Efficient and Adaptive Content Delivery of Linear and Interactive Branched Videos

Vengatanathan Krishnamoorthi
Licentiate presentation
4 November 2016
Video streaming landscape
Video streaming landscape
Video streaming landscape
Motivation

- Efficient and adaptive streaming
  - Streaming services contribute to over 60% of the global Internet traffic currently
  - By 2020, this share is expected to be over 80%
  - Systems need to be well understood, scalable, and efficient to match growth projections
Motivation

- Content personalization and personalized streaming
  - Regular web content is dynamic and personalized, while videos have remained largely unchanged
  - Viewer’s tastes vary significantly
  - Personalized streaming is relatively unexplored and several interesting questions remain open
Contributions: overview

• The contributions in this thesis are in the following areas related to efficient and adaptive content delivery:
Contributions: overview

• The contributions in this thesis are in the following areas related to efficient and adaptive content delivery:
  • Proxy-assisted delivery of linear (regular) videos
Contributions: overview

• The contributions in this thesis are in the following areas related to efficient and adaptive content delivery:
  • Proxy-assisted delivery of linear (regular) videos
  • Efficient and personalized streaming of interactive videos
Contributions: overview

• The contributions in this thesis are in the following areas related to efficient and adaptive content delivery:

Subtopic 1 • Proxy-assisted delivery of linear (regular) videos

• Efficient and personalized streaming of interactive videos
Contributions: overview

- The contributions in this thesis are in the following areas related to efficient and adaptive content delivery:
  - Proxy-assisted delivery of linear (regular) videos
  - Efficient and personalized streaming of interactive videos
Background
HTTP-based Streaming
HTTP-based Streaming

- HTTP-based streaming
HTTP-based Streaming

- HTTP-based streaming
  - Video is split into chunks
HTTP-based Streaming

- HTTP-based streaming
  - Video is split into chunks
  - Easy firewall traversal and caching
HTTP-based Streaming

- Video is split into chunks
- Easy firewall traversal and caching
- Support for interactive VoD (Video on Demand)
HTTP-based Adaptive Streaming (HAS)

- HTTP-based streaming
  - Video is split into chunks
  - Easy firewall traversal and caching
  - Support for interactive VoD (Video on Demand)
- HTTP-based **adaptive** streaming
  - Clients adapt quality encoding based on buffer/network conditions
HTTP-based Adaptive Streaming (HAS)

- **HTTP-based streaming**
  - Video is split into chunks
  - Easy firewall traversal and caching
  - Support for interactive VoD (Video on Demand)

- **HTTP-based adaptive streaming**
  - Clients adapt quality encoding based on buffer/network conditions
HTTP-based Adaptive Streaming (HAS)

- **HTTP-based streaming**
  - Video is split into chunks
  - Easy firewall traversal and caching
  - Support for interactive VoD (Video on Demand)

- **HTTP-based adaptive streaming**
  - Clients adapt quality encoding based on buffer/network conditions
HTTP-based Adaptive Streaming (HAS)

- **HTTP-based streaming**
  - Video is split into chunks
  - Easy firewall traversal and caching
  - Support for interactive VoD (Video on Demand)

- **HTTP-based adaptive streaming**
  - Clients adapt quality encoding based on buffer/network conditions
HTTP-based Adaptive Streaming (HAS)

- HTTP-based streaming
  - Video is split into chunks
  - Easy firewall traversal and caching
  - Support for interactive VoD (Video on Demand)

- HTTP-based adaptive streaming
  - Clients adapt quality encoding based on buffer/network conditions
HTTP-based Adaptive Streaming (HAS)

- **HTTP-based streaming**
  - Video is split into chunks
  - Easy firewall traversal and caching
  - Support for interactive VoD (Video on Demand)

- **HTTP-based adaptive streaming**
  - Clients adapt quality encoding based on buffer/network conditions
Background

Subtopic 1: Proxy caches
Proxy caches
Proxy caches
Proxy caches and HAS

• Clients typically want:
  – High playback quality
  – No buffer interruptions
  – Small stall times
  – Few quality switches
Proxy caches and HAS

• Clients typically want:
  – High playback quality
  – No buffer interruptions
  – Small stall times
  – Few quality switches

• Service providers typically want:
  – High QoE of customers/clients
  – Low bandwidth usage
Proxy caches and HAS
Proxy caches and HAS

HTTP Server → Proxy cache → Request → Response

Client 1

Quality

Proxy before

1,4 2,4 3,4 4,4 5,4 6,4 7,4...
1,3 2,3 3,3 4,3 5,3 6,3 7,3...
1,2 2,2 3,2 4,2 5,2 6,2 7,2...
1,1 2,1 3,1 4,1 5,1 6,1 7,1...

