Scheduling P2P Multimedia Streams: Can We Achieve Performance and Robustness?

Luca Abeni, Csaba Kiraly, Renato Lo Cigno
DISI – University of Trento, Italy
{luca.abeni,csaba.kiraly,renato.locigno}@disi.unitn.it

***Abstract*—Scheduling the transmission of information in P2P applications is one of the main challenges, and one of the keys to success, for these applications. We concentrate on low-latency streaming applications (e.g. TV) and explore different combinations of chunk and peer scheduling strategies, finding that the majority of the proposals found in the literature are not robust to changing conditions. We then propose a new scheduling combination that aims at robustness in face of heterogeneous distribution of peers' available bandwidth and we show, through a comprehensive set of simulated realistic scenarios, that this scheduler outperforms all other combinations in any scenario experimented.**

I. Introduction and background

P2P streaming and P2PTV systems are becoming the dominant multimedia applications in the Internet[1]. One of the challenges related to these systems is the scheduling strategy in mesh-based systems, which are the most widely used, due to their robustness and resilience.

Video streams are normally split in relatively small (a few seconds or less) pieces of information called *chunks*; scheduling refers to the decision (local to a peer) of which chunk will be sent to (or retrieved from) which peer in the system. Scheduling is extremely important because: i) it is one of the main drivers of the system performance; ii) performing, and robust distributed solutions are difficult to find due to the dynamic nature of the overlay topology and due to the tight deadlines typical of real-time systems; and iii) the networking resources required are large so the efficiency is crucial.

We consider unstructured mesh-based overlays (see [1], [2] for examples and comparison with structured systems). A topology management function builds a neighbourhood $\mathcal{N}_i$, used by peer P_i to take its scheduling decisions. We restrict the scope of the paper to push based systems. We focus on the algorithmic choices and decisions regarding the selection of peers and chunks and the effects of coupling different algorithms for chunk and peer selection.

One of the key points is that networks are neither ideal nor homogeneous, so that the choice of the destination peer will affect the performance in transferring the chunk to it, but also how this specific chunk will be diffused in the future: peers more endowed with resources will diffuse the chunk more efficiently than peers with scarce resources. Studies on this topic either focus on asymptotic properties of distributed gossipping algorithms [3], or probabilistic bounds for specific well known algorithms [4], or do not distinguish clearly the scheduling problem from the protocol to exchange the information ([7], [5], [6] just to cite a few recent ones) overlooking both fundamental bounds as a performance comparison means, and the interaction between the application and the network. Recently two works started looking at the problem with a different perspective. In [8] the resources (in terms of bandwidth) of the selected peer were included in the selection procedure, while in [9] we proved the existence of a class of *distributed* algorithms that in ideal networking scenarios achieve optimal distribution delay for a streaming system, thus setting a clear bound against which comparisons can be made in realistic scenarios. The optimal maximum distribution delay F we found in [9] is the well known generic (i.e., valid for any scheduler) lower bound

$$F = (\lceil \log_2(N) \rceil + 1)T \tag{1}$$

where T is the duration of a chunk in time, whose existence was before demonstrated only for centralized schedulers ([10], [11]).

In this paper, we consider the upload bandwidth of peers as their dominant characteristic, i.e., we disregard the impact of the difference in delay between peers and we assume that download bandwidth is much larger than upload bandwidth. We explore and assess the performance of a number of possible chunk and peer scheduling algorithms known in the literature, taking into account their combination as well as different bandwidth distribution models, including the extreme case of large bandwidth peers coupled with "free-riders." Then we propose a novel peer selection algorithm that makes our optimal scheduler proposed in [9] robust to heterogeneity. This scheduler proves to be robust and performs better than any other scheduler in all analysed scenarios.

II. Schedulers Definition and Composition

An unstructured P2P streaming system is modelled as a set $\mathcal{S} = \{P_1, \ldots P_N\}$ of N *peers* P_i, whose aim is receiving a stream from a *source*. The media stream is divided in M_c *chunks*; each peer P_i receives chunks C_j from other peers, and sends them out at a rate $s(P_i)$. The set of chunks already

This work is supported by the European Commission through the NAPA-WINE Project (Network-Aware P2P-TV Application over Wise Network – www.napa-wine.eu), ICT Call 1 FP7-ICT-2007-1, 1.5 Networked Media, grant No. 214412

[1]Sites like http://www.tvants.com, http://live.coolstreaming.us, http://www.zattoo.com, http://pplive.com, or http://www.pp.tv, just to cite a few, gives an idea of the size of the phenomenon and its impact on the Internet, both at the infrastructure level and at the services one.

