A performance comparison of hose rate controller approaches for P2P-TV applications

A Performance Comparison of Hose Rate Controller Approaches for P2P-TV Applications

S. Traverso[b,1,*], C. Kiraly[a], E. Leonardi[b], M. Mellia[b]

[a] *Bruno Kessler Foundation, Trento, Italy*
[b] *DET, Politecnico di Torino, Italy*

Abstract

The goal of this paper is to investigate rate control mechanisms for unstructured P2P-TV applications adopting UDP as transport protocol. We focus on a novel class of Hose Rate Controllers (HRC), which aim at regulating the aggregate upload rate of each peer. This choice is motivated by the peculiar P2P-TV needs: video content is not elastic but it is subject to real-time constraints, so that the epidemic chunk exchange mechanism is much more bursty for P2P-TV than file sharing applications. Furthermore, the peer uplink (e.g., ADSL/Cable) is typically the shared bottleneck for flows in real scenarios. We compare two classes of aggregate rate control mechanisms: Delay Based (DB) less-than-best-effort mechanisms, which aim at tightly controlling the chunk transfer delay, and loss-based Additive Increase Multiplicative Decrease (AIMD) rate controllers, which are designed to be more aggressive and can compete with other AIMD congestion controls, i.e., TCP.

Both families of mechanisms are implemented in a full-fledged P2P-TV application that we use to collect performance results. Only actual experiments – conducted both in a controlled test-bed and over the wild Internet, and involving up to 1800 peers – are presented to assess performance in realistic scenarios.

Results show that DB-HRC tends to outperform AIMD-HRC when tight buffering time constraints are imposed to the application, while AIMD-HRC tends to be preferable in severely congested scenarios, especially when the buffering time constraints are relaxed.

1. Introduction

P2P-TV applications are designed to offer real-time video streaming services exploiting the Peer-to-Peer (P2P) paradigm. In these systems, peers are arranged in a generic meshed overlay topology which is dynamically adapted and optimized by a

*Corresponding author
Email addresses: traverso@tlc.polito.it (S. Traverso), kiraly@fbk.eu (C. Kiraly), leonardi@tlc.polito.it (E. Leonardi), mellia@tlc.polito.it (M. Mellia)

[1] This work was funded by the European Commission under the FP7-Strep Project NAPA-WINE.

distributed algorithm, and neighboring peers are enabled to exchange data. The content to be delivered is chopped in segments, called *chunks* which are distributed among peers exploiting an epidemic approach. Because of this characteristics, unstructured P2P-TV systems may closely resemble P2P applications for file sharing, e.g., BitTorrent.

However, three peculiar aspects of P2P-TV applications mark a significant difference with respect to P2P file sharing applications like BitTorrent:

1) The content is not available in advance, but it is generated in real time at a single point (the source). As a consequence no seeds are available in the system.

2) New generated chunks must be delivered to all peers, within a deadline, the *total offset delay*, i.e., the lag between the chunk generation time at the source and the time the chunk has to be played at the peer. For live content distribution, it is highly desirable to keep this delay as small as possible so that chunks delivery times are minimized and peers can view the show in almost real-time. Total offset delay is controlled by limiting the number of chunks that peers can trade with other peers at any time.

3) As a consequence of the previous requirements, receiving data rate at peers can not be adapted, i.e., the application is not elastic: either the peer is in the condition of receiving chunks at the same rate they are generated at the source or it is not able to watch the show with an acceptable Quality of Experience (QoE). This implies that peers are not greedy (pre-fetching policies are indeed not possible, since content is generated in real-time), and the goal of the transmission rate controller is not to maximize the download/upload rate, but to guarantee that the total demand is sustained. That is, the system-wide total upload/download rate cannot exceed N times the video rate (where N is the number of peers in the system).

The above three characteristics heavily impact the overall design of P2P-TV applications. First, to control the delivery time, chunk size must be kept small. A common choice made in many P2P-TV systems is to deliver one video frame in a chunk. This causes the size of each chunks to be highly variable, due to the video encoding process nature [1, 2, 3].

Second, UDP is largely preferred to TCP, to avoid long retransmission delay and accelerate sending of chunks. Packets of the same chunk are transmitted back-to-back without sharing of the up-link capacity among different chunks [2, 3, 4, 5].

Third, to increase the ability of peers to retrieve chunks within a limited delay, every peer should rely on a sufficient large set of neighbors. Hence, the size of peers' neighborhoods to exchange data with is typically much larger than in BitTorrent (order of several tens versus five).

At last, scheduling algorithms must be designed having in mind the goal of distributing chunks to all peers within the limit imposed by the total offset delay [6, 7, 8, 9, 10, 11].

The literature about P2P-TV systems is mostly focused on the overlay topology design and on the modeling of scheduling algorithms to improve performance, and to the best of our knowledge, only a few works focus on transmission rate controllers for mesh-based P2P-TV systems (see Sec. 7 for a discussion). Indeed, most existing P2P live video systems do not consider the sender rate control problem explicitly, and they simply adopt a best-effort approach, where the senders try their best to serve the receivers by using as much up-link bandwidth as possible, without any rate control

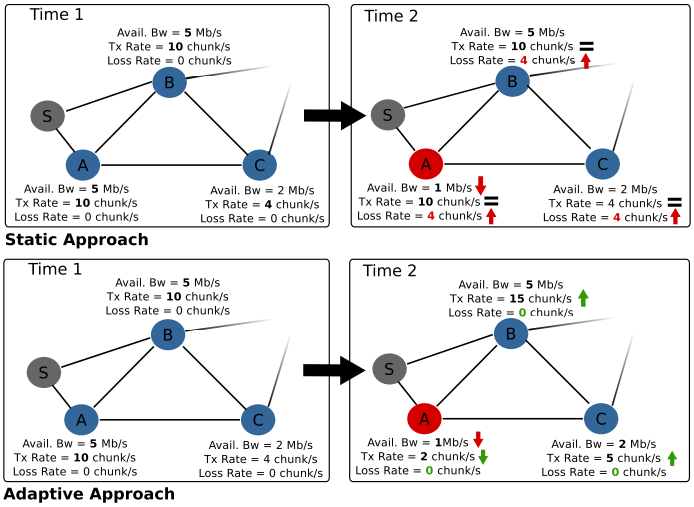

Figure 1: Simple example which illustrates the benefits of an adaptive mechanism for adjusting the transmission rate of chunks.

l. To the best of our knowledge, the only works that explicitly deal with the issue of regulating the transmission rate of peers are [12, 13]. In particular [13] compares a delay-based transmission rate controller against a simple scheme, where the chunk offering rate is fixed.

Considering instead the industrial side, the many commercial P2P-TV clients available in the Internet, e.g., PPLive, SopCast, etc., do not disclose their source code, making it impossible to understand which kind of rate controller they implement, and making unfeasible any experimental comparison with them.

In a nutshell, only little attention has been devoted to the problem of how to efficiently exploit the peer upload capacity by controlling the sending rate of peers. Considering the typical nowadays scenario faced by P2P applications, it is natural to assume that the bottleneck is provided by the peer up-link, e.g., it is located at the ADSL/cable up-link. In order to guess the right chunk sending rate most of P2P applications ask the user to manually set the available bandwidth, but bandwidth is known to vary in time due to background traffic condition (take the case of a shared wireless local network), and any static (mis)configuration often leads to an inefficient use of the available resources possibly inducing congestion and packet loss. Automatically tuning the transmission rate is thus essential i) to keep low chunk delivery delays, ii) to guarantee that all users fairly participate to the chunk dissemination process in a way proportional to their available resources, and iii) to let the system be more robust when faced with (abrupt) changes of the network conditions.

For instance, Fig. 1 describes by means of a simple example the benefits brought by an adaptive approach able to match the transmission rate of chunks to the available bandwidth at the peer. A few peers form a small mesh network where S represents the source peer: in the top part of the figure the sending rate of the chunks at each peer is fixed and does not change in time. Moving from Time 1 to Time 2, peer A

undergoes a bandwidth reduction from 5Mb/s to 1Mb/s, but its chunk transmission rate does not adapt to the new condition, introducing delays in the delivery of the chunks and, thus, losses. Notice, that peers B and C could actually replace peer A in the chunk distribution process (they have enough bandwidth), but they do not, since their transmission rate is fixed too. The bottom part of the figure depicts what happens when an adaptive regulation is available: in Time 2 peer A undergoes a bandwidth reduction, but its chunk transmission rate is nicely reduced to match its new upload capacity. Moreover, peers B and C increase their transmission rate to replace peer A in the chunk distribution. Observe, that by doing so, no delivery delays nor losses are introduced.

For these reasons, embedding an application level rate controller algorithm that is able to dynamically adapt the rate at which chunks are transmitted by peers becomes a crucial task in P2P-TV applications design. Observe that the same motivations drove to the development of Ledbat [14] in the context of P2P file sharing, and it is now embedded into the majority of BitTorrent clients.

Results presented in [13] show that in P2P-TV systems it is preferable to adopt transmission rate controllers rather than best-effort and static schemes. This paper goes further, and investigates two families of rate control algorithms for P2P-TV systems. We present, discuss and validate possible alternatives by means of thorough experimental campaigns. In particular, we address the following key points:

1. whether it is preferable to implement i) classical end-to-end transmission control mechanisms which manage the rate of individual flows as for TCP, Ledbat [15] or TCP Friendly Rate Control for multimedia, e.g., TFRC [16] [2]; or ii) aggregate forms of rate control which directly adjust the total chunk transmission rate at the peer, which we call "Hose Rate Control" - HRC [13].
2. whether it is preferable that the rate control mechanism attempts a tight control on the transmission queue delay (as for Ledbat or TCP Vegas) or embraces a more aggressive loss-based approach inspired to TCP Additive Increase Multiplicative Decrease (AIMD).

For what concerns the first design choice, we believe that hose aggregate controllers are more effective for P2P-TV applications as motivated in Sec. 3.1 and in Sec. 4.

For what concerns the second design dilemma, it is a priori much unclear which may be the more convenient choice. By controlling the transmission queue delay, indeed, we gain a direct control on the chunk transfer time. This can be particularly welcomed given the large queue size (and thus delay) that are encountered in today Internet, i.e., the so-called "bufferbloat" effect [17]. However as side effect, we obtain a less-than-best-effort rate controller compared, e.g., to TCP sources. Adopting an AIMD paradigm, on the contrary, we are potentially able to compete with TCP for the upload bandwidth of peers, at the risk of loosing the control on chunk transfer delay.

[2] In general, the end-to-end transmission control principle requires both peers, A and B, involved in the communication to participate at adjusting the transmission rate of the flow. A typical example is TCP, for which A regulates its sending rate by counting the number of ACKs sent back by B. Therefore, if peer A opens a connection towards each of its neighbors in parallel, it can actually regulate its total chunk transmission rate by adjusting the transmission rate of each single connection independently.

