Evaluating and Improving Push based Video Streaming with HTTP/2

Mengbai Xiao¹, Vishy Swaminathan², Sheng Wei²³, Songqing Chen¹

¹George Mason University, ²Adobe Systems, ³University of Nebraska-Lincoln
HTTP Streaming

- YouTube
- Netflix
- Vuclip
- Twitch
HTTP Streaming

HTTP request (Qlty#1) → Video segment (Qlty#1) → CDN

Video:
- Qlty#1
- Qlty#2
- Qlty#3
- Qlty#4

Cache
Downgraded Throughput

- **HTTP streaming** is build upon TCP
  - Sawtooth pattern traffic

- Short segment duration
  - live latency
  - High network adaptability
  - User abandonment behaviors lead to less waste of network resources
  - request explosion
Video Streaming enhanced by HTTP/2

Cable TV Users

HTTP Streaming Users

1:0

“Goal”
K-Push

HTTP 1.1

Client

Server

(a) No-Push

(b) All-Push

(c) K-Push

HTTP/2

Request

Response

Push

Client

Server

Client

Server

Client

Server

Client

Server
Demo

Playback

Regular Approach

~18.7 sec

Our Approach

~3.7 sec

Live Event
Server Push Mechanism and k-push

- The **server push mechanism** is proposed in **HTTP/2**
  - HTTP servers **speculatively** push back HTTP responses that the client does **not** request yet

- **K-push** is a solution to relieve the **request explosion** problem
  - One request for k+1 segments

- More practical than the **pipeline solution**
  - **Less requests** are delivered via network and are processed on servers
  - The **knowledge of future segment URLs** are not required
1. Analyze K-Push
2. Adaptive Push
3. Evaluate
K-push Model

- **K-push description**
  - A push cycle consists of a HTTP request and k+1 HTTP responses
  - The lead segment of a push cycle is the first segment transmitted
    - Lead segment implies the bit rate level selected for the push cycle
Verification

- Experimental parameters
  - Video length: 120s
  - Video qualities: 49 kbps, 217 kbps, 504 kbps, 752 kbps
  - Number of segments pushed: 0, 1, 4, 9
  - Bandwidth: 200 kbps, 560 kbps, 880 kbps
  - Round trip time: 20 milliseconds, 300 milliseconds, 500 milliseconds

- The bandwidth is carefully capped to make quality switch reflect the streaming throughput
Experimental Results

- Request overhead is caused by two dimensions
  - Request number
  - Round trip time

- Playback bandwidth is defined as the effective bandwidth to deliver video payload
  - Increasing $k$ substantially improves the playback bandwidth
Beyond Playback Bandwidth

- **Diminishing marginal returns** with the increasing number of segments pushed

- **Network adaptability** is not improved with reduced segment duration

- **Over-push problem**
  - User may decide not to continue watching a video after checking the first few seconds
Adaptive Push
Adaptive-push Design

- **Adaptive-push** features *dynamically* scaling the number of segments pushed in a video session.

- A small $k$ is selected for the first push cycle to alleviate over-push problem.
  - A number of users stop watching the videos after checking the first few seconds.

---

CDF

![Diagram showing the cumulative distribution function (CDF) of user watching portion. The x-axis represents the user watching portion ranging from 0 to 1, and the y-axis represents the CDF ranging from 0 to 1. The graph starts at (0, 0) and ends near (1, 1), indicating a positive trend.]
Adaptive-push Design (Cont.)

- The number of segments pushed is increased in various rates
  - When \( k \) is small, a high increment rate is preferred to improve the playback bandwidth
  - When \( k \) is large, a low increment rate or non-increment is more appropriate due to the diminishing playback bandwidth improvement

- The number of segments pushed is constraint by current buffer status
  - Long buffer length more efficiently absorbs network fluctuations

\[(k + 1)\left(\frac{bD}{B} - D\right) < L\]
Implementation

- The push directive is carried in the HTTP request header
  - PushDirective: k

- Jetty HTTP/2 server
  - Jetty project

- Video player: dash.js
  - Dynamically determined k

- tc is used to manipulate bandwidth and RTT
Evaluation
Evaluation

- **Same experimental parameters** as those in k-push analysis

- **Additional adaptive-push schemes**
  - Aggressive/moderate/conservative bandwidth prediction (880/415/49 kbps)
  - Two increment rates: 2k, k+1
    - Increment rate is decreased if the playback bandwidth improvement is detected less than 10%
    - Increment of k is stopped if the playback bandwidth improvement is detected less than 2%

- Evaluate playback bandwidth, network adaptability, and over-pushed content
Playback Bandwidth

- **Progressive** download the experimental video
  - Playback bandwidth is derived by dividing the overall downloaded segment size by the time consumed

- Adaptive-push outperforms regular HTTP/1.1 streaming, approaching k-push
K Variation

- Two increment rates are observed in the figures.
- K is always low if small RTT is observed, which means little playback bandwidth improvement when increasing k.
- Fluctuations occur when a large k is desired but the buffer length is low.

![Graph showing K Variation with different RTTs and bandwidths.](image-url)
Network Adaptability

- Experimental parameters
  - Length: 5 minutes
  - buffer length: 30 seconds
  - Network condition varies every 30 seconds
    - Bandwidth: 480 kbps, 640 kbps, 800 kbps
    - RTT: 20 ms, 300 ms, 500 ms

Results

<table>
<thead>
<tr>
<th></th>
<th>Buffer Length (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>E</td>
</tr>
<tr>
<td>no-push</td>
<td>29.08</td>
</tr>
<tr>
<td>9-push</td>
<td>16.46</td>
</tr>
<tr>
<td>a-push-con</td>
<td>23.55</td>
</tr>
</tbody>
</table>
Over-pushed Simulation

- Apple HLS trace collected at client side from Vuclip
  - 07/15/2015 ~ 08/31/2015
  - ~ 12 million video sessions

- Simulator implemented in perl
  - Requests are sent only if previous downloaded segments are watched
  - Requested bitrate is determined by the measured playback bandwidth of last push cycle
Over-pushed Content

CDF Over-pushed Video Length (s)

20ms-con  20ms-mod  20ms-agg  500ms-con  500ms-mod  500ms-agg  no-push  k-push

540 kbps actual bandwidth

880 kbps actual bandwidth
Dynamic Scaling of Number of Segments Pushed

Good Trade off between K-push and HTTP 1.1

Conclusion
Questions ?