Hose rate control for P2P-TV streaming systems

(Article begins on next page)

04 May 2024

Hose Rate Control for P2P-TV Streaming Systems

R. Birke[1], C. Kiraly[2], E. Leonardi[3], M. Mellia[3], M. Meo[3], S. Traverso[3]

[2]IBM Zurich Research Lab., Switzerland – email: bir@zurich.ibm.com

[2]University of Trento, Italy – email: kiraly@disi.unitn.it

[3]Politecnico di Torino, Italy – email: {firstname.lastname}@polito.it

Abstract—**In this paper we consider mesh based P2P streaming systems focusing on the problem of regulating peer upload 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 users' Quality of Experience is greatly enhanced. Both simulations and actual experiments involving up to 1000 peers are presented to assess performance in real scenarios. Results show that HRC consistently outperforms the Quality of Experience achieved by non-adaptive schemes.

I. Introduction

In mesh based Peer-to-Peer streaming (P2P-TV) systems, the real-time encoded video is sliced in small pieces called *chunks*, which are distributed over an overlay topology exploiting a fully distributed epidemic approach. Chunks have to be received by peers within a deadline of few seconds in order to guarantee real-time constraints. In these systems, download rate is dictated by video rate, which is limited by definition; the source peer emits chunks in real time at "constant" rate and all peers must trade them minimizing delays and losses to guarantee the best Quality of Experience (QoE) to users.

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 required between neighbor pairs to agree on the chunks to be sent. The trading phase requires the exchange of messages: an offer message sent by the sender a to the neighbor 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 which have been selected by b. A careful design of the trading scheme is needed to avoid that the additional signalling delay translates into an excessive delay. This paper focuses on the design of the trading scheme, proposing 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 application [1] to avoid both the burden of handling TCP and the unnecessary delay due to retransmissions and congestion control. However, this poses the problem of how to handle the congestion control and, in particular, how to limit the amount of information a peer transmits, being download rate limited by video-rate. Controlling therefore the uplink

bandwidth utilization is a key problem, which has been so far marginally considered by the research community. The design of trading mechanisms requires that a number of parameters is finely tuned to achieve optimal results, and the optimal setup, in its turn, depends on the specific scenario, which is typically unpredictable due to the variability of network conditions and to peer heterogeneity.

Considering the trading scheme, the most critical parameter is the number of "offers" (messages advertising the list of 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. 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 "WineStreamer", the new P2P-TV application of EU-FP7 NAPA-WINE STREP project Napa-Wine Project [2]. This allows us to provide experimental results on swarms of up to 1000 peers in a controlled environment. Simulation and experimental results 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) [3].

II. System Description

We consider a system in which a source segments the video stream into chunks and injects them in the overlay network. Let 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.
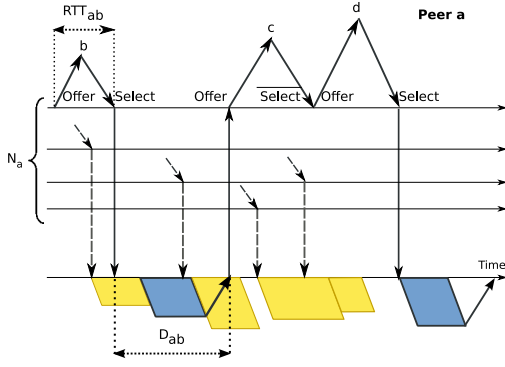
Figure 1: Schematic representation of the peer chunk trading mechanism.

Chunks are transmitted by peers to their neighbors in a swarm-like fashion; the overlay topology is defined by the set of peers and virtual links connecting them. Since the actual design of the overlay topology is out of the scope of this paper, we consider the simplest case in which the overlay network is built on a random basis, a common assumption in the literature [4], [5].

Since video chunks are transmitted over the network, the intuition suggests to keep them small, e.g., few IP packets, to minimize the packetization delay at the source, the store-and-forward delay at the peers and the chunk corruption probability due to packet loss. In what follows, we therefore 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, thus allowing a stricter real-time streaming. Furthermore, the rounding at frame boundaries minimizes the impact of losses, avoiding that a loss of a chunk affects several frames due to partial delivery of information, e.g, missing headers.

A. 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 [6], [7]. 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 signalling thread evolves as follows: peer a chooses one of its neighbors b and sends it an *offer* message that contains the set of chunks younger than D_{max} that a possesses. Upon receiving the offer message, b replies with a *select* message to request one desired chunk that the receiver marks as *pending* until it is correctly received; a pending chunk cannot be requested and cannot be published. When the select message is received by a:

- if no chunk was selected *negative select*, a new offer message can be sent.
- if a chunk was requested in the select message (*positive select*), a inserts it in its *transmission queue* that is served in a FIFO order.

Once b has completely received the previously selected chunk, it sends an *ACK* message to a. So that when a receives an ACK message, a new offer message can be sent.