In the following we summarize the main contributions of this paper:

- We describe two novel hose rate control mechanisms. Both algorithms have been implemented in PeerStreamer[3], the full-fledged P2P-TV application developed within the EU FP7 NAPA-WINE project[4].
- We present the results of an extensive experimental campaign, conducted in a large controlled test-bed involving up to 1800 peers, considering different scenarios, with and without TCP/UDP interfering traffic. Such results are complemented by running experiments in the wild Internet employing more than 400 PlanetLab nodes emulating a churning scenario. In this paper we present a subset of results equivalent to a total amount of 360 hours of tests. Furthermore, the performance of each scheme is assessed by measuring the actual QoE on delivered video streams.
- Finally our results show that the delay-based hose rate controllers (DB-HRC) outperform loss-based controllers (AIMD-HRC) when tight buffering delay constraints are imposed to applications and in not heavily congested conditions. In scenarios in which congestion is induced by large amount of competing TCP traffic, AIMD-HRC tends, instead, to provide better performance. We emphasize, however that in such scenarios peers typically experience severe QoE degradations.

The rest of the paper is organized as follows. In Sec. 2 we describe how P2P-TV systems work in general and which are the constraints designers have to face. In Sec. 3 we discuss whether end-to-end congestion controllers represent the best solution for P2P-TV systems with respect to aggregate type approaches. Secs. 3.2 and 3.3 describe the algorithms behind DB-HRC and AIMD-HRC, and we present in Sec. 4 toy-case experiments to show their differences in practice. Sec. 5 presents a comprehensive performance evaluation conducted both in a controlled environment and in PlanetLab (Sec. 5.1 to Sec. 5.4). In Sec. 6 we evaluate the hose rate controller sensitivity to the parameters and to the peer upload capacity distribution, their performance when increasing the scale and when adopting different chunk scheduler policies. Finally, we discuss the related work in Sec. 7, and we conclude the paper with Sec. 8.

2. System description

We consider a typical unstructured P2P-TV application in which a source segments the video stream into chunks and injects them into the system. Let N be the total number of peers which are logically arranged in a graph called *overlay topology*. The overlay topology is defined by the set of peers and virtual links connecting them and may be dynamically adapted using smart algorithms [18, 19]. Since the actual design of the overlay topology is out of the scope of this paper, we consider the simplest case

[3] Available at `http://www.peerstreamer.org`
[4] `www.napa-wine.eu/`

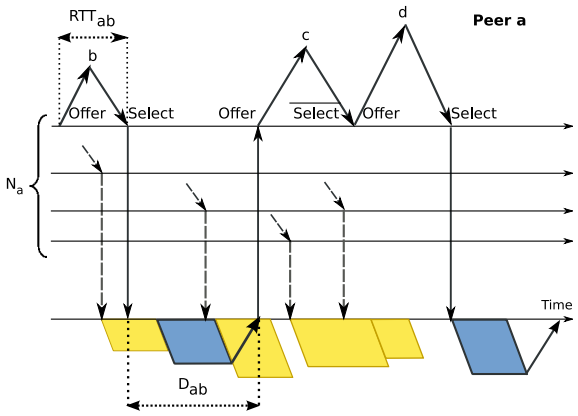

Figure 2: Schematic representation of the chunk trading mechanism.

in which the overlay network is built on a random basis, a common assumption in the literature [6, 7, 8].

Chunks are transmitted by peers to their neighbors in a swarm-like fashion. Since video chunks must meet strict delay constraints, 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 choose that 1 chunk contains exactly 1 video frame; thus the average chunk size is 5 kB for a 1 Mb/s encoding rate of a 25 fps video. 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 frame headers.

As already said, the system must deliver newly generated chunks to peers within the deadline provided by the *total offset delay*, D_{OS}, i.e., the lag elapsing between chunk generation time at the source and its playing time at the peer. If the chunk is received after its playing time, it is not anymore useful for the peer and it will be then skipped by the player.

To avoid complex synchronization procedures, typically the constraint on the total offset delay is indirectly imposed by limiting the number of chunks Z_0 that every peer can trade with other peers. Z_0 is often called *trading window*. At every time t, the trading window of peer a can contain the last Z_0 consecutive chunks which are in advance with respect to the chunk which is currently played by a. The resulting offset delay will be different for every peer, $D_{OS}(a)$. We notice that Z_0 (expressed in seconds of video) determines both i) the buffering time at the peer, which corresponds to the delay experienced by the peer when it joins the channel (i.e., its zapping time), and ii) $D_{OS}(a)$ that peer a suffers. Observe, however that the relation between Z_0 (in seconds) and $D_{OS}(a)$ is complex and not completely predictable, and only simple bounds can be derived: on the one hand, $D_{OS}(a)$ cannot be smaller than the local buffering time; on the other hand, chunks accumulate delay at every hop while traversing the overlay topology. Thus, $Z_0 \leq D_{OS}(a) \leq Z_0 H$, where H is the maximum distance of peers

from the source on the overlay topology (typically $H \approx \log N$). We need to reduce as much as possible both Z_0 and H to achieve a tight control on the buffering time and on D_{OS}.

Finally, we can define the overall system load ρ as in [20]

$$\rho = \frac{NV_r}{b_s + \sum_p b_p} \tag{1}$$

where V_r is the video encoding rate, b_s the bandwidth available at the source and b_p is the up-link capacity of peer p. Note that if load is smaller than one, peers cannot saturate their upload capacity.

2.1. Chunk Trading Mechanisms

The signaling mechanism used to exchange chunks is a trading scheme similar to the one used in other mesh-based P2P-TV systems [21, 22, 23]. A chunk is sent from a peer to one of its neighbors after a trading phase. In order to avoid idle times and long periods of inactivity, peer a maintains a certain number of parallel *trading threads*, N_a. Each of these evolves as follows:

1) Peer a chooses one of its neighbors b and sends it a signaling message, called *offer* message, that contains the set of chunks a possesses in its trading window.

2) Upon receiving the offer message, b replies with a *select* message to request one 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 again by b and it cannot be advertised until it has been fully downloaded.

3) When the select message is received by a

a) if a chunk was requested in the select message (*positive select*), a 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, or 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 requested chunks. Timers protect the waiting for messages so that in case no reply (or chunk) is received within a timeout, the status is reset. Fig. 2 represents the signaling messages and chunks exchanged by peer a with its neighbors over time. In particular, signaling 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 select chunks to download – known as the "chunk selection"; 3) the number N_a of parallel trading threads peer a handles.

For the *peer selection* and *chunk selection* rules we adopt the "Random Peer - Latest Useful Chunk" policy, which has been shown to provide good performance [9, 24]. According to this policy, 1) peer a chooses the peer b to contact uniformly at random within the set of its neighbors; 2) peer b always selects the most recent chunk it needs among the chunks offered by a. As for 3) N_a is the key parameter on which we focus our attention in the next sections.

3. Rate Controller Design

As already said, it is of paramount importance to promptly adapt the rate at which new offers are generated so to match peer's upload available bandwidth. If it is too small, the peer cannot effectively exploit its upload bandwidth. On the contrary, if the rate is too large, the bottleneck transmission queue of the peer, typically located at the ADSL/cable gateway, gets easily congested. This of course induces loss of packets that have to be recovered through packet and/or chunk based selective ARQ mechanisms. Resulting lengthy retransmissions would impair the chunk transferring delay.

In the following sections we critically discuss the main design choices for a possible rate controller. We present two possible schemes that embrace different philosophies.

3.1. Hose Vs End-to-End rate controllers

For what concerns the choice between classical end-to-end and hose (aggregate) rate controllers, our preference for hose rate controllers is motivated by the following considerations:

- The system bottleneck is typically located close to the peer, e.g., at the ADSL/cable up-link. Thus all flows would be bottlenecked at the same point.
- In P2P-TV applications, every peer exchanges simultaneously chunks with several tens of other peers in a rather intermittent fashion. In addition, the rate at which new neighbors are acquired can be very fast [3, 18]. Thus, several concurrent flows are present, and each carries a very bursty traffic, with low average rate, and maximum rate bounded by the encoding rate. For these reasons, individual flows hardly reach a steady state condition (in which the transmission rate can be effectively controlled). Furthermore, the actual working point results to be very low.
- Chunk scheduling algorithms determine the way in which the system-wide upload bandwidth of peers is shared. End-to-end rate controllers would necessarily interfere in a complex and unpredictable way with the dynamics of chunk scheduling algorithms. Hose rate controllers represent a much more natural choice for the P2P-TV applications, because they impose just a limit to the aggregate rate at which chunks can be transmitted at a peer, guaranteeing to the chunk scheduler the degree of freedom to share the peer bandwidth in an arbitrary way.

While an end-to-end rate controller a la TFRC requires to maintain the status of each single connection, the hose rate controller can be obtained by controlling the single state variable N_a, which is the equivalent of the window size in a window protocol. N_a, indeed, represents the maximum allowed number of pending chunks (i.e., chunks for which an offer has already been issued, but no acknowledgement has still been received). The rationale beneath controlling N_a is that it regulates the workload of peer a transmission queue: if N_a is too large, the queue gets congested deteriorating system performance; if it is too small, the queue gets frequently idle, and a fraction of the upload bandwidth of a is wasted (as depicted in Fig. 2).

3.2. Delay Based HRC

The basic idea of DB-HRC is rather simple: it adapts N_a on the basis of 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. (refer to Fig. 2): let W_a be the internal control variable (that we call window in the following), which takes real values: $N_a = \max(1, \lfloor W_a \rfloor)$.

For every neighbor peer b, peer a maintains an estimate of the minimum Round Trip Time, $minRTT_{ab}$. RTT estimate can be computed as the difference between the time a select message is received and the one the corresponding offer was sent,

$$RTT_{ab} = t^{(a)}_{\text{rx,select}} - t^{(a)}_{\text{tx,offer}} \tag{2}$$

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 obtains a sample of the delay D incurred by the chunk in the transmission queue, as

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

$\hat{D}$ is then compared with a prefixed target value, D_0, and W_a is updated according to a *simple proportional controller* rule:

$$W_a(n) \leftarrow W_a(n-1) - K_p(\hat{D} - D_0) \tag{4}$$

$$N_a(n) \leftarrow \max(1, \lfloor W_a(n) \rfloor) \tag{5}$$

D_0 represents the target queue delay which can be set in the order of hundreds of ms, and K_p is the proportionality gain. Interestingly, it is possible to prove that the DB-HRC controller is stable for a wide set of K_p [25].

We have $\Delta N_a = N_a(n) - N_a(n-1)$. If $\Delta N_a = 0$, the number of active threads is not changed, and peer a restarts the current thread by sending one new offer. If $\Delta N_a \geq 1$, the number of active threads is increased, and peer a sends $\Delta N_a + 1$ offers to its neighbors. At last, if $\Delta N_a \leq -1$, the number of active threads is decreased, and current thread is stopped (no new offer is sent).

