[Article] A delay-based aggregate rate control for P2P streaming systems

A Delay-Based Aggregate Rate Control for P2P Streaming Systems

Robert Birke, Csaba Kiraly, Emilio Leonardi, Marco Mellia, Michela Meo, Stefano Traverso

PII: S0140-3664(12)00233-2
DOI: http://dx.doi.org/10.1016/j.comcom.2012.07.005
Reference: COMCOM 4682

To appear in: *Computer Communications*

Received Date: 1 December 2011
Revised Date: 10 July 2012
Accepted Date: 11 July 2012

Please cite this article as: R. Birke, C. Kiraly, E. Leonardi, M. Mellia, M. Meo, S. Traverso, A Delay-Based Aggregate Rate Control for P2P Streaming Systems, *Computer Communications* (2012), doi: http://dx.doi.org/10.1016/j.comcom.2012.07.005

A Delay-Based Aggregate Rate Control for P2P Streaming Systems

Robert Birke[a], Csaba Kiraly[b], Emilio Leonardi[c], Marco Mellia[c], Michela Meo[c], Stefano Traverso[c,*]

[a] *IBM Research Zurich*
Saumerstrasse 4, 8803 Ruschlikon, Zurich - CH
[b] *DISI – Università di Trento*
Via Sommarive, 14, Povo, 38123 Trento - IT
[c] *Politecnico di Torino*
Corso Duca degli Abruzzi 24, 10129 - Torino - IT

Abstract

In this paper we consider mesh based P2P streaming systems focusing on the problem of regulating peer transmission rate to match the system demand while not overloading each peer upload link capacity. We propose Hose Rate Control (HRC), a novel scheme to control the speed at which peers offer chunks to other peers, ultimately controlling peer uplink capacity utilization. This is of critical importance for heterogeneous scenarios like the one faced in the Internet, where peer upload capacity is unknown and varies widely.

HRC nicely adapts to the actual peer available upload bandwidth and system demand, so that Quality of Experience is greatly enhanced. To support our claims we present both simulations and actual experiments involving more than 1000 peers to assess performance in real scenarios. Results show that HRC consistently outperforms the Quality of Experience achieved by non-adaptive schemes.

Keywords:
Peer-to-Peer, Video Streaming, Measurement, 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

*Corresponding Author: Stefano Traverso, Dipartimento di Elettronica, Politecnico di Torino, Corso Duca degli Abruzzi 24, 10124 - Torino, Italy email: traverso@tlc.polito.it, tel: +39-011-090-4243, Fax: +39-011-090-4099

Email addresses: bir@zurich.ibm.com (Robert Birke), kiraly@disi.unitn.it (Csaba Kiraly), leonardi@tlc.polito.it (Emilio Leonardi), mellia@tlc.polito.it (Marco Mellia), michela@tlc.polito.it (Michela Meo), traverso@tlc.polito.it (Stefano Traverso)

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 peers must trade them minimizing delays and losses to guarantee the best Quality of Experience (QoE) to users.

Common assumptions about P2P-TV 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 [1, 2, 3]. While the former assumption is often met in practice, latency between peers makes the latter unrealistic [4]. 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 to agree on the chunks to be exchanged. This trading phase requires the exchange of messages between peer pairs: an "offer" message sent by the potential sender a to the potential receiver b containing the list of the chunks a can offer (those within the deadline it possesses) and a reply message (called "select") containing the list of chunks selected to be downloaded by b. A careful design of the trading scheme is needed to avoid that the additional signalling delay translates into an excessive delay and that a peer overbooks its upload capacity by offering too many chunks. This paper focuses on the design of the trading scheme and proposes a simple algorithm to control the rate at which chunks are offered by peers to neighbors.

To transmit chunks, UDP is typically preferred by actual P2P-TV applications [5] to avoid both the burden of handling TCP connections and the unnecessary delay due to retransmissions and congestion control. However, this requires to handle the congestion control and, in particular, to limit the amount of content 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.

Considering the trading scheme, the most critical parameter is the number of "offers" (messages advertising the list of possessed chunks) 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 signalling threads to the current network 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 as much as possible chunk delivery delays. In other words, the scheme controls the bandwidth allocation on the peer uplink channel.

