Utility-based Packet Scheduling in P2P Mesh-based Multicast

Jacob Chakareski and Pascal Frossard

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

ABSTRACT

We consider streaming video content over an overlay network of peer nodes. We propose a novel streaming strategy that is built on utility-based packet scheduling and proportional resource sharing in order to fight against free-riders. Each of the peers employs a mesh-pull mechanism to organize the download of media packets from its neighbours. For efficient resource utilization, data units are requested from neighbours based on their utility. The packet utility is driven by both its importance for the video reconstruction quality at the receiving peer and its popularity within the peer neighbourhood. In order to discourage free-riding in the system, requesting peers then share the upload bandwidth of a sending peer in proportion to their transmission rate to that peer . Our simulation results show that the proposed protocols increase the performance of a mesh-pull P2P streaming system. Significant improvements are registered relative to existing solutions in terms of average quality and average decoding rate.

1. INTRODUCTION

Streaming video content over P2P networks is not any longer a distant prospect from the future, but rather a common place frequently encountered on the Internet. 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 P2P applications have emerged that enable sharing data packets with delivery deadlines over a network of peer nodes. 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.Among the main reasons for these apparent shortcomings we count the following. First, the design of the existing P2P streaming/multicast applications 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 end-to-end performance of 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.

In this paper, we present a streaming framework that attempts to address the issues in mesh-pull P2P systems described above. Specifically, we first design a receiver-driven algorithm for requesting media packets from neighbouring peers that takes into consideration the packets' utility for efficient use of the bandwidth resources. The utility of a packet is based on the packet's delivery deadline and its importance for the reconstruction quality of the media presentation, as well as on its popularity within the neighbourhood. This is done in order to facilitate the delivery of less frequently encountered data units in the network. A peer equipped with our algorithm can thus request the media packets that maximize the performance of its media presentation while contributing simultaneously toward the same goal at neighbouring peers. Therefore, a globally optimized performance of the presentation over the whole peer population is achieved. Then, we design a bandwidth sharing procedure that targets robustness against free-riders. In particular, a sending peers distributes its upload bandwidth among its requesting neighbours in proportion to their own contributions in terms of data rate 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. In addition, the bandwidth sharing procedure is augmented with a periodic search for new neighbours willing to contribute more rate to a peer where the newly selected neighbour replaces the least contributing peer from the original neighbourhood. Simulation results demonstrate that the proposed framework provides for efficient video streaming in P2P systems by outperforming substantially baseline mesh-pull based approaches in terms of average quality and useful bandwidth.

Due to its promise as a novel technology for delivering multimedia over the Internet at lower cost, P2P streaming has been studied considerably thus far. The solutions are generally built on either tree-based or mesh-based organizations of the peers. Despite the plethora of prior work on tree-push based P2P streaming, e.g.,,[4] we still have to wait to witness an actual deployment of such a system on the Internet. Virtually all such systems to date are rather mesh-pull based[*], and we will therefore focus on this type of solution, which generally offers increased robustness to the dynamics of a P2P system. From the extensive related work, we describe here the studies that are the most closely related to the present paper. In,[6] the authors address the mesh construction and 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[7] presents a method to monitor the network-wide quality of the media presentation, based on the buffer maps constructed by peers in mesh-based overlays in order to facilitate exchange of data with their neighbours. Finally,[8] is probably the most relevant prior work relative to our paper. The authors propose to use layered video in order to provide 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. Differently, the present work considers a generic video encoding that may not necessarily be layered. Furthermore, a more sophisticated model is proposed to determine the packets' importance when requesting data that may be overlooked if considering their layer index only. Lastly, we design a deterministic algorithm for upload bandwidth distribution over requesting peers which consistently rewards contributing peers instead of doing that on the average, as in.[8]

The rest of this paper is structured as follows. In Section 2, we describe the receiver-driven algorithm for requesting data from neighbours. The method for sharing the sending bandwidth of a peer among its neighbours is described in Section 3.2. Next, we examine the performance of our system via simulations in Section 4. Finally, we end the paper with concluding remarks in Section 5.

2. UTILITY-BASED SCHEDULING

In our system, each peers requests video data from a random subset of other peers that form its neighbourhood. Since video packets have typically very different importance for the reconstruction quality of the media presentation,[9] an effective usage of the bandwidth resources can be achieved by transmitting the most important packets first. Here, we propose a utility-based scheduling algorithm, where the utility of a packet is driven by its rate-distortion characteristics and its popularity within the neighbourhood.

2.1 Rate-distortion characteristics

A media presentation comprises data units that are output by an encoding algorithm when the content is originally compressed. The encoding process creates dependencies between the data units that can be abstracted as a directed acyclic graph, as illustrated in Figure 1. Each node in the graph represents a data unit, and an arrow from data unit l to data unit l' in the graph signifies that for decoding data unit l, data unit l' must be decoded first.

Each data unit in the presentation is characterized with the following quantities. B_l is the size of data unit l in bytes and $t_{d,l}$ is its *delivery deadline*. This is the time by which data unit l needs to be received in order to be usefully decoded[†]. Packets arriving after the delivery deadlines of the respective data units that they contain are discarded. Furthermore, $\mathcal{N}_c^{(l)} = \{1, \ldots, l\}$ is the set of data units that the receiver considers for error

[*]An interesting overview of mesh-pull based P2P video delivery and its commercial success for IPTV applications can be found in.[5]

[†]In MPEG terminology this is the so called decoder time-stamp.

