Adaptive Overlay Topology for Mesh-Based P2P-TV Systems

Richard Lobb, Ana Paula Couto da Silva, Emilio Leonardi, Marco Mellia, Michela Meo
Politecnico di Torino
Dipartimento di Elettronica
Corso Duca degli Abruzzi, 24 - 10129 Torino, ITALY
{firstname.lastname}@tlc.polito.it

ABSTRACT

In this paper, we propose a simple and fully distributed mechanism for constructing and maintaining the overlay topology in mesh-based P2P-TV systems. Our algorithm optimizes the topology to better exploit large bandwidth peers, so that they are automatically moved close to the source. This improves the chunk delivery delay so that all peers benefit, not just the high bandwidth ones. A key property of the proposed scheme is its ability to indirectly estimate the upload bandwidth of peers without explicitly knowing or measuring it. Simulation results show that our scheme significantly outperforms overlays with homogeneous properties, achieving up to 50% performance improvement. Moreover, the algorithm is robust to both parameter setting and changing conditions, e.g., peer churning.

1. INTRODUCTION

Peer-to-peer Live Streaming (P2P-TV) systems are candidates for becoming the next Internet killer applications, as testified by the growing success of commercial systems such as PPLive, SopCast and many others. These systems, which allow users to "watch television" over the Internet, have very low infrastructure costs, enabling almost anyone to become a content provider to the whole world. Freed from normal broadcasting constraints, any user can potentially watch any content anywhere.

In P2P-TV system, hosts running the application, called *peers*, form an *overlay topology* by setting up virtual links over which information is transmitted and received. A source peer is responsible for injecting the video stream, by chopping it into segments called *chunks* of a few kilobytes, which are then sent to a subset of its neighbouring peers, called *neighbours*. Each peer can then contribute to the chunk diffusion process by retransmitting chunks to its neighbours following a swarming like behavior, as in file sharing P2P systems like *BitTorrent*. The major differences between P2P-TV systems and traditional P2P file sharing applications are i) that the source is generating the stream in real time, ii) that data must be received by peers at constant rate, and iii) that chunks must arrive almost in sequence so that they can be immediately played at the receiver. Therefore, the main performance indices to be optimized in P2P-TV systems are the *chunk delivery delay*, i.e., the delay from when the source emits the chunk to when a peer receives it, and the *chunk loss rate*. To this extent, typical P2P file sharing applications mechanisms, such as the BitTorrent tit-for-tat, hardly adapt to this scenario, since they target throughput optimization rather than delay.

In this paper we consider unstructured (mesh-based) P2P-TV systems, focusing our attention on the design of efficient distributed mechanisms for the construction and the maintenance of the overlay topology, given nodes with heterogeneous upload bandwidths. The performance of these P2P-TV systems is mainly determined by two mechanisms: the *scheduling policy* with which a peer decides which chunk to distribute to which of its neighbours; and the *overlay construction* which determines the neighbourhood of each peer, i.e., the topology over which chunks are distributed. Several works have focused on chunk scheduling algorithms [1, 2, 3, 4, 5, 6], devising, proposing and analyzing numerous different 'optimal or near optimal' methods. For the case of bandwidth-heterogeneous networks it has been shown that it is important to preferentially distribute chunks to high-bandwidth nodes first, so that chunk replication occurs as rapidly as possible [4, 5].

From a chunk-scheduling perspective, a fully connected overlay is theoretically ideal in that a peer can then always make an optimal choice of peer as target for any chunk. From a practical standpoint, however, maintaining large neighbourhoods increases the complexity of the scheduling algorithm, forces each peer to maintain a large amount of information and wastes bandwidth since peers have to frequently exchange signalling messages with all their neighbours. Hence it is essential to restrict the overlay topology so that each peer has only a limited set of neighbours available. The overlay has to adapt to the peers' resources (e.g., bandwidth), to the "churn" of peers, and to variations in network status. In this paper we propose a simple distributed algorithm to achieve these goals and we show that we obtain significant performance improvement with respect to static overlays, up to 50% gain is achieved. Our scheme does not need an *a priori* knowledge of each peer's upload bandwidth, but simply responds to its observed performance, measured in terms of how many other peers it is able to serve. It thus

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
Copyright 200X ACM X-XXXXX-XX-X/XX/XX ...$5.00.

reacts appropriately when a peer's performance is substandard, e.g., due to high CPU load or network demands from other applications.