Peer a is committed to send all the chunks requested in all the received positive selects. Fig. 1 represents the signalling messages and chunks exchanged by peer a with its neighbors over time. The number N_a of signaling threads limits the number of chunks the peer can offer, and thus control the upload capacity utilization of the peer. The key parameter to set in this mechanism is N_a, which is the equivalent of the window size in a window protocol.

For the *peer selection* and the *chunk selection* policies we make the simplest possible choices: peer a chooses the peer to contact 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" [8].

B. The core of Hose Rate Control

The basic idea is to control the rate at which chunks are sent by peer a by adjusting the number of active threads N_a (equivalent to the window size in a window protocol), so that the queuing delay at the transmission queue is at a given (small) target. N_a controls the queue at peer a: if it is too large, delay increases, deteriorating performance; if it is too small, peer available upload bandwidth is not well exploited. Then, if the queuing delay is large, N_a is decreased, and vice-versa. More in detail, let W_a be the real internal control variable such that $N_a = \lfloor W_a \rfloor$. For every neighbor b, peer a maintains an estimate of the minimum Round Trip Time RTT_{ab} that can be computed/updated every time a receives a select message. When a receives an acknowledge from b, it estimates the queuing delay incurred by the chunk in the transmission queue, as $\hat{D} = t^{(a)}_{\text{rx,ack}} - t^{(a)}_{\text{tx,select}} - RTT_{ab}$, i.e., subtracting a RTT_{ab} 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: $W_a(n) \leftarrow W_a(n-1) - (\hat{D} - D_0)$. N_a is then increased/decreased by $\Delta N_a = \lfloor W_a(n) \rfloor - \lfloor W_a(n-1) \rfloor$. Then, 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 current thread is stopped (no new offer is sent).

III. EVALUATION BY SIMULATION

A. 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 [2]. In our scenario, peers are partitioned in four classes based on their upload capacity: 15% of peers are in Class 1 with upload bandwidth equal to 5Mb/s $\pm$ 20%, 35% in Class 2 with 1.6Mb/s $\pm$ 20%, 35% in Class 3 with 0.64Mb/s $\pm$ 20%, 15% in Class 4 with negligible upload bandwidth. The video source belongs to Class 1. The average bandwidth per peer is $E[B_a] = 1.25$Mb/s. In each simulation D_{max} is set to 6s and we consider $N = 2000$ peers. According to the assumption that the bottleneck is at
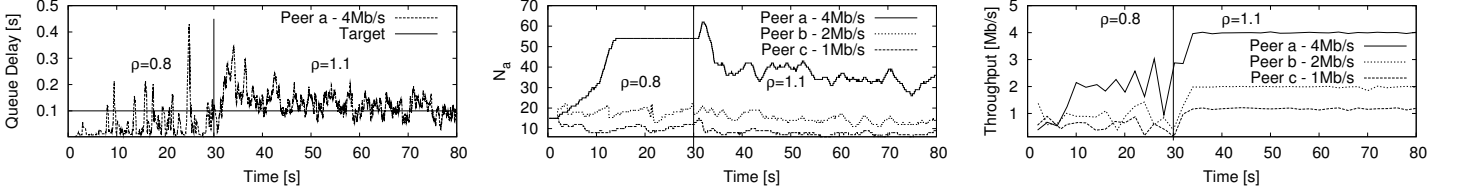
Figure 2: Queuing delay (left), N_a (center) and throughput (right) vs. time for flash crowd.

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 [9]; the overall mean latency is $E[l_{ab}] = 39$ms.

The well-known *Pink of the Aerosmith* video sequence (352x240p resolution, 25fps, 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 (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 $\rho = r_s/E[B_a]$. 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 randomly select 30 other peers as its neighbors. Since connections are bidirectional, the average number of neighbors for a peer is equal to 60. As we simulate a couple of minutes of the system behavior, we neglect the effect of churning so that the topology is static for the whole simulation run. Curves represented in Fig. 3 are obtained averaging the results of four random topologies.

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 multimedia and signal processing communities, instead, the evaluation of the perceived quality is considered mandatory [10], [11]. To this extent, performance is expressed in terms of average Structural Similarity Index (SSIM) [3] 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. It is a highly non linear metric in decimal values between -1 and 1. Values above 0.95 are typically considered of good quality. In our simulation scenario, SSIM has been computed considering video frames received by 100 peers (25 for each class), and then averaging among all of them. The initial and final 10s of video have been discarded to focus on steady state performance.