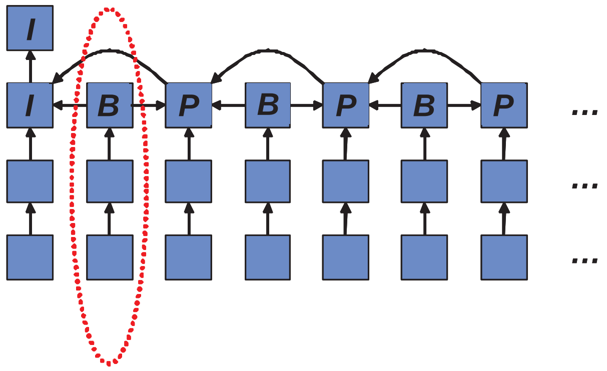

Figure 1. Dependency graph for data units of a media presentation.

concealment in case data unit l is not decodable by the receiver on time. Finally, $\Delta d_l^{(l_1)}$, for $l_1 \in \mathcal{N}_c^{(l)}$, is the reduction in reconstruction error (distortion) for the media presentation, when data unit l is not decodable but is concealed with data unit l_1 that is received and decoded on time. Note that $\Delta d_l^{(l)}$ denotes simply the reduction in reconstruction error when data unit l is in fact decodable. The media source model presented here has been adopted from,[10] which in turn represents a generalized version (accounting for decoder error concealment) of the model originally introduced in.[9]

2.2 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 respective windows. In this way, a peer can discover at its neighbours the availability of data units presently missing in its window.

More formally, let W denote the set of data units in the current sliding window at peer p and let $M \in W$ denote the subset of missing data units from W that the peer can request from its neighbours. The peer is interested in requesting these data units such that it maximizes its own utility of them. In particular, a peer may experience different reconstruction qualities of the media presentation played at its end commensurate to the media packets received in response to different request schedules. Therefore, peer n is interested in computing the optimal schedule for requesting data from neighbours that maximizes the reconstruction quality of its media presentation. In the following, we describe in detail the optimization problem that the peer is interested in solving.

2.2.1 Optimization Problem

Let $\boldsymbol{\pi}$ denote the collection of request schedules for the data units in M, i.e., $\boldsymbol{\pi} = \{\pi_1, \ldots, \pi_{|M|}\}$, where π_{l_m} is the request schedule for data unit $l_m \in M$, for $m = 1, \ldots, |M|$. Each request schedule π_m represents a matrix of size $|P| \times N$, where P denotes the set of neighbouring nodes of peer p. The row index n and the column index i of π_m correspond respectively to the neighbour $p_n \in P$ from which data unit l_m can be requested at time slot t_{i-1}, for $n = 1, \ldots, |P|$ and $i = 1, \ldots, N$. In particular, $t_0, t_1, \ldots, t_{N-1}$ represents a horizon of N time instances for requesting data units starting from the present time t_0. Given the above π_m comprises binary entries $a_{ni} \in \{0, 1\}$ that signify whether data unit l_m is requested at time slot t_{i-1} from neighbour p_n $(a_{ni} = 1)$ or the opposite is true, i.e., $a_{ni} = 0$. It should be noted that a data unit is not requested beyond its delivery deadline. Therefore, we set $a_{ni} = 0$ for $n = 1, \ldots, N$ and time slots $t_{i-1} \geq t_{d,l_m}$, for every data unit $l_m \in M$. Similarly, we set $a_{ni} = 0$ for $i = 1, \ldots, N$ for every neighbour p_n that does not have available data unit l_m in its current sliding window. Finally, let $\epsilon(\pi_{l_m})$ denote the *expected error* or the probability of not receiving data unit l_m on time given its request schedule π_{l_m}.

Following the approach in,[11] the expected distortion[‡] of the media presentation at peer p as a function of the request schedule $\boldsymbol{\pi}$ can be expressed as

$$\begin{aligned} D(\boldsymbol{\pi}) &= D_0 - \sum_{l_m \in M} \sum_{l_1 \in \mathcal{N}_c^{(l_m)}} \Delta d_{l_m}^{(l_1)} \prod_{j \in \mathcal{A}(l_1)} (1 - \epsilon(\pi_j)) \times \\ & \prod_{l_2 \in \mathcal{C}(l_m, l_1)} \left[1 - \prod_{l_3 \in \mathcal{A}(l_2) \setminus \mathcal{A}(l_1)} (1 - \epsilon(\pi_{l_3})) \right], \end{aligned} \tag{1}$$

where D_0 is the expected reconstruction error for the presentation if no data units are received. $\mathcal{A}(l_1)$ is the set of ancestors of l_1, including l_1. $\mathcal{C}(l_m, l_1)$ is the set of data units $j \in \mathcal{N}_c^{(l_m)} : j > l_1$ that are not mutual descendants, i.e., for $j, k \in \mathcal{C}(l, l_1) : j \notin \mathcal{D}(k), k \notin \mathcal{D}(j)$, where $\mathcal{D}(j)$ is the set of descendants of data unit j including data unit j itself. Finally, "$\setminus$" denotes the operator "set difference". We refer the reader to[11] for details on the derivation of the expression in (1).