3.2.1. Implementation Issues

The most critical part when undergoing the actual implementation of the DB-HRC scheme is the estimation of queuing delay (3). Two different cases can be considered in principle: i) the application can exploit priority mechanisms to differentiate signaling and data information at the ingress to the network, ii) the most challenging and intriguing case is the one in which no priority is available.

The first scenario is represented in Fig. 2. Here the bottleneck link supports separate queues: a high priority queue serves signaling packets, and a low priority queue serves data packets. Signalling packets thus are not delayed by queued chunks. This feature is offered by most of nowadays ADSL/cable devices that support multimedia services. In

this case, the estimation of $\hat{D}$ is straightforward as in (3), while (2) allows to estimate the RTT.

If no priority policy is provided, measurements include both the queuing delay at peer a, denoted by $D^{(a)}$, and the queuing delay at b, $D^{(b)}$, i.e., it is only possible for a to estimate the sum of the queuing delays,

$$\hat{D}^{(a+b)} = t^{(a)}_{\text{rx,ack}} - t^{(a)}_{\text{rx,select}} - minRTT_{ab} = D^{(a)} + D^{(b)} \tag{6}$$

$minRTT_{ab}$ can still be estimated as the minimum over all RTT samples (as in Ledbat), while it is impossible to decouple $D^{(a)}$ and $D^{(b)}$ from $D^{(a+b)}$. The HRC algorithm at peer a is coupled with the HRC control of all its neighbors.

The reader may be tempted to conclude that in this last scenario the DB-HRC mechanism will be hardly able to properly work, and it would be impossible to distinguish between a local congestion at a and a congestion at neighbor b. However, notice that the effect of the local congestion at a and the one present at b are differently weighted in (4). Indeed, only the *fraction* of delay samples $D^{(b)}$ related to b are biased by b congestion, while *all* the samples contain $D^{(a)}$ as component. For these reasons the algorithm is marginally affected by *remote* congestion at a few neighbors. We support this intuition by experimental evidences. At last, we emphasize that the implementation of DB-HRC scheme requires to make the system robust to losses of offer/select messages, through the use of opportune timeouts (set to 1.5 s in current implementation).

In this paper, we focus on the second and more challenging scenario where no priority support is offered.

3.3. AIMD-HRC

In this case, according to the AIMD philosophy, loss indications are used to control N_a.

To detect losses, a time-out τ is set for every chunk[5] that a enqueues in the transmission queue. If the timeout expires before the reception of the corresponding acknowledgement, the chunk is assumed to be lost. The timeout τ must be opportunely set up to guarantee a prompt reaction to congestion, while avoiding false alarms. For the sake of simplicity, we chose $\tau = 1.5$ s[6].

More in detail, upon acknowledge reception, a increases W_a according to

$$W_a(n) \leftarrow W_a(n-1) + K_A, \;\; K_A > 0 \tag{7}$$

Instead , upon chunk loss, W_a is reduced according to

$$W_a(n) \leftarrow \frac{1}{K_M} W_a(n-1), \;\; K_M > 1 \tag{8}$$

(7) leads to an approximate *additive increase* behavior in saturated conditions, i.e., when the transmission queue is continuously backlogged and the rate at which chunks

[5]We recall that our mechanism is designed to operate at chunk level, and not at packet level as in TCP.

[6]Tuning of timeout can be engineered as in TCP. While this would guarantee prompter loss detection, it would have no effect on congestion control.

that are transmitted is determined by the bottleneck queue service rate; in non saturated conditions, i.e., when the chunk transmission rate is constrained by W_a, (7) leads to a *multiplicative increase* [26]. Further observe how the additive rate in saturated conditions is, by construction, proportional to the peer up-link capacity. (8), instead, guarantees a *multiplicative decrease* with $K_M > 1.0$.

As in TCP, the AIMD-HRC rate control causes the bottleneck queue to exhibit the famous saw-tooth behavior alternating periods in which there is congestion (congestion epochs) to periods in which the queue is depleted because of window reductions. For effect of chunk loss burstiness, multiple chunks are typically lost within a congestion epoch, which would cause multiple window decreases. To prevent cascades of window reductions, we impose that decrease actions are inhibited for a time interval ΔT_{loss} after the first reduction, thus ignoring further loss indications. ΔT_{loss} must be properly match the typical duration of transmission queue congestion epochs.

3.4. Limiting the value of N_a at low loads

An important issue may arise at low loads. According to (1), if $\rho < 1$, any allocation of upload rate that meets the total download rate demand would be a feasible solution. Thus, peers may not be able to sustain a sufficient high transmission rate either to meet the D_0 target (DB-HRC) or to congest the transmission queue (AIMD-HRC). In this case the number of offers N_a will potentially grow large, inducing a useless increase of the signaling traffic, and eventually endangering the overall system performance. To avoid this phenomenon, it is desirable to clip the value of N_a with some ingenuity. Indeed, since typical expected values of N_a heavily depends from a set of a priori unpredictable parameters, such as peer's upload bandwidth and chunk size, it is difficult to choose an a priori maximum value for N_a.

We propose to use the ratio p_a between the received positive selects and the total offers issued by a. This ratio becomes significantly smaller than 1 when the peer is not able to effectively offer useful chunks, i.e., when there is a surplus of capacity in the systems and chunks are available at many neighbors.

In more details, we clip W_a (and consequently N_a) when the ratio p_a falls below a given threshold, p_0. That is:

$$W_a(n) \leftarrow \begin{cases} W'_a(n) & \text{if } p_a > p_0 \\ \min(W'_a(n), W_a(n-1)) & \text{if } p_a \leq p_0 \end{cases} \tag{9}$$

where $W'_a(n)$ is computed according either to DB-HRC update rule (4) or the AIMD-HRC ones, (7) and (8), and p_a is obtained by using a standard sliding window technique.

3.5. Delay based Vs AIMD: a critical discussion of Pros and Cons

DB-HRC and AIMD-HRC embrace two rather different and complementary philosophies to control rate at which peers send their data. When $\rho < 1$ and $p_a > p_0$, i.e., when the HRC is called to effectively control each peer upload rate, DB-HRC brings the system to work at an operating point in which no transmission queues are overloaded. This would be in principle a desirable property that guarantee a better control of the chunk transfer delay (which plays a fundamental rule in chunk distribution) with

respect to AIMD-HRC. Considering the bufferbloat phenomenon, i.e., the excessive presence of buffering at networking devices that can induce queuing delay to grow up to 10 s [17], tightly controlling the delay looks like an appealing opportunity. Under AIMD-HRC, indeed, the transmission queue of peers is subject to the classical sawtooth effect induced by AIMD sources, periodically exhibiting congestion epochs. As a consequence the chunk transfer delay is not well controlled by the AIMD-HRC mechanism as by DB-HRC. This could induce some performance degradation, especially when the sliding window Z_0 is very tight (and large buffers are encountered).

On the contrary, when the system has to compete with TCP flows, DB-HRC renounces to compete for the bandwidth while AIMD-HRC, if properly set, exhibits a more aggressive behavior that place it in a better position to rival TCP. However, in severely congested scenarios ($\rho > 1$), for effect of chunk losses and excessive delay, P2P-TV applications experience performance degradations that make the content almost non usable by the users. Thus it is questionable whether it is really beneficial to aggressively compete for upload capacity. We will discuss these issues in the experimental evaluation (Sec. 5).

4. Algorithm Behavior - Introductory Results

To better clarify how hose rate control schemes works, in this section, we report an introductory set of experimental results. We consider a simple overlay topology in which the source s is connected to a HRC-enabled peer a which in turn is connected to 30 other peers so that a upload capacity is used to serve them. We then focus on the upload link of a where we impose known conditions. In particular, the Linux Traffic Control tool `tc`, is used to shape the link capacity, queue size (fixed to 500ms) and latency (fixed to 25ms). The `iperf` traffic generator is used to inject competing TCP traffic flow. Video rate is $V_r = 0.8$Mb/s, encoded using H.264/AVC standard codec.

4.1. Transient analysis

Fig. 3 (for DB-HRC) and Fig. 4 (for AIMD-HRC) report the bottleneck link traffic in the top plot, the evolution of the $\hat{D}^{(a+b)}$ in the second plot, the evolution of N_a (left y-axis) and p_a (right y-axis) in the third plot, and the cumulative number of timeouts in the bottom plot. For DB-HRC, the target delay $D_0 = 50$ ms, while $K_p = 0.98$. For AIMD-HRC we use $K_A = 0.1$, $K_M = 1.5$, $\Delta T_{loss} = 1.5$ s, $T_{ack} = 1.5$ s. Finally, $p_0 = 0.5$ in both cases.

Clipping state - During the first 60 s, a up-link capacity is set to 40 Mb/s, i.e., large enough to transmit all required chunks to all 30 connected leechers. a thus cannot saturate its upload bandwidth and N_a regulation is mainly determined by the clipping mechanism associated to p_a that prevents N_a to grow without control. Notice the variability of offered traffic around the average nominal upload traffic of $30V_r \approx 32$ Mb/s (excluding overheads) which is due to the burstiness of the encoded video stream.

Decrease of available capacity - a up-link capacity is decreased first to 25 Mb/s at time $t = 60$ s, then to 15 Mb/s at time $t = 120$ s, and finally to 5 Mb/s only at time $t = 180$ s. In all cases, this results less than the required capacity, so that the offered traffic saturates the available capacity, and p_a goes to 1. Both algorithms react

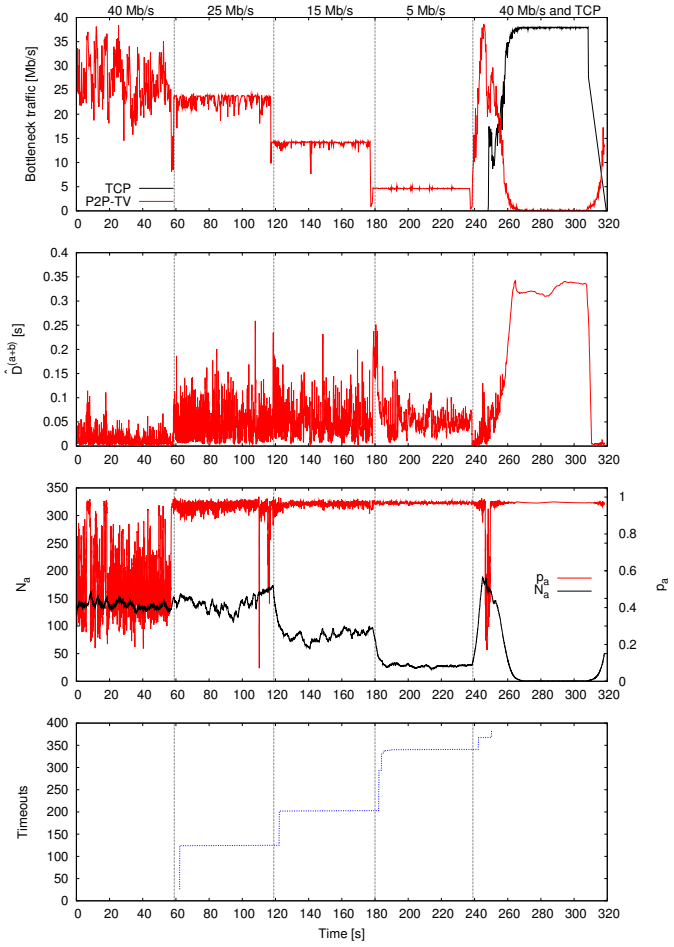

Figure 3: DB-HRC with $D_0 = 50$ ms; from the top: bottleneck link traffic, $\hat{D}^{(a+b)}$, N_a and p_a, cumulative number of timeouts.