The problem of the overlay topology optimization in mesh-based P2P-TV system has been almost neglected in the literature. To the best of our knowledge, apart from [9], performance of mesh-based systems has always been analyzed assuming the overlay topology to be either a fully connected mesh or a static random graph with given degree. In [9] the problem of building an efficient overlay topology has been formulated as an optimization problem. However, the proposed approach, which is deterministic and assumes perfect knowledge of the overlay edge costs, appears impracticable for large unstructured P2P-TV systems. Furthermore, the considered scenario in which the stream delivery delay is mainly due to the transport network latency, differs significantly from the scenario considered in this paper and more-commonly adopted in literature [1, 2, 3, 4, 6], where system performance is constrained by peer upload bandwidth.

2. ASSUMPTIONS

Consider a P2P-TV system composed of one video stream source and N peers interested in the video stream; the set of peers is denoted by $\mathcal{P}$. The stream is divided into chunks of fixed size of S bits. The source generates chunks at rate λ and sends them to some peers that, in their turn, forward the chunks to other peers in a P2P fashion. Peers have finite upload bandwidth, while download bandwidths are all sufficiently high that chunk delay is determined solely by upload bandwidth. We denote by B_p the upload bandwidth of peer p. The delay associated with chunk transmission is determined only by the upload bandwidth of the sender and is equal to S/B_p.

Peers are organized in an *overlay* network. The overlay can be represented by a directed graph $G(\mathcal{P}, E)$, where $(p, q) \in E$ if p can deliver the chunk to q; q is an out-neighbour of p (or, simply, a neighbour) and p is an in-neighbour of q. For each peer p, we denote by $\mathcal{N}^o(p)$ the set of its out-neighbours; $L(p) = card(\mathcal{N}^o(p))$ is the cardinality of this set, i.e., it is the number of output links of p. We denote by $\mathcal{C}(p)$ the set of neighbours of peer p's neighbours. The only knowledge that peer p has of the overlay topology is its own neighbourhood $\mathcal{N}^o(p)$ and the union of all its neighbours' neighbourhoods, $\mathcal{C}(p)$, assumed to be acquired by querying of its (p's) neighbours. Peers can join and leave the network independently.

Over a given overlay, the chunk distribution process is determined by the scheduling policy adopted at the peers. We consider the *latest useful chunk* strategy for the selection of the chunk, meaning that the peer delivers the most recent chunk; the peer to deliver the chunk to is chosen following a strategy inspired by the *bandwidth-aware* scheduling policy proposed in [4]. Details are given in section 3.3.

3. ADAPTIVE OVERLAY AND SCHEDULING ALGORITHMS

The adaptive overlay algorithm is independently performed by each peer $p \in \mathcal{P}$ in a fully distributed way. The main idea behind it is simple. After every δ_c received chunks, which we call a *time window*, peer p possibly adjusts its neighbourhood by growing or pruning it. The decision is taken based on the fraction of used output links.

- If, during the time window, p has sent at least one chunk to a large fraction of its neighbours, it is assumed that p can feed more peers and n^+ new neighbours are assigned to it. The new neighbours are chosen within the set $\mathcal{C}(p)$.
- On the contrary, if p has used only a small fraction of its output links, its neighbourhood is shrunk by culling n^- of the neighbours to which no chunk was sent.