The HRC objective is to exploit the upload bandwidth of peers while not increasing queuing delay, therefore targeting a less-than-best-effort policy being less aggressive than the TCP congestion control whose regulation scheme is loss-

based. This is an explicit design choice that aims at tightly controlling chunk delivery delay and chunk delivery probability, i.e., minimizing packet loss and lengthly retransmissions.

We implemented HRC in "PeerStreamer", the P2P-TV application developed within EU-FP7 NAPA-WINE STREP project [6]. This allows us to provide experimental results 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) [7].

2. Related Work

The literature on P2P-TV streaming systems is rich of works. Common assumptions are that the upload capacity of peers constitutes the main bottleneck to system performance, and each peer instantaneously has a perfect view of the internal state of other peers [1, 2, 3, 8]. While the former assumption is often met in reality, latency among peers makes the latter unfeasible. The impact of latency on system performance has been analyzed by means of a simple model corroborated by real measurements in PlanetLab in [4]. The authors propose a system that mitigates the effect of latency by exchanging state information via signalling messages. Little description is however given about the implemented signalling mechanisms details. [9] also evaluates the impact of latency on system performance, using a non-adaptive signalling mechanism.

Few papers specifically focus on the impact of peers bandwidth heterogeneity and how it can be exploited to improve system performance [8, 10, 11]. This aspect is crucial since peers homogeneity is hardly met in practice.

Given the small size of chunks, UDP is typically preferred by actual P2P-TV applications [5] to enforce serial chunk packet transmission and to avoid both unnecessary delay due to TCP retransmission and congestion control. Controlling the uplink bandwidth utilization is thus a key problem. To the best of our knowledge, the only work to explicitly deal with this issue is [12], in which authors present a preliminary scheme based on the periodic advertisement of peer's buffermap, offered to a subset of the peer's neighbors. However, the resulting scheme assumes essentially homogeneous latencies and may be difficult to implement in practice. In this paper, we propose a conceptually different scheme which proves to be much simpler to implement.

Considering file sharing P2P systems the problem of controlling the sending rate of peers is a timely problem. Recently, BitTorrent designers decided to adopt UDP in their client. Indeed, a new congestion control algorithm called LEDBAT (Low Extra Delay Background Transport) has been proposed [13]. LEDBAT targets a less aggressive congestion control mechanism than the one implemented in TCP.

To this extent HRC is somehow similar to LEDBAT, however, they differ in the following two key aspects: i) HRC is an *aggregate hose mechanism* that

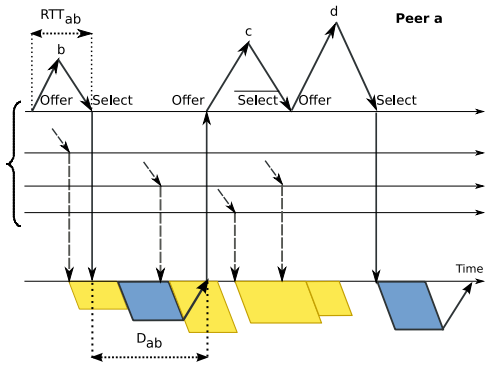

Figure 1: Schematic representation of the peer chunk trading mechanism.

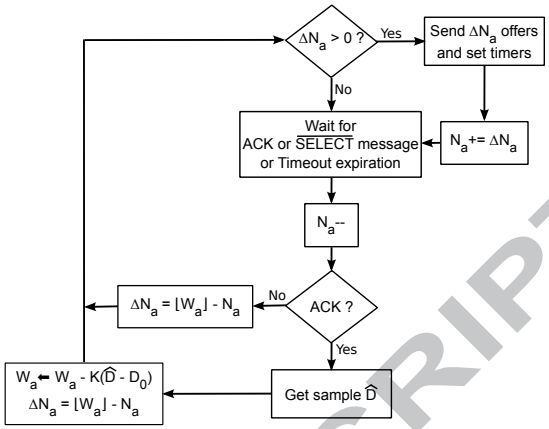

Figure 2: Flow chart representation of HRC control loop.

controls the overall sending rate of a peer, and not the per-flow end-to-end sending rate; ii) HRC is chunk based rather than packet based. As previously mentioned, this stems from the different goals of streaming versus file sharing P2P applications: the first targets live distribution of un-elastic content, while the latter targets the maximization of download throughput.

