

Aggregate Flow Control for P2P-TV Streaming Systems

R. Birke¹, C. Kiraly², E. Leonardi³, M. Mellia³, M. Meo³, S. Traverso³

¹IBM Zurich Research Lab. Switzerland; ²University of Trento, Italy; ³Politecnico di Torino, Italy.

E-mail: ¹bir@zurich.ibm.com, ²kiraly@disi.unitn.it, ³{lastname}@tlc.polito.it

Abstract: Peer-to-Peer streaming systems (or P2P-TV) provide low cost infrastructures for the support of live video distribution over the Internet. They are attracting an increasing attention from the research community, the developers and practitioners. In this paper we consider P2P streaming systems based on mesh overlays and focus on the matter of automatically adjusting peer transmission rate to match the demand of the system and not overloading the uplink capacity of peers. We propose Hose Rate Control (HRC), a novel scheme to control the speed at which peers offer chunks to other peers in order to control peer uplink capacity utilization. This is a critical issue to deal with in heterogeneous scenarios like the one faced in the Internet, where peer upload capacity is unknown and varies widely. Presenting extensive simulation and actual experimental results, we show that HRC nicely adapts to the actual peer available upload bandwidth and system demand, so that users' Quality of Experience is greatly enhanced.

Keywords: P2P-TV, streaming, P2P, flow control.

1 INTRODUCTION

In mesh based Peer-to-Peer streaming (P2P-TV) systems, the real-time encoded video is sliced in small pieces called *chunks*, which are distributed over an overlay topology exploiting a fully distributed epidemic approach. Chunks have to be received by peers within a deadline of few seconds in order to guarantee real-time constraints.

In these systems, download rate is dictated by video rate, which is limited by definition; the source peer emits chunks in real time at “constant” rate and all peers must trade them minimizing delays and chunk losses to guarantee real-time constraints and the best Quality of Experience (QoE) to users. Common assumptions about this kind of systems are that i) the upload capacity of peers constitutes the main bottleneck to system performance, and ii) each peer is supposed to instantaneously have a perfect view of the internal state of other peers [8], [2]. While the former assumption is often met in practice, latency among peers makes the latter unrealistic [12]. In mesh-based P2P-TV systems, peers are usually organized into a generic overlay topology, and neighboring peers exchange chunks periodically. To avoid chunk duplications at the receiver, a preliminary trading phase is thus required between neighbor pairs to agree on the chunks to be sent.

To avoid both the burden of handling TCP and the unnecessary delay due to retransmissions and congestion control, UDP is typically preferred by actual P2PTV application [4]. However, this poses the problem of how to handle the congestion control and, in particular, how to limit the amount of information a peer transmits, being download rate limited by video-rate. Controlling therefore the uplink bandwidth utilization is a key problem, which has been so far

marginally considered by the research community. To the best of our knowledge, the only work to explicitly deal with this issue is [3], in which we presented a scheme that assumes essentially homogeneous latencies and, therefore, may be difficult to implement in practice. In this paper, we propose a conceptually different algorithm which additionally results much simpler to implement.

Similarly, few papers specifically focus on the impact of peers bandwidth heterogeneity and how it can be exploited to improve system performance [6], [10]. This aspect is crucial since peers homogeneity is hardly met in practice. Therefore, the design of trading mechanisms requires that a number of parameters is finely tuned to achieve optimal results, and the optimal setup, in its turn, depends on the specific scenario, which is typically unpredictable due to the variability of network conditions and to peer heterogeneity.

Considering the trading scheme, the most critical parameter is the number of “offers” (messages advertising the list of chunks possessed) a peer should send in parallel, i.e., the number of active *signalling threads*. If this number is too small, the delivery rate of chunks is small, thus upload bandwidth is under-utilized. Conversely, if this number is too large, the committed workload would overflow transmission resources, impairing perceived quality of the video stream.

In this paper, we propose a scheme, called *Hose Rate Control*, HRC, to automatically adapt the number of signaling threads to the scenario. By doing so, the scheme implements a peer aggregate transmission rate control that aims at jointly exploiting the peer upload capacity and improving QoE of users reducing chunk delivery delays; in other words, the scheme controls the bandwidth allocation on the peer uplink channel.