to the various decreases by reducing N_a. After each bandwidth abrupt decrease, a temporarily congestion is experienced, which is reflected by the burst of threads timeouts caused by the loss of chunks. Both algorithms react to the new bandwidth conditions quite fast. After such transient phase, observe how DB-HRC adapts N_a to the new situation, sharply controlling the (average) chunk delay so that the queue can absorb the eventual burst of packets due to large chunks. Practically no loss (no timeouts) are registered. On the contrary, AIMD-HRC exhibits the classical saw-tooth behavior leading to periodic congestion epochs, reflected by a timeout expiration process which exhibits a strong burstiness. As said, this burstiness constitutes a challenge for the rate control scheme, which needs to be protected from consecutive timeouts by setting $\Delta T_{loss} = 1.5$ s.

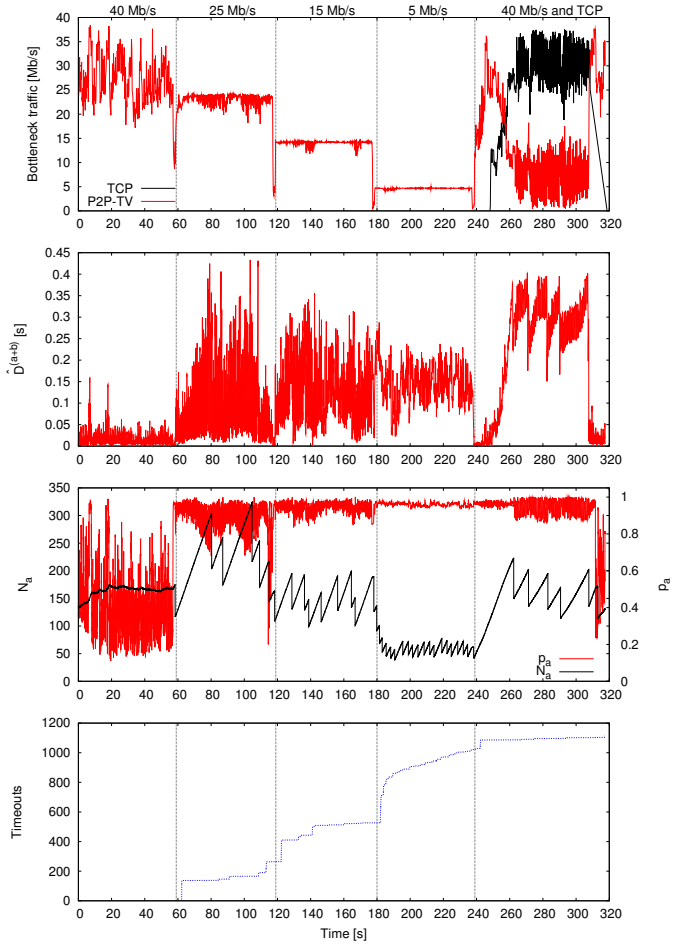

Figure 4: AIMD-HRC with $K_A = 0.1$, $K_M = 1.5$; from the top: bottleneck link traffic, $\hat{D}^{(a+b)}$, N_a and p_a, cumulative number of timeouts.

Competing TCP traffic - At time $t = 240$ s the full 40 Mb/s up-link bandwidth is available again; however, at $t = 250$ s a greedy TCP flow starts competing for the available upload bandwidth. Observe how DB-HRC reduces N_a to the minimum unitary value as expected, while AIMD-HRC is tries to compete against TCP (whose share of capacity is reported as well in top plot). Whereas AIMD-HRC vies obtaining some bandwidth share, DB-HRC renounces de facto to compete with TCP. Note however that AIMD-HRC tends to obtain less bandwidth than TCP for this parameters setting.

Fig. 5 shows the behavior of a more aggressive parameter setting for AIMD-HRC: $K_A = 0.5$, $K_M = 2.0$. This results in more frequent increase/decrease oscillations, which causes a higher frequency of congestion epochs (notice N_a curve, and the number of suffered timeouts). AIMD-HRC obtains thus larger share of capacity when competing against TCP traffic, at the cost of significantly increasing congestion at the

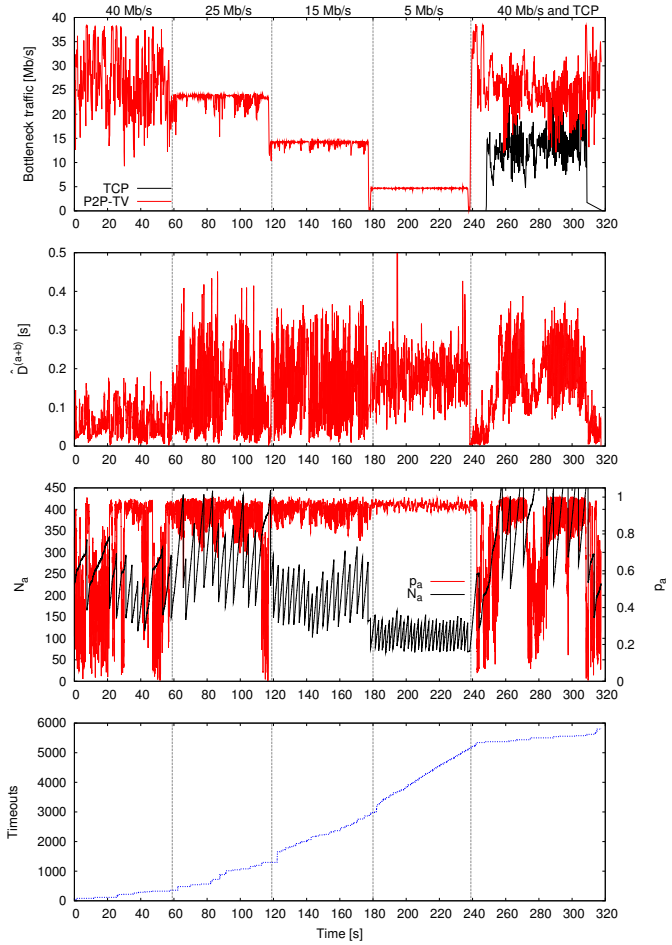

Figure 5: AIMD-HRC with $K_A = 0.5$, $K_M = 2.0$; from the top: bottleneck link traffic, $\hat{D}^{(a+b)}$, N_a and p_a, cumulative number of timeouts.

queue. In turn, the number of timeout expirations grows much faster, especially when no TCP flow is active. In the next section we will better investigate which are the possible negative consequences on the user perceived QoE when adopting a too aggressive AIMD-HRC scheme.

Our experiments with a third HRC driven by a TRFC-like controller (TFRC-HRC) show very poor performance (available in the supplemental material of this paper). Indeed, TFRC is designed to work for a context in which the the bandwidth bottleneck is in the network backbone. In such a context, the packet loss process can be considered as an exogenous process (i.e. a process whose dynamics are independent from the behavior of the *single* controlled source). In our case, the bottleneck is represented by the peer up-link, where multiplexing is very limited, and the packet loss process is induced by the sender behavior (self induced congestion). This makes hard formula-

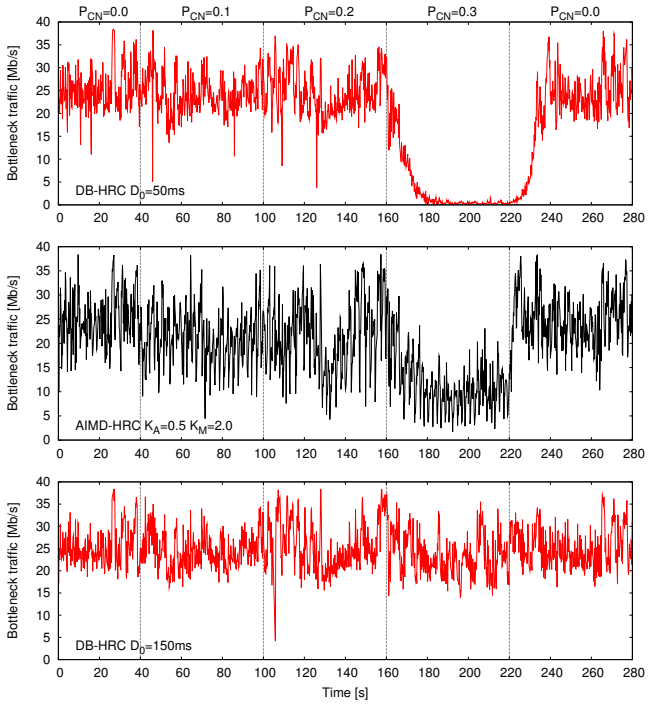

Figure 6: Evolution of the transmitted rate when an increasing number of neighbors gets congested. from the top: DB-HRC with $D_0 = 50$ ms, AIMD-HRC with $K_A = 0.5$, $K_M = 2.0$, DB-HRC with $D_0 = 150$ ms.

based congestion control like TFRC to work.

4.2. Impact of congestion at neighbors

Now we move to a slightly different scenario in which we vary a percentage P_{CN} of a neighbors which are affected by congestion on their up-link. Congestion is induced by competing greedy TCP flows sending data from the peer to a server. The goal is to verify whether DB-HRC and AIMD-HRC are able to discriminate between *local* and *remote* congestion and thus to exploit the upload bandwidth at peer a even when some of its neighbors are congested. As already explained, this scenario appears to be particularly critical for DB-HRC whose delay samples in (6) are directly affected by the congestion at neighbors. Top and middle plots of Fig. 6 reports the transmission rate for peer a for DB-HRC and AIMD-HRC, respectively. We have chosen the most aggressive parameter setting for AIMD-HRC. Available capacity is 40 Mb/s. For the first 40 s no neighbor peers are congested ($P_{CN} = 0.0$), then the fraction of congested neighbors P_{CN} increases so that $P_{CN} = \{0.1, 0.2, 0.3\}$ at time $t = \{40, 100, 160\}$ s; finally, at $t = 220$ s, $P_{CN} = 0.0$.