Quality

Proxy after

1,4 2,4 3,4 4,4 5,4 6,4 7,4...
1,3 2,3 3,3 4,3 5,3 6,3 7,3...
1,2 2,2 3,2 4,2 5,2 6,2 7,2...
1,1 2,1 3,1 4,1 5,1 6,1 7,1...
Proxy caches and HAS

HTTP Server → Proxy cache → Client

Client 1

Quality

Time

Proxy before

Proxy after
Proxy caches and HAS

Client 2
Proxy caches and HAS

HTTP Server → Proxy cache → Client

Client 2

Proxy before

Proxy after
Proxy caches and HAS

• However,
  – Proxy caches can also inflate client’s bandwidth estimates
  – Clients are exposed to actual end-to-end throughput only when cache misses occur
Contributions

• Our main contributions are:
  – Study on effects of proxy caches on HAS streams
Contributions

• Our main contributions are (subtopic 1):
  – Study on effects of proxy caches on HAS streams
  – Propose and evaluate HAS-aware proxy caches to improve bandwidth utilization and QoE
Background

Subtopic 2: Interactive branched video
Interactive branched video

• Video personalization through user interaction
Interactive branched video

- Video personalization through user interaction
  - Viewer streams a recorded video, with predefined branch points and branch options
Interactive branched video

• Video personalization through user interaction
  • Viewer streams a recorded video, with predefined branch points and branch options
  • Viewer interaction defines the chosen branch, and therefore the storyline
Interactive branched video

- Video personalization through user interaction
  - Viewer streams a recorded video, with predefined branch points and branch options
  - Viewer interaction defines the chosen branch, and therefore the storyline

```
Option A
  A1
  A2
  A3

Option B
  B1
  B2
  B3

C1
C2
C2
D1
D2

End 1
End 2
End 3
End 4
End 5
```
Interactive branched video

- Video personalization through user interaction
Interactive branched video

- Video personalization through user interaction

**What do you do?**

- Chase after the notes
- Make it up
- "Sorry, it's my first week, I have no idea."
Interactive branched video

• Regardless of interactivity, user experience and user satisfaction is greatly influenced by:
  – Playback stalls and quality fluctuations
Interactive branched video

• Regardless of interactivity, user experience and user satisfaction is greatly influenced by:
  – Playback stalls and quality fluctuations
  – Current interactive branched players split a video into many sub videos and then link them
Interactive branched video

• Regardless of interactivity, user experience and user satisfaction is greatly influenced by:
  – Playback stalls and quality fluctuations
  – Current interactive branched players split a video into many sub videos and then link them

• Issues
  – Playback stalls when playing a new video
  – Non-adaptive playback
Contributions

- Our main contributions are (subtopic 2):
  - Propose, implement and evaluate a framework for stall-free branched video streaming over HTTP
Subtopic 1: Proxy-assisted delivery of HAS videos
Establishing a baseline client

- At the time, several implementations of HAS players were available

<table>
<thead>
<tr>
<th>Player</th>
<th>Container</th>
<th>Open Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>Microsoft smooth streaming</td>
<td>Silverlight</td>
<td>✗</td>
</tr>
<tr>
<td>Netflix player</td>
<td>Silverlight</td>
<td>✗</td>
</tr>
<tr>
<td>Apple HLS</td>
<td>QuickTime</td>
<td>✗</td>
</tr>
<tr>
<td>Adobe OSMF</td>
<td>Flash</td>
<td>✓</td>
</tr>
<tr>
<td>Youtube player</td>
<td>HTML5/Flash</td>
<td>✗</td>
</tr>
</tbody>
</table>
Establishing a baseline client

Adobe’s OSMF (Open Source Media Framework) v1.6 and v2.0

- Instrumented the OSMF client to log internal parameters
  - Buffer occupancy
  - Playback quality
  - Stall occurrences and duration, etc.,
Establishing a baseline proxy

- We use a squid proxy and its default setting as the baseline
Simulating network characteristics

- We use dummynet to simulate varying network characteristic. We evaluate under different,
  - Bandwidths
  - RTTs
  - Packet loss rates
  - Bottleneck location (client-proxy and proxy-server)
Policies and classes

- Baseline policies
  - Empty cache
Policies and classes

- Baseline policies
  - Empty cache
  - Full cache (preload all versions)
Policies and classes

- **Baseline policies**
  - Empty cache
  - Full cache (preload all versions)
  - Best effort (default, as previous example)
Policies and classes