Now, a request schedule will also induce a certain data rate on the downlink of peer p. This is the expected amount of data that the neighbours in P will send in response to $\boldsymbol{\pi}$. This quantity can be computed as

$$R(\boldsymbol{\pi}) = \sum_{l_m \in M} B_{l_m} \rho_(\pi_{l_m}), \tag{2}$$

where B_{l_m} is the size of data unit l_m in bytes, as introduced before, and $\rho_(\pi_{l_m})$ is the *expected cost* or redundancy of requesting data unit l_m under policy π_{l_m}. Precisely, $\rho_(\pi_{l_m})$ denotes the expected number of bytes sent per source byte of l_m on the downlink of peer p.

Finally, the peer is interested in minimizing the expected distortion $D(\boldsymbol{\pi})$ such that its downlink capacity $C^{(d)}$ is not exceed as a result. In other words, peer p is interested in computing the optimal policy $\boldsymbol{\pi}^*$ given as

$$\boldsymbol{\pi}^* = \arg\min_{\boldsymbol{\pi}} D(\boldsymbol{\pi}), \qquad \text{s.t. } R(\boldsymbol{\pi}) \leq C^{(d)}. \tag{3}$$

Using the method of Lagrange multipliers, we can reformulate (3) as an unconstrained optimization problem. That is, we seek the policy vector $\boldsymbol{\pi}$ that minimizes the expected Lagrangian $J(\boldsymbol{\pi}) = D(\boldsymbol{\pi}) + \lambda R(\boldsymbol{\pi})$ for some Lagrange multipliers $\lambda > 0$, and therefore achieves a point on the lower convex hull of the set of all achievable distortion-rate pairs $(D(\boldsymbol{\pi}), R(\boldsymbol{\pi}))$. As shown in,[11] a policy vector $\boldsymbol{\pi}$ that minimizes the expected Lagrangian $J(\boldsymbol{\pi})$ can be computed using an iterative descent algorithm called Iterative Sensitivity Adjustment (ISA).[9] However, due to the complexity of such an approach, we propose instead an approximation to the solution in (3) that is more suitable for being incorporated as a part of an actual system. We describe this low complexity approach in the next section.

2.2.2 Practical Solution

For each data unit $l_m \in M$, we define S_{l_m} to be the sensitivity of the media presentation to not receiving data unit l_m on time. This quantity can be computed as the overall increase in distortion affecting the media presentation by the absence of l_m at decoding, i.e.,

$$S_{l_m} = \sum_{j \in \mathcal{D}(l_m)} \Delta d_j^{(j)}, \tag{4}$$

where $\mathcal{D}(l_m)$ is the set of descendants[§] of data unit l_m and $\Delta d_j^{(j)}$ is the reduction of reconstruction distortion associated with data unit j. Both of these quantities were introduced earlier. Furthermore, we define I_{l_m} to be

[‡]In this paper, the terms "distortion", "reconstruction quality", and "utility" can be used interchangeably.

[§]For example, for the first "B" data unit in Figure 1, this is the collection of data units encircled in dotted red.

the current importance of data unit l_m for the overall quality of the reconstructed presentation. Using (4) we compute this quantity as

$$\mathcal{I}_{l_m} = \frac{S_{l_m}}{B_{l_m}} \cdot P_{l_m}(k, |P|) \cdot U(t, t_{d,l_m}). \tag{5}$$

We explain each of the multiplicative factors in (5) in the following. The term S_{l_m}/B_{l_m} represents the sensitivity of the media presentation per source byte of data unit l_m. In other words, S_{l_m}/B_{l_m} describes the distortion-rate tradeoff for the media presentation associated with requesting data unit l_m or not. We denote the second term $P_{l_m}(k, |P|)$ the *popularity factor* for data unit l_m in the neighbourhood of peer p. This quantity describes how often this data unit is encountered among the peer nodes in P. Specifically, based on the number of replicas k of data unit l_m found in P and the size of the neighbourhood $|P|$, the popularity factor returns a number that is inversely proportional to the ratio $k/|P|$. When the frequency of coming across l_m in P increases, the popularity factor decreases and vice versa. The motivation behind using such a factor is to alleviate the dissemination of data units less frequently encountered among nodes in the overlay. Finally, the last multiplying factor in (5) accounts for the various delivery deadlines that different data units may have relative to the present time t. In particular, the *urgency factor* $U(t, t_{d,l_m})$ provides a measure of relative urgency of data unit l_m with respect to t and among the data units in M. As the deadline of a data unit approaches t, its urgency factor increases. Conversely, for data units with delivery deadlines far into the future, this factor should exhibit respectively smaller values. The idea for employing an urgency factor when evaluating the present importance of the data units in M is to be able to give preference to data units that need to be received sooner by peer p due to their more pressing delivery deadlines.