Observe that congestion at neighbors does not significantly affects a transmission rate, as long as P_{CN} keeps smaller than 0.3. Beyond this point, $D^{(a+b)}$ is strongly affected by the congestion at neighbors; thus, a reduces its transmission rate, maintaining

	DB-HRC		AIMD-HRC			Shared		
Parameter	D_0	K_p	K_A	K_M	ΔT_{loss}	T_{ack}	p_0	Z_0
Value	150-250 ms	0.98 ms^{-1}	0.3-0.5	1.5-2.0	1.5 s	1.5 s	0.5	50 (2 s) - 150 (6 s)

Table 1: Default values for DB-HRC and AIMD-HRC parameters.

only $N_a \simeq 1$ offer thread.

Interestingly, AIMD-HRC exhibits similar impairment, even if in a much less evident way. This is due losses of acknowledgement messages on the congested reverse paths, which causes timeouts to be triggered[7]. As depicted in the bottom plot of Fig. 6, the robustness of DB-HRC can be improved by increasing D_0 to 150 ms. Indeed, assuming $D^{(a)} \approx 0$ (no local congestion) and $D^{(b)} \leq 500$ ms (remote congestion with 500 ms queue size), we have that $D^{(a+b)} \approx P_{CN} D^b$ ms $< D_0$.

4.3. Final Remarks

In these simple introductory results, both DB-HRC and AIMD-HRC achieve the goal of adapting the chunk offering rate to the available bandwidth. When competing with TCP traffic, DB-HRC completely surrenders, whilst AIMD-HRC can maintain some bandwidth share, but tends to obtain less bandwidth than TCP.

5. Performance Comparison

In this section, we compare the performance of DB-HRC and AIMD-HRC in more significant scenarios. We start presenting results obtained over an emulated test-bed (Sec. 5.1, 5.4 and 5.4), then we move to a real Internet scenario (Sec. 5.4).

We emphasize that even if emulated test-bed scenarios may appear somehow artificial, they allow to have the full control of almost every network parameter (such as peer upload bandwidth and latency between peer pairs). In such a controlled environment we run different algorithms exactly in the same conditions obtaining results that are fully reproducible and directly comparable. On the wild Internet, instead, several parameters (such as the peer available bandwidth) are out of our control, thus performance of algorithms is affected by "unknown" environmental conditions that make results not completely reproducible. To increase the reliability of our predictions, all experiments were repeated 3 times, and averages are considered.

5.1. Emulated scenario - 200 peers

We start by presenting results collected by running the application in a controlled test-bed composed of 200 PCs. Each PC runs PeerStreamer, so that a swarm of 200 peers is obtained. Each peer's upload capacity has been artificially limited using the Linux Traffic Control tool `tc`: 10% of peers have 5 Mb/s, 35% have 1.6 Mb/s, 35%

[7]Notice that cumulative/selective acknowledgement policies are not straight-forward to protect from loss on the return path.

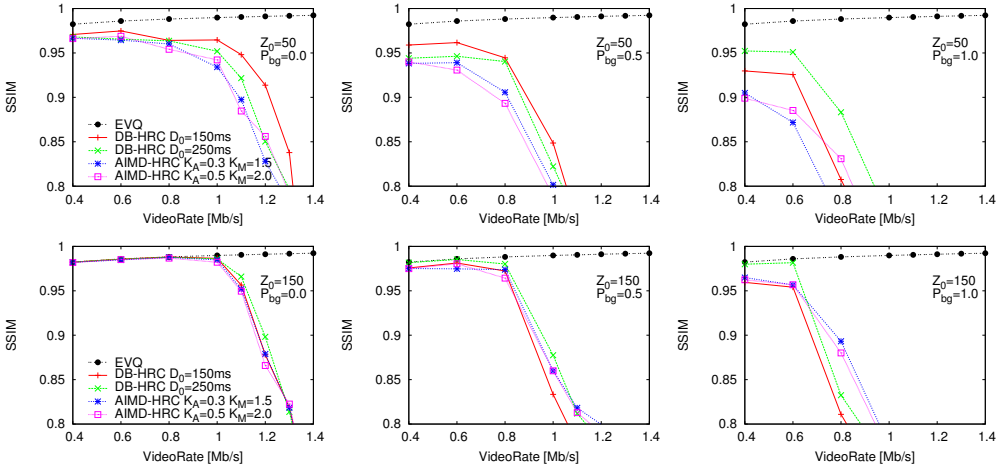

Figure 7: SSIM vs Video Rate when no, 50%, 100% of peers are affected by competing ON-OFF TCP sources for $Z_0 = 50$ (on the top) and $Z_0 = 150$ (on the bottom).

have 0.64 Mb/s and 20% have 0.20 Mb/s, corresponding to an average per peer Data-Link capacity of 1.32 Mb/s. Down-link capacity is 100 Mb/s, i.e., much higher than the video rate. Latencies among peers randomly varies uniformly in $[10, 30]$ ms (so that $minRTT$ varies in $[20, 60]$ ms). The *Pink of the Aerosmith* video (352x240 resolution, 25 fps, H.264/AVC Codec) has been encoded at different rates and "streamed" over the swarm. The whole video is looped 5 times, for a total duration of 20 m. After discarding the initial 12 m of each experiment, each peers saves 100 s of the received frames on disk. Structural Similarity index (SSIM) [27] is then computed against the original YUV video for each video trace and averaged over all peers. The SSIM is a highly non linear QoE metric that ranges from 0 to 1. Values higher than 0.95 are considered equivalent of good quality. Values smaller than 0.9 reflect already poor quality. For reference, the Encoded Video Quality (EVQ) reports the SSIM as computed at the source, i.e., considering the impairments due to encoding process, but not the one due to losses during the transmission. In this scenario, P_{bg} fraction of peers can be affected by TCP or UDP traffic competing for peer upload capacity. Interfering traffic follows an ON-OFF pattern, with ON and OFF periods that are exponentially distributed with average $E[T_{ON}] = 60$ s and $E[T_{OFF}] = 120$ s.

The results presented in this section constitute a small subset of the overall collected results, which amount to more than two months of equivalent time of testing. Additional results are presented in [25]. Both DB-HRC and AIMD-HRC have been tested under several parameter settings. As a result of our campaign, we observed that: performance of DB-HRC is weakly sensitive to the parameter settings; reasonable choices of D_0 in the range [150, 300] ms produce similar results in most of the cases of interests. We present results for two different DB-HRC configurations in which $D_0 = 150$ ms and $D_0 = 250$ ms respectively. For what concerns the parameter K_p, as long as K_p is selected sufficiently small ($K_p < 1$ s^{-1}), the DB-HRC results stable; this is confirmed

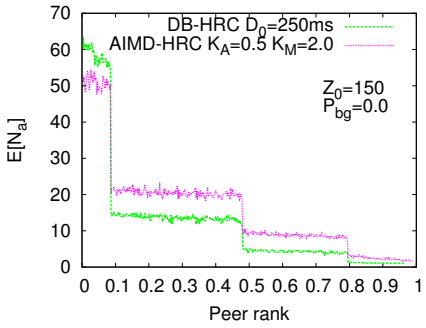

Figure 8: Average number of offers in flight for each peer. $Z_0 = 150$ and $P_{bg} = 0.0$

with simple control theoretical arguments, which are not reported for the sake of the brevity (the interested reader is referred to [25]). We fix $K_p = 0.98$ ms^{-1}. Parameter setting for AIMD-HRC is instead much more critical, since the overall performance of the system is strongly sensitive to choices. In this section we present results for the two different AIMD-HRC configurations that provided the overall best results. In the two proposed configurations we have set $K_A = 0.3$, $K_M = 2.0$ and $K_A = 0.5$, $K_M = 1.5$ respectively. Finally, p_0 has little impact on performance, but it greatly helps in limiting the number of offers when $\rho < 1$ (see Sec. 5.4). We choose $p_0 = 0.5$ for all experiments. Employed algorithm parameters are summarized in Table 1.

Fig.7 reports the average SSIM experienced by peers vs the video rate. Curves within each plot refer to different rate control schemes. Different plots refer to different network scenarios and Z_0 settings, with $Z_0 = 50$ (2 s) and $Z_0 = 150$ (6 s) on the top and bottom row, respectively.

We start considering left plots that refer to a scenario with no competing traffic. Observe that DB-HRC (especially when we set $D_0 = 150$ ms) in general outperforms AIMD-HRC for $Z_0 = 50$ (top plot). Indeed DB guarantees reasonable video quality (SSIM above 0.95) for video rate smaller or equal than 1.1 Mb/s (we recall that the average peer upload bandwidth bandwidth is approximately 1.32 Mb/s at Data-Link layer); AIMD-HRC performs worse, dropping below 0.95 already for $V_r = 1$ Mb/s. This is not surprising in light of the fact that under AIMD-HRC, peers greedily try to get bandwidth (even in absence of competing traffic) congesting their up-link. This leads to an increase of the chunk delivery delay, possibly inducing unnecessary chunk losses due to missed deadline. Increasing Z_0 to 150, the overall QoE performance significantly improves (so that the QoE matches the EVQ for V_r up to 1 Mb/s). This happens at the cost of tolerating a significantly larger buffering time and total offset delay. The performance gap between DB-HRC and AIMD-HRC is greatly reduced, because in this case the D_{OS} constraint is relaxed, and thus achieving a tight control of chunk delay becomes much less important.

5.2. Impact of TCP competing traffic

Central and right-most plots in Fig. 7 refer to scenarios in which an increasing fraction of peers, i.e. P_{bg}, is affected by ON-OFF TCP competing sources. $P_{bg} = 0.5$ in middle plots and 1.0 in right plots. Obviously, the SSIM starts degrading for smaller values of V_r, i.e., ρ increases due to the reduction of available average capacity. Observe that, in general, also in these scenarios, which should be in principle more favorable to AIMD-HRC, DB-HRC performs rather well and definitely better than AIMD-HRC when tight delay constraints are imposed (i.e. $Z_0 = 50$). DB-HRC with smaller target ($D_0 = 150$ ms) achieves better performance in the scenarios with moderate percentages of simultaneous peers affected by TCP traffic ($P_{bg} = 0.5$), while DB-HRC with larger target ($D_0 = 250$ ms) behaves better in scenarios in which the average number of TCP interfering flows increases ($P_{bg} = 1.0$). This because higher target delay makes DB-HRC more robust to neighbor congestion effects (as already observed in the previous section). Turning now our attention to AIMD-HRC, observe that the two selected configurations provide essentially the same performance; the more aggressive version of AIMD-HRC ($K_A = 0.5$, $K_M = 2.0$) tends to be preferable compared to the smoother version ($K_A = 0.3$, $K_M = 1.5$) as the amount of TCP traffic increases ($P_{bg} = 1.0$). Yet the higher greediness of AIMD-HRC does not pay off in terms of quality when tight delay constraints are imposed (i.e. $Z_0 = 50$): the higher greediness causes more losses, i.e., reduced useful delivery of timely chunks which overall provides worse QoE than DB-HRC. When we increase Z_0 to 150, the performance gap between AIMD-HRC and DB-HRC algorithms tend to disappear. Furthermore AIMD-HRC tends to perform better than DB-HRC in congested scenarios (i.e, for $V_r \geq 1$ Mb/s and $P_b = 1$) due to its ability to subtract bandwidth to competing TCP flows. Observe, however, that the overall QoE performance is globally rather bad (below 0.9) in these cases.

