

# Hose Rate Control for P2P-TV Systems

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

**Abstract**—In this paper, we focus on mesh based Peer-to-Peer live streaming systems (or P2P-TV) in which the peers form an overlay topology upon which they exchange small “chunks” of video. We propose Hose Rate Control (HRC), a scheme to control the speed at which peers offer chunks to other peers. Our solution achieves several desirable goals: i) it adapts to the actual peer available upload bandwidth and system demand, ii) it achieves a fair resources utilization, iii) it improves system performance with respect to non adaptive schemes.

## I. INTRODUCTION

In mesh-based P2P-TV live streaming systems, the encoded video-stream is sliced in real time into small pieces called *chunks*, which are distributed over an overlay topology accordingly to a distributed epidemic approach. Chunks have to be received by peers within a deadline of few seconds, starting from a *source* that emits chunks in real time. All peers must trade them minimizing delivery delays, to guarantee real-time-like constraints. Since peers are organized into a generic overlay topology, neighboring peers exchange periodically signaling messages to avoid duplicate deliveries. Considering that as transport protocol, UDP is preferred by actual P2P-TV applications [?], there is the need to handle the congestion control and to limit the amount of information a peer transmits, being the download rate limited by the video-rate. In current Internet, the typical capacity bottleneck is the peer upload bandwidth, so the latter must be carefully managed.

Due to the fact that network conditions and peers heterogeneity are typically unpredictable, the optimal setup is difficult to achieve. In particular, the most critical parameter is the number of neighbors to contact in parallel during their trading phases, which in turn guides the number of chunks a peer can transmit in a time interval. We propose a scheme, called *Hose Rate Control*, HRC, to automatically adapt the number of active *signalling threads* to the scenario. By doing so, the scheme implements a peer aggregate transmission rate control that aims at jointly exploiting the peer upload capacity and reducing chunk delivery delay and losses; HRC’s main objective is exploiting the bottleneck bandwidth while not increasing queuing delays (targeting a less than best effort policy). HRC algorithm has been modeled through fluid equations, deeply tested by simulations using realistic scenarios and it is currently being implemented in a P2P-TV application under development within the EU-FP7 NAPA-WINE STREP project (<http://www.napa-wine.eu/>). In this paper, we present a subset of simulation results.

## II. SYSTEM DESCRIPTION

Let us consider the simplest case in which the overlay topology defined by the set of peers and virtual links connecting them, is built once and on a random basis [?], [?]. Every

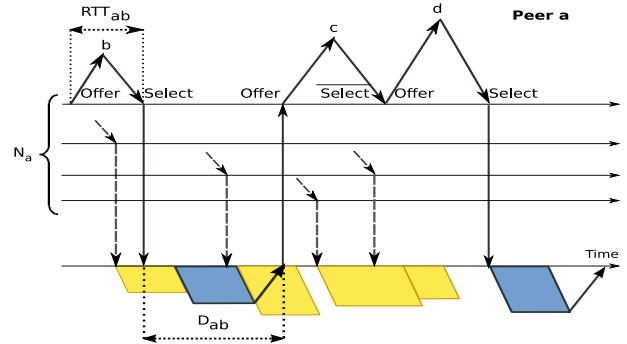


Figure 1. Schematic representation of the peer chunk trading mechanism.

chunk must be delivered within a deadline called *playout delay*,  $D_{max}$ , otherwise it becomes useless and cannot be traded anymore. To trade chunks, peer  $a$  maintains a number of *signalling 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 sets it 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 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  $a$ .
- If  $a$  receives an *ACK* message or a *negative select*, 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 signalling threads limits the number of chunks the peer can offer, and thus control the upload capacity utilization of the peer.

For *peer* and *chunk selection* policies we make the simplest possible choices:  $a$  chooses peers to contact uniformly at random within its neighbors, and they choose chunks to select randomly among the ones they need (“Random Peer - Random Useful Chunk selection”).

### A. 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

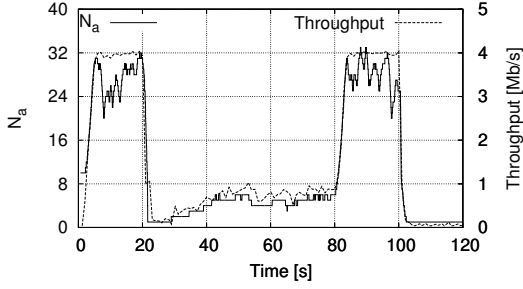


Figure 2. Value of  $N_a$  and throughput versus time with variations due to interfering traffic on upload link.  $\rho = 0.95$

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  $\hat{D} = t_{\text{rx,ack}}^{(a)} - t_{\text{tx,select}}^{(a)} - RTT_{ab}$ , 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  $\Delta N_a = \lfloor W_a(n) \rfloor - \lfloor W_a(n-1) \rfloor$ .