3. System Description

We consider a system in which a source segments the real-time generated 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 forms a random graph, i.e. peer connects to other peers on a random basis, a common assumption in the literature [1, 2, 10].

Since video chunks are transmitted over the network, the intuition suggests to keep them small, e.g., few kBytes, 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 choose that 1 chunk contains exactly 1 video frame, e.g., average chunk size is 5kB for a 1Mb/s encoding rate of a 25 fps video. This mapping scheme minimizes the chunk size and the impact of losses, avoiding that a loss of a chunk affects several frames due to partial delivery of information, e.g, missing headers.

3.1. Chunk Trading Mechanisms

The signalling mechanism used to exchange chunks is a trading scheme similar to the one used in other mesh-based P2P-TV systems [4, 14, 15]. 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 signalling message, called *offer* message; it contains the set of younger than D_{max} chunks a possesses.
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 yet.
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 selected chunk, it sends an *ACK* message to a.
  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. Timers protect the waiting for messages so that in case no reply is received within a timeout, the status is reset and a new thread can start. 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 which is somehow equivalent of the window size in a window protocol and represents the rate at which potential transmitting peers offer their chunks.

For the *peer selection* and the *chunk selection* policies we make the simplest possible choices: peer a chooses peers to contact uniformly at random within the set of its neighbors, and the 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" [16].

The key parameter that requires to be set in this mechanism is N_a.

3.2. The core of Hose Rate Control

HRC, the adaptive signal mechanism that we propose, stems from the basic idea 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, upload capacity is exceeded and delay increases, deteriorating performance; if it is too small, the peer available upload bandwidth could not be 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.

HRC mechanism has been studied analytically through a fluid model that we do not report in this paper due to the lack of space. However, the model is fully described in technical report [17].

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: $N_a = \max(1, \lfloor W_a \rfloor)$. $W_a < W_{\max}$, where $W_{\max}$ limits the maximum number of offers in flight, e.g., it can be constrained to the number of neighbors of peer a. 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,

$$RTT_{ab} = t^{(a)}_{\text{rx,select}} - t^{(a)}_{\text{tx,offer}}$$

where $t^{(p)}_{\text{action,type}}$ identifies the time of the "action" triggered by the message of "type" at peer p; action=$\{rx, tx\}$, type=$\{offer, select, chunk, ack\}$.

When a receives an acknowledge from b, it estimates the delay D incurred by the chunk in the transmission queue, as $\hat{D} = t^{(a)}_{\text{rx,ack}} - t^{(a)}_{\text{rx,select}} - RTT_{ab}$, 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:

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

N_a is then increased/decreased by $\Delta N_a = \lfloor W_a(n) \rfloor - \lfloor W_a(n-1) \rfloor$. Now, if $\Delta N_a = 0$, the number of active threads is not changed, and peer a is allowed to send a new offer to one of its neighbors. If $\Delta N_a > 0$, the number of active threads is increased, and peer a is allowed to send two or more offers to its neighbors. At last, if $\Delta N_a < 0$, the number of active threads is decreased and the current thread is stopped (no new offer is sent). A flow chart representation of the described algorithm is given in Fig. 2.

Influence of parameter K has been studied in technical report [17]. The stability of the controller has been analytically proven for any value of K in the interval $(0; 1)$, and numerically up to $K = 20$. However, even if parameter K regulates the reactiveness of the controller, due to the relatively slow dynamics of this kind of systems, it has a negligible effect on performance. For simulation and experimental results shown in this paper, we set K to 0.98.

Note that targeting queuing delay, HRC results less aggressive than TCP congestion control which reacts to congestion only after a packet has been dropped by the queue. This is an intended behavior of HRC since having a tight control of chunk delivery delay is fundamental to make the system to work properly in P2P-TV streaming applications. On this regards, observe that P2P streaming applications can effectively exploit spatial diversity (every peer can retrieve a chunk from several neighbors) to effectively face congestion that may arise at some of the peers.

