Efficient Uplink Bandwidth Utilization in P2P-TV Streaming Systems

This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE Globecom 2010 proceedings.

Efficient Uplink Bandwidth Utilization in P2P-TV Streaming Systems

Alessandra Carta, Marco Mellia, Michela Meo, Stefano Traverso
Politecnico di Torino, Italy – email: {firstname.lastname}@tlc.polito.it

***Abstract*—Peer-to-Peer streaming systems (or P2P-TV) have been studied in the literature for some time, and they are becoming popular among users as well. P2P-TV systems target the real time delivery of a video stream, therefore posing different challenges compared to more traditional P2P applications like the better known file sharing P2P application. In this paper, we focus on mesh based systems in which the peers form a generic overlay topology upon which peers exchange small "chunks" of video. In particular, we study the signaling mechanisms that must be in place to trade chunks and to match the demand from other peers in a quick and efficient way, by automatically adapting a peer service rate to its upload capacity. The goal is to maximize peer upload capacity utilization, while avoiding forming long transmission queue, therefore minimizing the chunk delivery time, a crucial parameter for P2P-TV systems. Our results show that the proposed solution achieves several desirable goals: i) it limits the overhead due to signaling messages, ii) it achieves a fair resource utilization, making peers contribute proportionally to their bandwidth, iii) it improves system performance, reducing loss probability and chunk delivery delay with respect to mechanisms with non adaptive number of contacted peers.**

I. Introduction

Mesh-based P2P streaming systems (P2P-TV) are among the most promising solutions for broadcasting real time video contents over the Internet [1]. They offer to content providers and broadcasters the opportunity of reaching a potentially unlimited audience without the necessity of expensive infrastructural investments. In mesh-based P2P streaming systems, the video content encoded in real time at the source is sliced in small pieces called *chunks*, which are distributed over a meshed overlay topology exploiting a fully distributed epidemic approach. Chunks should be received by the peers within a deadline from the instant of time they were generated, so that delivery delay is one of key aspects of these systems. There is a substantial difference between P2P systems for file-sharing and for streaming: the last ones have to guarantee real-time-like constraints, while delivering an almost constant bit rate stream of information. File sharing P2P systems have been engineered to maximize the download throughput, i.e., to minimize the download time of the overall content. In P2P-TV systems, on the contrary, the download rate is dictated by the video rate, which is limited by definition. Chunks are emitted by the source in real time, and must be delivered to all peers, minimizing the chunk delivery delay and losses.

The core of chunk distribution algorithms is the chunk scheduling policy, according to which the peers choose which chunks should be delivered to which peers. In the literature, there are two families of algorithms for practically implementing the chosen policy. The *push* based algorithms organize the peers in distribution trees which are rather static and over which a number of consecutive chunks are delivered; in *pull* based approaches, peers are organized in a generic overlay topology and a preliminary trading phase is required during which, before the actual chunk delivery, a peer advertises to some of its neighbors which chunks it possesses and the neighbors, in their turn, select the desired chunks. By avoiding the trading phase, push based algorithms typically achieve smaller chunk delivery delay than pull based approaches. The drawback is the higher complexity to manage the trees and a lower robustness to churning, which limits their scalability in terms of number of peers. Conversely, in pull based algorithms a careful design of the trading phase is needed to avoid that the additional signaling delay translates into an excessive cost to pay for better resource usage and resilience to churning.

This paper focuses on the design of the trading phase, which, to the best of our knowledge, has never been systematically studied in the literature. Each peer advertises to a subset of its neighbors the set of chunks it possesses through an *offer* message. Neighboring peers reply to it with a *select* message in which they specify the subset of chunks they are interested in. The transmitter then schedules the transmission of the selected chunks using a FIFO queue, from which chunks are served one after the other, since transmitting chunks in sequential order reduces the chunk delivery time with respect to parallel transmissions [2]. Finally, successfully received chunks are acknowledged to transmitters through an *ACK* message.

