SCHOOL OF ENGINEERING - STI
ELECTRICAL ENGINEERING INSTITUTE
SIGNAL PROCESSING LABORATORY

Jacob Chakareski

EPFL - FSTI - IEL - LTS
Station 11
Switzerland-1015 LAUSANNE

Phone: *+4121 6934709*
Fax: *+4121 6937600*
e-mail: `{jakov.cakareski}@epfl.ch`

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

P2P VIDEO BROADCAST WITH LATENCY AND UTILITY OPTIMIZATION

Jacob Chakareski and Pascal Frossard

Ecole Polytechnique Fédérale de Lausanne (EPFL)
Signal Processing Laboratory

Technical Report EPFL-LTS-2009-010

October 19, 2009

Part of this work has been submitted to IEEE Transactions on Networking.

P2P video broadcast with latency and utility optimization

Jacob Chakareski and Pascal Frossard
Signal Processing Laboratory - LTS4
Ecole Polytechnique Fédérale de Lausanne (EPFL)
1015 Lausanne, Switzerland

***Abstract*—We consider the problem of mesh-pull based video broadcast in peer-to-peer networks. We propose a novel algorithm for constructing the distribution overlay, where peers are arranged in neighborhoods that exhibit similar latency values from the origin media server. We analyze the properties of the resulting mesh and show that it increases data sharing between neighbors, hence improves the system performance compared to random constructions. The nodes are further equipped with a novel streaming strategy that is built on utility-based packet scheduling and proportional resource sharing in order to fight against free-riders. The utility is driven by both the packet importance for the video quality the packet popularity within the peer neighborhood. Our simulation results show that the proposed protocols increase the performance of a mesh-pull P2P broadcast system. Significant improvements are registered relative to existing solutions in terms of average quality and average decoding rate due to the packet scheduling and the mesh construction algorithm. The latter provides further gains in performance in terms of frame-freeze and playback latency relative to a conventional approach where peer neighbors are selected at random. Corresponding gains in video quality are registered due to the improved continuity of the playback experience.**

I. Introduction

With the rapid developments of P2P networks, overlay video streaming has been gaining a lot of interest and even becomes a common place that is 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, P2P video streaming applications have been successfully deployed and tested for broadcasting or multicasting pre-encoded content or live events to large audiences in the Internet. Systems like PPLive [1], PPStream [2], and Coolstreaming [3] are among the most popular solutions for P2P streaming.

Still, the present P2P multimedia experience is marred by uncontrollable start-up delays, frequent freezes of the multimedia playback, and significant fluctuations of audio and video quality. Among the reasons for these apparent shortcomings we count the following. First, the design of the existing P2P streaming applications have been mainly carried over from earlier P2P file sharing applications. As such they are ill equipped to deal with the specificities of multimedia data, such as delivery deadlines and unequal importance of packets for the video quality. In particular, the construction of the overlay delivery network and the data exchange mechanisms employed by the peers can contribute to poor P2P multimedia

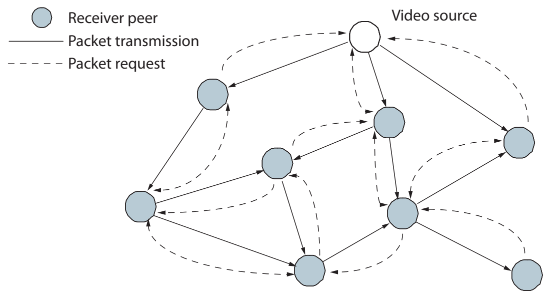


Fig. 1. Illustration of a mesh-pull based P2P broadcast system.

performance if they delay and latency constraints. Then, 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 where the available system bandwidth should increase 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, 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 comprehensive streaming framework that attempts to address the above issues in mesh pull-based P2P systems (see Fig. 1). Peers typically requests video packets from their neighbors in order to form the video stream that is passed to the decoder. At the same time, they try to satisfy the requests of other peers and forward requested packets as long as the bandwidth constraints permit it. We first design a mesh construction procedure that builds neighborhoods with peers that exhibit similar delivery delays relative to the broadcast media server. This increases the likelihood of data availability among neighbors, thereby reducing the playback latency and the frame freeze frequency of the media presentation at the peers. In addition, we design a receiver-driven algorithm for requesting media packets from neighboring peers that prioritizes packets based on their utility. The utility of a packet is based on the packet's delivery deadline and its importance for the reconstruction quality of the media presentation. The utility also depends on the popularity of a packet within the neighborhood so that the delivery of less frequently encountered data units is facilitated. A peer equipped with our algorithm can thus request the media packets that maximize the performance of its media presentation. This simultaneously contributes toward the same goal at neighboring peers. Therefore, a globally optimized performance of the presentation over the whole peer population is achieved. Finally, we design a bandwidth sharing procedure that targets robustness against free-riders. In particular, a sending peers distributes its upload bandwidth among its requesting neighbors 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 neighbors, as its rate contribution to them would typically be non-existing. We demonstrate through simulation that the proposed framework provides an efficient solution for video broadcast in P2P systems. It substantially outperforms existing mesh-pull based algorithms over several performance indices:

1) significant reductions in frame freeze and playback latency due to novel overlay organization
2) increased average video quality and useful decoding bandwidth thanks to utility-based packet scheduling
3) improved resiliency to free-riders due to proportional upload bandwidth sharing.

The rest of the paper is structured as follows. First, in Section II, we describe the proposed mesh construction procedure and analyze in details the properties of the resulting overlay network. Next, we describe the utility-based packet scheduling framework in Section III. Then, we describe the organization of the P2P streaming system in Section IV. Specific performance aspects of the whole system are examined in Section V, followed by a discussion of related work in Section VI.

II. Latency-based neighborhood construction

A. Motivation

The organization of the mesh that connects peers has a direct influence on the performance of the P2P video streaming system. In particular, an efficient construction should favor data exchange between peers that are direct neighbors in the mesh. Such packet exchanges typically happen when neighbor peers are synchronized or experience a similar latency to the streaming server. We propose in this section to organize the peers in neighborhoods that gather peers based on the latency they experience in order to make best use of the bandwidth resources.

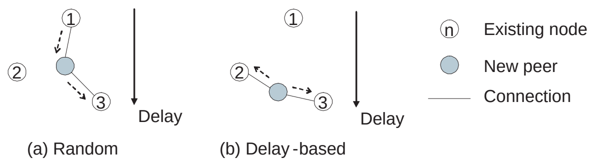


Fig. 2. Illustration of two possible configurations when a new node joins the P2P overlay.

As a motivation for such a mesh construction, we consider the following example shown in Figure 2. It illustrates the scenario of a new node joining an existing overlay of peers. The new node is represented as a full circle in Figure 2, while three of the peers already present in the overlay are denoted with white circles marked with numbers. The relative vertical position of the existing nodes in the overlay represents their respective latencies relative to the origin media server. Specifically, going top to bottom in the figure the delay in receiving original media packets becomes larger at each subsequent node.

Now, assume that in one case the new peer selects Node 1 and Node 3 as its neigbours. Then, the delay configuration between these three nodes may be as the one illustrated in Figure 2(a). Specifically, the new node exhibits longer latency relative to Node 1, which is expected, as it is connected to the media server through this node. For the same reason, the new node may exhibit shorter latency relative to Node 3. Hence, such a configuration typically creates a unidirectional flow of media packets from Node 1 through the new node to Node 3 in the end.

On the other hand, if the new peer selects Node 2 and Node 3 as its neighbor then the prospective latency disposition of the three nodes may be as the one shown in Figure 2(b). Here, Node 2 and Node 3 exhibit similar delays relative to the origin server and the newly joined node thus exhibits a similar latency too. Such a delay configuration may contribute to media packets being exchanged between Node 1 and Node 3 through the new peer in both directions, as illustrated in Figure 2(b). The Node 1 could further send packets to Node 2, Node 3, or the newly joined node, with a similar efficiency. In addition to contributing to a more balanced distribution of data flows in a neighborhood, the second configuration in Figure 2 provides a more consistent delivery of media packets in the advent of node departures. In particular, in the case examined in Figure 2(a) a sudden departure of Node 1 may disrupt (albeit temporarily) the timely delivery of media packets to the rest of the nodes in the neighborhood (in this case the new peer and Node 3). On the other hand, if Node 2 decides to leave the newly joined peer can continue to receive new media packets from Node 3 in a timely fashion due to the similar latencies that are involved in the scenario from Figure 2(b). This scenario is clearly more efficient for video streaming applications.

Overlay construction algorithms commonly select neighbors are random from an existing population of nodes. While the resulting overlay exhibits some appealing properties, such as strong network connectivity, it is likely to resemble the scenario described in Figure 2(a). Since random neighbor selection appears inappropriate for the context of data delivery with deadlines, we propose in this paper to select peer neighbors based on their delay characteristics relative to the media server, similarly to the scenario in Figure 2(b). We propose below an algorithm for delay-based neighborhood construction and we analyze the characteristics of the resulting overlays.