### III. PERFORMANCE EVALUATION

All results shown in this paper have been obtained with event driven simulator *P2PTV-sim* (available at <http://www.napa-wine.eu/cgi-bin/twiki/view/Public/P2PTVSim>). In our scenario, peers are partitioned in four classes based on their upload capacity: 15% of peers with  $5\text{Mb/s} \pm 20\%$ , 35% with  $1.6\text{Mb/s} \pm 20\%$ , 35% with  $0.64\text{Mb/s} \pm 20\%$ , 15% with negligible upload bandwidth. The video source belongs to the first class. The average bandwidth is  $E[B_a] = 1.25\text{Mb/s}$ . In each simulation we considered 2000 peers and set up static overlays given a number of neighbours for each peer equal to 60.  $D_{max}$  was equal to 6s and end-to-end latencies were taken from the experimental dataset of Meridian project (<http://www.cs.cornell.edu/People/egs/meridian>) whose overall mean value is  $E[l_{ab}] = 39\text{ms}$ .

As benchmark we adopted the well-known *Pink of the Aerosmith* video sequence encoded using the H.264/AVC Codec. The video consists of 3000 frames corresponding to 120s of visualization. The nominal video rate of the encoder  $r_s$  is a free parameter we vary to enforce different values of the system load defined as  $\rho = r_s/E[B_a]$ . The video-stream is chopped following a simple *one chunk to one frame* mapping.

Fig. 2 reports the evolution over time of the number of active threads  $N_a$  adjusted by HRC and the throughput for a given peer  $a$  whose upload bandwidth of  $4\text{Mb/s}$  varies due to interfering traffic. Parameter  $D_0$  is set to  $100\text{ms}$  and source has rate  $r_s = 1.2\text{Mb/s}$ , corresponding to  $\rho = 0.95$ . After an initial transient, HRC nicely adapts the offer rate to its  $4\text{Mb/s}$  uplink

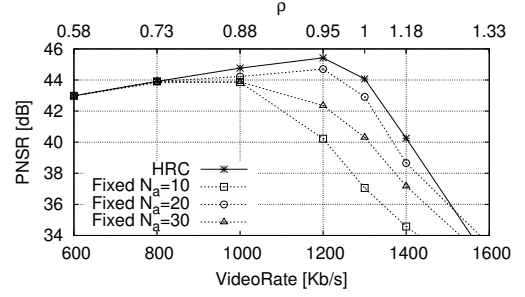


Figure 3. PSNR of HRC and non-adaptive schemes versus the system load. bandwidth ( $N_a = 25$  from  $t = 5\text{s}$  to  $t = 20\text{s}$ ).  $N_a$  is reduced to 5 when available bandwidth is dropped to  $1\text{Mb/s}$  because of an interfering Constant Bit Rate flow ( $t = 20\text{s}$  to  $t = 80\text{s}$ ). From  $t = 80\text{s}$  to  $t = 100\text{s}$  the whole bandwidth turns to be available again and  $N_a$  gets back to 25. Then, from  $t = 100\text{s}$  to  $t = 120\text{s}$ , a TCP-like flow consumes the whole upload bandwidth, dropping  $N_a$  to its minimal value and reducing throughput to negligible values. In conclusion, the control algorithm succeeds in reacting to bandwidth variations, and in achieving less-than-best effort bandwidth utilization.

Focusing on the steady-state performance of HRC, Fig. 3 reports results comparing HRC for  $D_0 = 200\text{ms}$  and schemes in which  $N_a$  is non-adaptive. The video rate is increased (bottom x-axis) to observe the performance of the system when increasing load  $\rho$  (top x-axis). As performance metric, we consider the video quality perceived by peers measured through PSNR, non linear metric in dB that reports of the distortion of the received image compared against the original source. Values above  $40\text{ dB}$  are typically considered of good quality. In all scenarios, HRC outperforms the non-adaptive scheme, for any values of  $N_a$ . Schemes with small values of  $N_a$  do not fully exploit the system bandwidth ( $N_a = 10$ ); schemes with large values of  $N_a$  overload the peer transmission queue leading to larger delivery delays, ( $N_a = 30$ ). Even if the performance of  $N_a = 20$  are comparable with that of HRC, we want to remark that setting the value of  $N_a$  is very critical, since, as said, the optimal value depends on many other system parameters, such as peers upload bandwidth distribution, that are difficult to know and variable in time.

### IV. CONCLUSIONS

Our results show that HRC achieves the goal of efficiently exploiting peers upload bandwidth in mesh-based P2P-TV systems, improving overall system performance: with respect to non-adaptive mechanisms, chunks delivery delays and loss probability are reduced, leading to a higher quality of the delivered video.