This pull mechanism requires a number of parameters to be tuned to reach optimal results, and the optimal setup, in its turn, depends on the specific scenario, which is typically highly variable due to the natural network variability and user heterogeneity.

In addition, video chunks must be small, e.g., less than 8 packets, to minimize the packetization delay at the source, the transmission delay on the network, and the store-and-forward delay at the peers. To avoid both the burdening of handling TCP, and unnecessary delay due to TCP retransmission and congestion control, UDP is typically preferred by actual P2P-TV application [3]. This poses the problem of how to handle the congestion control, and in particular, how to limit the amount of information a peer can transmit, since its download rate is in all cases limited by the stream rate. Controlling therefore the uplink bandwidth utilization is a key problem, which has been so far ignored by the research community.

Several proposals and actual implementations adopt pull

978-1-4244-5638-3/10/$26.00 ©2010 IEEE

based mechanisms (see [2], [4], [5], [6] for example), but, to the best of our knowledge, no discussion and tuning of the signaling mechanisms have ever been carried on. In this paper, we propose a scheme to automatically adapt the choice of: i) the frequency with which a peer advertises the chunks it possesses, ii) the number of peers to which the chunks are advertised. These two aspects are particularly critical since an inadequate setting can translate into performance degradation, due to excessive signaling overhead, waste of resources or queuing delay at the transmitting peer. We thus propose a solution to adapt the above mentioned parameters i) to the video rate and ii) to the upload capacity of the peers with the objective of jointly maximizing the exploitation of the peers' upload capacity and reducing chunk delivery delay and losses, i.e., carefully controlling the bandwidth allocation on the uplink channel of a peer. The proposed algorithm is extensively tested by simulations; the results show that, with respect to non adaptive mechanisms, it can consistently improve system performance in terms of chunk loss, delivery delay, and thus service quality. Furthermore, peers' upload capacity is used in such a way that system demand rate is satisfied in a fairer fashion, avoiding stressing low capacity peers, and avoiding concentrating the download from high capacity peers only.

The proposed algorithm is being implemented within the new P2P-TV application under development within the EU-FP7 NAPA-WINE STREP project [7].

II. System Description

We consider a system in which a source segments the video stream into chunks and injects them in the overlay network. Let $\mathcal{N}$ be the set of peers composing the overlay, with cardinality N. The application must deliver every generated chunk within a deadline starting from the instant in which it is emitted by the source; this deadline is called *playout delay*, D_{max}. If the chunk age is greater then the playout delay, the chunk cannot be traded anymore, as in a sliding window mechanism.

Chunks are transmitted by peers to their neighbors, i.e., they exchange chunks in a swarm-like fashion; the overlay topology is defined by the set of peers and virtual links connecting them. Let $\mathcal{C}_p$ be the set of p neighbors. The overlay topology changes its structure dynamically due to the churning and the possibly dynamic algorithms driving its maintenance and optimization [8]. Since the overlay dynamics are usually much slower than chunk distribution timings (minutes versus tens/hundreds of ms), we are going to neglect churning effects. The overlay can be built by assigning a certain set of neighbors to every peer p. Since the actual design of the overlay topology is out of the scope of this paper, we consider the simplest case in which the overlay network is built once and on a random basis (as we did in [9]).

As normally assumed in the literature of P2P-TV systems, we consider a case where peer's uplink capacity represents the bottleneck to system performance, and consider the chunk delivery loss as main performance indexes (this includes losses and chunks arrived after the playout deadline). In addition, we consider each peer uplink bandwidth utilization as an important index, which allows us to gauge the fairness and efficiency in allocating system upload capacity.