B. Neighborhood selection algorithm

We propose to organize peers into neighborhoods that feature similar delays from the media server for all their members. In other words, we aim to construct a neighborhood with K peers, such that each of these peers 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 that neighborhood exchanging packets with each other. Peers in a neighborhood are likely to be synchronised, as they request at the same time the packets that exhibit similar decoding timestamps. Increased data exchange within a neighourhood improves the performance of the whole system in terms of resource usage, delay and video quality.

Let assume an existing overlay formed by a set of nodes $P = \{p_1, \ldots, p_N\}$, with $N > 1$. Let $\Delta_P = \{\delta_1, \ldots, \delta_N\}$ be the corresponding latencies of these nodes relative to the origin server. Without loss of generality, we assume that the nodes are sorted according to increasing values of latency. A new node p can connect to the overlay via any node in P. Let $\Delta_{p \to P} = \{\delta_{p \to p_1}, \ldots, \delta_{p \to p_N}\}$ be the corresponding network delays between the new peer and the nodes in P. Finally, let $X = \{x_1, \ldots, x_N\}$ denote the relative latencies of the new peer to the server through the different nodes in P. Note that we have $x_i = \delta_{p \to p_i} + \delta_i$ or equivalently $X = \Delta_{p \to P} + \Delta_P$.

Recall that we want to favor data exchanges among nodes that exhibit similar latency. The new peer is therefore interested in finding the subset of peers $Z \subset P$ of size $K < N$ such that the distribution of latencies seen by the peer through these nodes relative to the origin server exhibits the smallest variance. In particular, let $Z = \{p_{l(1)}, \ldots, p_{l(K)}\}$ be a subset of peers from P, where $l(j)$ is a mapping function with $l(j) \in \{1, \ldots, N\}$ for $j = 1, \ldots, K$. Let $Y = \{y_1, \ldots, y_K\}$, with $y_j = \delta_{p \to p_{l(j)}} + \delta_{l(j)}$ be the relative latencies from the peer p to the server through the nodes in K. Then, let $\lambda^Z_{\min} = \min Y$ and $\lambda^Z_{\max} = \max Y$ denote respectively the minimum and maximum latencies that p sees to the server through the peers in Z. Finally, the spread of latencies characterizing the subset Z can be expressed as $|\lambda|^Z = \lambda^Z_{\max} - \lambda^Z_{\min}$. The subset Z^* that exhibits the smallest latency spread relative to the origin server is given by

$$Z^* = \arg\min_{Z \subset P} |\lambda|^Z. \quad (1)$$

The problem of Eq. (1) can be efficiently solved by performing the following three steps. First, we form subset of peers by sliding a window of size K on the set P. In other words, for $m = 1, \ldots, N-K+1$ we create the sets $Z_m \in P$ with the mapping function $l(j) = j+m-1$, for $j = 1, \ldots, K$. Then, for each set Z_m we compute the corresponding delay spread $|\delta|^{Z_m}$. Finally, we find the index m for which the latency spread is the smallest over all sets Z_m. Let this index be denoted m^*. The corresponding set Z_{m^*} represents the solution to Eq. (1). We summarize the algorithmic steps of the procedure described above for computing the solution of Eq. (1) in Algorithm 1. Note the in general, one may have to sort the peers according to latency values before running the Algorithm 1.

We show in Figure 3 an illustration of a mesh constructed using our procedure. Going radially from the center, where the media server is located, we can see that the latency from the server to the nodes increases. Still, nodes within a neighborhood, denoted as dashed clouds in Figure 3, 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.

Algorithm 1 Neighborhood selection

```
1:  **Initialization**: Set $p$, $P$, $X$
2:  **for** $m = 1, \ldots, N-K+1$ **do**
3:      **for** $j = 1, \ldots, K$ **do**
4:          $l(j) = j + m - 1$
5:          Construct set $Z_m \subset P$
6:          Compute latency spread $|\lambda|^{Z_m}$
7:      **end for**
8:  **end for**
9:  Find $m^* = \arg\min_m |\lambda|^{Z_m}$
10: Return: neighborhood of $p$ is $Z_{m^*}$
```

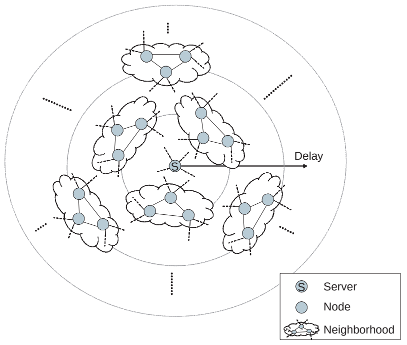


Fig. 3. An example of a minimum delay mesh construction. For the sake of clarity, the figure does not include every connection between the nodes in the network.

We analyze below the characteristics of the resulting overlay, both from statistical and graph theory perspectives.

C. Increased Likelihood of Data Sharing

We provide here a statistical characterization of the proposed neighborhood selection algorithm. Specifically, we analyze the latency distribution that a new peer experiences through its selected neighbors. We study the statistical properties of this distribution and we compare it to the corresponding distribution for the case when neighbors are selected at random. In summary, we show that the former distribution has a tighter/narrower support and features smaller mean and variance values related to the random selection case. This contributes to increased likelihood of data sharing among neighbors.

We look at the distribution of the latencies X that peers experience while receiving media packets from the server. We consider that this distribution is bounded, i.e., it exhibits a lower and an upper limit on the values it can produce. Specifically, the lower bound x_{min} can be related to the data rate at which packets are sent to the direct neighbors of the server. The upper bound x_{max} in turn can be related to the maximum distance or depth (in terms of number of hops) between the origin server and a node on the ”periphery” of the overlay. We further consider that X represents a sample set of realizations of N random variables X_i's that are independently and identically distributed over the range described by the two bounds, i.e., $[x_{min}, x_{max}]$. Sorting X along the increasing values of the delay then corresponds to creating an order statistics of the original N random variables $X_1, X_2, \ldots, X_N$ denoted as $X_{(1)}, X_{(2)}, \ldots, X_{(N)}$ [5]. Given the formalism of order statistics, the selection of a neighborhood Z_m as defined in Algorithm 1 corresponds to computing the realization of the random variable W_m defined as

$$W_m = X_{(n)} - X_{(m)}, \tag{2}$$

where the random variables $X_{(m)}$ and $X_{(n)}$ have realizations $x_{(m)}$ and $x_{(n)}$ that corresponds to the respective latency values for the first and last entries (member peers) in the set Z_m, as selected from the sorted array of delays X. Note that, when the size of the set Z_m is set to K, the difference between the indices m and n is always equal to $K-1$.

Next, let $f(x)$ and $F(x)$ denote respectively the probability density function (pdf) and the cumulative distribution function (cdf) for the random variables X_i, $i = 1, \ldots, N$, with respective realizations in X, as explained earlier. Then, it can be shown [5] that the m-th order statistics $X_{(m)}$ is characterized with the following pdf:

$$\begin{aligned} f_m(x) &= \frac{N!}{(m-1)!(N-m)!}\, F^{m-1}(x)\,(1-F(x))^{N-m}\, f(x) \\ &= \frac{1}{B(m, N-m+1)}\, F^{m-1}\,(1-F(x))^{N-m}\, f(x) \end{aligned} \tag{3}$$

Furthermore, given Eq. (3), we can derive the pdf for the random variable introduced in Eq. (2). Specifically, first it can be shown that the joint pdf $f_{mn}(x,y)$ of two random variables $X_{(m)}, X_{(n)}$, for $1 \le m < s \le N$ and $x \le y$ can be written as [5]

$$f_{mn}(x,y) = C_m\, F^{m-1}(x)\, f(x)\, (F(y){-}F(x))^{K-2}\, f(y)\, (1{-}F(y))^{N-m} \tag{4}$$

where we have used the fact that $n-m = K-1$. The constant term C_m is given as

$$C_m = \frac{N!}{(m-1)!(K-2)!(N-y)!}\,.$$

Then, using Eq. (4) and a transformation of variables $(x,y) \rightarrow (x, w_m)$[1] where $w_m = y - x$, we can obtain the pdf of W_m, $g(w_m)$, by integrating out the pdf $f_{mn}(x, w_m)$ over x, i.e.,

$$g(w_m) = C_m \int_{-\infty}^{\infty} f_{mn}(x, w_m)\, dx. \tag{5}$$

Recall that the proposed overlay construction technique computes the realizations of the variables W_m, for every set Z_m, with $m = 1, \ldots, N-K+1$. Therefore, it produces an array of values corresponding to these realizations, i.e., $(w_1, \ldots, w_{N-K+1})$. The algorithm then selects the smallest of these values whose corresponding index m^* determines the neighborhood of peers Z_{m^*} that the new node will join. This procedure corresponds in fact to finding the order statistics for the variables W_m, i.e., $W_{m(1)}, \ldots, W_{m(N-K+1)}$ and selecting the neighborhood Z_m that corresponds to the first order statistics $W_{m(1)}$. Finally, the cumulative distribution function of the first order statistics $W_{m(1)}$ can be computed from Eq.(3) by using $m = 1$ and $G(w_m) = \int g(w_m)$.