Basically, the number of used links is interpreted as an indication of the contribution that p can give to the stream delivery and p's neighbourhood size is adjusted accordingly. The objective of the algorithm is to make the neighbourhood size of p slightly larger than the number of used links, so as to trade off between two opposing needs: i) to reduce the possibility that some upload bandwidth of p is not used due to lack of neighbours (a larger neighbourhood is desired), ii) to reduce the possibility of overlay maintenance overload (a smaller neighbourhood is desired). In particular, let $U(p)$ be the set of p's *used* links and $\overline{U}(p)$ the set of *unused* links, with $card(U(p))$ and $card(\overline{U}(p))$ being their cardinalities; thus, $L(p) = card(U(p)) + card(\overline{U}(p))$. The algorithm aims at maintaining the following condition:

$$\alpha_L card(U(p)) < card(\overline{U}(p)) < \alpha_H card(U(p)) \qquad (1)$$

with α_L and α_H representing minimum and maximum thresholds that define the percentage of unused links in a neighbourhood. Note that the terms "used" and "unused" refer to the observations made by p during a time window corresponding to δ_c received chunks; thus, an unused link is not used during the current window but might be used in the following one. The algorithm is sketched below.

Algorithm 1: Adaptive Overlay Algorithm

```
for every $\delta_c$ received chunks do
    /* scan and adjust p's neighbourhood */
    if $card(\overline{U}(p)) < \alpha_L card(U(p))$ then
        /* grow the neighbourhood */
        Select $n^+$ neighbours from $\mathcal{C}(p) - \mathcal{N}^o(p) - \{p\}$
        Add the $n^+$ chosen neighbours to $\mathcal{N}^o(p)$
    end
    else if $card(\overline{U}(p)) > \alpha_H card(U(p))$ then
        /* shrink the neighbourhood */
        Select $n^-$ neighbours from $\overline{U}(p)$ for culling
        Cull the $n^-$ neighbours from $\mathcal{N}^o(p)$
    end
    else
        /* $p$'s neighbourhood does not change */
    end
end
```

A minimum number of out-neighbours e and in-neighbours f is enforced by the algorithm (not shown in the sketch above for the sake of simplicity) so as to avoid disconnecting some peers from the overlay. Throughout this paper we fixed $e = 1$ and $f = 2$.

3.1 Growing the neighbourhood

Let us turn our attention to the algorithm parameter settings. We start by considering the phase of neighbourhood growth.

At each peer bootstrap, the neighbourhood size is initially small, so that a slow peer does not risk becoming congested and overloaded due to maintaining too large a neighbourhood.

The parameter n^+, the increase in the neighbourhood size, is computed as $kL(p)$; k is a growth rate factor. k is initially k_i but linearly reduces to k_f, with $k_f < k_i$, over the first $N_{startup}$ chunks. The higher starting value is to ensure that high bandwidth peers, which are precious resources for the system and should have large neighbourhoods, can grow their neighbourhoods quickly.

The n^+ peers to be added to the neighbourhood of p are chosen from among the neighbours of peer p's neighbours, trying to favor those that can contribute the most to the chunk diffusion. Since the algorithm adjusts the neighbourhood size of a peer based on its contribution to the stream diffusion, the neighbourhood size of a peer (its out-degree) is an implicit measure of the peer's real upload performance. Thus, p chooses new neighbours with a selection probability that is a function of their out-degree.

Let p be, as before, a peer which decides to increase its neighbourhood size by n^+. A value $D(q)$ is associated with each peer $q \in \{\mathcal{C}(p) - \mathcal{N}^o(p) - \{p\}\}$; $D(q)$ is the *desirability* of q and is computed as a non-decreasing function of $L(q)$, the out-degree of q. Then, q is chosen with a probability that is proportional to $D(q)$.

The desirability function used in the rest of the paper is

$$D(q) = \sqrt{L(q)} \tag{2}$$

In our preliminary investigation (not reported here for the sake of brevity [10]) we tried different desirability functions. The selected function is indeed a good trade-off between chunk delay and loss probability. Intuitively, a too greedy policy risks to cluster high-bandwidth peers while leaving low-bandwidth peers at the overlay edge, thus causing higher chunk loss.

3.2 Shrinking the neighbourhood

n^-, the number of neighbours culled from the neighbourhood when shrinking is necessary, is computed as

$$n^- = \lfloor card(\overline{U}(p)) - \frac{\alpha_L + \alpha_H}{2} card(U(p)) \rfloor \tag{3}$$

The second term represents the desired number of unused links, which is the fraction $(\alpha_L + \alpha_H)/2$ of the number of used links. The difference between that number and the actual unused number is the number to cull.