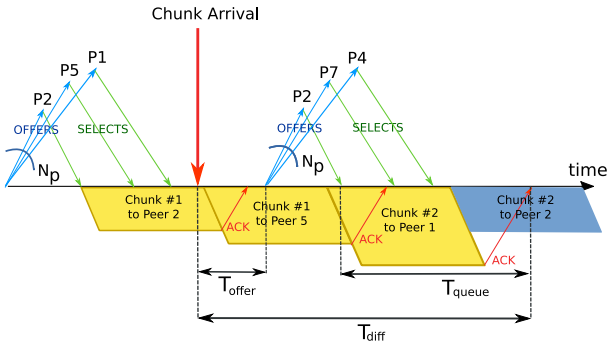

Figure 1. Schematic representation of the peer chunk trading mechanism.

The signaling mechanism used to trade chunk is a *pull* mechanism similar to the one used in other mesh-based P2P-TV systems, [2], [4], [5], [6]. A chunk is sent from a peer to one of its neighbors after a trading phase which is sketched in Fig. 1. In the figure, trading messages are represented above the time line and chunk transmissions are below it. The negotiation begins on the transmitter side: peer p periodically chooses a subset of its neighbors $\mathcal{N}_p$ (with $|\mathcal{N}_p| = N_p$) and sends them a special signaling message, called an *offer* message, containing the set of chunks it possesses and whose age is smaller than D_{max}. Every peer in $\mathcal{N}_p$ replies to the offer with a *select* message in which it indicates a subset of at most M desired chunks. Once a chunk has been "selected", the receiver will set it as *pending* to avoid requesting the same chunk from different peers at the same time[1].

As soon as p receives some *positive select* messages[2], it schedules the transmission of all requested chunks, maintaining a *transmission queue* of chunks to be sent that is served in a FIFO order. Peer p is committed to send all the chunks requested in all the received select messages. Chunks are small, to minimize the transmission delay and to quickly spread them through the overlay via the store-and-forward mechanism typical of the P2P systems.

Several design choices impact the performance of the pull mechanism: 1) the criterion to select peers belonging to $\mathcal{N}_p$ – known as the "peer selection"; 2) the strategy according to which peers in $\mathcal{N}_p$ select chunks to download – known as the "chunk selection"; 3) the frequency at which a peer p offers chunks to its neighbors; and 4) the values of the parameters M and N_p.

Since the objective of our study is to discuss the last two issues, for the *peer selection* and the *chunk selection* policies we make the simplest possible choices: peer p chooses the neighbors to contact uniformly at random within the set of its neighbors, and the neighbors choose the chunks to select at

[1]Note that *pending* chunks can not be published in *offer* messages yet.

[2]Positive means that at least one chunk was requested in the select message. In this paper we assume signaling messages are reliably delivered, e.g., an ARQ mechanism is present.

978-1-4244-5638-3/10/$26.00 ©2010 IEEE

random among the ones it needs. This policy is also known in the literature as "Random Peer - Random Useful Chunk selection" [10].

To keep the chunk delivery delay as low as possible, the length in chunks of the transmission queue must be kept as small as possible; this suggests to: i) set $M = 1$ to avoid filling the transmission queue with many chunks directed to the same neighbor, and ii) issue a new offer based on number of chunks waiting to be transmitted.

In next section we describe the algorithm, called ASP (Adaptive Signaling Protocol), we propose to automatically set N_p and decide the schedule of the offer messages.

III. The Adaptive Signaling Protocol

Consider a traditional sliding window algorithm adopted to perform congestion control in a end-to-end connection. It is well known that the transmitter window size has to be correctly set to match the actual available bandwidth and RTT. In our P2P-TV peer design, we have to decide the amount of information a peer can transmit to exploit its upload capacity. N_p is equivalent to the transmitter window, measured in chunks, which has to be correctly tuned to match peer p upload capacity, the actual system demand, and the RTT experienced between p and its neighbors. Differently from traditional congestion control algorithms, the overall system upload capacity has to be allocated to match the total download demand rate, since each peer has to contribute to the video distribution and each peer has to download an average amount of information equal to the video rate. Therefore, N_p determines also the bandwidth allocation among peers in the system.