As a further proof of the capability of HRC controller of adapting the chunk offering rate to the available up-link capacity, we report in Fig. 8 the average number of offers in flight $E[N_a]$ for all peers involved in our experiment, given the scenario with $P_{bg} = 0.0$ and $D_0 = 150$ ms. Peers were ranked according to their up-link capacity (peers with large up-link capacity to the left). Observe that DB-HRC tends to better exploit large bandwidth peers, reducing the stress on peers with lower resources.

5.3. Impact of UDP competing traffic

For completeness, Fig. 9 reports results for a scenario in which each peers shares its up-link capacity with a competing ON-OFF UDP source that grabs half of the peer nominal bandwidth when in ON state. In this case, DB-HRC is globally preferable to AIMD-HRC (especially for $Z_0 = 50$) since peers can quickly adapt their transmission rate to the available bandwidth left by UDP without greedily generating congestion on the up-link queue.

These experiment support the intuition that for *inelastic* P2P-TV applications, with near real-time constraints, it is hard to find a scenario in which a greedy behavior provides significant advantages for peers. Indeed when the bandwidth is made scarce for effect of elastic competing traffic, a greedy behavior does provide some advantage. However, this scenario is unfit for P2P-TV applications whose end-user QoE would remain in any case unsatisfactory. In less extreme scenarios, greedily competing

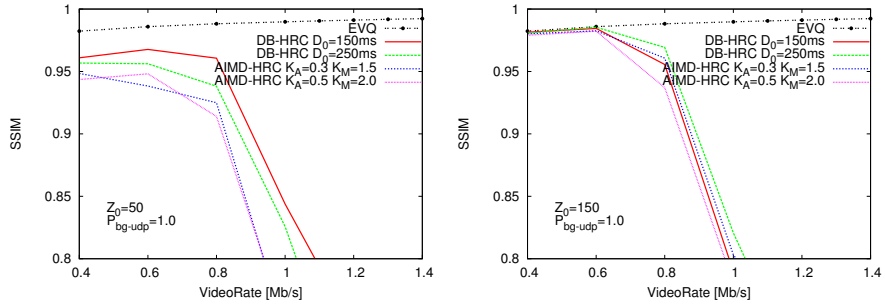

Figure 9: SSIM vs Video Rate when 100% of peers are affected by competing ON-OFF UDP sources for $Z_0 = 50$ (top plot) and $Z_0 = 150$ (bottom plot).

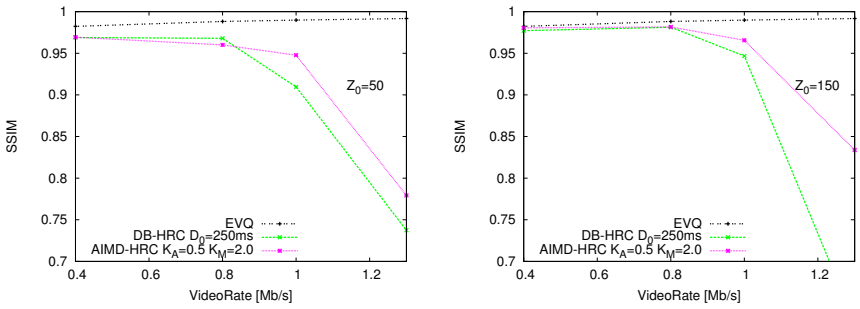

Figure 10: Average SSIM versus V_r for $Z_0 = 50$ (left plot) and $Z_0 = 150$ (right plot) on PlanetLab experiments.

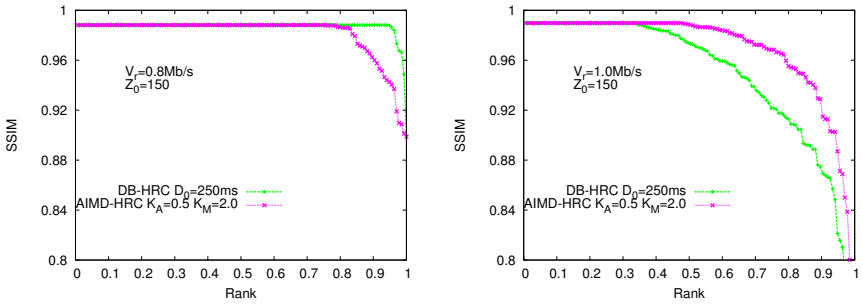

Figure 11: Per peer SSIM for $V_r = 0.8$Mb/s (left plot) and $V_r = 1.0$Mb/s (right plot) on PlanetLab experiments.

for bandwidth can induce unnecessary losses, and a more graceful rate controller that avoids transmission queue congestion typically achieves better performance.

5.4. Real Internet Results: PlanetLab

No churning - Now we move to a real Internet scenario. We selected about 500 PlanetLab nodes spread all over the world. Peer upload capacity has been limited by

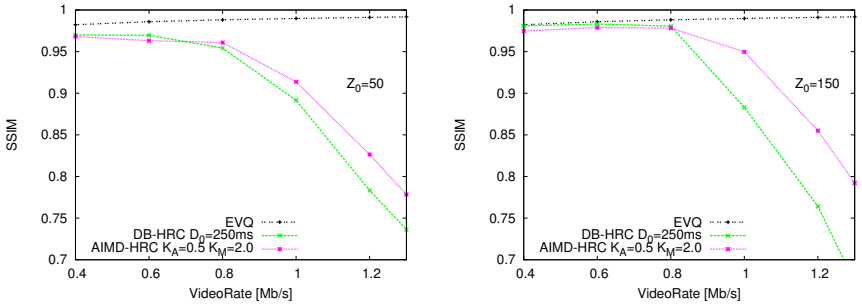

Figure 12: Average SSIM versus V_r for $Z_0 = 50$ (left plot) and $Z_0 = 150$ (right plot) on PlanetLab experiments with churning.

the PeerStreamer embedded rate limiter; 2 classes are present, with half of peers having 2 Mb/s at their up-link, and half with 0.64 Mb/s. Average upload capacity results to 1.32 Mb/s. Observe this is an upper-bound to the actual available peer upload bandwidth which may be reduced for effect of competing experiment running on the same PlanetLab node or due to other bottlenecks on the access links of the node. Similarly, download capacity and CPU availability at nodes may constitute a bottleneck. To avoid extreme outliers, we run a preliminary experiment with $V_r = 0.4$ Mb/s to discard the most congested and unreliable nodes, i.e., peers that received less than 90% of chunks even with this small ρ. 410 PlanetLab nodes were then selected.

Fig. 10 reports the average SSIM for different video rates for $Z_0 = 50$ (2 s) and $Z_0 = 150$ (6 s), on left and right plots, respectively. Curves refer to the more aggressive version of AIMD-HRC ($K_A = 0.5$, $K_M = 2.0$), while $D_0 = 250$ ms is considered for DB-HRC. Observe that for moderate $V_r < 1.0$ Mb/s (i.e., $\rho < 1$) there is little differences among the two cases, while AIMD-HRC takes the edge for $V_r \geq 1.0$ Mb/s. This is not surprising, since in congested scenario AIMD-HRC competes more aggressively with other traffic, getting a higher share than DB-HRC. Yet, in these conditions the QoE guaranteed to end-users is in any-case far from optimal. For completeness, Fig. 11 details each peer's individual SSIM performance for $V_r = 0.8$ Mb/s and $V_r = 1.0$ Mb/s, on left and right plot, respectively, and $Z_0 = 150$ for both. Peers have been sorted in decreasing SSIM to ease visualization. Observe that for $V_r = 0.8$ Mb/s, almost the totality of peers experiences the best QoE. Only some losses are experienced by a small fraction of peers (about 5 % of peers with DB-HRC, 15% with AIMD-HRC) whose SSIM is degraded. In this scenario DB-HRC is preferable. For $V_r = 1.0$ Mb/s, about 50% of peers already experiences significant losses with both controllers; AIMD-HRC provides slightly better QoE in this case.

Impact of churning - At last we consider a scenario in which peers are subject to churning to observe any negative impact on the performance of our schemes. In this scenario, one forth of peers are coming and going following an ON-OFF Markovian pattern. Churning peers' average sojourn time is 120 s. Off periods are set to only 10 s

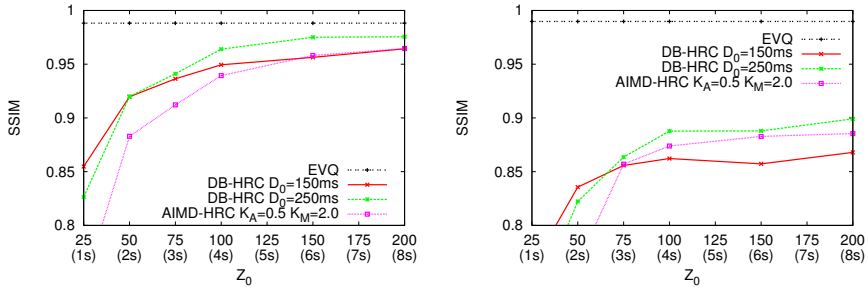

Figure 13: SSIM vs Z_0 when of 50% of peers is affected by competing ON-OFF TCP sources for $V_r = 0.8Mb/s$ (left) and $V_r = 1.0Mb/s$ (right).

to maximize the number of peers in the system[8].

We emphasize that SSIM can be computed only for the fraction of peers which is stable. Fig. 12 reports the results for the two considered algorithms versus V_r for $Z_0 = 50$ (left plot) and $Z_0 = 150$ (right plot). Observe that the effect of churning does not significantly change the way in which the HRC algorithms behave; similar considerations with respect to the previous case can be drawn. However a generalized performance degradation can be noticed by comparing Fig. 12 with Fig. 10. This is mostly unrelated to HRC since it is expected to impact overlay topology and scheduling algorithms effectiveness.

5.5. Final Remarks

From our experiments, we can conclude that DB-HRC and AIMD-HRC achieve similar performance when the constraints imposed by the total offset delay (i.e. the buffering time) are relaxed. In presence of tighter constraints, and when competing with UDP traffic DB-HRC is the preferable choice. When competing with TCP traffic (both in our test-bed and in the real Internet[9]) AIMD-HRC tends to perform better, but in a region in which the overall QoE is poor.