The output links to cull are selected within the set of unused links again based on the desirability function of the corresponding neighbour. In particular, if n^- neighbours to cull must be selected out of $\overline{U}(p)$, then $card(\overline{U}(p)) - n^-$ neighbours are chosen randomly with a probability that is proportional to their desirability function: these neighbours are maintained in the neighbourhood while the remaining neighbours are culled.

3.3 The chunk scheduler

Because the overlay algorithm requires that each peer retain all links used during a time window, it is in a sense driven by the underlying chunk scheduler, which decides which link to use for any given chunk. The overlay can be considered to be adapting to the needs of the chunk scheduler, whichever it is. Because of this coupling, it is appropriate to explain how the selected chunk scheduler works.

We use *push-based* chunk transmission, i.e., transmission of a chunk between any two peers is always initiated by the sender. Considering a chunk c available at peer p we denote with $\mathcal{N}(c,p)$ the set of neighbouring peers of p which are still missing chunk c.

Whenever a chunk arrives at a peer it is stored within the peer's playout buffer, a buffer of chunks available for redistribution, unless the chunk delay exceeds the global playout delay d_{max}. The peer p then performs a chunk scheduling operation as follows:

- p selects the most recent chunk c such that the set $\mathcal{N}(c,p)$ is not empty.

- The destination peer $q \in \mathcal{N}(c,p)$ is stochastically selected with probability

$$p_q = \frac{D(q)}{\sum_{r \in \mathcal{N}(c,p)} D(r)} \tag{4}$$

  where $D(q) = \sqrt{L(q)}$ is the desirability of q. We call this a *random weighted-neighbour* selection.

The chunk scheduler and the overlay algorithm currently use the same desirability function $D(q)$ for selecting links. We emphasize that $D(q)$ is used as an indirect estimate of the peer upload bandwidth B_q but it does not require any explicit bandwidth estimation technique. Indeed, as the presented numerical results will show, peer out-degree is tightly correlated with peer upload bandwidth.

4. THE SIMULATOR

An event-driven simulator was developed to evaluate the performance of the proposed adaptive overlay construction algorithm. For the sake of completeness, before presenting the results we discuss some simulation assumptions. The simulator, including the code that implements the algorithms described in this paper, is freely available from [7].

Our simulator allows us to specify C different peer bandwidth classes. For each class i of peer we specify what percentage of peers of the total of N peers are of that class and the upload bandwidth range, B_i of peers in that class.

The simulation starts with a random overlay topology, that belongs to the class of *Regular Random Graphs* (RRGs) in which the arcs are randomly placed [8]. Each peer has d^o randomly assigned neighbours; the out-degree $L(p)$ within the starting overlay is thus constant and equal to d^o.

Unless specified otherwise, the following parameter values were used throughout the various simulation runs.

- Number of peers: $N = 10,000$, with $C = 4$ classes as follows.
  - Class 1: 10% of N, $B_1 = 5$Mb/s $\pm 10\%$
  - Class 2: 40% of N, $B_2 = 1$Mb/s $\pm 10\%$
  - Class 3: 40% of N, $B_3 = 0.5$Mb/s $\pm 10\%$
  - Class 4: 10% of N, $B_4 = 0$Mb/s

  The last class may represent the set of peers that is behind a NAT and cannot contribute to the chunk diffusion.

  The average upload bandwidth is 1.1 Mbps.

The source is a special additional peer with a bandwidth of 5.5 Mb/s. It emits 0.1 Mb chunks at the rate $\lambda = 10$ chunk/s, corresponding to 1 Mb/s. Each peer starts with a neighbourhood of $d^o = 10$ and uses a playout delay of $d_{max} = 5$s, i.e., 50 chunks. Algorithm parameters, if not otherwise specified, are: $k_i = 0.4$, $k_f = 0.1$, δ_c =50 chunks, $N_{startup} = 750$, and $\alpha_L = 0.1, \alpha_H = 0.3$. Finally, the total chunks pushed through overlay is by default 10,000 chunks, i.e., 1000s of video[1]. Many of the algorithm's parameter values are somewhat arbitrary but, as will be demonstrated in the results section, seem to result in a rapid convergence and stable behaviour under a wide range of conditions.