978-1-4244-4793-0/09/$25.00 ©2009 IEEE

Symbol	Definition
$\mathcal{S}$	Set of all the peers
N	Number of peers in the system
M_c	Number of chunks in the stream
P_i	The i^{th} peer
C_h	The h^{th} chunk
r_h	Time when the source generates C_h
$\mathcal{N}_i$	Neighbourhood of peer P_i
f_h	Diffusion delay of C_h (time needed by C_h to reach all the peers)
$\mathcal{C}(P_i,t)$	Set of chunks owned by P_i at time t
$\mathcal{C}'(P_i,t)$	Set of chunks owned by P_i at time t which are needed by some of P_i's neighbours
$s(P_i)$	Upload bandwidth of P_i

Table I
SYMBOLS USED IN THE PAPER FOR THE MAIN SYSTEM PARAMETERS.

received by P_i at time t is indicated as $\mathcal{C}(P_i,t)$. The most important symbols used in this paper are recalled in Table I.

The source generates chunks in order, at a fixed rate λ and sends them with rate $s(source) = \lambda$, i.e., the source emits only one copy of every chunk. All the bitrates are normalised w.r.t. λ, so that the generation time of C_j is $r_j = j$.

$\mathcal{D}_j(t - r_j)$ is the set of nodes that already received chunk C_j at time t. The worst case diffusion delay f_j of chunk C_j is formally defined as the time needed by C_j to be distributed to every peer: $f_j = \min\{\delta : \mathcal{D}_j(\delta) = \mathcal{S}\}$. A generic peer P_i will receive chunk C_j at time t with $r_j + 1 \leq t \leq r_j + f_j$, hence any P_i is *guaranteed* to receive C_j at most at time $r_j + f_j$. To correctly reproduce the whole media stream, a peer must buffer chunks for a time of at least $F = \max_{1 \leq j \leq M_c}(f_j)$ before starting to play. For this reason, the worst case diffusion delay F is a fundamental performance metric for P2P streaming systems. We will also consider the 90-th percentile over chunks of the diffusion time, which is important, for instance, when FEC or any form of redundant coding is used[2].

Whenever a peer P_i decides to contribute to the streaming and pushes a chunk, it is responsible for selecting the chunk to be sent and the destination peer. These two decisions form the *scheduling logic*, and they are taken by the *chunk scheduler*, and *peer scheduler*. The peer scheduling algorithm can select a target peer $P_k \in \mathcal{N}_i$, where $\mathcal{N}_i$ is the set of its neighbours. The case in which $\forall i, \mathcal{N}_i = \mathcal{S} - P_i$ corresponds to a fully connected graph. Given an `X` chunk scheduler and a `Y` peer scheduler, applying first `X` and then `Y` one obtains a *chunk-first* `X/Y` scheduler; the other way around one obtains a *peer-first* `Y/X` scheduler. In this paper, due to space constraints, we only consider chunk-first compositions.

In the following, we briefly define the scheduling algorithms evaluated in this paper, referring to the relevant literature when appropriate. Since non-blind algorithms (i.e., those that exploit knowledge on the neighbours needs) are known to perform better than blind ones, this paper will only consider non-blind algorithms.

[2]Other percentiles, like 95, 99, or even 80 can be considered, depending on the specific redundancy of the system.

A. Chunk Schedulers

a) Random Useful (RUc): P_i randomly selects a chunk $C_j \in \mathcal{C}'(P_i,t)$ based on the knowledge of chunks required in $\mathcal{N}_i$.

b) Latest Useful (LUc): P_i selects the most recent chunk $C_j \in \mathcal{C}'(P_i,t)$ that is needed by some of the peers in $\mathcal{N}_i$.

c) Deadline-based scheduler (DLc): P_i selects the chunk $C_j \in \mathcal{C}'(P_i,t)$ having the earliest scheduling deadline. Scheduling deadlines are assigned to the various chunks instances present in the system, and each scheduling deadline is postponed by a fixed amount when sending a chunk. This algorithm [9] has been proved to be optimal in full meshes (when used in combination with the right peer scheduler), and to provide good performance when the neighbourhood size is reduced.