Selecting N_p is not easy. If N_p is too small, p upload bandwidth risks not to be exploited at best: the transmission queue empties quickly, causing long periods of inactivity (*idle times*); this can reduce system performance especially for high upload capacity peers. If, instead, N_p is too large, p transmission queue fills up, causing additional chunk delivery delay and, possibly, losses due to late delivery of chunks. Moreover, a lot of signaling overhead is produced. Thus, N_p must be adapted to the upload capacity of each peer, the average RTT, and the actual system demand rate. Starting from a default value, each peer modifies N_p according to the following algorithm:

```
if (Tdiff >= 2AvgRTT)
    Np--;
else if (PosSelectNum/OfferNum >= CR)
    Np++;
```

where

- T_{diff} is the time between a new chunk arrival and the moment in which the transmission queue becomes empty;
- $AvgRTT$ is the round trip time averaged among all peer's neighbors;
- $PosSelectNum$ and $OfferNum$ are respectively the number of received positive select messages and the number of offered messages sent for a given offer/select phase.
- CR is the *clipping ratio* that limits the growth of N_p when the number of positive select messages is small.

The algorithm is run every time a new chunk arrives and only once per trading phase. N_p is thus updated just before sending the offer messages. The algorithm aims at jointly using the available upload bandwidth and maintaining the queue as short as possible. If the transmission queue grows too long, the peer reduces the number of offer messages it sends. On the contrary, if the queue is too short (possible idle times and unused bandwidth), the number of neighbors to contact is increased. The decision is based on T_{diff} that, as sketched in Fig. 1; it represents the queue residual busy time at the chunk arrival. The optimal design should lead to have T_{diff} equal to twice the average RTT, so that by sending the offer messages a time equal to $2AvgRTT$ before the last chunk ACK message is received, the bandwidth results continuously utilized and the queue delay minimized[3], i.e., the queue residual time when the selects are returned (T_{queue} in the figure) tends to zero.

If at the chunk arrival the queue residual busy time T_{diff} is too small, say smaller than $2AvgRTT$, the algorithm can foreseen some idle time for the peer; thus, N_p can increase. However, when the peer is slow in distributing chunks or far away from the source, it tends to receive negative selects from neighbors and possibly to stay idle for long time. To avoid flooding the neighbors with an excessive number of useless signaling messages, N_p is increased only if the fraction of positive select messages is larger than the threshold CR.

IV. Performance Evaluation

A. Simulation scenario and assumptions

All results shown in this paper have been obtained through *P2PTV-sim*[4], an open source event driven simulator developed within the NAPA-WINE project. In our scenario, peers are partitioned in four classes based on their upload capacity:

- 15% of peers are in Class 1 with upload bandwidth equal to 5Mb/s $\pm$ 10%,
- 35% of peers are in Class 2 with upload bandwidth equal to 1Mb/s $\pm$ 10%,
- 30% of peers are in Class 3 with upload bandwidth equal to 0.64Mb/s $\pm$ 10%,
- 20% of peers are in Class 4 with negligible upload bandwidth.

The video source belongs to Class 1 peers. The corresponding average bandwidth is $E[B_p] = 1.3$Mb/s. To study the system under several values of network load we change the video rate r_s, so that the load is

$$\rho = r_s / E[B_p].$$

[3]Twice the minimum RTT would be enough to guarantee that a select message is received before the transmission queue empties. However, due to the variability of RTT and the randomness of peer selection process, using the average RTT is a safer choice.

[4]P2PTV-sim is available at http://www.napa-wine.eu/cgi-bin/twiki/view/Public/P2PTVSim.

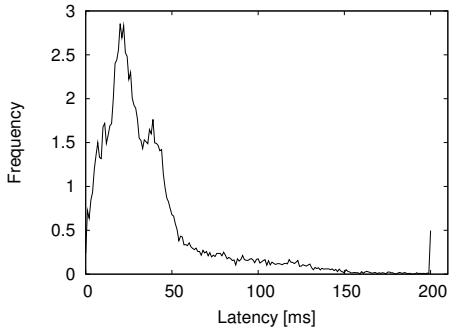

Figure 2. Latency distribution taken from Meridian Project [11].