[1]The Jacobian of this transformation has a unity modulus.

We analyze now the properties of the proposed overlay construction when the latencies are uniformly distributed. In this case, the distribution of the latency spreads $W_{m(1)}$ and W_m can be computed analytically, as given in Appendix A. These latency spreads correspond to the outcome of the overlay construction proposed in Section II-B, where the subset of neighbors Z_m is selected optimally or respectively by picking randomly any index $m \in \{1, \ldots, N-K+1\}$. We compare these statistics to the statistics of random overlay construction where K peers are selected at random from the set of peers P. The latter statistics are computed empirically since there is no closed form expressions for the mean and standard deviation when the overlay is constructed randomly.

	μ (%)	σ (%)
Random	152.64	45.50
W_m	13.61	8.77
$W_{m(1)}$	4.90	2.14

(a) Relative to a Uniform random variable

	μ (%)	σ (%)
W_m	8.91	19.28
$W_{m(1)}$	3.21	4.70

(b) Relative to a Random overlay construction

TABLE I
MEAN (μ) AND STANDARD DEVIATION (σ) FOR DIFFERENT NEIGHBOR SELECTION METHODS IN PERCENT OF (μ_x, σ_x) FOR A CHOSEN REFERENCE. NEIGHBORHOOD SIZE $K = 8$.

We compare the distribution of the latency spread in a neighborhood for different overlay construction methods to the values of μ_x and σ_x that represent respectively the mean and standard deviation of the uniformly distributed latency samples in X (see Table Ia). It can be observed that selecting neighbors at random contributes to a 52% higher mean value for the latency spread of the neighborhood relative to μ_x. On the other hand, the standard deviation of the latency spread is approximately 55% smaller than σ_x. Furthermore, it is encouraging that the latter two approaches create neighborhoods characterized with latency spread statistics that are much smaller than those for the origin delay distribution. Specifically, W_m exhibits mean and standard deviation that represent fractions of 14% and 9% only, relative to μ_x and σ_x, respectively. Further improvement is provided by $W_{m(1)}$ that is characterized with μ and σ representing very small portions of 5% and 2% of the corresponding statistics for the original distribution, as seen from Table Ia.

Next, in Table Ib we examine the relative values of μ and σ for W_m and $W_{m(1)}$ compared to the respective statistics in the random node selection algorithm (denoted Random in Table I). It can be seen that both methods significantly improve performance in terms of latency deviation within a neighborhood relative to Random. In particular, (μ, σ) for W_m are only 9% and 19% fractions, respectively, of the corresponding quantities for Random. In the case of $W_{m(1)}$ an even more significant reduction of the magnitude of the statistics of the latency spread is registered, as seen from Table Ib. The sizes of (μ, σ) for $W_{m(1)}$ represent 3% and 5% fractions of (μ, σ) for Random, respectively.

Finally, in Figure 4 we illustrate the cumulative distribution functions of the latency spread in the case of Random, W_m, and $W_{m(1)}$. As corroborated by the previous results from Table I, we can see from Figure 4 that the support of the cdf

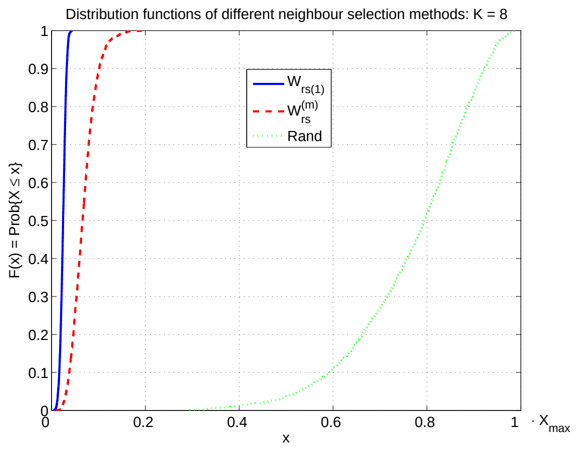


Fig. 4. Cumulative distribution function of different neighbor selection methods for $K = 8$.

and the mean and standard deviation values for $W_{m(1)}$ are the smallest, while those for Random are respectively the largest. This confirms that the latency-based mesh organization results in small latency difference between neighbors, hence increased likelihood of data exchange. Finally, we note that selecting the neighborhood by picking the index m randomly in the Algorithm 1 represents an interesting compromise between computational complexity[2] and performance. Although W_m performs relatively worse relative to $W_{m(1)}$, it still provides a substantial improvement in latency divergence performance relative to Random, as shown in Figure 4.

D. Small World Property

We show in this section that overlay networks constructed using the proposed procedure exhibit small world properties [4]. Small world networks are represented by a class of graphs that exhibit density of edges that is much higher than that of random graphs. This means that such networks feature clusters of highly interconnected nodes and a few connections between nodes of different clusters. The benefit of such a node configuration in the network is that once a packet reaches a cluster it can be readily obtained from the cluster members. Furthermore, the smaller number of connections between the dense clusters ensures short overall hop counts between different nodes, which in part can control the end-to-end latency.

We estimate the clustering coefficient in the proposed overlay construction algorithm. The clustering coefficient η describes the density of edges in the neighborhood of a peer, which is the ratio between the actual number of edges and the maximum number of edges in the neighborhood of the peer. We approximate the total population of peers in the overlay as belonging to two classes, $\mathcal{C}_1$ and $\mathcal{C}_2$. The first one comprises nodes that are mostly embedded in local dense clusters. And the second class features nodes that mostly serve as conduits, i.e., interconnects between different clusters. We then compute a statistical lower bound for η as the average of the clustering coefficients for each of the two classes. Finally, we show experimentally that the actual clustering coefficient established in simulations follows the computed bound closely.

[2]We only need to maintain an ordered list of peers in the network according to their respective latencies from the origin media server.

Let N_1 and N_2 denote respectively the minimum and the maximum sizes of a neighborhood in the overlay. These correspond respectively to the initial number of neighbors that a node can be associated with in the network and to the maximum number of neighbors that the node can possibly accept during his association with the overlay. In steady state, we model each peer in the class $\mathcal{C}_1$ as featuring dense local clustering of edges among N_1 of its neighbors, while the remaining $N_2 - N_1$ neighbors correspond to connections to other dense clusters. The remaining $N_2 - N_1$ neighbors that also feature a densely clustered set of nodes, though rather smaller than the former one. Therefore, the clustering coefficient η_1 for a node in $\mathcal{C}_1$ should be at least as large as

$$\eta_1 \geq \frac{\binom{N_1}{2} + \binom{N_2-N_1}{2}}{\binom{N_2}{2}} . \tag{6}$$

For nodes in the second class $\mathcal{C}_2$, it holds that they can be connected to as many as $N_2/(N_2 - N_1)$ different small dense cluster conduits featuring $N_2 - N_1$ neighbors each. Hence, a lower bound on the clustering coefficient η_2 for nodes in the second class can be computed in this case as

$$\eta_2 \geq \frac{\frac{N_2}{N2-N_1} \cdot \binom{N_2-N_1}{2}}{\binom{N_2}{2}} . \tag{7}$$

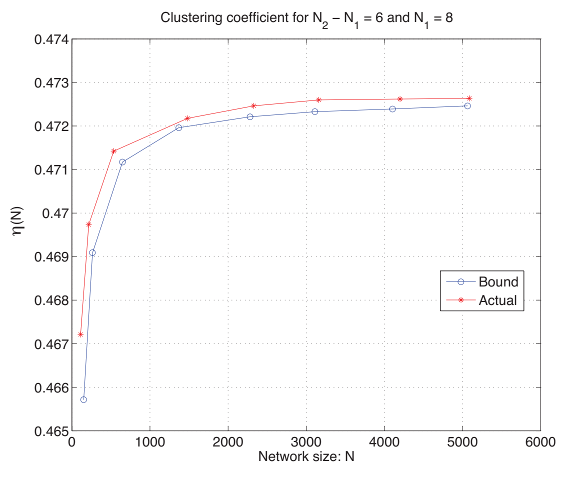
Finally, we estimate the overall clustering coefficient for the entire network as the statistical average of lower bounds for the clustering coefficients for the nodes in the two classes, given in (6) and (7). That is,

$$\eta = \gamma \cdot \eta_1 + (1 - \gamma) \cdot \eta_2 , \tag{8}$$

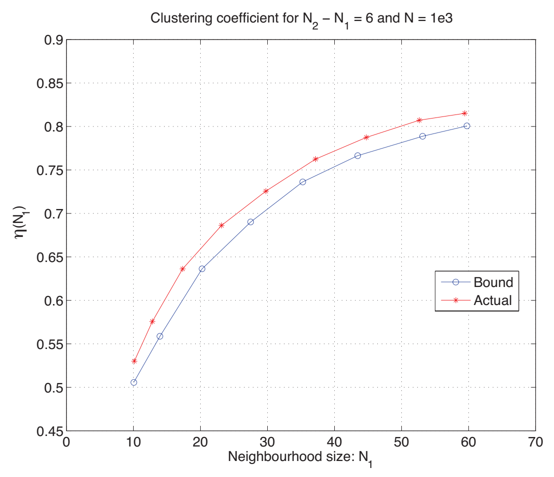
where $\gamma = \frac{N_1 \cdot \lfloor N/N_1 \rfloor}{N}$ represents the fraction of nodes of classes $\mathcal{C}_1$ in the overlay.