5. ALGORITHM OVERLAY TOPOLOGY

First, let us discuss the overlay evolution. Figure 1 shows the mean out-degree for each peer class as a function of time. The graph suggests that the overlay adapts to within about 10% of its final state in roughly $1,000$ chunks, i.e., about 100 seconds.

Table 1 shows a breakdown of links between classes, in percentage terms, at the end of the run of 10,000 chunks. The last column is the mean out-degree of peers in a given class. Both the adaptive topology algorithm and the scheduling mechanism are biased in favour of high bandwidth peers, because for optimal performance these should tend to receive chunks before low bandwidth peers. Hence we see that 34% of the source's outlinks go to class 1 peers, which constitute only 10% of the total, while only 0.9% go to class 4 peers, which also constitute 10%. Similar figures apply to class 1 peers. We have achieved a *clusterized* overlay topology with high bandwidth peers highly connected with each other. The case of class 4 peers is not reported since the corresponding links are irrelevant for chunk distribution; they exist because of mandatory requirement that all peers have at least e =1 out-going link. The final mean out-degree of the overlay is 20.7.

Table 1: Percentage breakdown of links from peers of one class to peers of another.

	To 1	To 2	To 3	To 4	Degree
From source	33.6	44.0	21.6	0.9	116
From class 1	25.8	42.9	29.2	2.1	120.4
From class 2	19.6	35.7	36.7	8.0	14.5
From class 3	19.4	31.7	37.4	11.4	6.9

We have tested with up to 50k peers and the transient behaviour and final topology characteristics change very little. Table 2 shows how the mean out-degrees, d_1, d_2 and d_3, of peer classes 1, 2 and 3 respectively vary with the number of peers, N; d_4 is not shown as it is always at the lower limit of 1 (see section 3.2).

The final topology is also insensitive to the initial topology of the random GNR network, as determined by the random number seed at the start of the run. In a series of 11 runs with $N = 1,000$ peers the mean out-degrees of the various classes of peer varied by at most ±3.2%.

6. OVERLAY PERFORMANCE

[1]Throughout the paper, time is always indicated in terms of the chunk number.

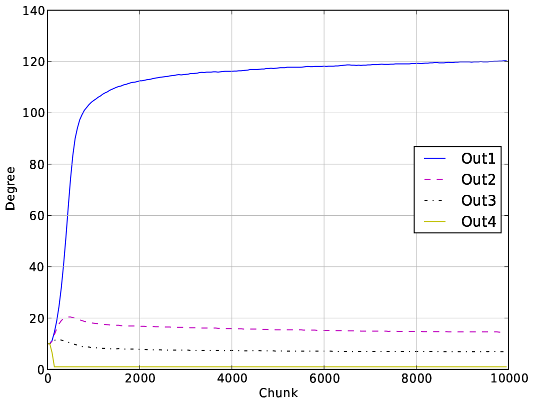


Figure 1: Mean out-degree versus time for the various classes of peer; $N = 10,000$.

Table 2: Mean out-degrees of the various peer classes as a function of N

N	d_1	d_2	d_3
10000	114.4	16.4	7.5
20000	111.9	16.5	7.6
30000	111.5	16.7	7.7
40000	111.2	16.6	7.7
50000	110.5	16.7	7.7

Figure 2 shows how chunk delivery delay varies with time. The figure reports the mean chunk delay computed over groups of 100 consecutive chunks. It can be seen that, apart from the very initial transient, delays decrease rapidly for all classes of peers during the initial adaptation phase, which, as noted before, lasts about 1000 chunks. As confirmation that high bandwidth peers tend to be served before low bandwidth peers, observe that the curves are ordered with respect to classes and class 1 peers have the lowest delay.

Figure 3 shows the number of lost chunks versus chunk number. Most of the losses occur during the very initial phase and no losses are observed after about chunk number 300.