B. Peer Schedulers

d) Random Useful Peer (RUp): P_i randomly selects a peer in $\mathcal{N}_i$ that needs the selected chunk (or a generic chunk $C_j \in \mathcal{C}(P_i,t)$ if the peer is selected first).

e) Most Deprived Peer (MDp): P_i selects the peer P_j in $\mathcal{N}_i$ which owns the smallest number of the chunks currently owned by P_i [12].

f) Earliest Latest scheduler (ELp): P_i selects the target peer owning the earliest latest chunk [9]. See Sect. IV-A for a description of the algorithm.

g) Bandwidth Aware scheduler (BA$_W$p): P_i randomly selects a target peer $P_k \in \mathcal{N}_i$ (as in RUp); the probability of selecting P_k is proportional to its output bitrate $s(P_k)$ [8].

Both ELp [9] and MDp [12] have been proved to have some optimality properties, hence in this paper they are considered in all the scheduling performance evaluation because of their formal properties. Besides, [8], [10] showed that, to be effective in an heterogeneous environment, a peer scheduler must consider also the upload bandwidth of the target peers. For this reason bandwidth aware schedulers are also considered in this paper. Finally a proposal to blend the best properties of bandwidth aware heuristics with ELp optimality will prove to inherit the best of both.

III. NETWORK AND BANDWIDTH MODELS

Assessing performances in a heterogeneous network requires a network . . . or a model of it. For our results we use the P2PTVSim simulation tool developed within the Napa-Wine EU project[3]. The simulator allows describing bandwidths and delays between peers and building many overlay topologies.

We restrict the analysis to the case when bandwidths are a property of the single peer and the overlay topology is n-regular[4] or a full mesh. The bandwidth being a property of the peer is typical of cases when the bottleneck is on the access link as in ADSL. We assume perfect knowledge of the

[3]The simulator is available at the project web-site www.napa-wine.eu, scripts and additional information can be obtained from the authors

[4]In n-regular topologies all nodes have the same connectivity degree, which means that the neighbourhood size N_N is equal for all peers.

neighbourhood status, which correspond to situations when the delays are small compared to chunk transmission times. The download bandwidth of a peer P_i is assumed to be much larger than its upload bandwidth $s(P_i)$.

Since the upload bandwidths of peers $P_i \in \mathcal{S}$ are not homogeneous, it is important to model their distribution. We consider two possible bandwidth distributions: *class-based distributions* (such as may arise from ADSL links), and *continuous distributions* (that may arise from user-imposed constraints). In a class-based distribution, the peers are grouped in a finite number of classes, and every class of peers is characterised by a different upload bandwidth. In a continuous distribution, the upload bandwidth of every single peer is randomly assigned according to a specified Probability Density Function (PDF).

Various models of class-based and continuous bandwidth distributions have been tested and simulated. We consider results generated in three different scenarios referred as "*3-class*", "*uniform*", and "*free-riders*". The *3-class* scenario is a class-based distribution with three classes: low-bandwidth peers (having an upload bandwidth equal to $0.5\overline{B}$), mid-bandwidth peers (having an upload bandwidth equal to $\overline{B}$), and high-bandwidth peers having an upload bandwidth equal to $2\overline{B}$. The fraction of high-bandwidth peers in the system is $h/3$ (where h is the *heterogeneity factor* of this scenario), the fraction of low-bandwidth peers is $2h/3$, and the fraction of mid-bandwidth peers is $1-h$; as a result, the average bandwidth is

$$2\overline{B} \cdot h/3 + \overline{B} \cdot (1-h) + 0.5\overline{B} \cdot 2h/3 = \overline{B} \qquad (2)$$

This scenario has been selected because it captures the most important properties of a class-based distribution, and a similar setup has already been used in literature [4].

The *uniform* scenario is an example of continuous distribution, in which $s(P_i)$ is uniformly distributed between a minimum value $B^{min} = (1-\Delta_B)\overline{B}$ and a maximum value $B^{max} = (1+\Delta_B)\overline{B}$, where $\overline{B}$ is the average upload bandwidth and Δ_B is the *bandwidth variability* (a measure of heterogeneity) of this scenario. This scenario is particularly interesting because it allows checking if some of the properties and results noticed in the 3-class scenario are due to artifacts caused by a class-based distribution, or are generic properties.

Finally, the *free-riders* scenario is based on a two-class distribution in which one class of peers is composed by "free riders", having upload bandwidth equal to 0. This scenario is important as it allows understanding what happens when some of the peers for some reasons do not contribute to the chunks diffusion. The average bandwidth $\overline{B}$ is kept constant also in this scenario.