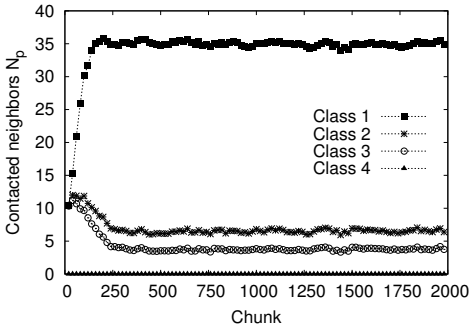

Figure 3. N_p evolution versus time with APS with $CR = 0.0$ and $\rho = 0.9$.

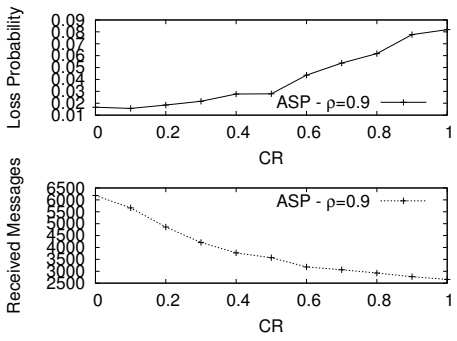

Figure 4. Chunk loss probability (top) and total number of signaling messages per peer versus CR (bottom) at $\rho = 0.9$.

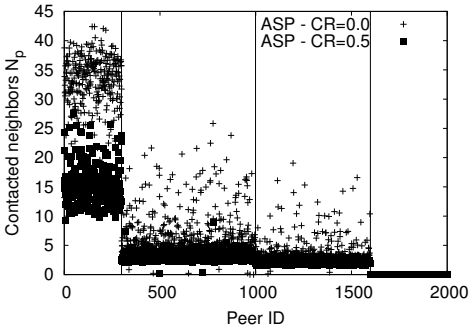

Figure 5. Average number of contacted neighbors N_p with ASP algorithms for all the peers, that are in decreasing order with their bandwidth at $\rho = 0.9$.

Chunk size is fixed and equal to $L = 100$kb, i.e., about 8 UDP packets (with typical 1500B size), while the inter-chunk time depends on the video rate. In each simulation the source emits 2000 chunks, which are equivalent to a video of about 4min at $r_s = 0.8$Mb/s. D_{max} is set to 7s. We consider $N = 2000$ peers. According to the assumption that the bottleneck is at the peer upload link, the model of the network end-to-end path is almost transparent: it is simply modeled by a delay l_{pq} that is added to the transmission time of all the packets flowing from p to q. End-to-end latencies l_{pq} are taken from the experimental data set of the Meridian project [11]. Latency frequencies are reported in Fig. 2, in which values of $l_{pq} \geq 200$ms are accumulated in the last bin for simplicity; the overall mean latency is $E[l_{pq}] = 39$ms.

The overlay topology is randomly generated at the beginning of a simulation by letting each peer randomly select $K = 20$ other peers as its neighbors. Since connections are bidirectional, the average number of neighbors for a peer is equal to $2K$. The topology is static for the whole simulation run (as already mentioned, since we simulate a few minutes of the system behavior, we neglect the effect of churning). All results presented below (except for the time evolution) are obtained averaging the results of four random topologies; when different systems are compared, they use the same four topologies.

B. ASP transient analysis

We first show the evolution of N_p with time. In Fig. 3 ($\rho = 0.9$ and $CR = 0.0$) N_p is averaged over all peers in the same class, considering time windows of 20 chunks (that corresponds to 1.7 s). Starting from the initial $N_p = 10$ for all the peers, the setting of N_p quickly converges (and then remains stable) to the proper value that depends on p upload bandwidth. Clearly, peers with negligible uplink capacity (class 4) do not generate any offer message.