4. Evaluation by simulation

4.1. 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 [6]. 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 ± 20%, 35% in Class 2 with 1.6Mb/s ± 20%, 35% in Class 3 with 0.64Mb/s ± 20%, 15% in Class 4 with negligible upload bandwidth. The video source belongs to Class 1. The average bandwidth per peer is $E[B_a] = 1.25$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 [18]; the overall mean latency is $E[l_{ab}] = 39$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 IDR x 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,

$$\rho = r_s / E[B_a] \tag{2}$$

The source node generates a new chunk at regular time, i.e., every new frame. 40B long signalling messages are considered.

The overlay topology is randomly generated at the beginning of a simulation by letting each peer selects 30 other peers at random as its neighbors. Since connections are bidirectional, the average number of neighbors for a peer is approximately 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 [19, 20] for notable examples. To this extent, performance is expressed in terms of average Structural Similarity Index (SSIM) [7] 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. In case of chunk loss, that is equivalent to a frame loss in our case, the employed codec replaces missing frames with the last successfully received video frame. The similarity of the replaced frame is then computed against the expected original one. SSIM 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 performance.

4.2. 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 $r_s = 1.2$Mb/s, corresponding to $\rho = 0.95$.

• **Simple scenario** - Let us focus on a randomly selected peer a with available upload bandwidth of 4Mb/s that varies due to interfering traffic. Fig. 3 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 $t = 20$s, a Constant Bit Rate flow starts injecting 3Mb/s of interfering traffic in the uplink. $\hat{D}(t)$ 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 $\hat{D}(t)$ over the control target D_0. As a consequence, 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.

• **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.

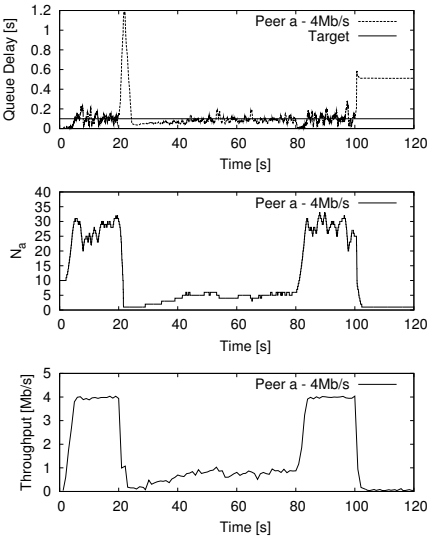

Figure 3: Queuing delay (top), value of N_a (center) and throughput (bottom) vs. time with variations due to interfering traffic on upload link. $\rho = 0.95$

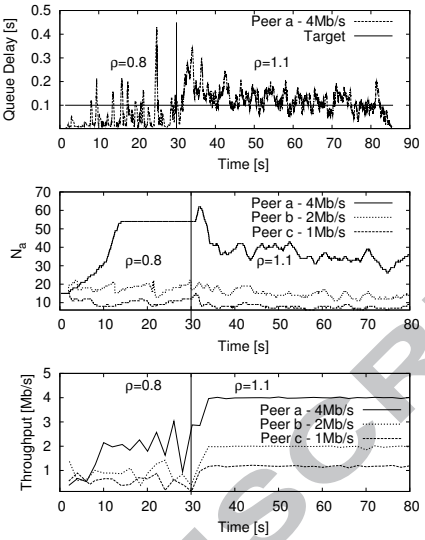

Figure 4: Queuing delay (top), N_a (center) and throughput (bottom) vs. time for flash crowd.

.

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. 4 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 $\hat{D}(t)$ is also reported (top). When $\rho = 0.8$, the number of parallel active threads 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 $N_a = W_{\max} = 53$, i.e., the number of a neighbors. At $t = 30s$, the HRC system reacts to the sudden system condition variation. In particular, for the high bandwidth peer a, N_a initially increases because its capacity was not fully exploited (its queuing delay still being smaller than D_0) and the number of its neighbors is increased from 53 to 61 due to the ingress of the 400 peers. Then, the increased system load has the effect of boosting the percentage of offers that are positively selected. Indeed, in heavy loaded conditions the source is generating more chunks than those the P2P system can actually deliver, thus inducing peers to accept all chunks offered by their neighbors. This effect obviously increases queuing delays experienced by chunks, so that after a quick transient, N_a nicely decreases, upload rate matches each peer's upload capacity, and queuing delay reaches the target D_0.