HRC has been implemented in “WineStreamer”, the new P2P-TV application under development within the EU-FP7 NAPA-WINE STREP project [1]. In this paper, we provide experimental results obtained by running experiments on swarms of up to 1000 peers in a controlled environment. Extensive experimental results obtained considering both simulations and the actual implementation show that, with respect to non adaptive mechanisms, HRC optimizes resource utilization, consistently improving system performance and QoE that we evaluate on real video sequences by computing the SSIM (Structural Similarity Index) [11].

2 SYSTEM DESCRIPTION

We consider a system in which a source segments the video stream into chunks and injects them in the overlay network. Let N be the number of peers composing the overlay. The system must deliver every generated chunk within a deadline

called *playout delay*, D_{max} . If the chunk age is greater than D_{max} , the chunk is useless and is not traded anymore. Chunks are transmitted by peers to their neighbors in a swarm-like fashion; the overlay topology is defined by the set of peers and virtual links connecting them. 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 on a random basis, a common assumption in the literature [8], [6].

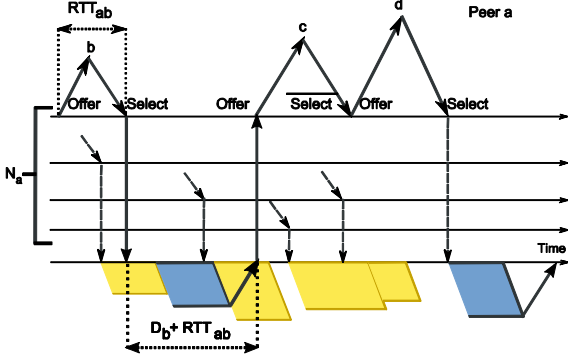


Figure 1: Schematic representation of the peer chunk trading mechanism.

Since video chunks are transmitted over the network, the intuition suggests to keep them small, e.g., few IP packets, to minimize the packetization delay at the source, the store-and-forward delay at the peers and the chunk corruption probability due to packet loss. In what follows, we therefore consider 1 video frame per chunk, e.g., average chunk size is 5.3 Kbyte with a 1Mbit/s encoding rate.

2.1 Chunk Trading Mechanism

The signalling mechanism used to exchange chunks is a trading scheme similar to the one used in other mesh-based P2P-TV systems [12], [13]. A chunk is sent from a peer to one of its neighbors after a trading phase. Peer a maintains a number of *trading threads*, N_a . Each trading thread evolves as follows:

- 1) peer a chooses one of its neighbors b and sends it a special signalling message, called *offer* message; it contains the set of chunks a possesses younger than D_{max} .
- 2) Upon receiving the offer message, b replies with a *select* message to request a desired chunk. Once a chunk has been “selected”, the receiver sets it as *pending* until it is correctly received; a pending chunk cannot be requested and cannot be published.
- 3) When the select message is received by a
 - a) if a chunk was requested in the select message (*positive select*), a schedules its transmission, inserts it in its *chunk transmission queue* that is served in a FIFO order.
 - b) Once b has completely received the chunk previously selected, it sends an *ACK* message.
 - c) When a receives the ACK message, it can send a new offer message and a new cycle starts.
 - d) If no chunk was requested in the select message (*negative select*), a can send a new offer message and a new cycle starts.

Peer a is committed to send all the chunks requested in all the received positive select messages. Fig. 1 represents the signalling messages and chunks exchanged by peer a with its neighbors over time. In particular signalling messages/chunks associated to one active thread are highlighted. Note that all N_a trading threads continue these cycles independent of each other.

Several design choices impact the performance of the trading mechanism: 1) the criterion to select destination peers for the offer message – known as the “peer selection”; 2) the strategy according to which peers receiving an offer message select chunks to download – known as the “chunk selection”; 3) the number N_a of threads peer a handles.

For the *peer selection* and the *chunk selection* policies we make the simplest possible choices: peer a chooses the peer to contact uniformly at random within the set of its neighbors, and neighbors choose the chunks to select at random among the ones they need. This policy is also known in the literature as “Random Peer - Random Useful Chunk selection” [5].

The key parameter to set in this mechanism is N_a , which is the equivalent of the window size in a window protocol.