Then, we analyze the impact of the clipping ratio on the performance of the ASP algorithm. We set the load to $\rho = 0.9$, i.e., video rate $r_s = 1.1$Mbps, and we plot the loss probability and the average number of signaling messages sent per peer in Fig. 4. The curves show that there is a trade-off between losses and signaling overhead. As CR increases, loss probability increases but the signaling overhead decreases; indeed, due to the epidemic and random chunk diffusion process, number of positive selects decreases by reducing the number of peers that are contacted. To achieve low loss probability, many peers should then be contacted, i.e., many messages should be sent, clearly increasing the signaling overhead. In the following, we will consider two cases: no clipping, $CR = 0$, and $CR = 0.5$, that seems a reasonable trade-off between signaling overhead and loss probability.

Fig. 5 reports the average number of contacted neighbors in every offer session ($\rho = 0.9$). Notice that peers are clusterized in the four different classes in decreasing order with their uplink capacity. ASP nicely adapts N_p to the peer upload bandwidth, and the number of contacted peers is roughly proportional to the peer upload bandwidth, The variability of N_p within the class is due to the different position of the peers in the overlay topology: peers that are close to the source

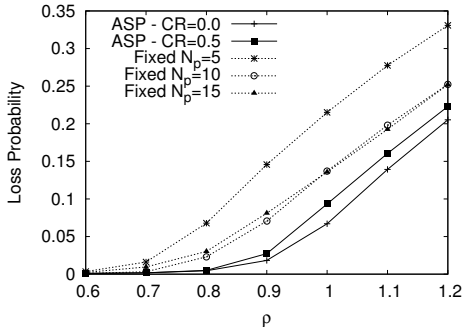

Figure 6. Chunk loss probability versus load.

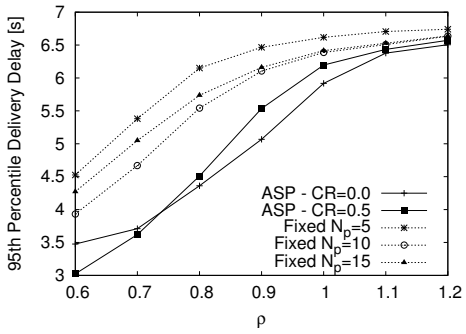

Figure 7. 95th percentile of chunks delivery delay versus load.

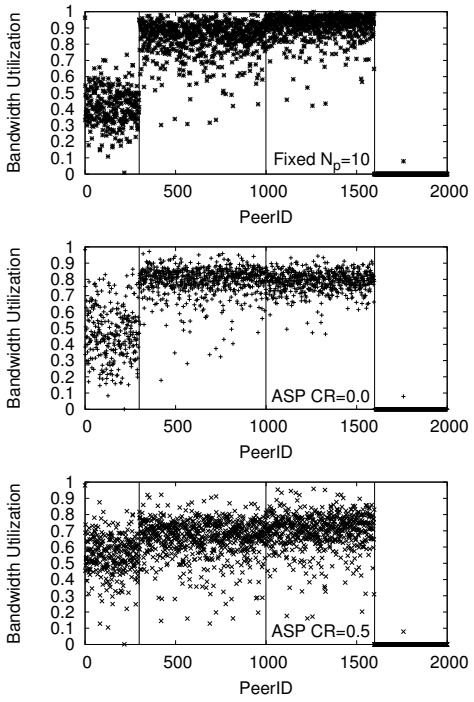

Figure 8. Bandwidth utilization with fixed $N_p = 10$ (top), ASP with $CR = 0$ (middle), ASP with $CR = 0.5$ (bottom) at $\rho = 0.6$.

Table I
AVERAGE BANDWIDTH UTILIZATION FOR CLASSES 1, 2, 3 AND JAIN FAIRNESS INDEX AT $\rho = 0.6$.