The proposed light weight optimization algorithm for computing the request schedule for the data units in M operates as follows. First, the current importance values for data unit $l_m \in M$ and $m = 1, \ldots, |M|$ are computed using (5). These quantities are then sorted in decreasing order. Let M^{sort} denote the corresponding set of 'sorted' data units. That is the index of data unit l_m in M^{sort}, for $m = 1, \ldots, |M|$, corresponds to the location/position of its current importance $\mathcal{I}_{l_m}$ in the sorted list of these values. Next, starting from the first element of M^{sort} and moving toward its last one, we compute for each entry in M^{sort} the likelihood of receiving this data unit at p before its delivery deadline. In particular, let $l_{m_j} \in M^{\text{sort}}$, for $j = 1, \ldots, |M|$, be the data unit considered in the algorithm presently. Furthermore, let $P_{(l_{m_j})} \subset P$ denote the subset of neighbours of p that have data unit l_{m_j} available in their sliding windows at present. Then, for every node $p_{n_k} \in P_{(l_{m_j})}$ we compute the probability that data unit l_{m_j} will arrive at peer p no later than $t + t_{d,l_{m_j}}$ in response to a request sent by p to node p_{n_k} at present, i.e., at time t. In other words, this is the probability of experiencing a delay shorter than $t_{d,l_{m_j}}$ between the events of sending the request on the forward channel $p \to p_{n_k}$ and receiving the data unit on the backward channel $p_{n_k} \to p$. In the terminology of computer networks this delay is called the round-trip time and we denote it here $RTT_{(p,p_{n_k})}$. Hence, we compute $Prob\{RTT_{(p,p_{n_k})} < t_{d,l_{m_j}}\}$ for $p_{n_k} \in P_{(l_{m_j})}$ and $k = 1, \ldots, |P_{(l_{m_j})}|$. The algorithm selects to send a request for l_{m_j} to the node p_{n_k} that exhibits the highest nonzero $Prob\{RTT_{(p,p_{n_k})} < t_{d,l_{m_j}}\}$. Otherwise, if there is no such value[¶], the data unit is not requested and the algorithm proceeds to the next element of M^{sort}. Finally, once p goes through all data units in the 'sorted' set, it sends the computed requests to the appropriate nodes in P. The major computational steps of the algorithm are summarized in Figure 2 below. Next, we describe the procedure for computing the probabilities $Prob\{RTT_{(p,p_{n_k})} < t_{d,l_{m_j}}\}$.

2.2.3 Computing $Prob\{RTT_{(p,p_{n_k})} < t_{d,l_{m_j}}\}$

It should be mentioned that for computing $Prob\{RTT_{(p,p_{n_k})} < t_{d,l_{m_j}}\}$, the algorithm takes into account (i) the statistics of the communication channel from node p_{n_k} to peer p, (ii) any previous (pending) requests to this sender for which peer p has not received yet the corresponding data units[‖], and (iii) the estimated transmission

[¶]The probability of receiving this data unit on time from any of the prospective senders is zero.

[‖]This includes any data units $l_{m_i} \in M^{\text{sort}}$, for $i = 1, \ldots, j-1$, that are going to be requested in this round prior to l_{m_j} from the same sender p_{n_k}.

```
Given $M$, $P$, $t$
  (0) Initialize schedules :
        NodeSchedule = {}, DataUnitSchedule = {}.
  (1) For $l_m \in M$ and $m = 1, \ldots, |M|$
        Compute $\mathcal{I}_{l_m}$.
  (2) Sort $\{\mathcal{I}_{l_m}\}$ in decreasing order.
        $\Rightarrow M^{\text{sort}}$.
  (3) For $l_{m_j} \in M^{\text{sort}}$ and $j = 1, \ldots, |M|$ do
        For $p_{n_k} \in P_{(l_{m_j})}$ and $k = 1, \ldots, |P_{(l_{m_j})}|$ do
            Compute $Prob\{RTT_{(p,p_{n_k})} < t_{d,l_{m_j}}\}$.
        Find $MAX = \max_{p_{n_k}}\{\, Prob\{RTT_{(p,p_{n_k})} < t_{d,l_{m_j}}\}\,\}$.
        If $MAX > 0$
            Update schedules :
                Find $p^*_{n_k} = \arg\max_{p_{n_k}}\{\, Prob\{RTT_{(p,p_{n_k})} < t_{d,l_{m_j}}\}\,\}$.
                NodeSchedule = {NodeSchedule, $p^*_{n_k}$}.
                DataUnitSchedule = {DataUnitSchedule, $l_{m_j}$}.
  (4) Execute schedules :
        For $n = 1, \ldots,$ |NodeSchedule| do
            $p_n$ = NodeSchedule(n); $l_n$ = DataUnitSchedule(n).
            Send request to node $p_n$ for data unit $l_n$.
```

Figure 2. Computing the optimal policy for requesting data units from neighbours.

bandwidth of the channel $p_{n_k} \rightarrow p$. In particular, requesting a data unit comprises sending a small control packet to a designated neighbour. Moreover, the frequency of sending such packets is typically much smaller than the rate at which the corresponding data units are returned in response. That is because multiple data units can be requested with a single request packet. Hence, requesting data units typically consumes a very small fraction of the transmission bandwidth between two peers. Thus, it is reasonable to assume that the network effects in terms of delay and packet loss that requests experience on the forward channel $p \rightarrow p_{n_k}$ are quite marginal and can be ignored for practical purposes. This is the approach that we follow here as we associate the overall delay $RTT_{(p,p_{n_k})}$ in receiving a requested data unit to the characteristics of the backward channel $p_{n_k} \rightarrow p$ only.