2.2 The core of Hose Rate Control

HRC, the adaptive signal mechanism that we propose, stems from the basic idea is to control the rate at which chunks are sent by peer a by controlling the number of parallel active threads N_a , so that the queuing delay at the transmission queue is at a given (small) target. The rationale is that N_a controls the amount of work that the peer a has to do: if it is too large, delay increases, deteriorating performance; if it is too small, the peer available upload bandwidth is not well exploited. The rule to decide N_a is based on an estimation of the queuing delay incurred by chunks in the transmission queue: if the queuing delay is large, N_a is decreased, and vice-versa.

More in detail, the algorithm according to which N_a is made adaptive is the following. Let W_a be the internal control variable, which takes real values: $N_a = \lfloor W_a \rfloor$. For every neighbor peer b , peer a maintains an estimate of the minimum Round Trip Time. This estimate can be computed/updated by a every time it receives a select message as the difference between the time the select message is received and the one the offer is sent, $RTT_{ab} = t_{rx,select}^{(a)} - t_{tx,offer}^{(a)}$ where $t_{rx,select}^{(a)}$ identifies the time at which the select message is received by a .

When a receives an acknowledge from b , it estimates the delay D incurred by the chunk in the transmission queue, as $\hat{D} = t_{rx,ack}^{(a)} - t_{rx,select}^{(a)} - RTT_{ab}$ i.e., subtracting a RTT from the difference between the time at which the acknowledge was received and the time at which the chunk was enqueued. \hat{D} is then compared with a prefixed target value, D_0 , and W_a is updated according to the following rule:

$$W_a(n) \leftarrow W_a(n-1) - K(\hat{D} - D_0) \quad (1)$$

N_a is then increased/decreased by $\Delta N_a = \lfloor W_a(n) \rfloor - \lfloor W_a(n-1) \rfloor$.

Note that targeting queuing delay, HRC results less aggressive than TCP congestion control which react to

congestion only after a packet has been dropped by the queue. This is an intended behavior of HRC since we want to achieve network friendliness, one of the major goal of the NAPA-WINE project [1], i.e., we target less than best-effort bandwidth utilization.

3 EVALUATION BY SIMULATION

3.1 Simulation scenario and assumptions

All simulation results shown in this paper have been obtained with *P2PTV-sim*, an open source event driven simulator available from [1]. 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 $5\text{Mb/s} \pm 20\%$, 35% in Class 2 with $1.6\text{Mb/s} \pm 20\%$, 35% in Class 3 with $0.64\text{Mb/s} \pm 20\%$, 15% in Class 4 with negligible upload bandwidth. The video source belongs to Class 1. The average bandwidth per peer is $E[B_a] = 1.25\text{Mb/s}$.

In each simulation D_{\max} is set to 6s if not otherwise stated. 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_{ab} that is added to the transmission time of all the packets flowing from a to b . End-to-end latencies are taken from the experimental dataset of the Meridian project (<http://www.cs.cornell.edu/People/egs/meridian>); the overall mean latency is $E[l_{ab}] = 39\text{ms}$.

The well-known *Pink of the Aerosmith* video sequence, encoded using the H.264/AVC Codec, is considered as benchmark. A hierarchical type-B frames prediction scheme has been used, obtaining 4 different kinds of frames that, in order of importance, are: IDR, P, B and b. The GOP structure has been set to $\text{IDR} \times 8 \{P, B, b, b\}$. The video consists of 3000 frames, which correspond to about 120s of visualization. The nominal video rate of the encoder r_s is a free parameter that we vary to enforce different values of the system load defined as,

$$\rho = r_s / E[B_a] \quad (2)$$

The source node then generates a new chunk at regular time, i.e., every new frame. This mapping scheme minimizes the chunk size, thus allowing a stricter real-time streaming. Furthermore, the rounding at frame boundaries minimizes the impact of losses, avoiding that a loss of a chunk affects several frames due to partial delivery of information, e.g, missing headers. 40B long signalling messages are considered.

The overlay topology is randomly generated at the beginning of a simulation by letting each peer randomly select 30 other peers as its neighbors. Since connections are bidirectional, the average number of neighbors for a peer is equal to 60. As we simulate a couple of minutes of the system behavior, we neglect the effect of churning so that the topology is static for the whole simulation run. All plots 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.

