Aggregate Flow Control for P2P-TV Streaming Systems

R. Birke, C. Kiraly, E. Leonardi, M. Mellia, M. Meo, S. Traverso

NEM Summit, 29 September 2011, Turin
Outline

1. Introduction
2. Design choices
3. Hose Rate Control
4. Performance Evaluation
5. Conclusions
Introduction
P2P Streaming Systems

- Cheap and stable technology
- Scale up to millions of users
- Many commercial solutions
How it works

- A video-stream is sliced in *chunks*
- Chunks are injected by the *source* into peers’ overlay
- Real time constraints!
Basic Concepts (cont’d)
Design Choices
Pull protocol

- **A trading phase** is required before each chunk transmission

- **Pros:**
  - peers can be organised in generic overlay topologies, i.e., *random graphs*
  - resilience to *churning*

- **But…**
  - design of the trading phase must guarantee *low signaling delays*
Signalling Thread

- A peer publishes the set of chunks it possesses (offer message).
- Peers specify the chunk they want in select messages.
- An ack is sent back once chunk is received.
- UDP as L4 protocol!
System Dynamics

RTT_{AB}

N_A

Peer A

d_{AB}

29/09/2011  NEM Summit 2011
Why this work?

- $N_A$ = number of parallel signalling thread
- $N_A$ is equivalent to a transmitter window:
  - it must match peers’ upload capacity.
- Optimal $N_A$ value depends on the network scenario which is unpredictable
- Auto-adjust $N_A$ to
  - exploit peers’ bandwidth
  - maintain short queues!
Hose Rate Control
Hose Rate Control

- Regulate the transmission rate of chunks adjusting the offer rate \( N_A \)!
- Offer more/less to transmit more/less
- Tunes offer rate looking at transmission delays of chunks
- Aggregate fashion: no end-to-end rate regulation
Hose Rate Control (cont’d)

- The algorithm runs everytime an **ACK** is received:
  1. \( D = t_{rx,ack} - t_{rx,sel} - RTT_{AB} \)
  2. \( W_A(n) = W_A(n-1) - K \times (D-D_0) \)
  3. \( \Delta N_A = \text{floor}(W_A(n)) - \text{floor}(W_A(n-1)) \)

- Where:
  - \( D \) is a queue delay estimation
  - \( D_0 \) is a given target
  - \( K \) is a scaling factor
  - \( W_A \) is the real internal control variable \( (N_A = \text{floor}(W_A)) \)
  - \( t_{rx,ack} \) is the time at which the ACK is received
  - \( t_{rx,sel} \) is the time at which related Select was received
  - \( RTT_{AB} \) is the round trip time between involved peers
Performance Evaluation
Simulation/Testbed Scenario

- Peers have been partitioned in four classes:
  - 15% of peers with upload bandwidth equal to 5 Mb/s;
  - 35% of peers with upload bandwidth equal to 1.6 Mb/s;
  - 30% of peers with upload bandwidth equal to 0.64 Mb/s;
  - 20% of peers with upload bandwidth equal to 0 Mb/s.

- Experiments involved thousands of peers and 3000 chunks
- Real H.264 encoded video sequences used as benchmark
- End-to-end latencies are taken from an experimental data set
- Overlay graph is randomly generated; degree $E[K] = 40$
- **Peer selection** and **chunk scheduling** policies are randomly based
HRC vs Fixed $N_A$: QoE

SSIM index varying $\rho = r_s / E[BU]$.
Queue delay ($D$), number of active signalling threads ($N_A$) and throughput evolution during time adopting HRC ($\rho = 0.9$, $D_0=150\text{ms}$).
HRC Performance (cont’d)

- HRC nicely adapts $N_A$ to the system load!
Conclusions
Conclusions

- **Hose Rate Control** can tune the number of chunks peers can **offer** to their neighbors...
- …to efficiently **exploit** peers **upload bandwidth**…
- …by controlling the **queuing delay** of transmitted chunks
- It improves **system performance** and **Quality of Experience** of users!
Future works

- Already implemented in WineStreamer
- Need to
  - Test HRC AIMD version to compete with TCP
  - Launch HRC into the wild (PlanetLab)
Thank you.
Basic Concepts (cont’d)

- Two families of algorithms for implementing a P2P-TV system:
  - Push scheme (trees).
  - Pull scheme (swarms).