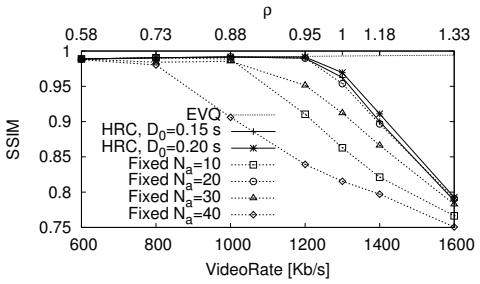

Figure 5: Average SSIM of HRC and non-adaptive schemes versus the system load. Simulation results, 2000 peers.

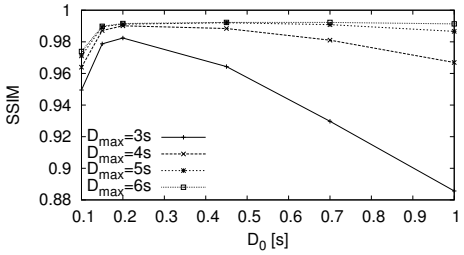

Figure 6: Average SSIM for different values of D_0 and D_{max}. $\rho = 0.95$. Simulation results, 2000 peers.

4.3. 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. 5 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 r_s 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 (Encoded Video Quality, EVQ, curve in the plot). 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., $N_a = 40$. 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, are variable in time due to interfering traffic, as seen in Figs. 3 and 4.

Fig. 6 reports the average SSIM obtained with HRC versus D_0 for different values of the playout delay D_{max}. $\rho = 0.95$. On the one hand, small values of D_0 lead to inefficient exploitation of peer upload capacity. Being the system load quite large, this leads to loss of chunks that impairs the video quality. On the other hand, D_0 represents the average queue delay that a chunk suffers at every hop it traverses; larger values of D_0 lead to larger delivery delays that might translate into chunk losses for small values of D_{max}, again impairing the QoE represented by SSIM index. Thus, smaller values of D_0 should be preferred. A good tradeoff is obtained for $D_0 \in [150, 200]$ms.

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.

5. Evaluation by Experiment

The HRC controller has been implemented into *PeerStreamer* [1] 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.

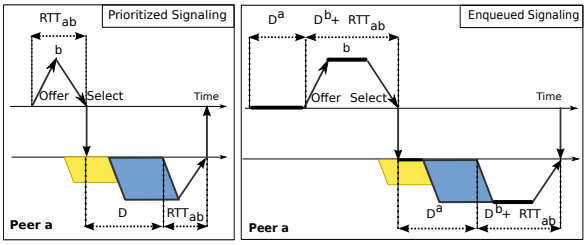

Figure 7: Schematic representation of the peer chunk trading mechanism with prioritized signalling -PS- (left) and enqueued signaling -ES- (right).

5.1. 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) let us first consider the scenario in which the access device of the bottleneck node supports separate queues: a high priority queue serves signalling packets, and a low priority queue serves data packets. The estimation of the round-trip time between a and b, RTT_{ab} (which does not include the queuing delay), could be easily carried out by exploiting the higher priority service offered to signalling packets: $RTT_{ab} = t^{(a)}_{\text{rx,select}} - t^{(a)}_{\text{tx,offer}}$;
ii) in the second scenario a single class of service is offered by the network devices, but peers are synchronized. Here, signalling messages are delayed by the transmission queue too, as sketched in left part of Fig. 7, but synchronization allows to measure the One-Way-Delay between a and b, OWD_{ab}, as the minimum of all $t^{(a)}_{\text{rx,select}} - t^{(b)}_{\text{tx,select}}$ estimates. Even some coarse synchronization, as the one provided by the NTP protocol, would suffice and errors (in the order of 1ms) would only marginally affect the HRC control, since its target queuing delay D_0 is of the order of 100ms;
iii) 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,

$$\hat{D}_{(a+b)} = D^{(a)} + D^{(b)} = t^{(a)}_{\text{rx,ack}} - t^{(a)}_{\text{rx,select}} - RTT_{ab} \tag{3}$$

[1] Source code is available from `http://peerstreamer.org`.