IV. The Quest for a Robust Scheduler

Each of the schedulers presented in Sect. II has been found, in some context or another, to perform well. However, a scheduling logic composed of a peer and a chunk scheduler must be, first of all, robust to different network conditions. We have evaluated all the schedulers under many conditions through extensive simulations: The amount of data produced during such simulations is so large (hundreds of plots!) that it is important to identify the most relevant results, and to focus on them. In particular, it is important to understand:

- Which metric to use for comparing the algorithms;
- What algorithms are the best candidates for further refinement and development.

Often scheduling algorithms are compared using the *average* chunk diffusion time as a metric. Albeit this metric allows to evaluate the long-term stability of a streaming system, and it gives an idea of the average performance, using the average diffusion time to predict the QoS experienced by a user is not easy, because it does not provide enough information about the chunks dropped in the system, or delayed beyond their target playout time. As noticed in Sect. II, the probability to miss a chunk C_j depends on the diffusion time f_j: if f_j is larger than the playout delay, then C_j is missed. If some chunk loss can be tolerated, then the metric to be measured is some percentile. We selected, somewhat arbitrarily, the maximum (F) and 90-th percentile (F_{90}) of the delay as dominant parameters, but with the warning that, to gain insight in schedulers' performance it is often necessary to analyse the entire distribution correlated to the relevant parameter under study, further details on this topic can be found in [13].

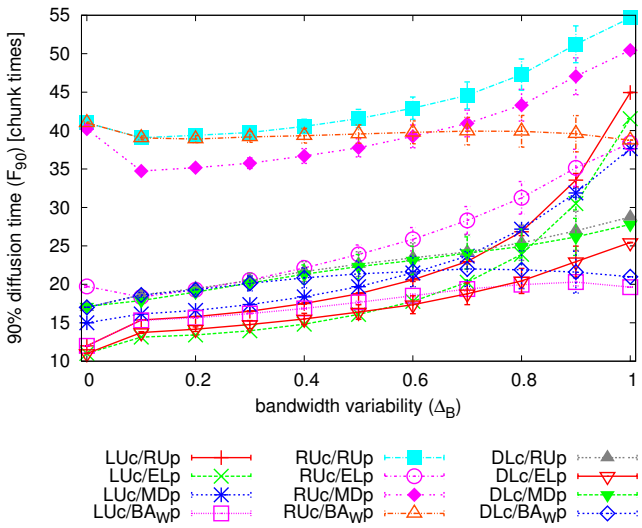

Figure 1. 90-th percentile of delay for selected chunk-first schedulers as a function of the bandwidth variability in the uniform scenario; $\overline{B} = 1$, $N = 600$ peers in full mesh, $M_c = 1200$ chunks.

In order to exemplify the problem of heterogeneity, Fig. 1 reports F_{90} measured in the *uniform* scenario with full mesh connectivity and average bandwidth $\overline{B} = 1$. The plot is function of the bandwidth variability Δ_B, with $\Delta_B = 0$ corresponding to a homogeneous scenario ($B^{min} = B^{max} = 1$) and $\Delta_B = 1$ corresponding to the peers upload bandwidths uniformly distributed in $[B^{min} = 0, B^{max} = 2]$.

Analysing the figure it is clear that LUc and DLc chunk schedulers ensure the best performance (but LUc is fragile reducing the neighbourhood size as shown in [9]), while in

```
function L(P_k, t)
    max = 0
    for all C_h in C(P_k, t) do
        if r_h > max then
            max = r_h
        end if
    end for
    return max
end function
function ELP(P_i, t)
    int min;
    peer target;
    for all P_k in N_i do
        if L(P_k, t) < min then
            min = L(P_k, t);
            target = P_k;
        end if
    end for
    return target
end function
```

Figure 2. The ELp algorithm.

peer scheduling there is no clear winner: bandwidth awareness guarantees good performances in highly non homogeneous scenarios, while ELp performs the best (as it must be) in case of homogeneous networks.

From these initial results it seems that deadline-based schedulers are the most promising to obtain robust and performing scheduling procedures. For this reason, and because of its formal optimality in full meshes, this paper will use DLc as a chunk scheduler and will focus on comparing different peer schedulers. DLc has a parameter, the deadline postponing value, which in the homogeneous case has no influence on the performance. In case of heterogeneous systems instead, it does influence results. Further details on this topic can be found in Technical Reports of the authors available through University of Trento e-prints[5].