Now, in order to be able to compute $Prob\{RTT_{(p,p_{n_k})} < t_{d,l_{m_j}}\}$ we need a statistical characterization for the backward channel. Here, we model $p_{n_k} \rightarrow p$ as a packet erasure channel with random transmission delays.[9] Specifically, packets carrying requested data units sent on this channel are either lost with a probability ϵ_B or otherwise they experience a random transmission delay y generated according to a certain probability distribution $f(y)$. Then, $Prob\{RTT_{(p,p_{n_k})} < t_{d,l_{m_j}}\}$ can be written as

$$Prob\{RTT_{(p,p_{n_k})} < t_{d,l_{m_j}}\} = (1 - \epsilon_B) \int_{y < t_{d,l_{m_j}}} f(y)dy. \tag{6}$$

In our case, we characterize the delay as exponentially distributed with a right shift of κ. This means that the delay y comprises a constant component associated with κ and a random component x exhibiting a exponential distribution with a parameter θ. Thus, $f(y)$ can be written as

$$f(y) = \begin{cases} \theta e^{-\theta(y-\kappa)} & : \quad y \geq \kappa, \\ 0 & : \quad \text{otherwise.} \end{cases} \tag{7}$$

We attribute the existence of κ to the prospective backlog of previously requested data units from p_{n_k} that has not been received yet by p and in addition to the required amount of time to empty out data unit l_{m_j} itself from the transmission buffer of node p_{n_k}. Furthermore, we relate the random component of the delay x to transient bandwidth variations of the network links comprising the channel $p_{n_k} \rightarrow p$ which in turn are caused by random occurrences of cross traffic on these links. The requesting peer estimates ϵ_B based on gaps in sequence numbers of arriving data units from p_{n_k} and similarly it estimates the parameter θ based on the jitter of the inter-arrival times of these data units. Finally, let $\tilde{r}_{(p_{n_k},p)}$ denote the present estimate of the download rate from node p_{n_k} that peer p has** and let $\mathcal{DU}$ denote the set of data units previously requested from p_{n_k} that has not been received yet. Then, p computes κ as

$$\kappa = \frac{\sum_{l \in \mathcal{DU}} B_l + B_{l_{m_j}}}{\tilde{r}_{(p_{n_k},p)}} \tag{8}$$

Once the peer has values for κ, θ, and ϵ_B, it can compute $Prob\{RTT_{(p,p_{n_k})} < t_{d,l_{m_j}}\}$ using (6) and (7) as

$$Prob\{RTT_{(p,p_{n_k})} < t_{d,l_{m_j}}\} = (1 - \epsilon_B) \int_{\kappa}^{t_{d,l_{m_j}}} \theta \, e^{-\theta(y-\kappa)} dy \; . \tag{9}$$

3. NEIGHBOURHOOD CONSTRUCTION AND RESOURCE SHARING

In our system, each peer selects a random subset of other peers in the network as its neighbours and use them for data exchange. Addresses of peers in the network are obtained from a tracking server with which each peer registers upon joining the network. In order to deal with the dynamics of P2P systems, the neighbourhood has to be redefined continuously. Each peer periodically discards the neighbour with the smallest rate contribution, and selects at random a new peer to be included in its neighbourhood. Then, an efficient resource sharing strategy is proposed in order to fight against free-riders.

3.1 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. For example, let $\mathcal{DU}^{(p_k)}$ represents the set of data units that peer p has received from its neighbour p_k within the last download rate estimation period T. Then, p computes the the received rate contribution from p_k as

$$\tilde{r}_{(p_k,p)} = \frac{\sum_{l \in \mathcal{DU}^{(p_k)}} B_l}{T} \, . \tag{10}$$

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 at random. 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.

**In the next section, we explain how p computes $\tilde{r}_{(p_{n_k},p)}$ periodically.

3.2 Sender Upload Bandwidth Sharing

The algorithm for sharing the upload bandwidth of a peer among its requesting neighbours operates as follows. Let $C^{(u)}$ be the upload bandwidth of peer p, and let PR denote the subset of neighbours from which p has pending requests at present. Then, to every node $p_k \in PR$, peer p allocates a share of its upload bandwidth computed as

$$r_{(p,p_k)} = \frac{\tilde{r}_{(p_k,p)}}{\sum_{p_k \in PR} \tilde{r}_{(p_k,p)}} \cdot C^{(u)}, \tag{11}$$

where $\tilde{r}_{p_k,p}$ denotes the present estimate of the sending rate from node p_k to peer p. Hence, nodes that contribute more of their sending rate to peer p will receive in return a larger share of its own upload bandwidth, as provided through Equation (11).

4. SIMULATIONS

4.1 Setup

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.[12] 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 uplink data rate of free-riders is irrelevant for the investigation here. In the simulations, we measure performance as the average Y-PSNR (dB) of the reconstructed video frames at each peer. 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 2.2 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. Initially each peer selects 8 other peers at random as its neighbours. The size of a neighbourhood for a peer can grow subsequently to contain up to 14 other peers.

4.2 Influence of Urgency and Popularity Factors

In this section, we examine the influence of the urgency factor $U(t, t_{d,l_m})$ and the popularity factor $P_{l_m}(k, |P|)$ on the overall performance of our system. As introduced in Section 2.2.2, these quantities convey respectively the relative temporal and the relative spatial importance of a data unit. That is, how soon the data unit is due to expire and how often the data unit is encountered in a neighbourhood. In the following, we describe our specific choices for $U(t, t_{d,l_m})$ and $P_{l_m}(k, |P|)$.