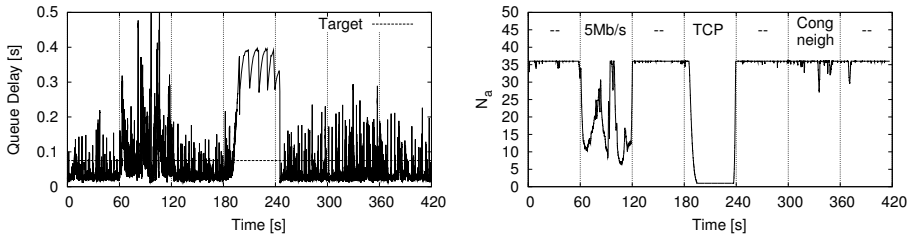

Figure 8: Queue delay (left) and N_a (right).

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 the queuing delay is estimated as in (3). At last we emphasize that the implementation of HRC requires to make the system robust to losses of signalling messages and chunks through the use of opportune timeouts (set to 1.5s in current implementation).

5.2. Experimental results

• **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 0.6Mb/s, 20 ms RTT is imposed on all links and $D_0 = 75$ms.

Fig. 8 reports the evolution of queuing delay (left) and of N_a (right) for peer a. During the first 60s, peer a uplink bandwidth (100Mb/s) is large enough to transmit all committed chunks. Since a queuing delay cannot reach the target, N_a stabilizes at the neighborhood size ($N_a = W_{\max} = 36$).

Decrease of available capacity - At time $t = 60$s, a uplink capacity is limited to 5Mb/s, inducing HRC to reduce the number of parallel signalling threads while queuing delay varies around the target value. From $t = 120$s to $t = 180$s, N_a stabilizes at the maximum allowed value, being uplink bottleneck removed.

Competing TCP traffic - 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.

Congestion at neighbor - From $t = 300s$ to $t = 360s$, peer b, one of a's neighbors, suffers congestion in its uplink: a TCP flow starts sending data from b to a, so that $D^{(b)}$ grows, possibly impairing $\hat{D}_{(a+b)}$. 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

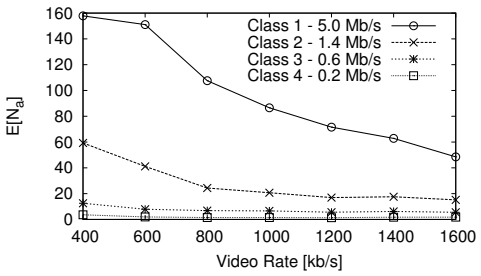

Figure 9: Average N_a for HRC when varying video rate r_s. Experimental results in swarm of 1000 peers.

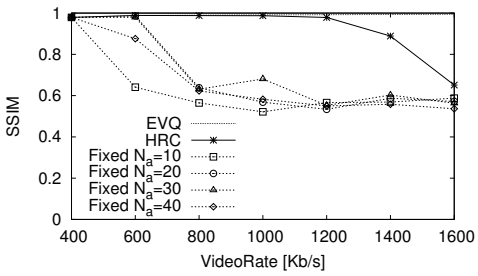

Figure 10: Average SSIM for HRC and non-adaptive schemes when varying video rate r_s. Experimental results in swarm of 1000 peers.

N_a is basically unaffected. The intuition indeed suggests that if the number of "biased" $\hat{D}$ estimates at peer a is limited, the system is still able to control the peer uplink queue by "filtering" out the few overestimated samples, thus limiting the impairment on N_a that adjusts the number of active signalling threads.

• **Large swarm** - We now present results collected by running the application in a controlled test-bed composed of 200 PCs. Each PC runs 5 copies of the application simultaneously, creating a swarm of 1000 peers. Also in this case, the average number of neighbors for each peer was approximately equal to 60. Each peer upload capacity has been artificially limited using a rate limiter embedded in our P2P-TV application which runs at packet level: 10% of peers have 5Mb/s, 35% have 1.6Mb/s, 35% have 0.64Mb/s and 20% have 0.20Mb/s, corresponding to an average per peer data link capacity of 1.32Mb/s. Latencies among peers randomly varies between 10ms and 20ms (so that the RTT varies in $[20, 40]$ms). The *Pink of the Aerosmith* video (352x240p resolution, 25fps, H.264/AVC Codec) has been encoded at different rates and "streamed" over the swarm looping the video 5 times. After discarding the initial 12min of video, each peers saves 100s of the received frames on disk. SSIM is then computed against the original YUV video for all video traces; then average SSIM is computed over all peers. Simple random overlay topology and random peer/random chunks selection are adopted. The playout delay D_{max} is set to 6s, the HRC queuing target D_0 is set to 200ms, and the maximum number of offer threads $W_{\max}$ is set to twice the number of current neighbors.