- Quality and content-aware prefetching policies
  - 1-ahead
  - N-ahead
  - Priority-based
Policies and classes

- Quality and content-aware prefetching policies
  - 1-ahead
  - N-ahead
  - Priority-based

If client switches to a higher encoding and it is not the first time, then prefetch:
Policies and classes

• Quality and content-aware prefetching policies
  — 1-ahead
  — N-ahead
  — Priority-based

If client switches to a higher encoding and it is not the first time, then prefetch:
  (i) current Q, (ii) one Q level below, (iii) one Q level above, and (iv) no prefetching.
Policies and classes

• Quality and content-aware prefetching policies
  – 1-ahead
  – N-ahead
  – Priority-based

If client switches to a higher encoding and it is not the first time, then prefetch:
  (i) current Q, (ii) one Q level below, (iii) one Q level above, and (iv) no prefetching.
Else prefetch:
Policies and classes

• Quality and content-aware prefetching policies
  — 1-ahead
  — N-ahead
  — Priority-based

If client switches to a higher encoding and it is not the first time, then prefetch:
  (i) current Q, (ii) one Q level below, (iii) one Q level above, and (iv) no prefetching.
Else prefetch:
  (i) current Q, (ii) one Q level above, (iii) one Q level below and (iv) no prefetching.
Policies and classes

- **Client-proxy cooperation policies**
  - Buffer oblivious (priority-based prefetching)
  - Buffer aware (conservative quality during low buffer conditions)
Policies: overview

• **Baseline policies**
  — Empty cache
  — Full cache (preload all versions)
  — Best effort (default, as previous example)

• **Quality and content-aware prefetching policies**
  — 1-ahead
  — N-ahead
  — Priority-based

• **Client-proxy cooperation policies**
  — Buffer oblivious (priority-based prefetching)
  — Buffer aware (conservative quality during low buffer conditions)
Evaluation: Client-proxy bottleneck

Quality level

Stall times
Evaluation: Client-proxy bottleneck

- Proxies provide only limited performance advantages under client-proxy bottleneck
Evaluation: Client-proxy bottleneck

- Proxies provide only limited performance advantages under client-proxy bottleneck
- Some performance improvements with prefetching (but penalty for excessive prefetching)
Evaluation: Proxy-server bottleneck

- Quality level
- Stall times
Evaluation: Proxy-server bottleneck

• Large performance potential for proxy caching
Evaluation: Proxy-server bottleneck

- Large performance potential for proxy caching
- Significant performance improvement with the best effort policy
Evaluation: Proxy-server bottleneck

- Large performance potential for proxy caching
- Significant performance improvement with the best effort policy
- Naive prefetching results in penalty. Need for more intelligent prefetching policies (cooperative)
Evaluation: co-operative policies

- For client-proxy bottleneck, both policies slightly outperform all baseline and quality-aware prefetching policies (right)
Evaluation: co-operative policies

- For proxy-server bottleneck, both policies **vastly outperform** all baseline and quality-aware prefetching policies (right)
Proxy-assisted HAS: Conclusions

• Performance impact of HAS-aware proxy policies
  – Baseline policies
  – Quality and content-aware prefetching
  – Client-proxy cooperation
Proxy-assisted HAS: Conclusions

• Performance impact of HAS-aware proxy policies
  – Baseline policies
  – Quality and content-aware prefetching
  – Client-proxy cooperation

• Bottleneck location and network conditions play central roles in which policies are most advantageous
Proxy-assisted HAS: Conclusions

• Performance impact of HAS-aware proxy policies
  – Baseline policies
  – Quality and content-aware prefetching
  – Client-proxy cooperation

• Bottleneck location and network conditions play central roles in which policies are most advantageous

• Proxy design and policy selection is very important
Subtopic 2: Interactive branched videos
• Branched video and branch points
  – The video can include branch points, with multiple branch choices
  – User selects which segment to play back next
HAS-based interactive branched video

- Branched video and branch points
  - The video can include branch points, with multiple branch choices
  - User selects which segment to play back next
- Our solution: Combine branched video and HAS
HAS-based interactive branched video

- Branched video and branch points
  - The video can include branch points, with multiple branch choices
  - User selects which segment to play back next
- Our solution: Combine branched video and HAS
- Goal: Seamless playback even if user decision at last possible moment
Problem description and constraints
Problem description and constraints

- Problem: Maximize quality, given playback deadlines and bandwidth conditions
Problem description and constraints