We model these two factors as simple polynomials composed of a single term that satisfy the functional requirements on $U(t, t_{d,l_m})$ and $P_{l_m}(k, |P|)$ described in Section 2.2.2. Specifically, for data unit l_m they are computed at time t as

$$\begin{aligned} U(t, t_{d,l_m}) &= \left(\frac{t}{t_{d,l_m}}\right)^{\alpha}, \\ P_{l_m}(k, |P|) &= \left(\frac{|P|}{k}\right)^{\beta} \end{aligned} \tag{12}$$

where the parameters $\alpha, \beta \geq 0$ are the powers of the polynomials for $U(t, t_{d,l_m})$ and $P_{l_m}(k, |P|)$, respectively. Using the formulation in (13) for these factors allows for a simple implementation that at the same time provides a lot of flexibility in terms of the range of values that can be covered by $U(t, t_{d,l_m})$ and $P_{l_m}(k, |P|)$ as a function of the parameters α and β. In Figure 3 below, we illustrate the forms that these polynomials can attain as a function of the power parameter. Specifically, we select $\alpha(\beta) \in \{1/4, 1/2, 1, 2, 4\}$ and compute the corresponding functions for $U(t, t_{d,l_m})$ (left) and $P_{l_m}(k, |P|)$ (right) in Figure 3. For ease of presentation, all polynomials in the case of each factor attain the same maximum value that is additionally normalized to one in Figure 3. In brief, it can be seen from Figure 3 that $U(t, t_{d,l_m})$ (left) and $P_{l_m}(k, |P|)$ can indeed place a great degree of relative importance between different data units depending on their respective power parameters α and β, and arguments $(t/t_{d,l_m})$ and $(k/|P|)$.

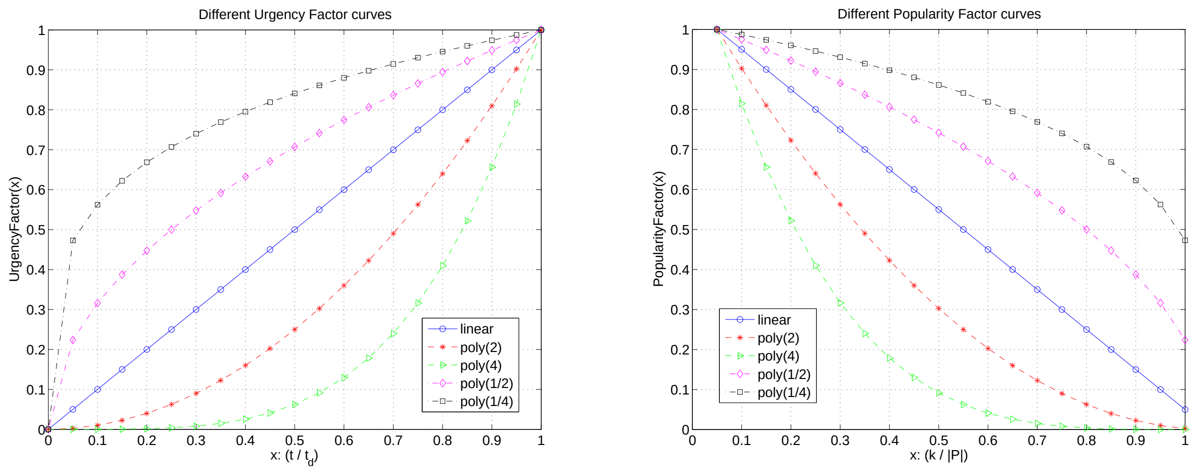

Figure 3. Different curves for $U(t, t_{d,l_m})$ (left) and $P_{l_m}(k, |P|)$ (right).

Next, we consider how the specific forms of the polynomials for the urgency factor and the popularity factor, i.e., their respective polynomial power parameters, influence the quality of the reconstructed media presentation at each peer. In particular, we conduct experiments where we vary α and β in the range [0,2] and measure the corresponding average video quality and its standard deviation for the populations of cable/dsl peers and ethernet peers, respectively.

In Figure 4, we show these quantities in the case of cable/dsl peers. In particular, in Figure 4 (left) we show the gain in dB of the average video quality (Y-PSNR) relative to the case when the urgency and the popularity factors are not employed, i.e., $\alpha, \beta = 0$. It can be seen that the gain can reach as high as 1.3 dB when the power parameters are in the range [1,1.5]. As the range of variations for the Y-PSNR gain in this range ($\alpha \in [1, 1.5], \beta \in [1, 1.5]$) is quite small, to maximize performance the values for these parameters can be selected to correspond anywhere in this plateau. Hence, for ease of implementation we opted to select the same value for both of them, and that is one, in the rest of the experiments in this paper. Finally, we can see from Figure 4 (right) that the optimum range for α and β in the case of Y-PSNR gain also corresponds to the biggest reduction in standard deviation of this quantity relative to the case when $U(t, t_{d,l_m})$ and $P_{l_m}(k, |P|)$ are not employed. Specifically, for $\alpha \in [1, 1.5], \beta \in [1, 1.5]$ we observe a plateau of maximum standard deviation reduction of 25% in Figure 4 (right).