Fig. 9 reports the average number of active signalling threads N_a for each class of peers when HRC is enabled and video rate r_s is increased. The first evident thing is that HRC achieves the objective of adapting N_a to actual peer's upload bandwidth: e.g., for video rate 1.0Mb/s, $E[N_a]$ is 90 for high bandwidth peers, $E[N_a]$ is 22 for mid bandwidth peers, $E[N_a]$ is 5 for the low bandwidth peers and $E[N_a]$ is 1 for those peers with very low bandwidth. The second thing to notice is how N_a decreases when r_s increases and, therefore, the load ρ of the system grows; for higher loads, e.g., $r_s = 1.6$Mb/s, the chunks acceptance probability becomes higher inducing peers' uplink queue to grow so that N_a decreases. Instead, when system conditions are relaxed, e.g., $r_s = 0.6$Mb/s, queues

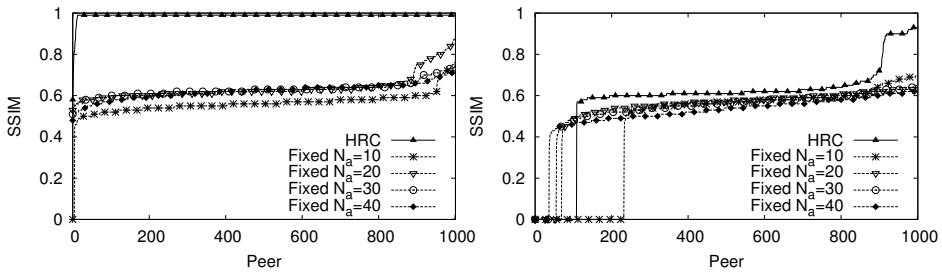

Figure 11: Average SSIM per peer distribution for HRC and non-adaptive schemes for video rates $r_s = 800$kb/s (left) and $r_s = 1600$kb/s (right). Experimental results in swarm of 1000 peers.

are short and peers - especially those whose upload bandwidth is large - increase the number of active trading threads to raise their transmission workload. Note that no fixed values would be suitable for any scenario.

Focusing on the Quality of Experience, again expressed with SSIM index [2], Fig. 10 compares HRC behavior against non adaptive schemes in which $N_a = 10, 30, 20, 40$ respectively. Results are similar to the one of Fig. 5: all schemes perform similarly when the system is under-loaded, e.g, $r_s = 400$kb/s, but as soon as r_s increases, HRC dramatically 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. Any fixed values would cause a mismatch, impairing the overall system performance. This is the fundamental concept behind HRC algorithm and it is even clearer when combining results in Fig. 10 with those shown in Fig. 9: by simply adapting the transmission window represented by N_a to peer's upload bandwidth, performances are greatly improved.

In Fig. 11 we report SSIM index for each peer for video rate of 0.8Mb/s (left plot) and 1.6Mb/s (right plot), respectively. The former corresponds to a case in which the system is under-loaded, the latter refers to a scenario in which system conditions are heavily stressed. Again HRC shows much better performance: if system load is low ($r_s = 0.8$Mb/s) HRC guarantees each peer to successfully play the whole video. On the contrary, any fixed value of N_a fails at successfully delivering the content, severely impairing the video quality; finally, when $r_s = 1.6$Mb/s the system has not enough resources to satisfy all peers demand, but, still, HRC shows better performance respect to whatever fixed scheme by efficiently allocating peers' upload capacity.

[2] SSIM is smaller than 1 since we are considering the encoding loss too.