In Figure 5 below, we examine the bound for η from Eq. (8) computed for different N and N_1 values together with the actual values for the clustering coefficient determined in experiments for the same network and neighborhood sizes. In particular, in Figure 5a we show the bound and the actual value for η as a function of the network size N for $N_1 = 8$ and $N_2 = N_1 + 6$. It can be seen the actual value of η follows closely our estimate, which justifies the modeling employed in our analysis for computing the bound in Eq. (8). Similar behaviour can be observed from Figure 5b where we compute the bound and the actual value for η as a function of N_1, while keeping the network size at $N = 10^3$ and $N_2 \leq N_1 + 6$. It can be seen from Figure 5b that again the actual value of η follows closely the estimate.

We conclude the section by comparing the values of η computed for an overlay constructed using the proposed approach and for a random graph of the same vertex size. Specifically, we set the number of peers in the network N to 10^3 and the initial and maximum neighborhoood size for a node to $N_1 = 8$ and $N_2 = 14$ peers. The corresponding clustering coefficient for this parameter configuration was computed to be 0.472.



(a) $\eta(N)$



(b) $\eta(N_1)$

Fig. 5. Clustering coefficient η as a function of (a) Network size N, for $N_1 = 8$, and $N_2 = 14$, and (b) neighborhood size N_1, for $N_2 = N_1 + 6$ and $N = 10^3$.

The corresponding value for η in the case of the random graph is only 0.0122. The significantly larger clustering coefficient in the former case confirms that the proposed delay-based technique creates overlays that indeed belong to the class of small-world networks.

III. Utility-based packet scheduling

A. Receiver-driven packet scheduling

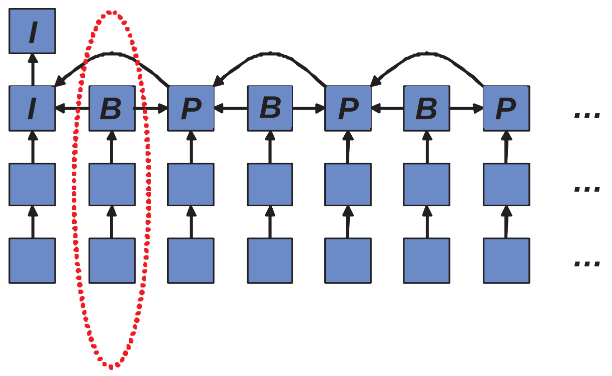


Fig. 6. Dependency graph for data units of a video presentation.

The delay-based overlay construction algorithm proposed above increases data sharing among neighbor peers. However, it is known that data packets in multimedia streams have very heterogeneous characteristics in terms of importance and timing information [7]. It becomes therefore primordial to carefully prioritize packets so that data exchanges in a neighborhood maximizes the video quality at all the peers. We propose a utility-based receiver-driven packet scheduling algorithm, where the utility of a packet is driven by its rate-distortion characteristics and its popularity within the neighborhood.

First, we consider that the video stream 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 [7]. , as illustrated in Figure 6. 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: (i) B_l, the size of data unit l in bytes and (ii) $t_{d,l}$ is its *delivery deadline*. This is the time by which data unit l needs to be received in order to be usefully decoded[3]. Packets arriving after the delivery deadlines of the respective data units that they contain are discarded. Furthermore, $\mathcal{N}_c^{(l)} = \{1, \dots, l\}$ is the set of data units that the receiver considers for error 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 [8], which in turn represents a generalized version (accounting for decoder error concealment) of the model originally introduced in [7].

Then, each peer maintains a sliding window S of packets that are useful to him at a given point in time. The window typically comprises the packets with the video information that has to be played back in the near future. The peers buffer the already received data units from this window, while they seek to request the rest of them from its neighbors. Let $M \subset S$ denote the set of missing data units at peer p. The 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, the peer p is interested in computing the optimal schedule for requesting data in M from its neighbors, such that the reconstruction quality of its media presentation is

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

maximized. Typically, the peer p tries to receive the most important packets first. We formalize below the problem for rate-distortion optimal packet scheduling. We then propose a low-complexity solution based on the notion of packet utility.

B. Packet scheduling optimization problem

Let $\boldsymbol{\pi}$ denote the collection of request schedules for the data units in M, i.e., $\boldsymbol{\pi} = \{\pi_1, \dots, \pi_{|M|}\}$, where π_{l_m} is the request schedule for data unit $l_m \in M$, for $m = 1, \dots, |M|$. Each request schedule π_m represents a matrix of size $|\mathcal{P}| \times N$, where $\mathcal{P}$ denotes the set of neighboring nodes of peer p. The row index n and the column index i of π_m correspond respectively to the neighbor $p_n \in \mathcal{P}$ from which data unit l_m can be requested at time slot t_{i-1}, for $n = 1, \dots, |\mathcal{P}|$ and $i = 1, \dots, N$. In particular, $t_0, t_1, \dots, t_{N-1}$ 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 neighbor 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 $n = 1, \dots, N$ 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 $i = 1, \dots, N$ for every neighbor 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 [9], the expected distortionof the media presentation at peer p as a function of the request schedule $\boldsymbol{\pi}$ can be expressed as

$$\begin{aligned} D(\boldsymbol{\pi}) \quad &= \quad D_0 - \sum_{l_m \in M} \sum_{l_1 \in \mathcal{N}_c^{(l_m)}} \Delta d_{l_m}^{(l_1)} \prod_{j \in \mathcal{A}(l_1)} (1 - \epsilon(\pi_j)) \times \\ & \prod_{l_2 \in \mathcal{C}(l_m, l_1)} \left(1 - \prod_{l_3 \in \mathcal{A}(l_2) \setminus \mathcal{A}(l_1)} (1 - \epsilon(\pi_{l_3})) \right), \end{aligned} \tag{9}$$

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, "\" denotes the operator "set difference". We refer the reader to [9] for details on the derivation of the expression in (9).

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 neighbors in $\mathcal{P}$ will send in response to $\boldsymbol{\pi}$. This quantity can be computed as

$$R(\boldsymbol{\pi}) = \sum_{l_m \in M} B_{l_m} \rho(\pi_{l_m}), \tag{10}$$

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 exceeded as a result. In other words, the peer p is interested in computing the optimal policy $\boldsymbol{\pi}^*$ given as

$$\boldsymbol{\pi}^* = \arg\min_{\boldsymbol{\pi}} D(\boldsymbol{\pi}), \qquad \text{s.t. } R(\boldsymbol{\pi}) \leq C^{(d)}. \tag{11}$$

Using the method of Lagrange multipliers, we can reformulate (11) 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 [9], 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) [7]. However, due to the complexity of such an approach, we propose instead an approximate yet efficient solution to the problem of Eq. (11), which is more suitable for being incorporated as a part of an actual system. We describe this low complexity approach in the next section.

C. Low complexity packet scheduling algorithm

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.,

$$S_{l_m} = \sum_{j \in \mathcal{D}(l_m)} \Delta d_j^{(j)}, \tag{12}$$

where $\mathcal{D}(l_m)$ is the set of descendants[4] of data unit l_m and $\Delta d_j^{(j)}$ is the reduction of reconstruction distortion associated with data unit j. Furthermore, we define I_{l_m} to be the current importance of data unit l_m for the overall quality of the reconstructed presentation. Using (12) we compute this quantity as

$$\mathcal{I}_{l_m} = \frac{S_{l_m}}{B_{l_m}} \cdot Q_{l_m}(k, |\mathcal{P}|) \cdot U(t, t_{d,l_m}). \tag{13}$$

We explain each of the multiplicative factors in (13) 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 $Q_{l_m}(k, |\mathcal{P}|)$ the *popularity factor* for data unit l_m in the neighborhood of peer p. This quantity describes how often this data unit is encountered among the peer nodes in $\mathcal{P}$. Specifically, based on the number of replicas k of data unit l_m found in $\mathcal{P}$ and the size of the neighborhood $|\mathcal{P}|$, the popularity factor returns a number that is inversely proportional to the ratio $k/|\mathcal{P}|$. When the frequency of coming across l_m in $\mathcal{P}$ increases, the popularity factor decreases and vice versa. The motivation behind using

[4]For example, for the first "B" data unit in Figure 6 this is the collection of data units encircled in dotted red.

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 (13) 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 then operates as follows. First, the current importance values for data unit $l_m \in M$ and $m = 1, \ldots, |M|$ are computed using (13). 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 $\mathcal{P}_{(l_{m_j})} \subset \mathcal{P}$ denote the subset of neighbors 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 $p \rightarrow p_{n_k}$ and receiving the data unit on the backward channel $p_{n_k} \rightarrow p$. 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 $p_{n_k} \in \mathcal{P}_{(l_{m_j})}$ and $k = 1, \ldots, |\mathcal{P}_{(l_{m_j})}|$ from statistics of the channel and estimate of the bandwidth, as described in Appendix B 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[5], 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 $\mathcal{P}$. The major computational steps are summarized in Algorithm 2.

