ELASTIC: a Client-side Controller for Dynamic Adaptive Streaming over HTTP (DASH)

Packet Video Workshop - San Jose, CA, USA

12-13 December 2013

L. De Cicco, V. Caldaralo, V. Palmisano, S. Mascolo

Politecnico di Bari, Dipartimento di Ingegneria Elettrica e dell'Informazione
Adaptive Streaming today…

Video distribution platforms use adaptive video streaming, instead of progressive download streaming, to improve the QoE.

Multi-bitrate (stream-switching) is the mainstream approach used to implement adaptive streaming over HTTP (MPEG-DASH, HLS).
How does it work?

- Video is encoded at different bitrate and resolutions, the *video levels*

<table>
<thead>
<tr>
<th>Level</th>
<th>Bitrate</th>
<th>Resolution</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>300 kbps</td>
<td>320x180</td>
</tr>
<tr>
<td>1</td>
<td>700 kbps</td>
<td>640x360</td>
</tr>
<tr>
<td>2</td>
<td>1500 kbps</td>
<td>640x360</td>
</tr>
<tr>
<td>3</td>
<td>2500 kbps</td>
<td>640x360</td>
</tr>
<tr>
<td>4</td>
<td>3500 kbps</td>
<td>1280x720</td>
</tr>
</tbody>
</table>
How does it work?

- Video is encoded at different bitrate and resolutions, the *video levels*
- Each video level is divided into segments
How does it work?

- Video is encoded at different bitrate and resolutions: the *video levels*
- Each video level is divided into segments
- For each video segment the stream-switching controller selects the video level according to channel conditions

Leading architectural approach *(DASH, HLS)*

- The controller is placed at the client
- An HTTP server streams the video
Introduction

The **HTTP server** sends video through an Internet connection with an end-to-end bandwidth \( b(t) \).

The **Client** receives the video segments at a rate \( r(t) \), and stores them in a **playout buffer**.

The **measurement** module feeds the controller with the estimated bandwidth \( \hat{b}(t) \) and the playout buffer level \( q(t) \).

The **controller** dynamically selects the video level \( l(t) \) by sending a HTTP GET request to the HTTP Server.
The player can be in two different states:

- **Buffering phase**: Segments requests are performed back-to-back to quickly fill the playout buffer. This state is left when $q(t) > q_T$
- **Steady-state**: segment requests are spaced to keep the playout buffer level constant.

This generates an **ON-OFF traffic pattern**:

- **ON** when the segments are downloaded
- **OFF** when the player is idle
This ON-OFF traffic pattern causes three problems:

1. Unfair bandwidth utilization when many video share a bottleneck [Akhshabi et al, 2012]
2. Flickering of the requested video level [Akhshabi et al, 2012]
3. Video is not able to get the fair share when in the presence of long-lived TCP flows (*downward spiral effect*) [Huang et al, 2012]
ELASTIC
(fEedback Linearization Adaptive STreamIng Controller)
Conventional approach
Two controllers are used:
1) The first selects the video level to match the available bandwidth
2) The second controls the playout buffer length by regulating the idle period between the download of two segments (on-off traffic pattern).

Proposed approach
► Design one controller that throttles the video level $l(t)$ to drive the playout buffer length $q(t)$ to a set-point $q_T$.
► This eliminates the ON-OFF traffic pattern

The player is always in ON phase unless $l(t)$ is the highest level and $q > Q_{\text{max}} (>q_T)$
Playout buffer model

\[ \dot{q}(t) = \frac{r(t)}{l(t)} - d(t) \]

**Draining rate**

\[ d(t) = \begin{cases} 1 & \text{playing} \\ 0 & \text{paused or } q(t) = 0 \end{cases} \]

**Idea**: Based on the playout buffer model, design a feedback control system that computes \( l(t) \) to steer \( q(t) \) to a threshold \( q_T \).

**Received rate** \( r(t) \): considered as a (measurable) disturbance since it cannot be manipulated.

**\( l(t) \)**: output of the controller
ELASTIC: the control law

\[ \dot{q}(t) = \frac{r(t)}{l(t)} - d(t) ~ (1) \text{ nonlinear system} \]

We impose a linear second order dynamics for the queue

\[ \begin{cases} \dot{q}(t) = -k_p q(t) - k_i q_I(t) ~ (2) \\ \dot{q_I}(t) = q(t) - q_T \end{cases} \text{ integral error dynamics} \]

The additional dynamics is added to make \( q \) converge to \( q_T \) at steady state

By equating (1) and (2) and solving for \( l(t) \) we get the following law for the stream-switching controller:

\[ l(t) = \frac{r(t)}{d(t) - k_p q(t) - k_i q_I(t)} \]
On segment download:

1. $DT = \text{getDownloadTime}(); /* \text{Downl. Time of last segm. (seconds)} */$
2. $S = \text{getSegmentSize}(); /* \text{Last downl. segment Size (bytes)} */$
3. $q = \text{getQueueLength}(); /* \text{current queue length (seconds)} */$
4. $r = h(S/DT); /* \text{Harmonic filter of last 5 received rate samples} */$
5. $qI = qI + DT*(q-qT) /* \text{Update integral error} */$
6. $l = Q(r / ( 1 – kp*q – ki * qI)) /* \text{Control law} */$

The quantizer $Q(x)$ associates to $x$ the highest video level $l$ that is less than $x$
TESTBED
Server Host
► Apache as HTTP Server. Connections are persistent.