6. Conclusions

In this paper we focused on the trading phase of mesh-based P2P-TV systems. We 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 uplink, which is today the typical bottleneck for P2P-TV systems. We implemented the proposed mechanism in a real client, coping with the actual implementation issues and presenting actual experimental results considering swarms up to 1000 peers. Our results show that HRC reduces chunks delivery delay and loss probability, providing much better Quality of Experience for users even when the system load is close to 1.

Acknowledgment

This work was supported by the EC under the FP7 STREP "Network-Aware P2P-TV Application over Wise Networks".

References

[1] L. Massoulié, A. Twigg, C. Gkantsidis, P. Rodriguez, Randomized decentralized broadcasting algorithms, in: IEEE INFOCOM, Anchorage, AK, 2007.

[2] S. Sanghavi, B. Hajek, L. Massoulié, Gossiping with multiple messages, in: IEEE INFOCOM, Anchorage, AK, 2007.

[3] T. Bonald, L. Massoulié, F. Mathieu, D. Perino, A. Twigg, Epidemic live streaming: optimal performance trade-offs., in: SIGMETRICS, Annapolis, MD, 2008.

[4] M. Zhang, L. Zhao, Y. Tang, J. Luo, S. Yang, Large-scale live media streaming over Peer-to-Peer networks through global Internet, in: P2PMMS, Singapore, 2005.

[5] D. Ciullo, M. A. Garcia, A. Horvath, E. Leonardi, M. Mellia, D. Rossi, M. Telek, P. Veglia, Network awareness of P2P live streaming applications: a measurement study, IEEE Transanctions on Multimedia 12 (1) (2010) 54–63.

[6] NAPA-WINE, http://www.napa-wine.eu (2008-2011).

[7] Z. Wang, A. C. Bovik2, H. R. Sheikh, E. P. Simoncelli, Image quality assessment: From error visibility to structural similarity, IEEE Transactions on Image Processing 13 (4) (2004) 600–612.

[8] A. C. da Silva, E. Leonardi, M. Mellia, M. Meo, A bandwidth-aware scheduling strategy for P2P-TV systems, in: IEEE P2P, Aachen, DE, 2008.

[9] R. Fortuna, E. Leonardi, M. Mellia, M. Meo, S. Traverso, QoE in Pull Based P2P-TV Systems: Overlay Topology Design Tradeoffs, in: IEEE P2P, Delft, The Netherlands, 2010.

[10] Y. Liu, On the minimum delay peer-to-peer video streaming: how realtime can it be?, in: ACM Multimedia, Augsburg, DE, 2007.

[11] T. Small, B. Liang, B. Li, Scaling laws and tradeoffs in Peer-to-Peer live multimedia streaming, in: ACM Multimedia, Santa Barbara, CA, 2006.
[12] A. Carta, M. Mellia, M. Meo, S. Traverso, Efficient uplink bandwidth utilization in p2p-tv streaming systems, in: IEEE Globecom, Miami, FL, 2010.
[13] S. Shalunov, G. Hazel, Low Extra Delay Background Transport (LEDBAT), Internet Draft draft-ietf-ledbat-congestion-02, IETF (July 2010).
[14] X. Zhang, J. Liu, T. Yum, Coolstreaming/donet: A data-driven overlay network for peer-to-peer live media streaming, in: IEEE INFOCOM, Miami, FL, 2005.
[15] F. Picconi, L. Massoulié, Is there a future for mesh-based live video streaming?, in: IEEE P2P, Aachen, DE, 2008.
[16] A. Couto da Silva, E. Leonardi, M. Mellia, M. Meo, Exploiting Heterogeneity in P2P Video Streaming, IEEE Transactions on Computers.
[17] http://www.tlc-networks.polito.it/traverso/papers/tr-2010-30-08.pdf.
[18] Meridian, http://www.cs.cornell.edu/people/egs/meridian/ (2008).
[19] E. Setton, J. Noh, B. Girod, Low latency video streaming over peer-to-peer networks, in: IEEE ICME, Toronto, CA, 2006.
[20] Z. Liu, Y. Shen, K. W. Ross, S. S. Panwar, Y. Wang, Layerp2p: using layered video chunks in p2p live streaming, IEEE Transaction on Multimedia. 11 (7) (2009) 1340–1352.