Real video streams carry highly structured information, part of which is more important than other, with high variability in the generated bit-rate. Chunk loss probability and delivery delay are the performance indexes typically adopted by the networking community, but they provide only a partial view of the actual performance of a P2P-TV system, the user perceived quality. In the multimedia and signal processing communities, instead, the evaluation of the perceived quality is considered mandatory, see [9], [7] for notable examples. To this extent, performance is expressed in terms of average Structural Similarity Index (SSIM) [11] which has been designed to improve on traditional methods like Peak Signal-to-Noise Ratio (PSNR) and Mean Squared Error (MSE), which have proved to be inconsistent with human eye perception. The SSIM is a measure of the similarity of the received image compared against the original source. It is a highly non linear metric in decimal values between -1 and 1. Values above 0.95 are typically considered of good quality. In our simulation scenario, SSIM has been computed considering video frames received by 100 peers (25 for each class), and then averaging among all of them. The initial and final 10s of video have been discarded to focus on steady state.

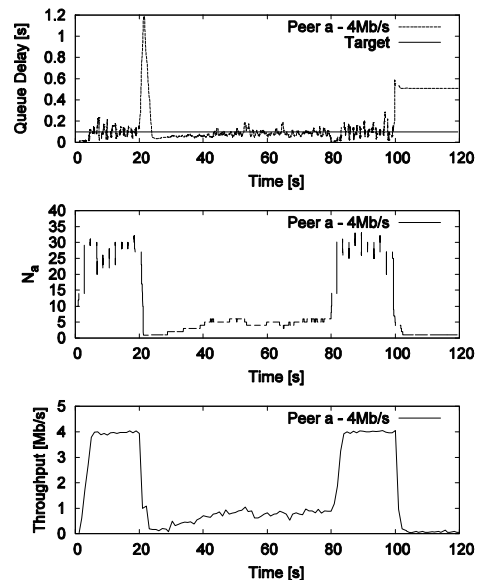


Figure 2: Queuing delay (top), value of N_a (center), and throughput (bottom) vs time with variations due to interfering traffic on upload link. $\rho = 0.95$.

3.2 Transient analysis

In the following, we show results about the HRC algorithm obtained through simulations; D_0 is set to 100ms. The source has rate $r_s = 1.2\text{Mb/s}$, corresponding to $\rho = 0.95$.

3.2.1 Simple scenario

Let us focus on a peer a with available upload bandwidth of 4Mb/s that varies due to interfering traffic. Fig. 2 reports the evolution over time of queue delay $\hat{D}(t)$ (top), the number of active threads N_a (center) and the throughput (bottom) for peer a . During the first 20s, no interfering traffic is present; after an initial transient the value of N_a stabilizes around 25, so that the peer can exploit at best its upload bandwidth. At time $t = 20\text{s}$,

a Constant Bit Rate flow starts injecting 3Mb/s of interfering traffic in the uplink. $\widehat{D}(t)$ abruptly grows; thus, N_a is reduced and stabilized around 5 and the upload throughput decreases to 1Mb/s. At $t = 80$ s the whole upload bandwidth turns to be available again, inducing an increase of N_a which gets back to 25. Then, from $t = 100$ s to $t = 120$ s, a TCP-like flow is present, consuming the whole peer's upload bandwidth. In this period, the number of active threads drops to its minimal possible value, $N_a = 1$, because the congestion due to the TCP-like flow pushes $\widehat{D}(t)$ over the control target D_0 . As a consequence, even the application throughput is reduced to negligible values. In conclusion, the control algorithm succeeds in promptly reacting to bandwidth variations, and in achieving less than best-effort bandwidth utilization.

