Caching in HTTP Adaptive Streaming: Friend or Foe?

Danny Lee
Constantine Dovrolis
Ali C. Begen
Objectives

• What happens when a cache lies between a content server and client?

• What is the simplest scenario that results in bitrate oscillations?

• How can we prevent bitrate oscillations in the presence of caching?
Outline

- Overview of adaptive streaming over HTTP
- Oscillations due to interaction between cache and client
- A traffic shaping solution
- Simulation description
- Experiments and Results
- Conclusions
Outline

• Overview of adaptive streaming over HTTP
• Oscillations due to interaction between cache and client
• A traffic shaping solution
• Simulation description
• Experiments and Results
• Conclusions
Adaptive Streaming over HTTP

- Media is split into “segments”, encoded in multiple bitrates
- Clients adaptively request segments based on their estimate of available bandwidth
Outline

• Overview of adaptive streaming over HTTP
• Oscillations due to interaction between cache and client
  – How does it cause problems?
• A traffic shaping solution
• Simulation description
• Experiments and Results
• Conclusions
Caching - A Simple Model

- Caches may be deployed to reduce upstream bandwidth usage, and provide better downstream latency and bandwidth to clients.

- Media segments are transferred over HTTP, and may be cached in the cache server.

- For a cache in an access network, typically $C_S < C_C$. 
Erroneous Bandwidth Estimation due to Cache Hit

- Clients retrieving cached segments will receive them at $C_C$
  - Causes an artificially high bandwidth estimation

- But faster is better, right?
Bitrate Oscillations

- Problems arise when a mid-quality bitrate video is cached.

- Typically, continuous segments are cached.

- Clients switch between high and low quality video every few seconds – annoying!
Outline

• Overview of adaptive streaming over HTTP
• Oscillations due to interaction between cache and client
• A traffic shaping solution
• Simulation description
• Experiments and Results
• Conclusions
Video Shaping Intelligent Cache (ViSIC)

• Control the bitrates requested by the client by shaping the download speed
  – Prevent erroneous bandwidth estimates
  – Smooth fluctuations in available bandwidth

• Cache Server Implementation
  – Independent of client and origin server
  – Reduce upstream bandwidth usage
  – Serve cached segments faster than no-cache
The Shaping Algorithm - High Level Description

• Estimate $C_S$ and $C_C$ from all traffic passing through cache server

• Select a target bitrate we want the client to use
  – Detect long duration changes in available bandwidth and allow increase/decrease in target bitrate
  – Otherwise favor bitrates of cached segments

• Shape downloads from the cache
  – Use higher rate than target bitrate
  – But lower than what causes clients to switch rates
Outline

• Overview of adaptive streaming over HTTP
• Interaction between cache and client
• A traffic shaping solution
• Simulation description
  • Setup
  • Client and Standard Cache
• Experiments and Results
• Conclusions
Simulation Description - Setup

- Compare different Cache Server scenarios: ViSIC, Standard Cache and No-cache
- We varied $C_S$ and $C_C$
- Representation Bitrates: 256 Kbps, 768 Kbps, 1.5 Mbps, 2.8 Mbps, 4.5 Mbps
Simulation Description – Client and Cache

• **Client**
  – Simulates a typical adaptive streaming player
  – Adjusts requested bitrates based on average segment throughput
  – If video buffer level falls below a low threshold, engage “panic mode”

• **Standard Cache**
  – Simulates a traditional Web cache
  – Cache hit: Serve files at maximum speed
  – Cache miss: Serve files at upstream speed
Outline

• Overview of adaptive streaming over HTTP
• Oscillations due to interaction between cache and client
• A traffic shaping solution
• Simulation description
• Experiments and Results
  • Constant Bandwidth
  • Fluctuating Bandwidth
• Conclusions
Constant Bandwidth

Requested Representation Bitrate (Mbps)

Available Bandwidth
No-Cache
Standard Cache
ViSIC

Time (seconds)
Fluctuating Bandwidth - ViSIC

Requested Representation Bitrate (Mbps)

Time (seconds)

Available Bandwidth
ViSIC

BW Spikes w/ cached segments
BW Drops w/ cached segments
BW Drops w/ uncached segments
BW Spikes w/ uncached segments
Fluctuating Bandwidth - Stability

![Graph showing instability metrics for different segment counts. The y-axis represents Instability Metric, ranging from 0.0 to 1.0, and the x-axis represents Segment Count, ranging from 0 to 250. The graph compares No-Cache, Standard Cache, and ViSIC.]
Outline