- Objective function:

\[
\text{maximize } \sum_{i=1}^{n_e} q_i l_i + \sum_{i=n_e+1}^{n_e+|\mathcal{E}^b|} w_e^b q_i l_i
\]
Problem description and constraints

- Objective function:

$$\text{maximize } \sum_{i=1}^{n_e} q_i l_i + \sum_{i=n_e+1}^{n_e + |E^b|} w_i^b q_i l_i$$

Beginning of next segment
Problem description and constraints

- Download order: round robin (optimal)
Problem description and constraints

- Download order: round robin (optimal)
Problem description and constraints

- Download order: round robin (optimal)
Problem description and constraints

- Download order: round robin *(extra workahead)*
 Once branch point has been traversed, move on to next segment ...
Problem description and constraints

Download schedule: 1 2 3 4 7 10
Problem description and constraints

Playback deadlines:

- For seamless playback without stalls, e.g., chunks 2 and 3,

\[ t^c_i \leq t^d_i = \tau + \sum_{j=1}^{i-1} l_j, \quad \text{if } 1 \leq i \leq n_e \]
Problem description and constraints

- Download schedule:

```
1  2  3  4  7  10
```

- Download completion times

- Playback deadlines:
  - For seamless playback without stalls, e.g., chunks 2 and 3,
    \[
    t_i^c \leq t_i^d = \tau + \sum_{j=1}^{i-1} l_j, \quad \text{if } 1 \leq i \leq n_e
    \]
Problem description and constraints

Playback deadlines:

- For seamless playback without stalls, e.g., chunks 2 and 3,

\[
\left( t_i^c \right) \leq t_i^d = \tau + \sum_{j=1}^{i-1} l_j, \quad \text{if } 1 \leq i \leq n_e
\]
Playback deadlines:

- For seamless playback of first chunks in next segment: e.g., 4, 7, and 10

\[ t^c_i \leq t^d_i = \tau + \sum_{j=1}^{n_e} l_j, \quad \text{if } n_e < i \leq n_e + |\mathcal{E}^b| \]
Problem description and constraints

- Playback deadlines:
  - For seamless playback of first chunks in next segment: e.g., 4, 7, and 10

\[ t^c_i \leq t^d_i = \tau + \sum_{j=1}^{n_e} l_j, \quad \text{if } n_e < i \leq n_e + |E^b| \]
Prefetching policies

- At download completion
  - Decide number of chunks to download next (number of connections)
  - Decide quality level of chunks
  - Maximize expected weighted playback
Prefetching policies

• At download completion
  – Decide number of chunks to download next (number of connections)
  – Decide quality level of chunks
  – Maximize expected weighted playback
• Exponential number of candidate schedules
Prefetching policies

- At download completion
  - Decide number of chunks to download next (number of connections)
  - Decide quality level of chunks
  - Maximize expected weighted playback
- Exponential number of candidate schedules
- Our optimized policies restrict the number of candidate schedules to consider
  - Policies differ in number of candidate schedules and how aggressive they are (quality choice)
### Comparison between policies

<table>
<thead>
<tr>
<th>Policy</th>
<th>Connections</th>
<th>Schedules considered</th>
<th>Objective</th>
</tr>
</thead>
<tbody>
<tr>
<td>All schedules</td>
<td>$1 \leq c_i \leq C_{\text{max}}$</td>
<td>$Q^M$, where $M=n_e+</td>
<td>\xi_b</td>
</tr>
<tr>
<td>Optimized non-increasing quality</td>
<td>$1 \leq c_i \leq C_{\text{max}}$</td>
<td>$\begin{cases} M+Q-1 \ Q-1 \end{cases}$</td>
<td>$\sum_{i=1}^{n_e} q_{i/l_{i}} + \sum_{i=n_e+1}^{n_e+</td>
</tr>
<tr>
<td>Optimized maintainable quality</td>
<td>$1 \leq c_i \leq C_{\text{max}}$</td>
<td>$Q$</td>
<td></td>
</tr>
</tbody>
</table>

- **Total number of schedules:** $Q^M$
- **Optimized non-increasing quality:**
  - Constraint: Qualities of consecutive chunks are non-increasing
### Comparison between policies