Fig. 4 shows the effect on chunk loss rate and chunk delay of varying the video rate from 1.0 Mb/s (a load factor of approximately 0.9) through to 1.2 Mb/s (a load factor of approximately 1.1). Curves of median chunk delay (seconds) and chunk loss rate (percent) are shown for the standard playout delay of 5 seconds and also for a playout delay of 10 seconds. Data relates to the last 7,000 chunks out of 10,000 chunk runs, with 10,000 peers. All other parameters are as before. It can be seen that the chunk delay is unaffected by both playout delay and video rate but that the loss rate rises sharply as the load factor approaches 1. Increasing the playout delay significantly improves the chunk loss rate only near the critical load factor.

All calculations in this paper so far have used the values $\alpha_L = 0.1$, $\alpha_H = 0.3$, the goal being to achieve an "oversupply" of out-links of around 20%. Table 3 shows how the median chunk delay for a 1000-peer overlay varies with α_L and α_H. The results, which are calculated over the last 1000

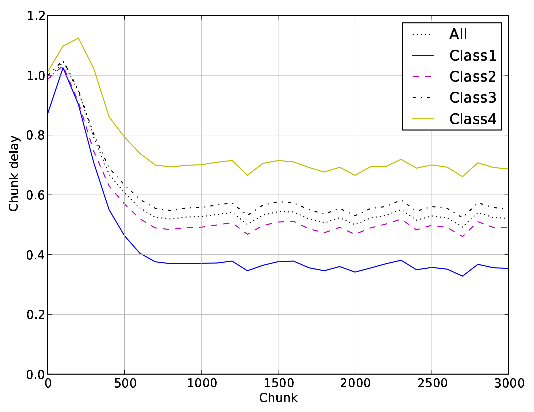


Figure 2: Chunk delivery delay (secs), averaged over 100 chunks, versus time and per peer class.

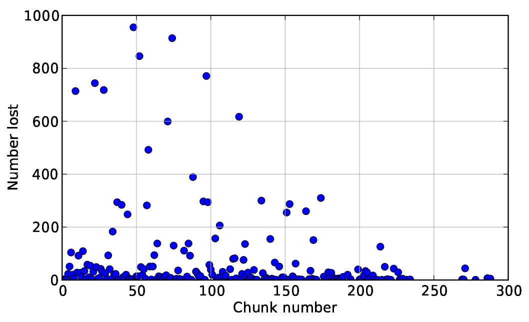


Figure 3: Number of chunks lost versus time.

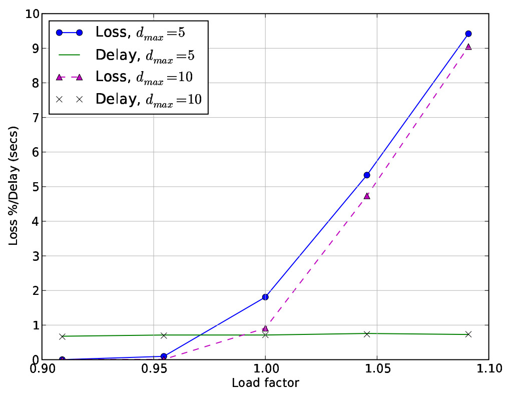


Figure 4: The effects of varying video rate and playout delay on chunk loss rate and delay.

chunks of runs with 3000 chunks, suggest that, in this scenario, our choice is a little sub-optimal from a delay standpoint. Best results for delay are obtained with $(\alpha_L,\alpha_H) = (0.4, 0.5)$ or $(0.5, 0.6)$; those values reduce the delay by just over 13%. However, they also result in considerably more links: the average degree rises from 23.2 with $(\alpha_L, \alpha_H) = (0.1, 0.3)$ to 34.5 and 39.1 with $(\alpha_L, \alpha_H) = (0.4, 0.5)$ and $(0.5, 0.6)$, respectively. A 50% increase in the number of links seems a rather high price to pay for a 13% delay reduction.