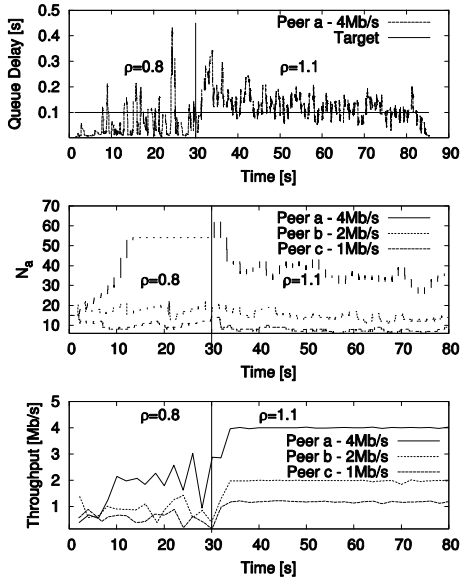


Figure 3: Queuing delay (top), N_a (center), and throughput (bottom) vs time with variations due to flash crowd event.

3.2.2 Flash crowd scenario

We now consider the scenario in which the system operating point is abruptly modified at $t = 30$ s: a sudden ingress of 400 new peers with negligible upload-bandwidth and a contemporary reduction by 50% of the available upload bandwidth of all peers belonging to Class 3 happens. Given video rate $r_s = 1$ Mb/s, this causes the system load ρ to shift from 0.8 to 1.1. Even if this scenario is rather artificial, it has been selected because it maximizes the stress on the control scheme. Fig. 3 reports the evolution of N_a (center) and throughput (bottom) for three sample peers, a , b and c , with upload bandwidth of 4, 2 and 1Mb/s, respectively. The evolution of peer a queue delay $\widehat{D}(t)$ is also reported (top). When $\rho = 0.8$, N_a , N_b and N_c adapt to different values, reflecting each peer's ability to contribute to chunk diffusion. Since $\rho < 1$, not all system capacity is required, and N_a rapidly grows to its maximum value $N_a = 53$, i.e., the number of a neighbors. At $t = 30$ s, the HRC system reacts to the sudden system condition variations. In particular, for the high bandwidth peer a , N_a initially increases, since the number of its neighbors increases and its capacity was not fully exploited

(its queuing delay still being smaller than D_0). Then, the increased system load boosts the percentage of offers that are positively selected, causing additional queuing delay, so that N_a decreases. After a quick transient, upload rate matches each peer upload capacity, and queuing delay reaches the target D_0 .

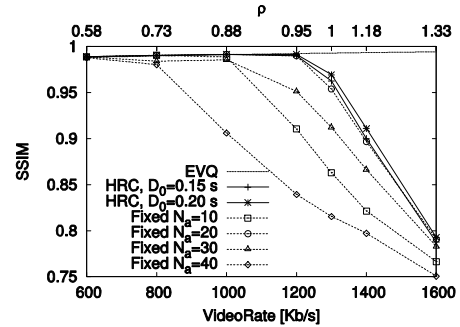


Figure 4: SSIM of HRC and non-adaptive schemes vs system load.

3.2.3 Steady-state analysis

In this section, we focus on the steady-state performance of HRC and we compare it with non-adaptive schemes that use a fixed value of N_a .

Fig. 4 compares the HRC system for $D_0 = 150$ ms and 200ms and the non-adaptive schemes in which N_a is fixed. The video rate is increased (reported on bottom x-axis) to observe the performance of the system of increasing ρ (reported on top x-axis). When $\rho < 1$, the SSIM increases for increasing r_s thanks to the higher quality of the encoded video. As soon as the system is overloaded, the SSIM rapidly drops due to missing chunks which impair the quality of the received video. In all scenarios, HRC outperforms the non-adaptive scheme, for any values of N_a . Schemes with too small values of N_a do not fully exploit the system bandwidth, e.g., $N_a = 10$; schemes with too large values of N_a tend to overload the peer transmission queue leading to an unnecessary increase of the chunk delivery delay, e.g., $N_a = 30$. The performance of the scheme with $N_a = 20$ are comparable with that of HRC. However, setting the value of N_a is very critical, since the optimal value depends on many other system parameters, such as the peers upload bandwidth distribution, that, besides being difficult to know, is variable in time due to interfering traffic, as seen in Figs. 2 and 3.

We have performed a more extensive set of simulations to assess the benefits of HRC. Due to lack of space we do not report them here, but we prefer to present some experimental results we collected from real implementation of HRC.

4 EVALUATION BY EXPERIMENT

The HRC controller has been implemented into WineStreamer (source code is available at <http://peerstreamer.org>) P2P-TV application. In the following we briefly discuss the key aspects of the implementation and provide some experimental evidence of the benefits of HRC.

