that significant reductions in frame-freeze time and play-out delay can be achieved if peer neighbours are selected such that they share common latency distance relative to the origin media server.

6. REFERENCES

[1] PPLive, ," http://www.pplive.com/.

[2] PPStream, ," http://www.ppstream.com/.

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

[4] N. Magharei and R. Rejaie, "Understanding mesh-based peer-to-peer streaming," in *Proc. Int'l Workshop on Network and Operating Systems Support for Digital Audio and Video*, Newport, RI, USA, May 2006, ACM, pp. 56–61.

[5] X. Hei, Y. Liu, and K.W. Ross, "Inferring network-wide quality in p2p live streaming systems," *IEEE J. Selected Areas in Communications*, vol. 25, no. 10, pp. 1640–1654, Dec. 2007.

[6] Z. Liu, Y. Shen, S. Panwar, K. Ross, and Y. Wang, "Using Layered Video to Provide Incentives in P2P Live Streaming," in *Proc. Workshop on Peer-to-Peer Streaming and IP-TV*, Kyoto, Japan, Aug. 2007, ACM SIGCOMM, pp. 311–316.

[7] J. Chakareski and P. Frossard, "Efficient video streaming in p2p networks," in *Proc. Conf. on Visual Communications and Image Processing*, San Jose, CA, USA, Jan. 2009, SPIE, to appear.

[8] ITU-T and ISO/IEC JTC 1, "Advanced video coding for generic audiovisual services, amendment 3: Scalable video coding," *Draft ITU-T Recommendation H.264 - ISO/IEC 14496-10(AVC)*, Apr. 2005.