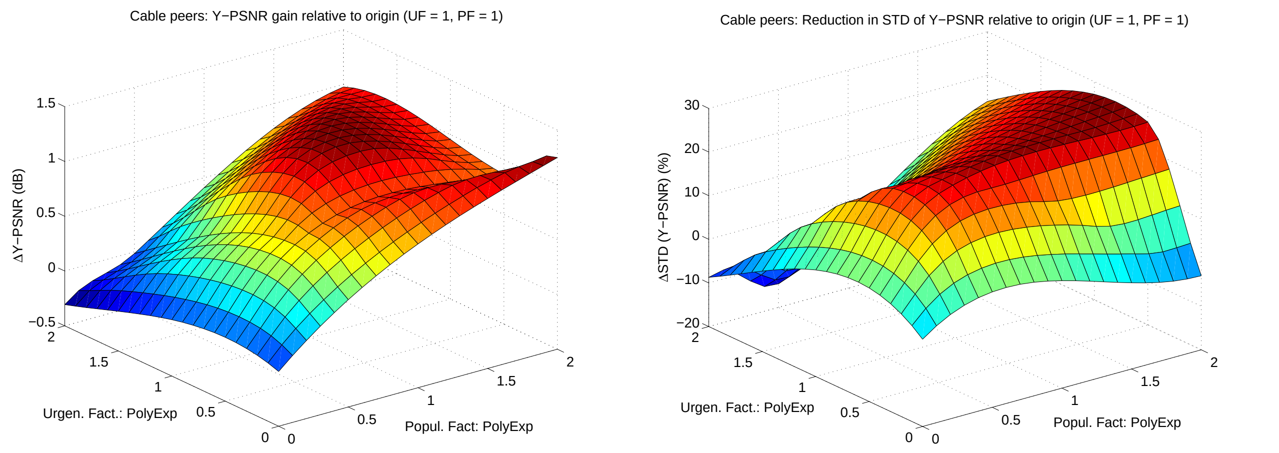

Figure 4. Gain in Y-PSNR (dB) (left) and reduction (in %) of the standard deviation of Y-PSNR (right) when using the popularity and urgency factors in the case of cable peers.

4.3 Utility-based scheduling

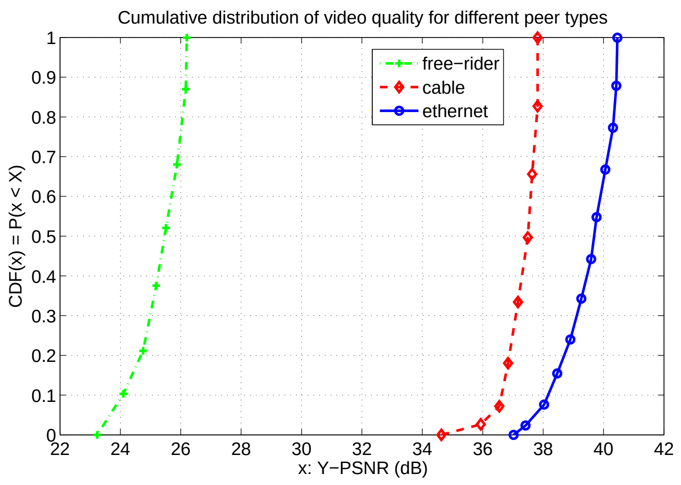

Figure 5. Cumulative distribution function of average video quality (Y-PSNR) for different peer types.

In Figure 5, we show the cumulative distribution function of the average video quality for each peer type. It can be seen from the figure that free-riders experience the media presentation at a very low quality. This is actually desirable as these peers do not contribute their upload bandwidth resources to serving data to other peers in the network, as explained earlier. Hence, the degraded video quality that they receive may in fact contribute to them changing their bandwidth sharing policy when they connect to the network next time. On the other hand, cable peers and ethernet peers exhibit distributions of video quality that are quite narrow in range and steep in slope, and most importantly of much higher amplitude relative to that of free-riders, as also seen from Figure 5. Furthermore, the cumulative distributions of video quality for cable peers and ethernet peers are commensurate to their bandwidth capabilities, as ethernet peers can receive more video quality layers from their neighbours and correspondingly serve more layers to them in return. Hence, ethernet peers exhibit video quality that is on the average 2 dB higher than that for cable peers. In particular, the average video quality for most of the cable peers ranges between 37 dB and 38 dB, while the average video quality for most of the ethernet peers is in the range 39 - 40.5 dB, as shown in Figure 5.

Next, we briefly go over the cumulative distribution of video decode rate for the different peer types. This is

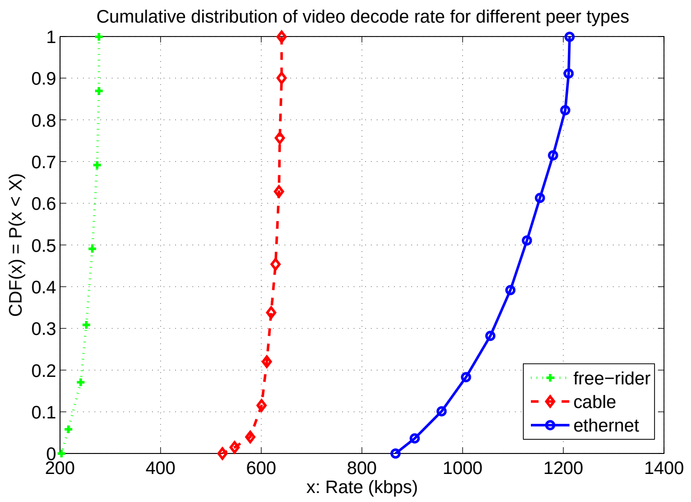

Figure 6. Cumulative distribution function of average video decode rate (kbps) for different peer types.