Algorithm 2 Packet scheduling algorithm

```
1:  Initialization: NodeSchedule = {}, DataUnitSchedule = {}
2:  for $m = 1, \ldots, |M|$ do
3:      for $l_m \in M$ do
4:          Compute $\mathcal{I}_{l_m}$
5:      end for
6:  end for
7:  Sort $\{\mathcal{I}_{l_m}\}$ in decreasing order $\Rightarrow M^{\text{sort}}$
8:  for $j = 1, \ldots, |M|$ do
9:      for $l_{m_j} \in M^{\text{sort}}$ do
10:         for $k = 1, \ldots, |\mathcal{P}_{(l_{m_j})}|$ do
11:             for $p_{n_k} \in \mathcal{P}_{(l_{m_j})}$ do
12:                 Compute $Prob\{RTT_{(p,p_{n_k})} < t_{d,l_{m_j}}\}$
13:             end for
14:         end for
15:         Find $MAX = \max_{p_{n_k}}\{\, Prob\{RTT_{(p,p_{n_k})} < t_{d,l_{m_j}}\}\,\}$
16:         if $MAX > 0$ then
17:             Update schedules:
18:             Find $p^*_{n_k} = \arg\max_{p_{n_k}}\{\, Prob\{RTT_{(p,p_{n_k})} < t_{d,l_{m_j}}\}\,\}$
19:             NodeSchedule = {NodeSchedule, $p^*_{n_k}$}
20:             DataUnitSchedule = {DataUnitSchedule, $l_{m_j}$}
21:         end if
22:     end for
23: end for
24: Execute schedules :
25: for $n = 1, \ldots, |\text{NodeSchedule}|$ do
26:     $p_n$ = NodeSchedule(n); $l_n$ = DataUnitSchedule(n)
27:     Send request to node $p_n$ for data unit $l_n$
28: end for
```

IV. Utility-based P2P streaming system

A. System description

We describe in this section a P2P streaming system that integrates the mesh construction and the utility-based scheduling described above. First, we assume that a registry server keeps track of the peers 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 registry server maintains a sorted list of the registered peers in increasing order of their minimum latency to the server.

A connect procedure for a peer joining the network comprises the following steps. The peer contacts the registry server and provides it its IP address. In return, the server provides the sorted list of peers that the connecting peer can use to select its own neighborhood in the network, according to Algorithm 1. In order to trade off computational complexity and performance, the algorithm does not necessarily select the neighborhood with the smallest latency spread but rather one of the latency-based neighborhoods that still accept nodes, as discussed in Section II-C. 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 first if the contacted node can accept a new neighbor[6]. At

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

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

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 neighbor, the peer creates its own neighborhood 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 measures from neighbors to the values in the sorted list provided by the registry server. The smallest of these values for the neighborhood of the peer is returned to the registry server[7]. This concludes the connect procedure for a peer. Each neighborhood can accept a maximal number of peers that is equal to $K_{max} \leq K$.

Note here that for the peers that first join the network, the delays that they measure and store with the registry server actually represent the latency at which they receive data directly from the media server. Subsequent peers however cannot directly access the media server any longer, otherwise the server would be overwhelmed with requesting connections. These peers register their network delay from the media server indirectly, through nodes to which they connect to the network when establishing their own neighborhoods.

Once the overlay is established, each peer requests video data from a subset of other peers that form its neighborhood. The peer is interested in requesting these data units such that it maximizes its video quality. Each peer maintains a sliding window S of data units that periodically advances. It buffers the already received data units from this window and periodically exchanges maps describing the presence/absence of data units with its neighbors. In this way, a peer can discover at its neighbors the availability of data units presently missing in its window. The peer then computes the optimal schedule for requesting missing data from neighbors that maximizes the reconstruction quality of its media presentation. It periodically runs the Algorithm 2 in order to compute the sequence of requests it should send to its neighbors.

B. Connection updates

In order to deal with the dynamics of P2P systems, the neighborhood of a node in the overlay needs to be redefined continuously. Each peer periodically discards the neighbor with the smallest rate contribution, and selects a new peer to be included in its neighborhood using the procedure from Algorithm 1.

A peer periodically estimates the respective download rates from its neighbors. This is done by computing the total amount of data received from each neighbor 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 neighbor p_k within the last download rate estimation period T. Then, p computes the the received rate contribution from p_k as

$$\tilde{r}_{(p_k,p)} = \frac{\sum_{l \in \mathcal{DU}^{(p_k)}} B_l}{T}. \tag{14}$$

[7]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.

In this way, a peer can sort its neighbors based on their send rate contributions to this peer. Then, the peer can periodically replace the least contributing neighbor with a new peer selected at random. Furthermore, if the peer experiences multiple neighbor nodes with no rate contribution, it will simultaneously replace all of them with newly selected neighbors.

At the same time, the nodes can also check periodically if their minimum delays have changed. This can happen as the data rates at which they receive data from their neighbors can change over time. Hence, a node can update then its minimum delay value stored with the registry server. At the same time, the node informs its neighbors that its own minimum delay has changed so that they can revaluate in turn and if needed the minimum delay values for their respective neighborhoods. Note that the case of departure of one or multiple neighbors is covered with the above consideration as then the data rate(s) of the departed neighbor(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.

C. Resiliency to free-riders

In order to fight efficiently against free-riders that consume bandwidth without participating to the stream delivery, the nodes share their upload bandwidth in proportion to the number of bits received from the neighbors. Typically, a peer sends more packets to a neighbor that provides it with a lot of data. In particular, the algorithm for sharing the upload bandwidth of a peer among its requesting neighbors operates as follows. Let $C^{(u)}$ be the upload bandwidth of peer p, and let PR denote the subset of neighbors 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

$$r_{(p,p_k)} = \frac{\tilde{r}_{(p_k,p)}}{\sum_{p_k \in PR} \tilde{r}_{(p_k,p)}} \cdot C^{(u)}, \tag{15}$$

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 Eq. (15).

V. Simulation results

A. Simulation setup

In this section, we examine the performance of the proposed framework for streaming actual video content. We first analyze the influence of the urgency and popularity factors in the performance of the utility-based packet scheduling algorithm. Then we study the benefits of utiliy-based packet scheduling and efficient upload bandwidth sharing on the performance of the streaming system. Finally, we look at the advantages offered by the delay-based mesh construction over random overlay architectures.

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 [10]. 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 S for keeping track of data units at each peer is 30 seconds of data. Sending requests to its neighbors 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 neighborhood and the consecutive selection of a new replacement neighbor is done by a peer every 30 sec. Initially each peer selects 8 other peers as its neighbors. The size K of a neighborhood for a peer can grow subsequently to contain up to $K_{max} = 14$ other peers.

B. 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 $Q_{l_m}(k, |\mathcal{P}|)$ on the overall performance of our system. 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 $Q_{l_m}(k, |\mathcal{P}|)$ described in Section III-C. Specifically, for data unit l_m they are computed at time t as

$$U(t, t_{d,l_m}) = \left(\frac{t}{t_{d,l_m}}\right)^{\alpha}, \tag{16}$$

$$Q_{l_m}(k, |\mathcal{P}|) = \left(\frac{|\mathcal{P}|}{k}\right)^{\beta} \tag{17}$$

where the parameters $\alpha, \beta \geq 0$ are the powers of the polynomials for $U(t, t_{d,l_m})$ and $Q_{l_m}(k, |\mathcal{P}|)$, respectively. This formulation 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 $Q_{l_m}(k, |\mathcal{P}|)$ as a function of the parameters α and β. In Figure 7, 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 $Q_{l_m}(k, |\mathcal{P}|)$ (right) in Figure 7. 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 7. In brief, it can be seen from Figure 7 that $U(t, t_{d,l_m})$ (left) and $Q_{l_m}(k, |\mathcal{P}|)$ 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 $(k/|\mathcal{P}|)$.

We observe 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. In Figure 8, we show these quantities in the case of cable/dsl peers. In particular, in Figure 8 (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 8 (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 $Q_{l_m}(k, |\mathcal{P}|)$ 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 8 (right). It should be mentioned that similar observations and conclusions can be made for the case of ethernet peers. These results are not included here due to space considerations.