6. Sensitivity to Parameters and Scales

6.1. Impact of Z_0

To provide a more complete view of the impact of Z_0 parameter, Fig. 13 reports the average SSIM vs Z_0. For the sake of clarity, the latter is expressed in number of chunks with the corresponding amount of buffering time within parentheses.

[8]When a peer enters the overlay after an off period, it takes a different ID.

[9]Notice that in this case the competing traffic is given by the natural working of the Internet, i.e., traffic generated by Internet users, that we expect to be largely composed by TCP.

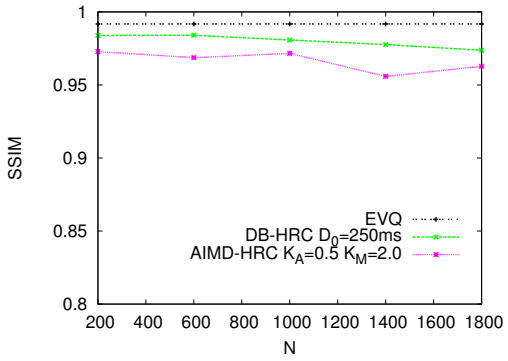

Figure 14: Average SSIM when increasing the number of peers in the swarm N. $V_r = 1.3$ Mb/s, $Z_0 = 150$ (6 s).

We report results for two configurations of DB-HRC and one configuration of AIMD-HRC. We consider a scenario in which $P_{bg} = 0.5$ (i.e., 50% peers competing with ON-OFF TCP sources). Left plot of Fig. 13 refers to $V_r = 0.8$ Mb/s, while $V_r = 1.0$ Mb/s is considered on right plot. Results confirm the expected intuition: when achieving a tight control on the chunk transfer time is of paramount importance, i.e., Z_0[10] forced small, DB-HRC outperforms AIMD-HRC, and a smaller target ($D_0 = 150$ ms) provides better performance than the more aggressive settings ($D_0 = 250$ ms). As the Z_0 constraint is relaxed and larger delays are tolerated, the performance gap between DB-HRC and AIMD-HRC tends to vanish, and more aggressive settings (DB-HRC with $D_0 = 250$ ms) guarantees better overall performance.

These figures clearly enlighten that a challenging issue for P2P-TV designers is to find a reasonable compromise between the opposite needs of guaranteeing good QoE performance while keeping the real-time delay as short as possible.

6.2. Scaling the emulated scenario up to 1800 peers.

Fig. 14 shows the average SSIM for increasing number of peers N for DB-HRC ($D_0 = 150$ ms) and AIMD-HRC ($K_A = 0.5$, $K_M = 2.0$). In this scenario, each PC runs from 1 to 9 PeerStreamer instances, thus the swarm size varies from 200 to 1800 total peers[11]. To limit the up-link capacity, we adopt the rate limiter embedded in PeerStreamer. The bandwidth distribution is heterogeneous as described in previous section, but the available capacity results constrained at the application layer and not at the Data-Link layer as before, allowing for a slightly higher average total upload capacity. There is no competing traffic ($P_{bg} = 0.0$), $Z_0 = 150$ (6 s), and $V_r = 1.3$ Mb/s, corresponding to nominal $\rho = 0.98$, a rather critical scenario. Result shows that the system is able to practically deliver very good performance for ρ very close to 1. As expected, the average QoE perceived by users is substantially independent from

[10] Observe that in general all peers participating to the swarm share the same Z_0.

[11] Instantiating more than 9 PeerStreamer per PC causes congestion on the local O.S.

N, with a marginal decrease for larger population (related to the increase of number of hops in the overlay). In general, observe again that DB-HRC performs better than AIMD-HRC since it tries its best to avoid overflowing the transmission queue at peers.

6.3. Upload capacity heterogeneity.

The heterogeneity of the peers in terms of upload capacity has in general some impact on the QoE which is perceived by the users. This is confirmed by Fig. 15 where peers are ranked according to their upload capacity (peers with larger upload capacity to the left) and which reports the per peer SSIM. Experiment refers to a scenario where Vr = 1.3Mb/s, corresponding to a system load ρ close to 1. Observe that peers whose upload capacity is poorer experience more congestion at the uplink, and thus encounter some difficulties at replying to offer messages promoting new chunks. However, observe that both the HRC controllers are affected by this effect, being AIMD-HRC and DB-HRC showing very similar patterns. This demonstrates that the difference in performance between the two HRC controllers is not affected by the upload capacity distribution, and, therefore, the results presented in this work are not biased.

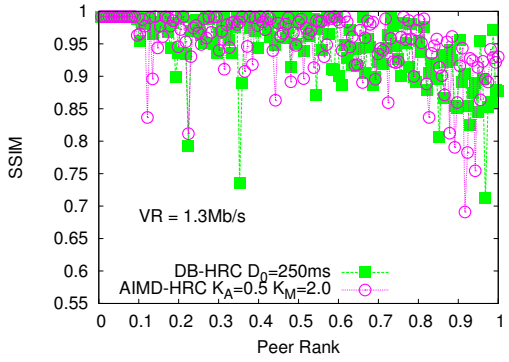

Figure 15: Per peer SSIM for Vr = 1.3Mb/s ($\rho = 0.98$) in controlled test-bed. Peers are ranked according to their upload capacity. $Z_0 = 150$.

6.4. Signalling overhead.

One important aspect of a P2P-TV system is the evaluation of signaling overhead that is required to deliver the video content. In our case, both the overlay maintenance and trading scheme require peers to exchange signaling messages. Among the two, the second one is expected to be more critical. Notice that when $\rho < 1$, overhead is potentially not critical, and when $\rho = 0.98$ as in Fig. 14, we have already observed that the system provides overall excellent performance. Yet, HRC entails a mechanism to self limit the amount of offers which depends on p_0 - see Sec 3.4. We thus run a set of experiments to gauge the percentage of overhead that is required to execute the offer-select scheme. Details can be found in [25]. Overall, when ρ approaches 1, the overhead (in bytes) never exceed 5-10%. When ρ is small, the clipping effect due to

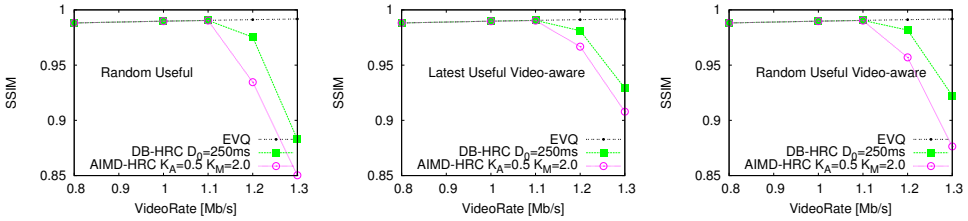

Figure 16: SSIM vs Video Rate comparison between DB-HRC and DB-AIMD, when adopting different chunk scheduling policies: "Random Useful" (left), "Latest Useful Video-Aware" (center) and "Random Useful Video-Aware" (right). $Z_0 = 150$.

p_0 successfully limits the signaling impact which decreases with increasing p_0. Values in the range $0.4 \leq p_0 \leq 0.8$ have little impact on the QoE, and guarantee that the overhead is limited to 30% - 15% for $V_r = 0.6$ Mb/s.

6.5. Impact of chunk scheduling policies

Finally, the reader can argue that the performance of the transmission rate controllers may be different for different peer and chunk scheduling policies running above it. We thus conclude our experimental campaign showing that the relative performance of the transmission control mechanisms proposed in this paper are independent on the logic adopted by the chunk scheduling algorithms.

We focus on the impact of enhanced and well-understood chunk scheduling policies that have been proposed in the literature [9, 24, 28]. Implementing them is a fairly simple task in PeerStreamer. We thus show results obtained comparing different chunk schedulers on the top of both DB-HRC and AIMD-HRC.

We consider the following four different chunk scheduling policies:

1. "Latest Useful" (LU) - already adopted in experiments shown in previous sections: a peer selects the most recent chunk (i.e., with the largest sequence number) among the useful chunks offered by a neighbor.
2. "Random Useful Chunk" (RU): a peer selects at random a chunk among the useful chunks offered by a neighbor.
3. "Latest Useful Video-Aware Chunk" (LUV): for each frame class (I, P, B), a peer selects the latest-useful chunks defining the set $LUV = \{I_{lu}, P_{lu}, B_{lu}\}$. Then, a weighted random choice is performed to select one chunk from LUV. Weights are proportional to the importance of the frame type, i.e., $(W_I = 5,\ W_P = 2,\ W_B = 1)$, so that higher preference is given to the most recent chunk carrying an I-frame [29].
4. "Random Useful Video-Aware Chunk" (RUV): useful chunks form the set RUV, in which each chunk is weighted based on the type of frame it carries $(W_I = 5,\ W_P = 2,\ W_B = 1)$. Then, one chunk is selected using a weighted random choice among chunks in RUV.

We run experiments in our controlled environment as considered in Sec. 5.1 ($P_{bg} = 0.0$, i.e., no background traffic) and compute the resulting SSIM for both DB-HRC ($D_0 = 250$ms) and AIMD-HRC ($K_A = 0.5$, $K_M = 2.0$). Results are plotted in

Fig. 16 for the three new schedulers (since LU has been already studied in the previous sections). In this scenario, DB-HRC performs better than AIMD-HRC, confirming results reported in Fig. 7. While the chunk schedulers do have an impact on absolute performance (with LUV offering the best results), the ranking is respected independently on which chunk scheduler algorithm is adopted. This confirms that the hose rate controller can be adopted on the top of different chunk schedulers with good performance.

6.6. Final Remarks

From the results presented in this section, we can conclude that DB-HRC outperforms AIMD-HRC when the total offset delay is above 3s in the scenario considered in Sec. 5.4. Moreover, the difference in performance between DB-HRC and AIMD-HRC is independent from i) the number of peers involved in the swarm, ii) from the kind of chunk scheduler that is built upon them and iii) the peer upload capacity distribution.

7. Related Works

In P2P-TV systems chunks present a smaller size with respect to other P2P applications, and UDP is typically preferred by commercial solutions [2, 3, 4, 5]. Handling smaller chunks allows to enforce serial chunk packet transmission and to avoid unnecessary delays due to TCP retransmission and congestion control. However, adopting UDP at transport level poses the problem of how to regulate the amount of information a peer can transmit, since the download rate in P2P-TV systems is in all cases limited by the video stream rate. Controlling therefore the up-link bandwidth allocation is a key problem.