the amount of data received by a peer that the peer can actually use toward reconstructing the media presentation at its end. In other words, a peer may decode only a part of its received data, as duplicate packets constitute an unnecessary redundancy. It can be seen from Figure 6 that ethernet peers exhibit much higher decode rates than the cable peers do, which is expected and is due to the different bandwidth capabilities for these two peers types, as explained earlier. Furthermore, both cable and ethernet peers receive substantially larger amounts of useful video data relative to free-riders, as observed from Figure 6. This is desirable, as we would like ethernet peers and cable peers to spend as much as possible of their bandwidth resources between them, i.e., to share as little as possible of them with the non-contributing free-riders, as discussed previously. Finally, note that the results on video decode rate from Figure 6 correspond to those on average video quality from Figure 5.

4.4 Robustness to free-riders

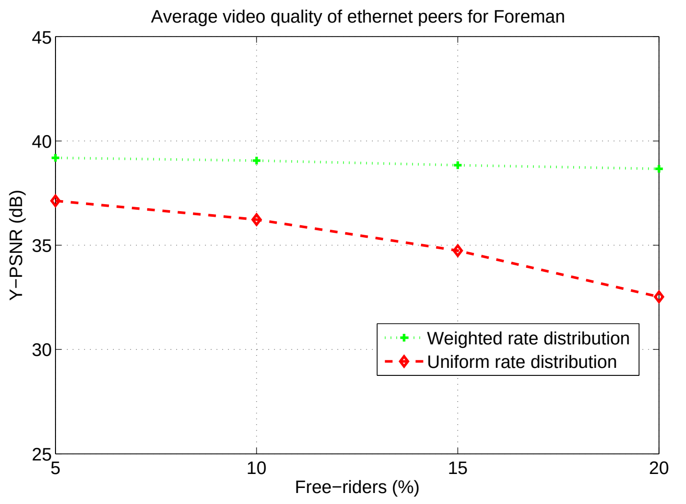

Figure 7. Influence of free-riders with and without proportional upload bandwidth sharing.

Finally, to examine the resilience of the proposed framework to the influence of free-riders, we conducted the following experiment. We increase the percentage of free-riders in the overall population to 10, 15 and 20 percent, and we measured the corresponding average video quality for the three peer types. In Figure 7, we show these results for ethernet peers, together with the corresponding performance for ethernet peers in the case when sending peers share their upload resources uniformly. In other words, in this latter case, peers send data to their neighbours at same outgoing data rates. It can be seen from Figure 7 that when our framework is employed, the average performance of the video presentation for the ethernet peers does not vary substantially, as the number of free-riders in the network is increased. However, in the case of uniform send-rate distribution we can see that the average video quality of the ethernet peer population degrades substantially, as more and more resources in the network are consumed by the non-responding free-rider peers. For example, even at 20% free-riders in the network the reduction in average video quality for ethernet peers does not exceed 0.1 - 0.2 dB under the weighted send-rate allocation of our framework, while it reaches around 6 dB in the case of uniform allocation, as evident from Figure 7. Note that similar observations can be drawn when comparing the corresponding results for the cable peer type. These results are not included here for space considerations.

5. CONCLUSIONS

We have proposed a mesh-pull based P2P streaming framework. The framework comprises an utility-based 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. The algorithm works in conjunction with a technique for sharing the upload bandwidth of a sending peer that effectively marginalizes the influence of free-riding in the system. Through simulations, we demonstrate that these two algorithms permit to increase the performance of a mesh-pull P2P streaming system. Average quality and average decoding bandwidth increases with respect to the ones of baseline P2P streaming systems, and free-riders are effectively shut down. In our future work, we will consider designing an algorithm for creating and maintaining a peer neighbourhood that contributes to further improvements in performance of the system over previous approaches for mesh-pull based streaming.

REFERENCES

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

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

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

[4] Padmanabhan, V., Wang, H., and Chou, P., "Resilient peer-to-peer streaming," in [*Proc. Int'l Conf. on Network Protocols*], 16–17, IEEE (Nov. 2003).

[5] Hei, X., Liu, Y., and Ross, K., "IPTV over P2P streaming networks: the mesh-pull approach," *IEEE Communications Magazine* **46**, 86–92 (Feb. 2008).

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

[7] Hei, X., Liu, Y., and Ross, K., "Inferring network-wide quality in p2p live streaming systems," *IEEE J. Selected Areas in Communications* **25**, 1640–1654 (Dec. 2007).

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

[9] Chou, P. A. and Miao, Z., "Rate-distortion optimized streaming of packetized media," Tech. Rep. MSR-TR-2001-35, Microsoft Research, Redmond, WA (Feb. 2001).

[10] Chakareski, J. and Girod, B., "Rate-distortion optimized packet scheduling and routing for media streaming with path diversity," in [*Proc. Data Compression Conference*], 203–212, IEEE Computer Society (Mar. 2003).

[11] Chakareski, J. and Girod, B., "Server diversity in rate-distortion optimized streaming of multimedia," in [*Proc. Int'l Conf. Image Processing*], **3**, 645–648, IEEE (Sept. 2003).

[12] 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).