C. Utility-based scheduling

Here, we are interested in studying exclusively the performance advantages that our utility-based scheduling and bandwidth sharing techniques from Section IV provide. Therefore, in this section we opt to employ a standard technique for mesh construction and maintenance, where each peer selects randomly other nodes in the overlay as its neighbors. In Figure 9, 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 9. Furthermore, the cumulative

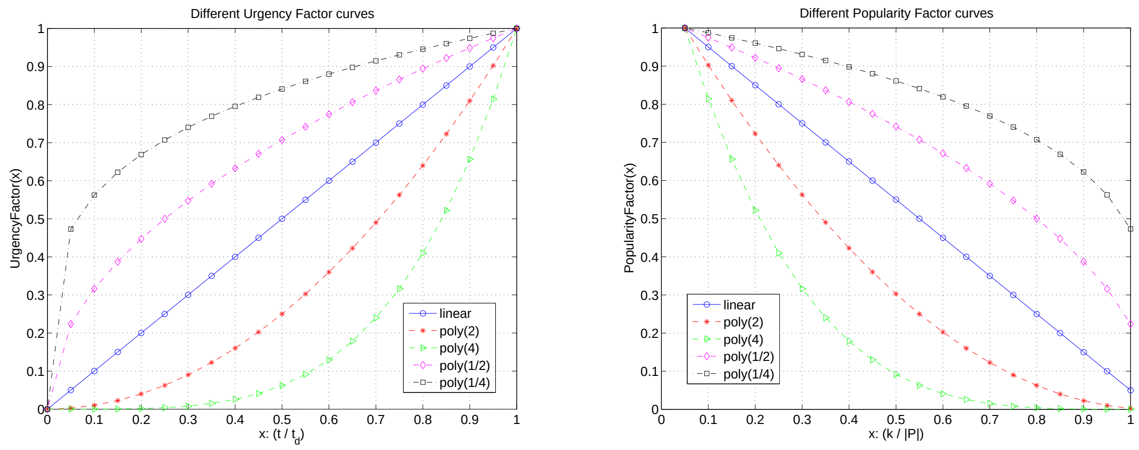


Fig. 7. Sample functions for the urgency and popularity factors $U(t, t_{d,l_m})$ (left) and $Q_{l_m}(k, |\mathcal{P}|)$ (right).

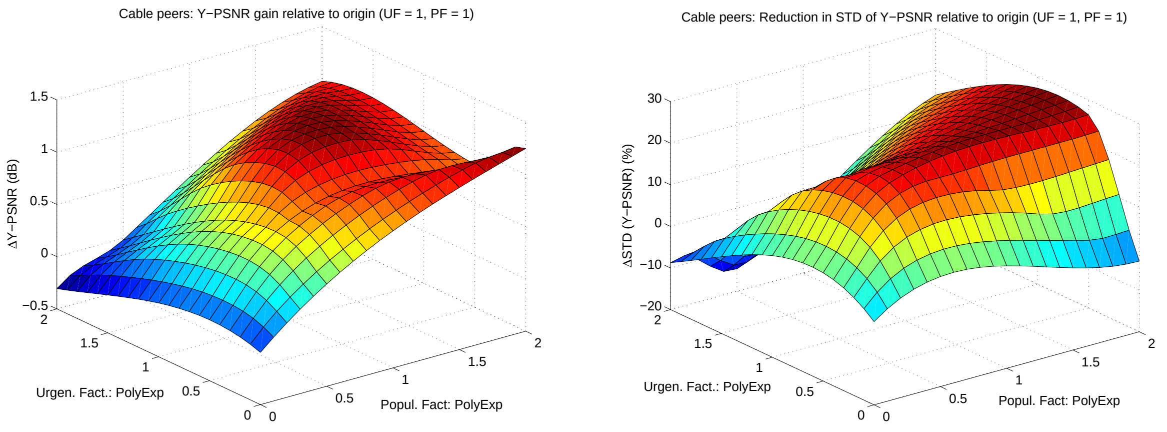


Fig. 8. Gain in Y-PSNR (dB) (left) and reduction (in %) of the standard deviation of Y-PSNR (right) when using the popularity and urgency factors in the case of cable peers.

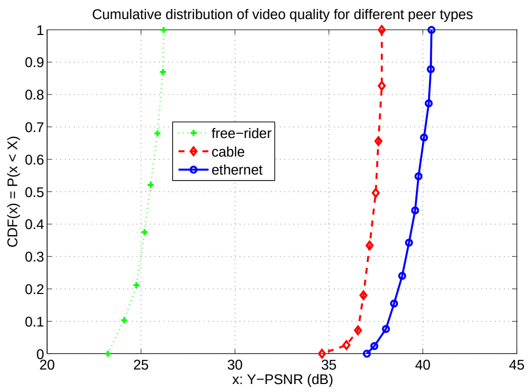


Fig. 9. Cumulative distribution function of average video quality (Y-PSNR) for different peer types.

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 neighbors 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 9.

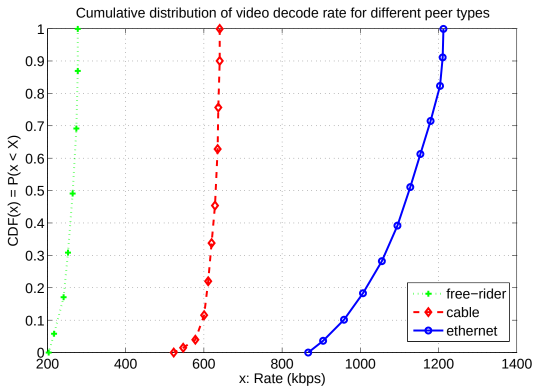


Fig. 10. Cumulative distribution function of average video decoding rate (kbps) for different peer types.

Next, we briefly go over the cumulative distribution of video decoding rate for the different peer types. This is 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 10 that ethernet peers exhibit much higher decoding 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 10. 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 decoding rate from Figure 10 correspond to those on average video quality from Figure 9.

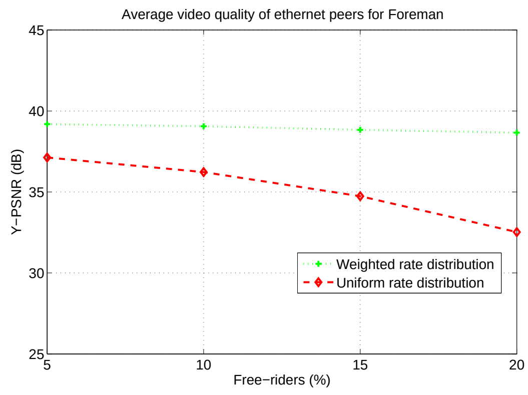


Fig. 11. Influence of free-riders with and without proportional upload bandwidth sharing.

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 11, 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 neighbors at same outgoing data rates. It can be seen from Figure 11 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 11. 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.

D. Minimum latency spread neighborhoods

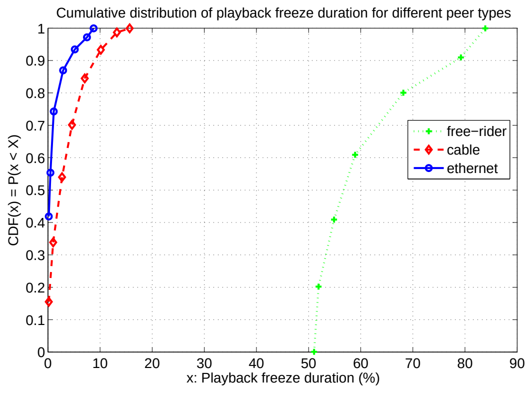
Here, we present the relative improvements in performance when the mesh construction algorithm (Min Spread) from Section II is employed, instead of the standard (Random) mesh construction technique where each peer selects its neighbors at random. The metrics over which we measure performance in this section are (i) frame freeze duration, (ii) normalized play-out time, and (iii) ratio ”send rate / receive rate”.

First, in Figure 12 we compare the difference in frame freeze duration experienced by the peers when each of the two mesh construction algorithms is used. 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 for this concealment method) until a subsequent frame is decodable and therefore ready to be displayed next.

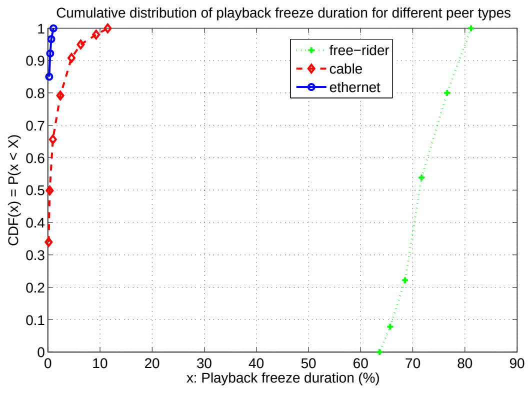
Specifically, in Figure 12a we show the cumulative distribution functions of the frame freeze duration for the three peer types in the case of random mesh construction, while in Figure 12b we show the corresponding results for the case of minimum latency spread mesh construction. It can be seen that 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 12b and 12a, respectively. Similarly, when the minimum latency spread 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.

The reduction in frame freeze duration when the Min Spread algorithm is used should result into a corresponding improvement in average video quality observed by the peers when playing the media presentation. This is confirmed with the graphs shown in Figure 13 that represent the distribution of video quality for the different peer types when our algorithm for mesh construction is used.

Next, we study the differences in normalized play-out time for a peer between random and minimum latency spread mesh constructions. 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



(a) Random



(b) Min Spread

Fig. 12. Frame freeze duration (%) for different peer types and mesh construction algorithms.