B. Transient analysis

We consider a scenario where D_0 is set to 100ms and in which the system operating point is abruptly modified at $t = 30$s: a sudden ingress of 400 new peers with negligible upload-bandwidth and a contemporary reduction by 50% of the available upload bandwidth of all peers belonging to Class 3 happens. Given video rate $r_s = 1$Mb/s, this causes the system load ρ to shift from 0.8 to 1.1. Even if this scenario is rather artificial, it has been selected because it maximizes the stress on the control scheme. Fig. 2 reports the evolution of N_a (center) and throughput (right) 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 (left). When $\rho = 0.8$, N_a, N_b and N_c adapt to different values, reflecting each peer's ability to contribute to chunk diffusion. Since $\rho < 1$, not all system capacity is required, and N_a rapidly grows to its maximum value $N_a = W_{\max} = 53$, i.e., the number of a's neighbors. At $t = 30s$, the HRC system reacts to the sudden system condition variations. In particular, for the high bandwidth peer a, N_a initially increases, since the number of its neighbors increases and its capacity was not fully exploited (its queuing delay still being smaller than D_0). Then, the increased system load boosts the percentage of offers that are positively selected, causing additional queuing delay, so that N_a decreases. After a quick transient, upload rate matches each peer upload capacity, and queuing delay reaches the target D_0.

C. 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. 3 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., $N_a = 40$. Even if performance of the scheme with $N_a = 20$ are comparable with that of HRC, setting the value of N_a is very critical, since the optimal value depends on other system parameters, such as the peers upload bandwidth distribution, that, besides being difficult to know,
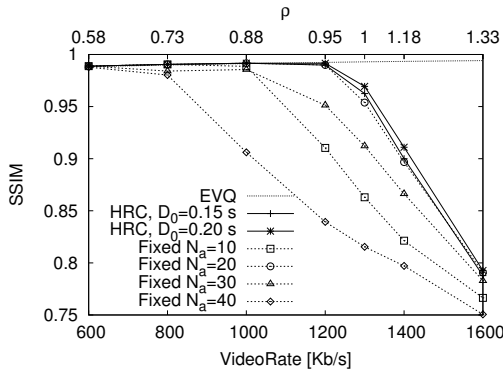
Figure 3: Average SSIM of HRC and non-adaptive schemes versus the system load. Simulation results, 2000 peers.
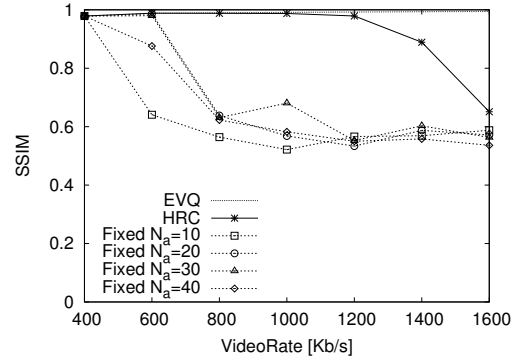

Figure 4: Average SSIM for HRC and non-adaptive schemes when varying video rate r_s. Experimental results in swarm of 1000 peers.

are variable in time due to interfering traffic, as seen in Fig. 2. 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.

IV. EVALUATION BY EXPERIMENT

The HRC controller has been implemented into "WineStreamer" P2P-TV application and we 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. Each peer upload capacity has been artificially limited using a packet level rate limiter embedded in our P2P-TV application: 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 capacity of 1.32Mb/s. Latencies among peers randomly varies between 10ms and 20ms (so that the RTT varies in $[20, 40]$ms). Againt the *Pink of the Aerosmith* video 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.

Focusing on the Quality of Experience, again expressed with SSIM index[1], Fig. 4 compares HRC behavior against non adaptive schemes in which $N_a = 10, 30, 20, 40$ respectively. Results are similar to the one of Fig. 3: 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.

V. 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.

REFERENCES

[1] D. Ciullo, M. A. Garcia, A. Horvath, E. Leonardi, M. Mellia, D. Rossi, M. Telek, and P. Veglia, "Network awareness of P2P live streaming applications: a measurement study," *IEEE Transanctions on Multimedia*, vol. 12, no. 1, pp. 54–63, January 2010.

[2] NAPA-WINE, http://www.napa-wine.eu.

[3] Z. Wang, A. C. Bovik2, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: From error visibility to structural similarity," *IEEE Transactions on Image Processing*, vol. 13, no. 4, pp. 600–612, April 2004.

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

[5] S. Sanghavi, B. Hajek, and L. Massoulié, "Gossiping with multiple messages," in *IEEE INFOCOM*, Anchorage, AK, US, May 2007.

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

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

[8] A. Couto da Silva, E. Leonardi, M. Mellia, and M. Meo, "Exploiting Heterogeneity in P2P Video Streaming," *IEEE Transactions on Computers*, December 2010. [Online]. Available: http://www.tlc-networks.polito.it/leonardi/papers/TC_P2P.pdf

[9] Meridian, http://www.cs.cornell.edu/People/egs/meridian/.

[10] E. Setton, J. Noh, and B. Girod, "Low latency video streaming over peer-to-peer networks," in *IEEE ICME*, Toronto, CA, July 2006.

[11] Z. Liu, Y. Shen, K. W. Ross, S. S. Panwar, and Y. Wang, "Layerp2p: using layered video chunks in p2p live streaming," *IEEE Transaction on Multimedia.*, vol. 11, no. 7, pp. 1340–1352, 2009.

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