However, many works in the literature consider the general problem of optimizing the resource allocation in media streaming [30, 31, 32]. This problem is commonly known with the name of rate allocation, but with different meaning [33]: the first one, mostly signal-processing oriented, focuses on the problem of allocating the bitrate during source and channel coding phases, with the objective of optimizing the quality of the reconstructed video. The second one, is devoted to the problem of exploiting the available bandwidth, with the objective of maximizing some utility function. This work clearly falls in the latter, more network oriented, category. However, all the controllers proposed in this branch do not fit with the needs of the system we propose: i.e., they rely on multicast [32], they propose end-to-end rate controllers [30], or they rely on structured overlays (trees) [33]. In a nutshell, the literature about transmission rate controllers for mesh-based P2P-TV systems is very limited and we are not aware of other proposals. This is justified since most existing P2P live video systems do not consider the sender rate control problem explicitly, and they simply adopt a best-effort approach. Furthermore, any comparison with other solutions is impracticable, since popular applications (e.g., PPLive, SopCast, etc.) do not disclose their source code, so that it is impossible to understand whether they implement any rate controller, and, if so, which kind. As said, to the best of our knowledge, the only works that explicitly deal with the issue of regulating the transmission rate of peers in unstructured P2P-TV

systems are [12, 13]. The algorithm shown in [12] is based on the periodic advertisement of peer's buffermaps, offered to a subset of the peer's neighbors. However, the resulting scheme which assumes essentially homogeneous latencies may be difficult to implement in practice. The transmission rate control proposed in [13] represents a preliminary performance evaluation work about Hose Rate Control. The delay-based HRC version is compared against fixed naive schemes and a comprehensive comparison with loss-based schemes is missing. In this paper we fill this gap: we present results obtained from experimental test-beds run both on PlanetLab and in a controlled environment to compare a less-than-best-effort philosophy (DB-HRC) with a TCP-competitive loss-based one (AIMD-HRC).

About congestion control protocols, reader may refer to Ledbat [14] and TCP-Vegas [34] as notable examples of delay-based approaches. About loss-based congestion controllers, instead, many works have been proposed, from early proposals such as TCP-Reno and TCP-Sack to more recent solutions like TCP-Cubic [35]. A specific solution for media streaming is represented by TFRC [16] whose transmission controller built at application level guarantees a smooth sending rate and a fair competition with TCP sources. However, all these solutions rely on some end-to-end rate regulation mechanism and, as shown in 3.1, this makes them hardly adequate for live P2P-TV applications.

Common assumptions in literature about P2P-TV systems are that i) the main bottleneck to system performance is given by the upload capacity of peers, ii) peers are expected to have homogeneous and stable-in-time upload bandwidths, and iii) each peer is assumed to have a perfect up-to-date view of the internal state of other peers [7, 8, 9, 10]. While assumption i) is often met in practice, ii) is unrealistic since peers are distributed in heterogeneous capacity scenarios and, moreover, available upload bandwidth varies in time due to user's activity. Even assumption iii) is unfeasible in practice, since peers have to face with latencies separating each other.

Few papers focus on the impact of peers bandwidth heterogeneity and how it can be exploited to improve system performance [10, 6, 36]. How the latency can impact on system performance is a problem that has been studied through a simple model corroborated by real measurements in PlanetLab in [21]. The authors propose a system that overcomes the effect of latency by exchanging state information via signaling messages. Little description is however given about the implemented signaling mechanisms details. In [28] and [19] we also evaluated the impact of latency on system performance, using a non-adaptive signaling mechanism. Considering then both peers heterogeneity and latencies, an accurate design of the signaling mechanism is mandatory.

8. Conclusions

In this paper we have addressed issues related to how to design simple and efficient rate controller mechanisms for P2P-TV applications. Given the almost un-elastic video-rate, and non greediness of receiving peers, transmission rate controllers for P2P-TV are intrinsically different from P2P file sharing applications. We have discussed and motivated all the design choices proposing two simple aggregate rate controller schemes that embrace two rather different philosophies. The first scheme DB-HRC

attempts a tight control of the transmission queue delay, the second AIMD-HRC mimics the behavior of TCP falling in the class of the Additive Increase Multiplicative Decrease (AIMD) schemes.

Our main conclusions are:

- Having a tight control of the chunk delivery delay is of fundamental importance when the buffering time at peers is tightly constrained (smaller than 5 s). In this conditions DB-HRC scheme is in general preferable to AIMD-HRC, leading to better global QoE performance. This because delay based schemes are able to gently adapt the transmission rate of peers to the available bandwidth without congesting the up-link, and thus keeping the chunk transfer time under control.

- The performance gap between AIMD-HRC and DB-HRC tends to vanish when the constraints on the buffering time are relaxed, i.e., buffering time is set greater or equal than 5 seconds.

In conclusion, we believe that P2P-TV designers should target non extremely adverse scenarios where enough bandwidth is globally available to the application even if a fraction of the peers may experience congestion due to crossing traffic; in such conditions DB-HRC rate control appears preferable to AIMD-HRC, as confirmed by our experimental results.

References

[1] X. Hei, C. Liang, J. Liang, Y. Liu, K. W. Ross, A measurement study of a large-scale p2p iptv system, Multimedia, IEEE Transactions on 9 (8) (2007) 1672 – 1687. doi:10.1109/TMM.2007.907451.

[2] X. Hei, Y. Liu, K. W. Ross, Iptv over p2p streaming networks: The mesh-pull approach, IEEE Communication Magazine 46 (2) (2008) 86–92.

[3] 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.

[4] 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. doi:http://dx.doi.org/10.1109/TMM.2009.2030656.

[5] Y. G. et al., Survey of P2P Streaming Applications, Internet Draft draft-ietf-ppsp-survey-02, IETF (July 2011).

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

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

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

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

[10] 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.

[11] L. Abeni, C. Kiraly, R. Lo Cigno, On the optimal scheduling of streaming applications in unstructured meshes, in: IFIP Networking, Aachen, DE, 2009.

[12] A. Carta, M. Mellia, M. Meo, S. Traverso, Efficient uplink bandwidth utilization in p2p-tv streaming systems, in: IEEE Globecom, Miami, FL, US, 2010.

[13] C. Kiraly, R. Birke, E. Leonardi, M. Mellia, M. Meo, S. Traverso, Hose rate control for p2p streaming systems, in: IEEE P2P, Kyoto, JP, 2011.

[14] S. Shalunov, G. Hazel, Low Extra Delay Background Transport (LEDBAT), Internet Draft draft-ietf-ledbat-congestion-02, IETF (July 2010).

[15] D. Rossi, C. Testa, S. Valenti, L. Muscariello, Ledbat: The new bittorrent congestion control protocol, in: Computer Communications and Networks (ICCCN), 2010 Proceedings of 19th International Conference on, 2010, pp. 1 –6. doi:10.1109/ICCCN.2010.5560080.

[16] S. Floyd, M. Handley, J. Padhye, J. Widmer, TCP Friendly Rate Control (TFRC): Protocol Specification, Internet Draft draft-ietf-dccp-rfc5348, IETF (September 2008).

[17] J. Gettys, Bufferbloat: Dark buffers in the internet, Internet Computing, IEEE 15 (3) (2011) 96. doi:10.1109/MIC.2011.56.

[18] L. Vu, I. Gupta, K. Nahrstedt, J. Liang, Understanding overlay characteristics of a large-scale peer-to-peer iptv system, ACM Trans. Multimedia Comput. Commun. Appl. 6 (2010) 31:1–31:24. doi:http://doi.acm.org/10.1145/1865106.1865115.
URL http://doi.acm.org/10.1145/1865106.1865115

[19] S. Traverso, L. Abeni, R. Birke, C. Kiraly, E. Leonardi, R. Lo Cigno, M. Mellia, Experimental comparison of neighborhood filtering strategies in unstructured p2p-tv systems, in: Peer-to-Peer Computing (P2P), 2012 IEEE 12th International Conference on, 2012, pp. 13–24.

[20] R. Kumar, Y. Liu, K. Ross, Stochastic fluid theory for p2p streaming systems, in: INFOCOM 2007. 26th IEEE International Conference on Computer Communications. IEEE, 2007, pp. 919 –927. doi:10.1109/INFCOM.2007.112.

[21] 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.

[22] 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, US, 2005.

[23] F. Picconi, L. Massoulié, Is there a future for mesh-based live video streaming?, in: IEEE P2P, Aachen, DE, 2008.

[24] A. Couto da Silva, E. Leonardi, M. Mellia, M. Meo, Exploiting heterogeneity in p2p video streaming, Computers, IEEE Transactions on 60 (5) (2011) 667 –679. doi:10.1109/TC.2010.244.

[25] S. Traverso, C. Kiraly, E. Leonardi, M. Mellia, Hose rate control for p2p-tv streaming systems.
URL http://www.tlc-networks.polito.it/traverso/papers/tr-2010-30-08.pdf

[26] F. Baccelli, D. Hong, Interaction of tcp flows as billiards, IEEE/ACM Trans.

Netw. 13 (2005) 841–853. doi:http://dx.doi.org/10.1109/TNET.2005.852883.
URL http://dx.doi.org/10.1109/TNET.2005.852883

[27] Z. Wang, A. C. Bovik, 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.

[28] 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.
URL http://www.telematica.polito.it/traverso/papers/p2p10.pdf

[29] E. Setton, J. Noh, B. Girod, Congestion-distortion optimized peer-to-peer video streaming, in: Image Processing, 2006 IEEE International Conference on, 2006, pp. 721–724. doi:10.1109/ICIP.2006.312442.

[30] P. Zhu, W. Zeng, C. Li, Joint design of source rate control and qos-aware congestion control for video streaming over the internet, Multimedia, IEEE Transactions on 9 (2) (2007) 366–376. doi:10.1109/TMM.2006.886284.

[31] P. Troubil, H. Rudová, et al., Integer linear programming models for media streams planning, ICAOR 11 (2011) 509–522.

[32] K. Kar, L. Tassiulas, Layered multicast rate control based on lagrangian relaxation and dynamic programming, Selected Areas in Communications, IEEE Journal on 24 (8) (2006) 1464–1474. doi:10.1109/JSAC.2006.879353.

[33] L. Lima, M. Dalai, R. Leonardi, P. Migliorati, R. Bernardini, R. Rinaldo, Optimal rate allocation for p2p video streaming, Selected Areas in Communications, IEEE Journal on 31 (9) (2013) 200–213.

[34] L. S. Brakmo, S. W. O'Malley, L. L. Peterson, Tcp vegas: new techniques for congestion detection and avoidance, SIGCOMM Comput. Commun. Rev. 24 (1994) 24–35. doi:http://doi.acm.org/10.1145/190809.190317.
URL http://doi.acm.org/10.1145/190809.190317

[35] S. Ha, I. Rhee, L. Xu, Cubic: a new tcp-friendly high-speed tcp variant, SIGOPS Oper. Syst. Rev. 42 (2008) 64–74. doi:http://doi.acm.org/10.1145/1400097.1400105.
URL http://doi.acm.org/10.1145/1400097.1400105

[36] T. Small, B. Liang, B. Li, Scaling laws and tradeoffs in Peer-to-Peer live multimedia streaming, in: ACM Multimedia, Santa Barbara, CA, US, 2006.