	Class 1	Class 2	Class 3	Fairness Index
N_p = 10	0.412	0.836	0.904	0.934
ASP (CR = 0.0)	0.464	0.792	0.785	0.951
ASP (CR = 0.5)	0.550	0.657	0.695	0.962

tend to have a large number of positive selects from their neighbors and they can effectively exploit their bandwidth by only emitting a limited number of offer messages (N_p is small); on the contrary, peers that are far away from the source end up emitting a large number of offer messages (N_p is large). Squared markers refer to a scenario in which the clipping ratio, CR, is set to 0.5, while crosses indicate no clipping ratio. The absence of clipping ratio makes peers that are far from the source pointlessly increase N_p to very large values.

C. Performance analysis and comparison with fixed N_p schemes

We now consider the performance of ASP with respect to schemes in which N_p is fixed, so that every peer always generates the same amount of offer messages, independently on its upload capacity and status of the transmission queue. Fig. 6 reports loss probability versus load for the case of ASP with $CR = 0$ or $CR = 0.5$ (solid lines) and N_p fixed to 5, 10, or 15 (dashed lines). Loss probability is averaged over all chunks and all peers. Observe that by guaranteeing a better utilization of the bandwidth, ASP always achieves better performance than the scheme with fixed N_p, e.g., losses are reduced by a factor up to 4 for $\rho > 0.9$. Moreover, improvements are equal to all classes of peers, so that loss probabilities are the same for all classes. Interestingly, ASP outperforms also the system with $N_p = 15$ that corresponds to the value achieved by high bandwidth peers under the ASP (as can be observed by Fig. 5). The reason is that $N_p = 15$ is too large a value for low bandwidth peers that end up transmitting lots of chunks, but introducing additional queuing delays to the chunk delivery time. Conversely, when N_p is fixed and equal to 5, peers cannot fully exploit their bandwidth, and this explains the higher loss probability. Same considerations can be achieved from Fig. 7 where the 95th percentile of delivery delays of chunks is reported. Again ASP with $CR = 0$ or $CR = 0.5$ show lower delivery delays and, therefore, better performance than schemes with fixed N_p. An interesting fact to point out is that a larger clipping ratio can actually help in reducing delays when the system is under loaded ($\rho < 0.7$. Indeed the $CR = 0.5$ curve exhibits smaller delivery delays respect to $CR = 0.0$ one if $\rho < 0.75$. This is due to the fairer peer uplink capacity utilization as discussed in the following.

D. Bandwidth allocation among peers

Fig. 8 reports the bandwidth utilization per peer measured as the fraction of time the uplink channel is used to transmit chunks; average values per class and Jain's fairness index are reported in Tab. I. We consider a scenario in which the video rate is 0.7Mbps, corresponding to $\rho = 0.6$, meaning that each peer can contribute to the chunk distribution by spending only 60% of its upload capacity. Top plot refers to the case of fix $N_p = 10$. High bandwidth peers have a low

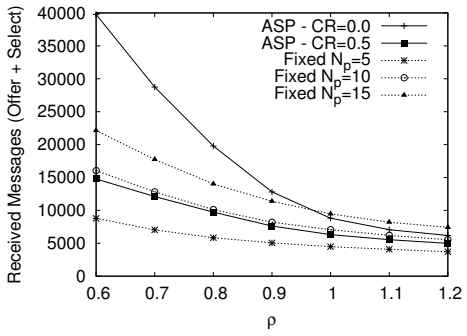

Figure 9. Total number of received signaling messages per peer versus load.

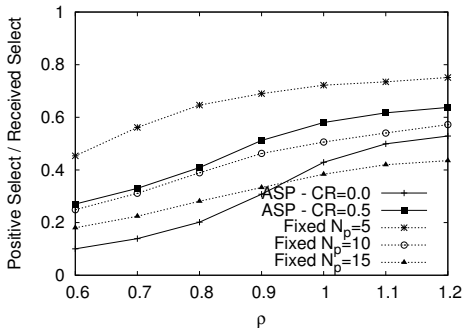

Figure 10. Fraction of positive select messages versus load.

utilization of about 40% of their bandwidth, meaning that they are basically underutilized due to the low number of contacted neighbors (N_P is low). Class 2 and 3 peers compensate by working most of the time, so that they experience utilization higher than 80%. While this bandwidth allocation still allows to successfully deliver all chunks to all peers (no losses are experienced), the overall delay is quite large, due to the slow store-and-forward at low bandwidth peers.