• Overview of adaptive streaming over HTTP
• Oscillations due to interaction between cache and client
• A traffic shaping solution
• Simulation description
• Experiments and Results
• Conclusions
Conclusions

• A cache server in the path of an HTTP adaptive streaming client can cause problems
  – Bitrate oscillations, buffer draining

• Cause: Cache hits cause erroneous bandwidth estimations
  – Clients overestimate actual path bandwidth
  – Clients request segments that are unsustainable

• Traffic shaping at the cache can prevent oscillations and buffer drains
  – Maintains cache benefits over no-cache
Acknowledgments

Ashok Narayanan

Ashok presented the cache-induced instability problem at the Adaptive Media Transport Workshop, organized by Cisco, in June 2012
In Memory Of

Saamer Akhshabi
1 April 1987 - 6 March 2014
Thanks
Typical Behavior of a Player

• Estimates available bandwidth using running average of per-segment TCP throughput measurements

• Adaptive segment bitrate selection
  – Increase if throughput is high
    (i.e., can support higher bitrate segments)
  – Decrease if throughput is lower than current bitrate
    (i.e., transfer is slower than real time)

• Client buffer levels affect the state
  – “Panic mode” to recover from low buffer situation
Selecting the Target Bitrate - More Details (i)

• By shaping the download speed, we can cause the client to select specific bitrates - Target Bitrate

• Two possibilities:
  1. **Stay at current segment bitrate**
     • Cache Hit: Avoid erroneous high-bandwidth estimation
     • Absorb short term $C_S$ bandwidth fluctuations
     • Guard against $C_S$ bandwidth decreases when serving cached segments
  2. **Change to bitrate supported by available path bandwidth**
     • Allow client to adapt to long term bandwidth increases/decreases
Selecting the Target Bitrate - More Details (ii)

Case I: When $C_S \leq C_C$
- For a cache hit
  - If $C_S$ has a long term increase, use current path bw
  - Else shape at current segment bitrate
- For a cache miss
  - If $C_S$ has a long term increase or decrease, use current path bw
  - Else shape at current segment bitrate

Case II: When $C_S > C_C$
- Whether cache hit or miss
  - If $C_C$ has a long term increase, use current path bw
  - Else shape at current segment bitrate
The Shaping Algorithm - More Details

• Make use of higher available bandwidth between cache server and client
  – Serve a segment at a higher speed than the target bitrate

• Solution:
  1. Select the next higher representation bitrate
  2. Multiply it by a factor $\beta$ (we used 0.9)

• Shape at a higher speed than current bitrate but lower than the next higher representation
  – Make use of $C_S < C_C$
  – Better performance than No-Cache
The Shaping Algorithm - Example

- **Representation Bitrates:** 256 Kbps, 768 Kbps, 1.5 Mbps, 2.8 Mbps, 4.5 Mbps
- **$C_C$:** 4.0 Mbps
- **Target Bitrate:** 1.5 Mbps
- **Shaped Bitrate:** $0.9 \times 2.8 \text{ Mbps} = 2.52 \text{ Mbps}$
- **Client will continue to request 1.5 Mbps segments, but receive them at a higher rate!**
Simulation Description - Standard Cache

• Functions as a cut-through cache
  – Intercepts HTTP requests to the server
  – Files not in cache don’t need to be fully downloaded to be served

• List of files that represents cached segments

• If a file exists on disk, cache hit: served at $C_C$

• Cache miss
  – Starts a new transfer, serves file one RTT later
  – Effective bandwidth is $C_S$
Fluctuating Bandwidth - Full Results

Requested Representation Bitrate (Mbps) vs. Time (seconds)

- Available Bandwidth
- No-Cache
- Standard Cache
- ViSIC
Fluctuating Bandwidth - Playback Start Time

- Playback starts after client buffer is full

<table>
<thead>
<tr>
<th>Scenario</th>
<th>Playback Start (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ViSIC</td>
<td>11.954</td>
</tr>
<tr>
<td>No-cache</td>
<td>14.258</td>
</tr>
<tr>
<td>Standard Cache</td>
<td>15.664</td>
</tr>
</tbody>
</table>
Fluctuating Bandwidth - Buffer Fullness

We aim to keep the buffer full
- Allows user to seek in buffered region
- Minimal quality disruptions during playback

Playback buffer is near full for ViSIC in all scenarios

Panic Mode engaged multiple times for no-cache and standard cache