Client Host
► NetShaper: we developed this tool to set bandwidth and propagation delays
► AVP uses GStreamer libraries and supports HLS format. Implements several client-side algorithms as AVP plugins.
► IPerf is used to inject long-lived TCP flows
Experiments

Scenarios
S1. Two connections (1 video + 1 video or 1 video + 1 TCP) over a 4Mbps bottleneck
S2. A variable number of video and TCP connections over a 40Mbps bottleneck

Metrics
S1. received video level $l(t)$, received rate $r(t)$, channel utilization.
S2. We consider:
   - $r$: average received rate
   - $U$: channel utilization
   - $JFI$: Jain Fairness Index
   - $RB$: Rebuffering-ratio
   - $n_s$: number of video level switches
Video specifics

- Video encoded at five different bitrates (like Akamai does):

<table>
<thead>
<tr>
<th>Video level</th>
<th>$l_0$</th>
<th>$l_1$</th>
<th>$l_2$</th>
<th>$l_3$</th>
<th>$l_4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bitrate (kbps)</td>
<td>300</td>
<td>700</td>
<td>1500</td>
<td>2500</td>
<td>3500</td>
</tr>
<tr>
<td>Resolution</td>
<td>320x180</td>
<td>640x360</td>
<td>640x360</td>
<td>1280x720</td>
<td>1280x720</td>
</tr>
</tbody>
</table>

- Fragment duration of 2s.
ALGORITHMS

- ELASTIC: $kp = 1/100$, $ki = 1/1000$
- FESTIVE: specifically designed to counteract unfairness in multi-client scenario. Implemented according to the Conext 2012 paper
- PANDA: proposed by Li et al, dynamically computes the duration of OFF phases based on a control law. We employ the parameters suggested in the paper
- Conventional: implemented as described in PANDA paper, it selects the video level as a function of the estimated bandwidth
RESULTS
TWO VIDEOS SHARING A 4 Mbps BOTTLENECK

Results

- Conventional and PANDA: the two videos fairly share the channel with some link underutilization.
- ELASTIC and FESTIVE provide a received video rate that oscillates around the fair share, with an increased number of video level switches.

<table>
<thead>
<tr>
<th>Method</th>
<th>Video 1</th>
<th>Video 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conventional</td>
<td>0.82</td>
<td></td>
</tr>
<tr>
<td>PANDA</td>
<td>0.85</td>
<td></td>
</tr>
<tr>
<td>FESTIVE</td>
<td>0.93</td>
<td></td>
</tr>
<tr>
<td>ELASTIC</td>
<td>0.95</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Level Changes</th>
<th>Video 1</th>
<th>Video 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conventional</td>
<td>2, 2.3</td>
<td>1.9, 2.5</td>
</tr>
<tr>
<td>PANDA</td>
<td>1.6, 2.2</td>
<td>1.6, 2.1</td>
</tr>
<tr>
<td>FESTIVE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ELASTIC</td>
<td>2.3, 2.3</td>
<td>2.5, 2.5</td>
</tr>
</tbody>
</table>
ONE VIDEO AND ONE TCP FLOW SHARING A 4 Mbps BOTTLENECK

- Conventional, PANDA and FESTIVE are not able to obtain the fair share when coexisting with a TCP flow (downward spiral effect)
- ELASTIC avoids the downward spiral effect since it does not produce ON-OFF traffic and behaves as the long-lived TCP flow
CU for ELASTIC is independent of $N_V$ and around 0.96.

Other algorithms exhibit a lower channel utilization for $N_V = 11$.

JFI is in the range $[0.96, 1]$ for all considered algorithms.
Results

$N_V$ VIDEOS SHARING A 40 Mbps LINK WITH $N_{TCP}$ TCP FLOWS

AVERAGE VIDEO FLOW RECEIVED RATE

- ELASTIC is able to get the fair share regardless of the total number of flows and of the fraction of TCP flows
- The per-flow average level obtained by PANDA and Conventional decreases when the number of concurrent flow increases

The elimination of the ON-OFF pattern avoids the «downward spiral effect»
Results

NV VIDEOS SHARING A 40 Mbps LINK WITH NTCP TCP FLOWS

NUMBER OF PER-FLOW LEVEL SWITCHES

- ELASTIC performance is comparable to that of Conventional (slightly worse)
- PANDA provides the best results (under 2 switches) but at the expense of a reduced received video bitrate
Results

For N<50 flows we measured RB ratio less than 0.01
ELASTIC and Conventional provide RB ratios < 0.05
FESTIVE and PANDA exhibit increasing RB ratios.
CONCLUSIONS

► We have proposed ELASTIC a client-side controller which does not generate an ON-OFF traffic pattern
► An experimental evaluation has shown that ELASTIC is able to get the fair share when competing with TCP long-lived flows