A. A Bandwidth-Aware ELp Algorithm

As highlighted in Sect. IV, a peer scheduler, in order to be robust to different bandwidth distribution scenarios, should be both bandwidth aware [8] and select peers that are in the best conditions to redistribute the chunk as ELp does [9].

A first way to integrate ELp with some kind of bandwidth awareness is to use hierarchical scheduling. Two possible hierarchical combinations can be implemented: EL_{BA}p and BA_{EL}p. EL_{BA}p uses ELp to select a set of peers having the earliest latest chunk, and then uses a bandwidth aware strategy to select the peer having the highest output bandwidth among them. On the other hand, BA_{EL}p first uses a bandwidth aware scheduler to select the set of peers having the highest output bandwidths and then applies ELp scheduling to this set.

Although hierarchical scheduling can be very effective in some situations, a better integration of the two scheduling algorithms can improve the system's performance (as shown in Sect. V). To understand how to integrate ELp and bandwidth aware scheduling, it is important to better understand the details of earliest-latest scheduling. Fig. 2 reports the algorithmic definition of ELp.

Let $L(P_i, t)$ be the latest chunk owned by P_i at time t. ELp minimises $L(P_j, t)$ in $\mathcal{N}_i$, and this is equivalent to maximising $t - L(P_j, t)$. The meaning of this rule is that ELp tries to select the target peer P_i having the latest chunk that has been distributed within the overlay for more time (hence, it can be argued that such a chunk will be useful for less time, and the peer can soon pass to distribute another one). However, *the peer status at scheduling time t is not fundamental*, but what matters is the target peer status when the currently scheduled chunk will be received (and hence redistributed). The original ELp algorithm has been developed assuming that all the peers have the same output bandwidth $s(P_j) = 1$, so every chunk was diffused in 1 time unit and the quantity to be maximised was $t + 1 - L(P_j, t)$, which is equivalent to maximising $t - L(P_j, t)$.

When the output bandwidths $s(P_j)$ of the various peers are not uniform, maximising $t - L(P_j, t)$ is not meaningful. When P_i sends a chunk to P_j at time t, this chunk will be received at time $t' = t + 1/s(P_i)$ (measuring bandwidths in chunks per second). Hence, we can maximise

$$[t - L(P_j, t) + B_w(s(P_j)/s(P_i))] \tag{3}$$

where B_w is a *weight* assigned to the upload bandwidth information. Note that the weight B_w enables to realise trade-offs between BA_{EL}p and EL_{BA}p: for $B_w >> 1$ maximizing (3) tends to BA_{EL}p, whereas for $B_w << 1$ the algorithm tends to EL_{BA}p. Finally, note that if all the peers have the same output bandwidth then maximizing (3) is equivalent to ELp regardless of B_w. The peer scheduling algorithm resulting from this heuristic is named BAELp (Bandwidth Aware Earliest Latest peer scheduler).

We found that the algorithm is not very sensitive to B_w (see [13] for details) and that $B_w = 3$ is a robust value for all scenarios.

V. Algorithms Comparison

All results in this Section are averaged over 100 simulation runs (with different topologies generated at random); the error bars indicate the confidence interval with 95% confidence level.

Fig. 3 plots the worst case (F) and the 90-th percentile (F_{90}) of the diffusion delay for the scheduling algorithms under analysis, as a function of the heterogeneity h in a 3-class scenario. The overlay has $N = 600$ peers connected in full mesh and $M_c = 1200$ chunks are distributed. The average upload bandwidth is 1.

First of all, focusing on DLc, F_{90} and F behaviour are very similar (for this reason, in the rest of the paper only F_{90} will be shown). Second, the peer scheduling algorithms can be classified in 3 groups. The first group, including BAELp, EL_{BA}p, and ELp is the one showing consistently the best performance