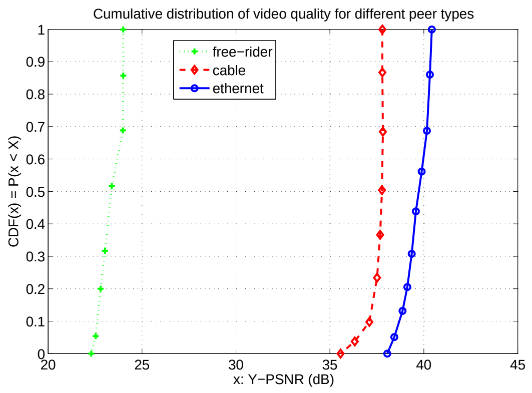


Fig. 13. CDF of average video quality (Y-PSNR) for different peer types in the case of Min Spread mesh construction.

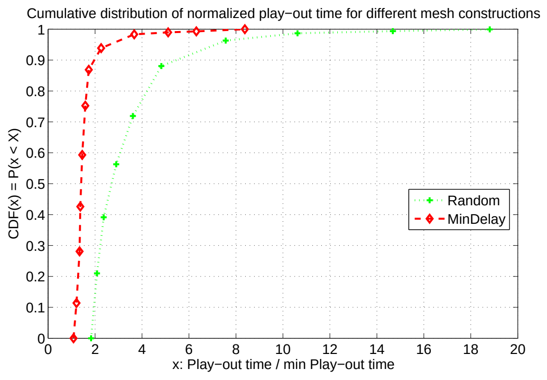


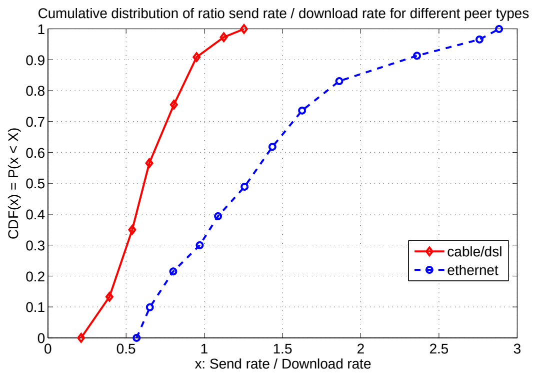
Fig. 14. CDF of normalized play-out time of a peer for different mesh construction techniques.

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.

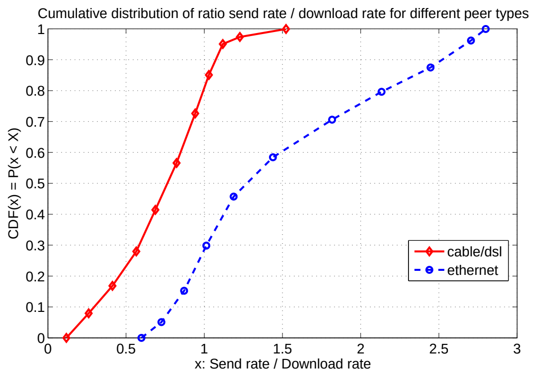
In Figure 14, we show the cumulative distribution functions of the normalized play-out time for a peer for each of the two mesh construction algorithms. As expected, we can see from Figure 14 that when our algorithm is employed the peers observe much shorter play-out times which in turn improves their 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 latency spread mesh construction, compared to about six times the preset play-out delay for the same percentage of peers for random mesh construction.

In the last results in this section, we examine the differences in the distribution of the ratio between the send rate and the receive rate for the different peer types and for each mesh construction algorithm. With the ratio ”send rate / receive rate” we actually measure how much of its uplink bandwidth a peer contributes over time for sending data to its neighbors relative to the amount of data the peer has received from them during the same period. This ratio indicates how much the peers share their bandwidth resources in the network under each mesh construction algorithm.

The results are shown in Figure 15. It can be seen that cable/dsl peers contribute more rate (hence data) to their neighbors in the case of Min Spread mesh construction. This is because the CDF of the ratio ”send rate / receive rate” for cable/dsl peers in Figure 15b is more skewed toward larger values relative to the corresponding graph in Figure 15a. Furthermore, the distribution of this ratio for ethernet peers remains more of less the same for both mesh construction algorithms, as evident from Figure 15. Hence, when Min Spread mesh construction is employed, the joint distribution of the ratio ”send rate / receive rate” for both cable/dsl and ethernet peers exhibits an average value of one, while for the case of random mesh construction this average value is smaller than one (around 0.88). Similarly, in the former case the standard deviation of this distribution is 30% smaller relative to the latter case. Therefore, given these observations it can be said that Min Spread mesh construction provides for a more



(a) Random



(b) Min Spread

Fig. 15. Ratio ”send rate / receive rate” for cable/dsl and ethernet peers and different mesh construction methods.

desirable distribution of bandwidth sharing in the network. It should be noted that the CDF graphs of the ratio ”send rate / receive rate” for free-riders are not shown in Figure 15 as they represent an impulse value of one at the value of zero for this ratio[8]. Hence, as such these results are irrelevant for the analysis presented above. Finally, it should also be noted that the statistical quantities described and discussed above are computed empirically based on our experimental data.

VI. Related Work

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 organization of the peers. Despite the plethora of prior work on tree-push based P2P streaming, e.g., [11], 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[9], and we will therefore focus on this type of solution, which generally offers increased robustness to the dynamics of a P2P system.

From the plethora of prior work, we describe first the studies that are the most closely related to the present paper from a system-wise perspective. In [13], 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 [14] 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 neighbors. Finally, [15] 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 neighbors in prioritized order based on their layer index and the probability of serving a neighbor is commensurate to the rate contribution received from this neighbor. Still, there are several significant differences between the two works, e.g., the delay-based overlay construction proposed in the present paper relative to the random neighbor selection employed in [15]. In addition, the present work considers content created using generic video encoding that may not be layered necessarily. Furthermore, a more sophisticated utility-based model is proposed here to determine the packets’ importance when requesting data that may be overlooked if only the index of the video layers to which the packets belong is considered, as proposed by the authors of [15]. Lastly, we design a deterministic algorithm for upload bandwidth distribution over the requesting peers which consistently, i.e., all of the time, rewards contributing peers instead of doing that on the average, as in [15].

Another body of prior work that needs to be mentioned in the context of the present paper is on exploiting locality information for improving the efficiency of P2P systems. Specifically, random peer selection combined with flooding-based approaches for content search in overlay networks contribute to excessive amounts of traffic even in moderate size networks [16, 17]. Most of this traffic is unnecessary and is caused by the mismatch between the logical topology of the overlay and the actual underlying physical network. The problem is exacerbated by blindly flooding message on multiple paths that in effect may have the same destination in the end. Therefore, such networks can suffer from scalability issues [18]. Another study that illuminates this mismatch problem is [19] which shows that only a small percentage of the overall number of connections in a Gnutella [20] overlay link peers within a single autonomous system. In order to account for this, there have been proposals for constructing more efficient overlay topologies in recent years, such as [21, 22] that cluster peers based on the closeness of their IP addresses, [23][36] that determines the closeness between two peers based on their latency to common landmark servers, and [24][x] that selects neighbors based on estimates of delay distances between peers. The works outlined above exhibit an analogy with our overlay construction procedure in the sense that they all attempt to cluster peers based on some notion of distance. Still, in the present paper we create neighborhoods exhibiting min latency

[8]Remember here that free-riders do not share their uplink bandwidth resources with other peers in the network

[9]An interesting overview of mesh-pull based P2P video delivery and its commercial success for IPTV applications can be found in [12].

spread across their member nodes relative to the origin server. This does not necessarily mean that two neighbors will be close to another according to the distance metrics employed above.

Lastly, from the perspective of packet scheduling in P2P systems, the present paper is related to prior works on optimized video streaming in tree-based overlays such as [25] that proposes to employ priority-based mechanisms for allocating resources across different packets at a node in a multicast tree. Differently, the present paper considers utility based scheduling for efficient video streaming in mesh-pull based P2P overlays. Maximizing a utility function of other parameters in a P2P streaming system, such as the playback buffer content reserve or the processor task scheduling at a peer, has been investigated previously in [26] and [27], respectively.

VII. 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 neighborhoods that exhibit similar latencies from the media server across their member nodes; (ii) an algorithm for requesting data from neighbors that maximizes the video quality at the peer while taking into account the popularity of the data units within the neighborhood; (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 established that utility-based packet scheduling and effective upload bandwidth sharing provide a significant improvement of the performance of a mesh-pull P2P streaming system. Both average video quality and decoding bandwidth increase with respect to the corresponding ones for a baseline P2P streaming systems. In addition, free-riders are effectively shut down from degrading the performance of the system by wasting its resources unnecessarily. The new mesh construction procedure further provides significant reductions in frame-freeze time and play-out delay. The improved continuity of the playback experience in this case provides for further gains in video quality in the media presentation.