4.1 Implementation issues

The most critical part when undergoing the actual engineering of the HRC scheme is the estimation process of queuing delay which is at the core of HRC scheme. Three different cases can

be considered: i) the presence of priority policies that differentiate signalling and data information at the ingress to the network, ii) no priority policy but synchronization among peers achieved at the application layer, iii) no priority and no synchronization.

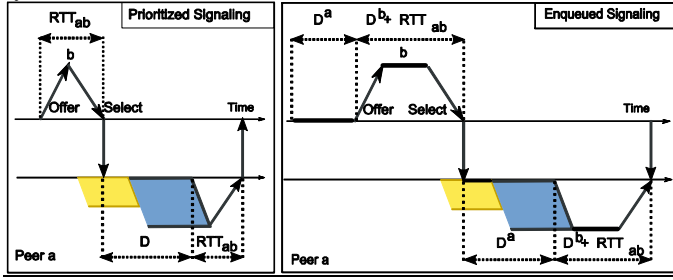


Figure 5: Schematic representation of the peer chunk trading mechanism with prioritized signalling –PS– (left) and enqueued signalling –ES– (right).

Let us first consider the scenario in which the access device of the bottlenecked node supports separate queues: a high priority queue serves signalling packets, and a low priority queue serves data packets. This feature is offered by most of nowadays ADSL/cable access devices that support multimedia services. Referring to left part of Fig. 5, consider a peer a that is sending an offer to a neighbor b ; in a , the estimation of the queuing delay \hat{D} is then straightforward $\hat{D} = t_{rx,ack}^{(a)} - t_{rx,select}^{(a)} - RTT_{ab}$.

The estimation of the round-trip time between a and b , RTT_{ab} (which does not include the queuing delay), can be easily carried out by exploiting the higher priority service offered to signalling packets: $RTT_{ab} = t_{rx,select}^{(a)} - t_{tx,offer}^{(a)}$.

Consider now the second scenario, in which a single class of service is offered by the network devices and peers are synchronized. Here, signalling messages are delayed by the transmission queue too, as sketched in left part of Fig. 5. Synchronization allows to measure the One-Way-Delay between a and b , OWD_{ab} , as the minimum of all $t_{rx,select}^{(a)} - t_{tx,select}^{(b)}$ estimates. The estimation is then given by $\hat{D} = t_{rx,ack}^{(a)} - t_{rx,select}^{(a)} - OWD_{ab}$.

Coarse synchronization, as the one provided by the NTP protocol, would suffice, and errors in the order of 1ms would marginally affect the HRC control, given the target queuing delay D_0 is of the order of 100ms.

Finally, in the third scenario peers are not synchronized and no priority policy is provided. \hat{D} cannot be estimated anymore, since any RTT measurement includes both the queuing delay at peer a , denote it by $D^{(a)}$, and the queuing delay at b , $D^{(b)}$; i.e., it is only possible to estimate the sum of the queuing delays,

$$\hat{D}_{(a+b)} = D^{(a)} + D^{(b)} = t_{rx,ack}^{(a)} - t_{rx,select}^{(a)} - RTT_{ab}. \quad (3)$$

RTT_{ab} can still be estimated as the minimum over all RTT samples, while it is impossible to decouple $D_{(a+b)}$ from $D^{(a)}$ and $D^{(b)}$. Thus, the HRC algorithm at peer a controls the sum of the queuing delays, and it is coupled with the HRC control

of all its neighbors. We consider this latter scenario in our implementation so that queuing delay is estimated as in (3).

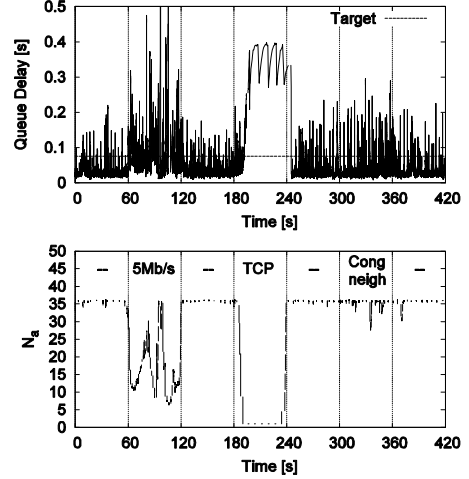


Figure 6: Queue delay (top), N_a (bottom) vs time with variations due to interfering traffic on upload link in experimental setup.