<table>
<thead>
<tr>
<th>Policy</th>
<th>Connections</th>
<th>Schedules considered</th>
<th>Objective</th>
</tr>
</thead>
<tbody>
<tr>
<td>All schedules</td>
<td>$1 \leq c_i \leq C_{\text{max}}$</td>
<td>$Q^M$, where $M = n_e +</td>
<td>\xi_b</td>
</tr>
<tr>
<td>Optimized non-increasing quality</td>
<td>$1 \leq c_i \leq C_{\text{max}}$</td>
<td>$\begin{pmatrix} M+Q-1 \ Q-1 \end{pmatrix}$</td>
<td>$\sum_{i=1}^{n_e} q_i l_i + \sum_{i=n_e+1}^{n_e+</td>
</tr>
<tr>
<td>Optimized maintainable quality</td>
<td>$1 \leq c_i \leq C_{\text{max}}$</td>
<td>$Q$</td>
<td>$\sum_{i=1}^{n_e} q_i l_i + \sum_{i=n_e+1}^{n_e+</td>
</tr>
</tbody>
</table>

- **Total number of schedules:** $Q^M$
- **Optimized non-increasing quality:**
  - Constraint: Qualities of consecutive chunks are non-increasing
- **Optimized maintainable quality:**
  - Constraint: Chosen quality must be sustainable
Comparison between policies

<table>
<thead>
<tr>
<th>Policy</th>
<th>Connections</th>
<th>Schedules considered</th>
<th>Objective</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single connection</td>
<td>1</td>
<td>Q</td>
<td>$n_e \sum_{i=1}^{q_i}</td>
</tr>
</tbody>
</table>

- Single connection: baseline comparing to policies which do not use multiple connections
Comparison between policies

<table>
<thead>
<tr>
<th>Policy</th>
<th>Connections</th>
<th>Schedules considered</th>
<th>Objective</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single connection</td>
<td>1</td>
<td>Q</td>
<td>$\sum_{i=1}^{n_e} q_i</td>
</tr>
</tbody>
</table>

- Single connection: baseline comparing to policies which do not use multiple connections
- Naïve: benchmark to regular branched video players
Test scenario
Test scenario

Worst case scenario

• always pick the last segment
• at last possible moment
Test scenario

• Default scenario:
  – Segment length: 5
  – Branch options per branch point: 4
  – Branch points: 3
Policy comparison

- Naïve policy: does not perform prefetching
  - Stalls at every branch point
  - Note: High playback rate is misleading on its own
Policy comparison

- **Optimized maintainable quality** provides best tradeoff
Policy comparison

- **Optimized maintainable quality** provides best tradeoff
  - Much lower stall probability
Policy comparison

- **Optimized maintainable quality** provides best tradeoff
  - Much lower stall probability
  - Tradeoff is somewhat lower playback quality
Policy comparison

- **Optimized non-increasing quality** is aggressive
  - Higher playback rate
  - More stalls
Policy comparison

- **Optimized non-increasing quality** is aggressive
  - Higher playback rate
  - More stalls
Policy comparison

- **Optimized non-increasing quality** is aggressive
  - Higher playback rate
  - More stalls
Policy comparison

- **Single connection** does not use parallel connections
  - Good (slightly higher) playback rate
  - Much more stalls
• *Single connection* does not use parallel connections
  – Good (slightly higher) playback rate
  – Much more stalls
Policy comparison

- **Single connection** does not use parallel connections
  - Good (slightly higher) playback rate
  - Much more stalls
Impact of segment lengths

Segment length
Impact of segment lengths

- Quality increases with more chunks per segment
- Very many stalls if segments are too short
Impact of branch options

Branch options
Impact of branch options

- Stalls frequent when too many branch options
  - Single connection struggles the most
HAS-based branched video: Conclusion

• Designed and implemented branched video player that achieve seamless branched streaming

• Designed optimized policies that maximize playback quality while ensuring sufficient workahead

• Evaluation shows that solution effectively adapt to varying conditions

Our interactive branched video implementation can be downloaded from: http://www.ida.liu.se/~nikca89/papers/mm14.html
Summary
Summary

- In this thesis, we have:
  - Evaluated the performance impact of proxy caches on HAS clients
  - Designed and evaluated collaborative policies between HAS clients and proxy caches
Summary

• In this thesis, we have:
  – Evaluated the performance impact of proxy caches on HAS clients
  – Designed and evaluated collaborative policies between HAS clients and proxy caches
  – Proposed, designed, implemented and evaluated stall-free HAS-based branched streaming
Works presented were in collaboration with ...

• Patrik Bergström (Linköping University, Sweden)
• Niklas Carlsson (Linköping University, Sweden)
• Derek Eager (University of Saskatchewan, Canada)
• Anirban Mahanti (NICTA, Australia)
• Nahid Shahmehri (Linköping University, Sweden)
Papers in this thesis


Efficient and Adaptive Content Delivery of Linear and Interactive Branched Videos

Vengatanathan Krishnamoorthi