[5] http://eprints.biblio.unitn.it/

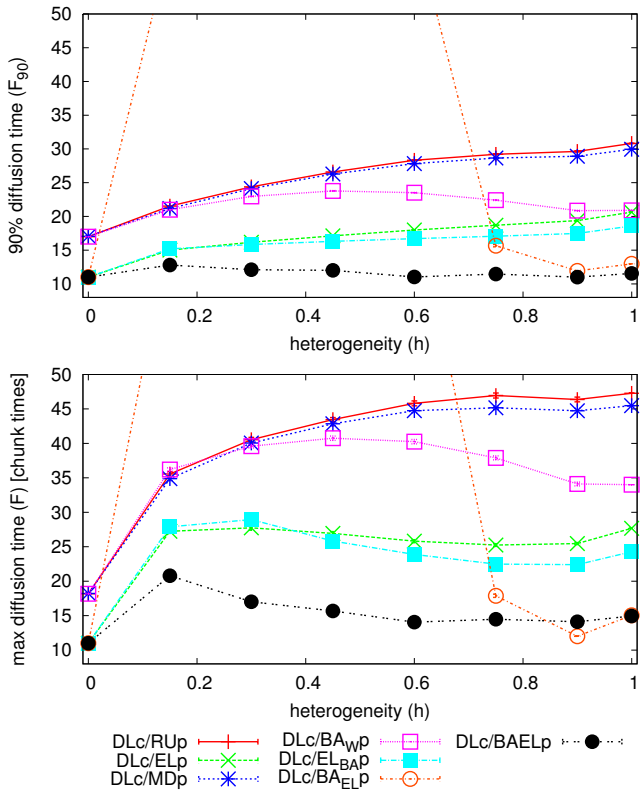

Figure 3. 3-class scenario with full mesh: diffusion delay as a function of the heterogeneity factor h; Upper plot: F_{90}, Lower plot: F; $\overline{B} = 1$, $N = 600$ peers in full mesh, $M_c = 1200$ chunks.

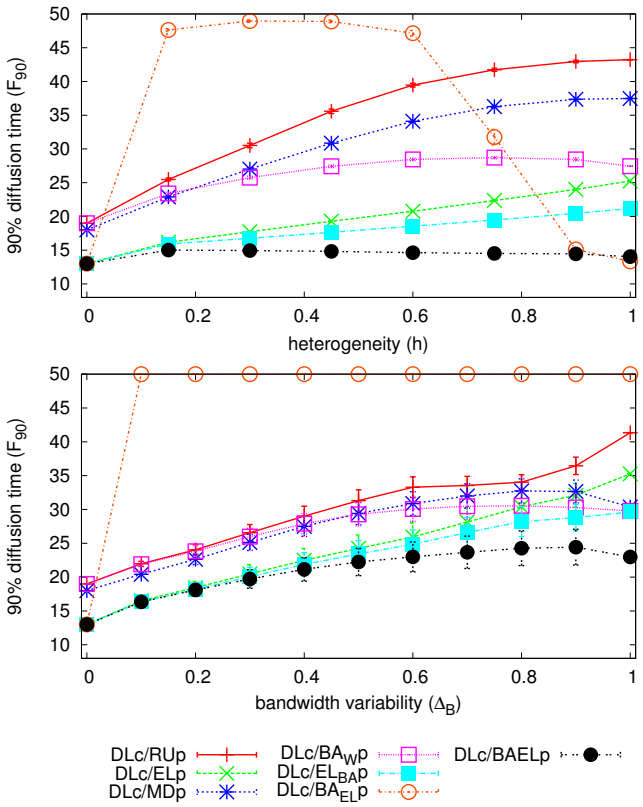

Figure 4. F_{90} as a function of heterogeneity for neighbourhood size 20 and playout delay 50; Upper plot: 3-class, Lower plot: Uniform; $\overline{B} = 1$, $N = 600$ peers, $M_c = 2000$ chunks.

regardless of the inhomogeneity factor. Schedulers of the second group (MDp, RUp, and BA$_W$p) perform identically to each other when the peers are homogeneous ($h = 0$), however BA$_W$p handles heterogeneity better than the other two. The third group, finally, which only contains BA$_{EL}$p, provides reasonable performance only for homogeneous system and very inhomogeneous ones.

Building and maintaining a full mesh overlay is not realistic as the number of peers increases; moreover, the playout delay is finite and small, limiting the maximum delay a chunk can have before being discarded. As an example of a more realistic scenario, we selected a neighbourhood size equal to 20 and a playout delay of 50 (other numbers would not change the meaning and quality of the results).

Results are shown in Fig. 4, which reports F_{90} for the 3-class (upper plot) and uniform (lower plot) scenarios. In both plots BAELp outperforms all other algorithms. BA$_{EL}$p shows an F_{90} equal to the playout delay, which means chunks are missed, and the quality will be poor.

Observing both Figs. 3 and 4 two considerations are in order. First of all, reducing the neighbourhood size tends to increase the gap between the first and the second group of schedulers. Second, in the uniform scenario, while the order of schedulers is unchanged, the difference decreases. This second result is due to a smaller actual heterogeneity of the uniform scenario w.r.t. the 3-class one.