4.2 Experimental results

4.2.1 Simple scenario

We first consider a simple scenario in which the source s is connected to a HRC-enabled peer a only. 36 other peers are then attached to a , so that its upload capacity is used to feed all neighbors. We then impose transient conditions to the upload link of a . The Linux `tc` tool is used to limit the upload capacity and delay, while the `iperf` tool is used to inject artificial traffic. Video rate is 0.6Mbps, 20ms RTT is imposed on all links and $D_0 = 75$ ms.

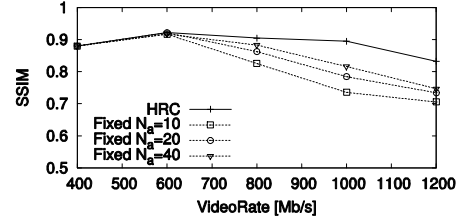


Figure 7: SSIM vs video rate for HRC and non-adaptive schemes. Experimental results in swarm of 74 peers.

Fig. 6 reports the evolution of queuing delay (top) and of N_a (bottom) for peer a . During the first 60s, peer a uplink bandwidth is large enough to transmit all committed chunks. Since a queuing delay cannot reach the target, N_a stabilizes at the maximum possible value ($N_a = 36$) which represents the size of peer's neighborhood. At time $t = 60$ s, a uplink capacity is limited to 5Mb/s, inducing HRC to nicely reduce the number of parallel active signalling threads while queuing delay varies around the target value. From $t = 120$ s to $t = 180$ s, N_a stabilizes at the neighborhood size, being uplink bottleneck removed. At time $t = 180$ s a competing TCP flow starts consuming the link capacity, increasing the queuing delay so that N_a reduces to 1, the minimum possible value. At the end of the TCP flow, HRC controls N_a value increasing it to 36 again. Finally, from $t = 300$ s to $t = 360$ s, peer b , one of a 's neighbors, suffer congestion in its uplink: a

TCP flow starts sending data from b to a , so that $D^{(b)}$ grows, possibly impairing $\widehat{D}_{(a+b)}$.

The plot shows that the estimated queuing delay at peer a is slightly affected by the presence of congestion on its neighbor, indeed some larger oscillations are visible. However N_a is basically unaffected. The intuition indeed suggests that if the number of “biased” estimates at peer a is limited, the system is still able to control the peer uplink queue by “filtering” out the few overestimated samples.

4.2.2 Small swarm

In this second experiment, we consider 74 PCs each running a peer. Upload capacity of peers has been limited using τc : 10% of peers have 5Mb/s, 50% have 1Mb/s, 40% have 0.5Mb/s only, corresponding to an average per peer data link capacity of 1.2Mb/s. Delay among peers randomly varies between 10ms and 50ms. The “Big Buck Bunny” video (480p resolution, 24fps, H.264/AVC Codec) has been encoded at different rates and “streamed” to all peers. After discarding the initial 3000 frames (125s), each peer save 1500 (75s) of the received frames on disk. SSIM is then computed against the original YUV video for all peers, and the average SSIM is then computed. Simple random overlay topology (average neighbors size=40), random peer/random chunks selection are adopted. D_{max} was 3s.

Fig. 7 compares the HRC performance against that of non adaptive schemes in which $N_a = 10, 20, 40$ respectively. $D_0 = 250$ ms in this case. Results are similar to the one of Fig. 4: HRC performs similarly when the system is underloaded, while it outperforms any fixed schemes. Indeed, the correct choice of N_a is critical: it must be small to prevent from overloading low bandwidth peers, while it must be large to avoid under-utilizing high bandwidth peers. HRC achieves this goal: for example, for video rate 1Mb/s, $N_a \in [60, 64]$ for high bandwidth peers, $N_a \in [10, 15]$ for mid bandwidth peers, and $N_a \in [5, 8]$ for the low bandwidth peers. Any fixed values would cause a mismatch, impairing the overall system performance. Recall that SSIM is smaller than 1 since we are considering the encoding loss too.