Table 3: Median chunk delay (secs) as a function of α_L and α_H

	α_H					
α_L	**0.2**	**0.3**	**0.4**	**0.5**	**0.6**	**0.7**
0.1	0.438	0.453	0.478	0.478	0.463	0.478
0.2		0.408	0.438	0.468	0.498	0.473
0.3			0.398	0.443	0.453	0.468
0.4				0.393	0.433	0.468
0.5					0.393	0.423
0.6						0.398

6.1 Comparison against random topology

In this section, we evaluate the performance improvement achieved by distributing the chunks over an adaptive overlay topology instead of a static random overlay topology with constant degree, which is the typical overlay topology considered in previous literature.

Fig. 5 shows the variation in both median chunk delay and 95-percentile chunk delay through a fixed-degree GNR overlay as the degree, k, is increased. The horizontal lines at 0.408 secs and 0.818 secs show the performance of the adaptive overlay, using the simulation run shown in Table 3 with $(\alpha_L, \alpha_H) = (0.2, 0.3)$. For that run the mean degree was 25.3, as represented by the single plotted marker on each horizontal line. The adaptive overlay delay measures are approximately 60% of the GNR overlay at comparable degree and even very high-degree GNR overlays have inferior performance. The fact that both median and 95-percentile delays have improved by a similar amount confirms that the improvement in median chunk delay has not been at the expense of low-bandwidth peers.

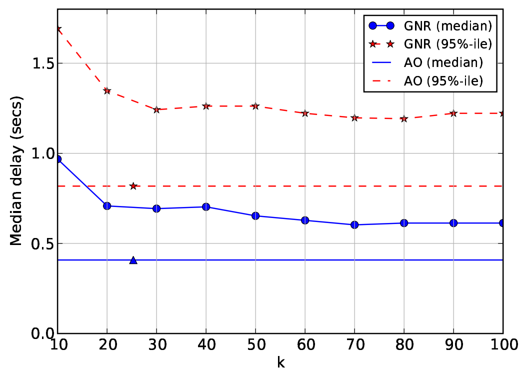


Figure 5: Median chunk delays through GNR overlays of various degrees, k.

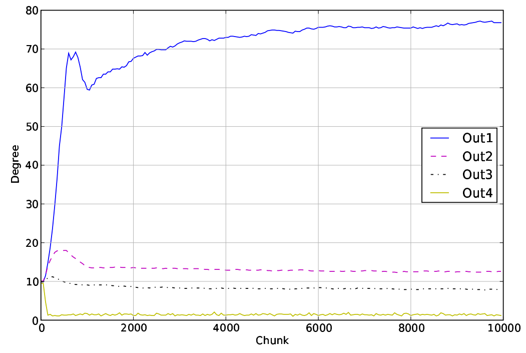
Intuitively, the delay for static topologies is larger because the peers' upload bandwidth is not optimally used. Consider for example class 1 peers: with their upload bandwidth, these peers can deliver a chunk to several neighbours; thus, to fully exploit their bandwidth, these peers need that they always have some neighbour needing the chunk they are distributing. However, imposing the same large neighbourhood on low bandwidth peers is both useless for the chunk diffusion process (they can serve only a few, if any, other peers) and harmful for them, as they have to manage large status and signaling traffic with only a low bandwidth. An additional reason for the improvement is that the scheduler and the adaptive overlay algorithms work together to create a somewhat clusterized overlay topology, in which high bandwidth peers are highly connected with each other.

6.2 The impact of churning

We now consider a scenario in which 50% of the peers are "churning", i.e., are intermittently disconnecting and reconnecting. Each churning peer generates a random time interval in the range 10 to 100 seconds (100 to $1,000$ chunk arrivals) and at the end of that time it resets its neighbourhood by discarding all in-coming and out-going links and then connecting with both an in-link and an out-link to 10 randomly-selected peers, which may include the source. A new random time interval in the same range is generated and the process continues throughout the simulation. We used $N = 1,000$ and kept all other parameters as specified in section 4.

Figure 6 shows the out-degree versus time on the top plot and the mean delay per chunk for this new scenario[2]; compare the results to Figures 1 and 2 where no churning was present.