In the next set of experiments, the stringent (and somewhat arbitrary) $\overline{B} = 1$ assumption has been removed, comparing the behaviour of the schedulers when the average upload bandwidth is increased. Fig. 5 reports F_{90} as a function of the average bandwidth $\overline{B}$. The neighbourhood size is 20 and the playout delay is 50. The upper plot refers to the 3-class scenario with $h = 0.5$, and the lower plot to the uniform scenario with $\Delta_B = 0.8$. Again, BAELp is the best performing algorithm (clearly, when the average bandwidth increases too much the differences between the various algorithms vanish).

Finally, we explore the ability to cope with peers that do not participate to the stream distribution. In this "free-riders" scenario the neighbourhood size must be increased, otherwise the system results fragile when too many peers do not cooperate: we set it to 100 peers. Fig. 6 reports F_{90} as a function of the fraction of peers that do not redistribute chunks (free riders, having upload bandwidth equal to 0). Note that curves relative to the ELp, RUp, MDp, and EL$_{BA}$p schedulers stop between 15% and 20% of free riders, because with higher fractions of free riders such schedulers are not able to properly diffuse the stream. Again, this experiment shows that BAELp is the most robust scheduler in face of changing conditions (more simulations in the free-riders scenario have been performed, and they confirmed the results of this experiment).

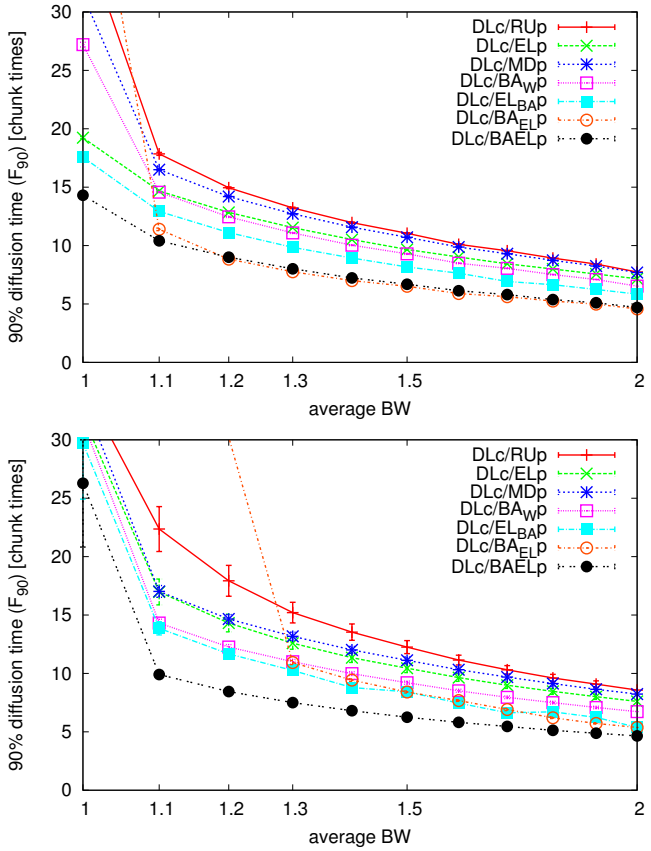

Figure 5. F_{90} as a function of $\overline{B}$ for neighbourhood size 20 and playout delay 50; Upper plot: 3-class with $h = 0.5$, Lower plot: Uniform with $\Delta_B = 0.8$; $N = 1000$ peers, $M_c = 2000$ chunks.

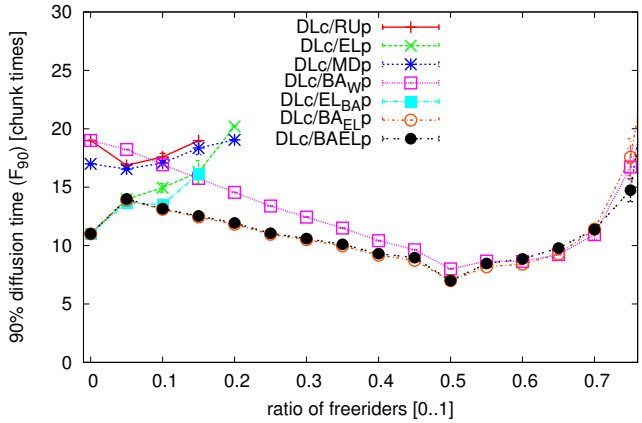

Figure 6. Free Riders scenario, $\overline{B} = 1$, neighbourhood size 100 and playout delay 50: F_{90} versus the fraction of the free riders. $\overline{B} = 1$, $N = 1000$ peers, $M_c = 2000$ chunks.