Appendix

A. Statistics of the latency spread for uniform delay distribution

We can compute the pdf of W_m when the pdf of the delay $f(x)$ is uniform in the interval $[x_{min}, x_{max}]$. In particular, we can obtain closed form expressions in (4) and (5) and $f_{mn}(x,y)$ becomes

$$f_{mn}(x,y) = C_m\, x^{m-1}(y-x)^{K-2}(1-y)^{N-n} \tag{18}$$

when $x_{min} \leq x \leq y \leq x_{max}$, and 0 otherwise. Similarly, with some work and by noting that $f(y) = f(x+w_m) = 0$ for $x \geq x_{max} - w_m$, $g(w_m)$ can be succinctly written as

$$g(w_m) = C_m \int_{x_{min}}^{x_{max}-w_m} x^{m-1} w_m^{K-2}(1-x-w_m)^{N-n}\, dx. \tag{19}$$

Now, the integral in (19) can be solved by employing the transformation of variables $x = y(1-w_m)$ in which case one obtains the following expression

$$g(w_m) = I(w_m) w_m^{K-2}(1-w_m)^{N-K+1}, \tag{20}$$

for $0 \leq w_m \leq x_{max}$ and where the term $I(w_m)$ is given as

$$I(w_m) = C_m \int_{\frac{x_{min}}{1-w_m}}^{\frac{x_{max}-w_m}{1-w_m}} y^{m-1}(1-y)^{N-n} dy\,. \tag{21}$$

We note here that the variables W_m are identically distributed with a pdf given in (20).

For ease of presentation we can normalize the interval $[x_{min}, x_{max}]$ and respectively the samples in X to the unit range $[0,1]$. Consequently, this restricts the support of the functions $f(x)$ and $F(x)$ to the latter range, which in turn allows for simplification of some of the expressions developed thus far. In particular, the integral $I(w_m)$ in (21) simplifies to

$$\begin{aligned} I(w_m) &= C_m \int_0^1 y^{m-1}(1-y)^{N-n} dy \\ &= C_m\, B(m, N-n+1) \\ &= \frac{1}{B(K-1, N-K+2)}\,, \end{aligned} \tag{22}$$

where the constant $B(a,b)$ replacing the integral in the first line of (22) is also known as the Beta function [6], i.e.,

$$B(a,b) = \int_0^1 x^{a-1}(1-x)^{b-1} dx$$

and for which it holds

$$\binom{i}{j} = \frac{1}{(i+1)B(i-j+1, j+1)}\,,$$

a property that we used earlier in Equation (3). Consequently, $f(w_m)$ in (20) becomes the density of a beta $\beta(K.1, N-n+2)$ variate. Finally, given the above the pdf of the first order statistics $W_{m(1)}$ obtains the following form:

$$f_{W_{m(1)}}(w) = \frac{1}{B(1,N)}\,(1-I_w(K-1, N-K+2))^{N-1}\, f(w_m)\,, \tag{23}$$

where $I_x(a,b)$ in (23) denotes the incomplete beta function [6] that is sometimes also known as the regularized (due to the denominator term $B(a,b)$) incomplete beta function. Its expression is provided below

$$I_x(a,b) = \frac{1}{B(a,b)} \int_0^x t^{a-1}(1-t)^{b-1} dt\,.$$

Almost surprisingly, the above integral can be worked out, in the case of integer a, b, for example by using integration by parts. This is exactly what we have in (23). Therefore, for completeness we include the solution here

$$I_x(a,b) = \sum_{j=a}^{a+b-1} \frac{(a+b-1)!}{j!\,(a+b-1-j)!}\, x^j\, (1-x)^{a+b-1-j}.$$

Replacing in Eq. (23), we obtain the pdf of the first order statistics $W_{m(1)}$, in the case where delays are distributed uniformly.

B. Distribution of the RTT

We describe here how the statistical distribution of the round trip time is computed in the packet scheduling algorithm. It should be mentioned first 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[10], and (iii) the estimated transmission bandwidth of the channel $p_{n_k} \to p$. In particular, requesting a data unit comprises sending a small control packet to a designated neighbor. 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 $p \to p_{n_k}$ 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 $p_{n_k} \to p$ 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 $p_{n_k} \to p$ as a packet erasure channel with random transmission delays [7]. 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

$$Prob\{RTT_{(p,p_{n_k})} < t_{d,l_{m_j}}\} = (1-\epsilon_B)\int_{y<t_{d,l_{m_j}}} f(y)dy. \tag{24}$$

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

$$f(y) = \begin{cases} \theta e^{-\theta(y-\kappa)} & : \quad y \geq \kappa, \\ 0 & : \quad \text{otherwise.} \end{cases} \tag{25}$$

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 $p_{n_k} \to p$ 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[11] 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

$$\kappa = \frac{\sum_{l\in\mathcal{DU}} B_l + B_{l_{m_j}}}{\tilde{r}_{(p_{n_k},p)}} \tag{26}$$

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 (24) and (25) as

$$Prob\{RTT_{(p,p_{n_k})} < t_{d,l_{m_j}}\} = (1-\epsilon_B)\int_{\kappa}^{t_{d,l_{m_j}}} \theta\, e^{-\theta(y-\kappa)}dy\ . \tag{27}$$

[10]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}.

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

References

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

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

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

[4] D. J. Watts and S. H. Strogatz, "Collective dynamics of 'small-world' networks," *Nature*, vol. 393, no. 6684, pp. 440–442, Jun. 1998.

[5] H. A. David and H. N. Nagaraja, *Order Statistics*, 3rd ed. Wiley-Interscience, Aug. 2003.

[6] G. Casella and R. L. Berger, *Statistical Inference*, 2nd ed. Duxbury Press, Jun. 2001.

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

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

[9] ——, "Server diversity in rate-distortion optimized streaming of multimedia," in *Proc. Int'l Conf. Image Processing*, vol. 3. Barcelona, Spain: IEEE, Sep. 2003, pp. 645–648.

[10] 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.

[11] V. Padmanabhan, H. Wang, and P. Chou, "Resilient peer-to-peer streaming," in *Proc. Int'l Conf. on Network Protocols*. Atlanta, GA, USA: IEEE, Nov. 2003, pp. 16–17.

[12] X. Hei, Y. Liu, and K. Ross, "IPTV over P2P streaming networks: the mesh-pull approach," *IEEE Communications Magazine*, vol. 46, no. 2, pp. 86–92, Feb. 2008.

[13] 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: ACM, May 2006, pp. 56–61.

[14] X. Hei, Y. Liu, and K. 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.

[15] 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: ACM SIGCOMM, Aug. 2007, pp. 311–316.

[16] S. Saroiu, K. P. Gummadi, R. J. Dunn, S. D. Gribble, and H. M. Levy, "An analysis of internet content delivery systems," in *Proc. 5th Symp. Operating Systems Design and Implementation*. Boston, MA, United States: USENIX, Dec. 2002.

[17] S. Sen and J. Wang, "Analyzing peer-to-peer traffic across large networks," *IEEE/ACM Trans. Networking*, vol. 12, no. 2, pp. 219–232, Apr. 2004.

[18] J. Ritter, "Why Gnutella can't scale. No, really." 2001, http://www.darkridge.com/∼jpr5/doc/gnutella.html.

[19] M. Ripeanu, I. Foster, and A. Iamnitchi, "Mapping the Gnutella Network: Properties of Large-Scale Peer-to-Peer Systems and Implications for System Design," *IEEE Internet Computing Journal*, vol. 6, no. 1, pp. 50–57, Jan. 2002.

[20] "Gnutella," http://en.wikipedia.org/wiki/Gnutella.

[21] B. Krishnamurthy and J. Wang, "Topology modeling via cluster graphs," in *Proc. 1st Workshop on Internet Measurement*. San Francisco, CA, USA: ACM SIGCOMM, Nov. 2001, pp. 19–23.

[22] V. N. Padmanabhan and L. Subramanian, "An investigation of geographic mapping techniques for internet hosts," in *Proc. Data Communication, Ann. Conf. Series*. San Diego, CA, USA: ACM, Aug. 2001, pp. 173–185.

[23] Z. Xu, C. Tang, and Z. Zhang, "Building topology-aware overlays using global soft-state," in *Proc. 23rd Int'l Conf. Distributed Computing Systems*. Providence, RI, USA: IEEE Computer Society, May 2003, pp. 500–508.

[24] Y. Liu, L. Xiao, X. Liu, L. M. Ni, and X. Zhang, "Location awareness in unstructured peer-to-peer systems," *IEEE Trans. Parallel and Distributed Systems*, vol. 16, no. 2, pp. 163–174, Feb. 2005.

[25] J. Chakareski and P. Frossard, "Adaptive p2p video streaming via packet labeling," in *Proc. Conf. on Visual Communications and Image Processing*. San Jose, CA, USA: SPIE, Jan. 2007.

[26] Z. Li, J. Huang, and A. K. Katsaggelos, "Content reserve utility based video segment transmission scheduling for peer-to-peer live video streaming system," in *Proc. 45th Allerton Conf. on Communications, Control, and Computing*. Monticello, IL: IEEE, Sep. 2007, pp. 563–567.

[27] F. Chen, "A utility-based approach to scheduling multimedia streams in peer-to-peer systems," in *Proc. 18th Int'l Symp. Parallel and Distributed Processing*. Santa Fe, NM, USA: IEEE Computer Society, Apr. 2004.