The ASP scheme is fairer in distributing workload among peers proportionally to their bandwidth, as can be observed by middle and bottom plots, respectively referring to $CR = 0$ and $CR = 0.5$. In this case, the high bandwidth peers automatically increase the number of offer messages, trying to increase the uplink bandwidth utilization. Notice the beneficial impact of the $CR = 0.5$, which, by limiting N_p, reduces the utilization of the low bandwidth peers. This speeds up the chunk delivery time, as already noted in Fig. 7.

E. Signaling overhead

Let us now focus on the signaling overhead. Fig. 9 reports the average number of exchanged signaling messages per peer. The signaling overhead decreases with the load due to the larger probability of positive selects per offer message, as confirmed also by Fig. 10 which reports the fraction of positive select messages versus all select messages. The case of fixed $N_p = 5$ leads to the smallest number of signaling messages. However, $N_p = 5$ leads also to the worst performance (see Fig. 6). Improvements can be achieved for higher values of N_p and for the ASP algorithm. When the load is low, queues are short, chunks are distributed quickly, and peers get them easily, so that most of select messages are negative; when no clipping is used, $CR = 0$, the number of signaling messages is very high and the fraction of positive selects is low. If, instead, clipping is active (ASP with CR =0.5), the mechanism is much more efficient and the waste due to signaling reduces by a factor 3.

V. Conclusions

In this paper, we focused on the trading phase of pull-based P2P-TV systems. While a large amount of messages exchanged during the trading phase causes transmission queues to grow, increasing chunks delivery delays, losses and wasted time and resources, a strong limitation to the number of these signaling messages leads to bad performance in terms of losses. To find the best trade-off, we proposed a distributed algorithm to determine when a peer must start publishing its content and the number of neighbors it must contact; the trade-off is decided on the basis of the peer upload bandwidth and status of the transmission queue. Our results prove that the proposed algorithm actually reduces the amount of signaling overhead, introduces a fair and efficient upload bandwidth utilization in heterogeneous scenarios, and improved system performance in terms of delay and losses.

We are currently implementing the proposed mechanism in the NAPA-WINE client, solving a number of implementation issues, such as how to estimate the average RTT and the T_{diff}.

Acknowledgment

This work was supported by the European Commission under the FP7 STREP Project Network-Aware P2P-TV Application over Wise Networks (NAPA-WINE).

References

[1] J. Liu, S. G. Rao, B. Li, and H. Zhang, "Opportunities and challenges of peer-to-peer internet video broadcast," in *Proceedings of the IEEE*, vol. 96, no. 1, 2008, pp. 11–24. [Online]. Available: http://dx.doi.org/10.1109/JPROC.2007.909921

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

[3] D. Ciullo, M. A. Garcia, A. Horvath, E. Leonardi, M. Mellia, D. Rossi, M. Telek, and P. Veglia, "Network awareness of P2P live streaming applications: a measurement study," *submitted to IEEE Transactions on Multimedia*, 2009.

[4] M. Zhang, L. Zhao, Y. Tang, J. Luo, and S. Yang, "Large-scale live media streaming over Peer-to-Peer networks through global Internet," in *Workshop on advances in Peer-to-Peer multimedia streaming*, Singapore, November 2005.

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

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

[7] [Online]. Available: http://www.napa-wine.eu

[8] R. Lobb, A. P. Couto da Silva, E. Leonardi, M. Mellia, and M. Meo, "Adaptive overlay topology for mesh-based p2p-tv systems," in *ACM NOSSDAV*, Williamsburg, Virginia, USA, June 2009.

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

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

[11] [Online]. Available: http://www.cs.cornell.edu/People/egs/meridian/