VI. Conclusions and Future Work

This paper compared a large number of scheduling strategies for P2P streaming systems in the presence of network heterogeneity (peers having different upload bandwidths) and considering different conditions and scenarios, highlighting the fragility of all of them in light of different scenarios and conditions variability. A new peer scheduling algorithm, named BAELp, has been developed and compared with the other ones.

BAELp outperforms all the other scheduling algorithms in a large number of different conditions and scenarios (in all the ones that have been tested): for example, in homogeneous networks BAELp is equivalent to ELp, which is optimal, and in highly-heterogeneous networks it performs better than other bandwidth aware schedulers. This peer scheduler is also able to cope with the presence of free riders, namely peers that do not contribute to uploading chunks and sustaining the stream.

As a future work, a more formal analysis of the BAELp algorithm will be performed, and some of its theoretical properties will be analysed. Moreover, when using a DLc/BAELp scheduler there are two parameters that can be tuned: the amount of deadline postponing for DLc and the bandwidth weight B_w for BAELp. In this paper, both values have been empirically tuned to reasonable values, but a more detailed analysis of the effects of the two parameters is needed. In particular, the optimal values might depend on the network conditions; in this case, some kind of self-adaptation of the scheduling parameters (based on a feedback loop) should be developed.

References

[1] M. Castro, P. Druschel, A.-M. Kermarrec, A. Nandi, A. Rowstron, and A. Singh, "Splitstream: high-bandwidth multicast in cooperative environments," in *SOSP 03*. New York, NY, USA, 2003.

[2] Y. H. Chu, S. G. Rao, and H. Zhang, "A case for end system multicast," *IEEE JSAC, Special Issue on Networking Support for Multicast*, vol. 20, no. 8, 2002.

[3] S. Sanghavi, B. Hajek, and L. Massoulié, "Gossiping with multiple messages," in *IEEE INFOCOM 2007*, Anchorage, Alaska, USA, May 2007.

[4] T. Bonald, L. Massoulié, F. Mathieu, D. Perino, and A. Twigg, "Epidemic live streaming: optimal performance trade-offs." in *ACM SIGMETRICS 2008*, Annapolis, Maryland, USA, June 2008.

[5] M. Zhang, Q. Zhang, L. Sun, and S. Yang, "Understanding the power of pull-based streaming protocol: Can we do better?" in *IEEE JSAC, special issue on Advances in Peer-to-Peer Streaming Systems*, vol. 25, no. 10, pp. 16401654, Dec. 2007.

[6] X. Zhang, J. Liu, and T. shing Peter Yum, "Coolstreaming/donet: A data-driven overlay network for peer-to-peer live media streaming," in *in IEEE INFOCOM 2005*, Miami, FL, USA, March 2005.

[7] D. Ren, Y. T. H. Li, and S. H. G. Chan, "On reducing mesh delay for peer-to-peer live streaming," in *IEEE INFOCOM 2008*, Phoenix, AZ, USA, April 2008.

[8] A. Couto da Silva, E. Leonardi, M. Mellia, and M. Meo, "A bandwidth-aware scheduling strategy for P2P-TV systems," in *P2P'08*, Aachen, Germany, September 2008.

[9] L. Abeni, C. Kiraly, and R. Lo Cigno, "On the optimal scheduling of streaming applications in unstructured meshes," in *IFIP Networking*, Aachen, Germany, May 2009.

[10] Y. Liu, "On the minimum delay peer-to-peer video streaming: how realtime can it be?" in *ACM MULTIMEDIA '07*. Augsburg, Germany, September 2007.

[11] J. Mundinger, R. Weber, and G. Weiss, "Optimal scheduling of peer-to-peer file dissemination," *J. of Scheduling*, vol. 11, no. 2, pp. 105–120, 2008.

[12] L. Massoulié, A. Twigg, C. Gkantsidis, and P. Rodriguez, "Randomized decentralized broadcasting algorithms," in *IEEE INFOCOM 2007*, Anchorage, Alaska, USA, May 2007.

[13] L. Abeni, C. Kiraly, and R. Lo Cigno, "Achieving performance and robustness in P2P streaming systems," University or Trento, Tech. Rep. TR-DISI-09-041, 2009. [Online]. Available: http://disi.unitn.it/locigno/preprints/TR-DISI-09-041.pdf