Although the traces are now somewhat noisy, as would be expected, the algorithm is managing to maintain a fairly high mean out-degree for the critical class 1 peers even though 50% of them are churning. Similarly, the chunk delay suffered by peers still improves from the initial values for all classes of peers. Comparing to improvements in a scenario when no churning is present (Fig. 5, which also considers 1000 peers), the average delay the system can achieve grows to 0.6 due to churning effects (it was 0.4 when considering no churning). In other results, not shown here for the sake of brevity, we observed that the algorithm was robust and stable even with 90% of churning peers [10].



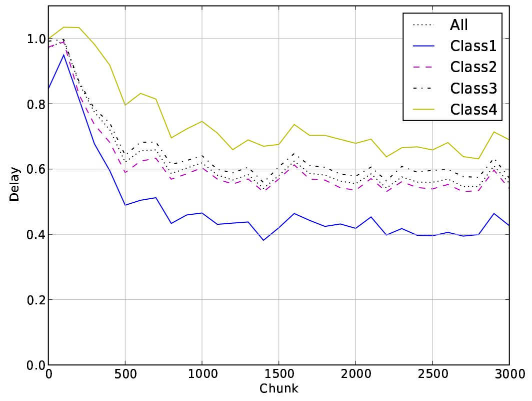


Figure 6: Mean out-degree (on the top) and delay (on the bottom) versus time for the various classes of peer when 50% of peers are churning.

7. ACNOWLEDGEMENTS

This work was funded by the European Commission under the 7th Framework Programme Strep Project "NAPA-WINE" (Network Aware Peer-to-Peer Application over Wise Network).

8. CONCLUSION

We have presented a distributed algorithm for P2P streaming TV that dynamically builds and maintains an efficient overlay using only local information. The algorithm allocates large and small neighbourhoods to high and low bandwidth nodes respectively based on their measured chunk-delivery performance rather than on theoretical bandwidth specifications. The algorithm has been extensively tested on a simulator and has behaved stably and efficiently over a wide range of parameters. Because the algorithm is distributed and operates continuously, it is robust against churning and other performance disruptions like network failures.

9. REFERENCES

[1] L. Massoulie, A. Twigg, C. Gkantsidis, and P. Rodriguez. "Randomized decentralized broadcasting algorithms" *INFOCOM, 2007*, Anchorage, AK, May 2007.

[2] S. Sanghavi, B. Hajek, and L. Massoulie, "Gossiping with multiple messages", *INFOCOM, 2007*, Anchorage, AK, May 2007.

[3] T. Bonald, L. Massoulié, F. Mathieu, D. Perino, A, Twigg, "Epidemic Live Streaming: Optimal Performance Trade-Offs", Sigmetrics 2008, Annapolis, ML, June 2008.

[4] A. P. C.da Silva, E. Leonardi, M. Mellia, M. Meo. "A Bandwidth-Aware Scheduling Strategy for P2P-TV Systems", *8th International Conference on Peer-to-Peer Computing 2008 (P2P'08)*, Aachen, September 8-11, 2008.

[5] Yong Liu, "On the minimum delay peer-to-peer video streaming: how realtime can it be?", *ACM Multimedia 2007*, Augsburg, Germany, September 2007.

[6] F. Picconi and L. Massoulie, "Is there a future for mesh-based live video streaming?", *IEEE P2P 2008*, Aachen, Germany, September 2008.

[7] "Network-Aware P2P-TV Application over Wise Networks ", `http://www.napa-wine.eu`

[8] Béla Bollobás. "Random Graphs", Cambridge

[2]Chunk delay is averaged over all peers that received that chunk, whether they are churning peers or not.

University Press, 2001.

[9] Dongni Ren, Y.T. Hillman Li, S.H. Gary Chan, "On Reducing Mesh Delay for Peer-to-Peer Live Streaming", *IEEE INFOCOM 2008*, Phoenix, AZ, April 2008.

[10] R.J.Lobb, A. P. C.da Silva, E. Leonardi, M. Mellia, M. Meo. "Adaptive overlay Topology for P2P-TV systems", *Technical Report - Polito092802-1. Available on line from* `http://www.tlc.polito.it/mellia/Polito092802-1.pdf`