4.2.3 Large swarm

Finally, we presents results considering an experiment involving 200 PCs, each running 5 peers simultaneously, i.e., a swarm of 1000 peers. Peer upload capacities are artificially limited as above. Due to practical constraints, we present results considering as performance index the per-peer average played frame rate. Fig. 8 reports results considering video rate of 1Mb/s, which corresponds to a scenario in which the system is already overloaded. Notice that HRC guarantees much better performance than any fixed scheme. Interestingly, better video quality is guaranteed to higher bandwidth peers, as highlighted by the “clustering” in the results. Investigating further, we observed that this is due to the “maximum” queueing delay that packets of different capacity peers may face: low capacity peer packets will stay in the upload queue for longer, so that scheduler timers are negatively affected causing artificial frame drops due to $D_{max} = 3$ s only.

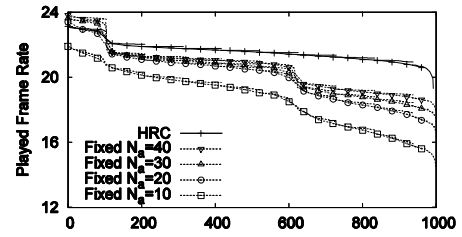


Figure 8: Average played frame rate for HRC and non-adaptive schemes. Experimental results in swarm of 1000 peers. $r_s = 1$ Mbps.

5 CONCLUSIONS

We focused on the trading phase of mesh-based P2P-TV systems and proposed Hose Rate Control, an algorithm to tune the number of chunks a peer offers to its neighbors. HRC aims at efficiently exploiting the peer upload bandwidth by controlling the queuing delay suffered by transmitted chunks in the peer upload access link, which is today the typical bottleneck for P2P-TV systems. Our results show that HRC reduces chunks delivery delay leading to better QoE for users. We implemented HRC in the NAPA-WINE client, coping with the actual implementation issues and presenting experimental results considering swarms up to 1000 peers.

6 ACKNOWLEDGMENT

This work was supported by the EC under the FP7 TREP “Network-Aware P2P-TV Application over Wise networks”.

References

- [1] NAPA-WINE, <http://www.napa-wine.eu>.
- [2] T. Bonald, L. Massoulié, F. Mathieu, D. Perino, and A. Twigg. Epidemic live streaming: optimal performance trade-offs. In *SIGMETRICS*, Annapolis, MD, US, June 2008.
- [3] A. Carta, M. Mellia, M. Meo, and S. Traverso. Efficient uplink bandwidth utilization in P2P-TV streaming systems. In *IEEE Globecom*, Miami, FL, US, December 2010.
- [4] 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. *IEEE Transactions on Multimedia*, 12(1):54–63, January 2010.
- [5] A. Couto da Silva, E. Leonardi, M. Mellia, and M. Meo. Exploiting Heterogeneity in P2P Video Streaming, *IEEE Transactions on Computers*, December 2010.
- [6] Y. Liu. On the minimum delay peer-to-peer video streaming: how realtime can it be? In *ACM Multimedia*, Augsburg, DE, September 2007.
- [7] Z. Liu, Y. Shen, K. W. Ross, S. S. Panwar, and Y. Wang. Layerp2p: using layered video chunks in p2p live streaming. *IEEE Transaction on Multimedia*, 11(7):1340–1352, 2009.
- [8] S. Sanghavi, B. Hajek, and L. Massoulié. Gossiping with multiple messages. In *IEEE INFOCOM*, Anchorage, AK, US, May 2007.
- [9] E. Setton, J. Noh, and B. Girod. Low latency video streaming over peer-to-peer networks. In *IEEE ICME*, Toronto, CA, July 2006.
- [10] T. Small, B. Liang, and B. Li. Scaling laws and tradeoffs in Peer-to-Peer live multimedia streaming. In *ACM Multimedia*, Santa Barbara, CA, US, October 2006.
- [11] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. Image quality assessment: From error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, April 2004.
- [12] 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 *P2PMMS*, Singapore, November 2005.
- [13] 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, FL, US, March 2005.