Towards a Small Buffering Delay in Adaptive Video Streaming

Presenter: Tobias Lange

Yongtao Shuai, Thorsten Herfet

{shuai, herfet, lange}@nt.uni-saarland.de
Motivation

State-of-the-art rate adaptation is not suitable for low-latency dynamic streaming, due to a lack of explicit stabilization of client buffer dynamics.

In case the client buffer is at its maximum level (the maximal buffering delay),

- interactions with TCP’s flow control may lead to a biased throughput feedback, and result in undesirably variable and low video quality;[1]
- \textit{ON-OFF streaming pattern} occurs, and may cause unfairness with multiple video streaming sessions.[2]

In contrast to existing solutions that focus on buffer control at near-zero buffer levels, a stabilization of buffer dynamics with a filled buffer is an open issue.

Outline

✓ Motivation

☐ Rate control for buffer stabilization
  - Buffering delay
  - Modeling buffer dynamics
  - Rate selection

☐ Prototype implementation
  - Server-based streaming architecture (Open-Loop rAte Control, OLAC)
  - Transport protocol configuration

☐ Results
Buffering Delay

- **Buffering Delay** is buffered video in seconds.
- We achieve *low-latency* dynamic video streaming with buffering delays as low as the chunk-duration.
A Model of Buffer Dynamics

Client buffer

Fill-rate (network throughput)

Drain-rate (video playback rate)

Too high

Desired buffer level

Too low

Quality Selection

Stabilizing the buffer to the desired level by regulating the drain-rate, i.e. by selecting a video bit rate for the chunk.
A Model of Buffer Dynamics

Express the buffer level in **seconds of video**.

\[
a(t) \quad \text{the throughput rate achieved at the time } t
\]

\[
r(t) \quad \text{the selected video bit rate at the time } t
\]
A Model of Buffer Dynamics

Compute the buffer level every discrete chunk.

\[
\text{Fill-rate} \quad \frac{a[i]}{r[i]} \cdot t[i] \\
\text{Drain-rate} \quad 1 \cdot t[i]
\]

\(a[i]\) : the throughput rate achieved during the reception of chunk \(i\)

\(r[i]\) : the selected video bit rate of chunk \(i\)

\(t[i]\) : the reception duration of chunk \(i\)
A Model of Buffer Dynamics

Client buffer

Fill-rate
\[
\frac{a[i]}{r[i]} \cdot T_c \cdot \frac{r[i]}{a[i]}
\]

Drain-rate
\[
T_c \cdot \frac{r[i]}{a[i]}
\]

Desired buffer level

Compute the buffer level *every discrete chunk*.

- \(a[i]\): the throughput rate achieved during the reception of chunk \(i\)
- \(r[i]\): the selected video bit rate of chunk \(i\)
- \(T_c\): the chunk duration
A Model of Buffer Dynamics

Client buffer

- **Fill-rate**: $T_c$
- **Drain-rate**: $T_c \cdot \frac{r[i]}{a[i]}$

**Current buffer level**

$$b[i] = b[i-1] + T_c - T_c \cdot \frac{r[i]}{a[i]}$$

$\text{Desired buffer level}$

$b[i]$ : the buffer level (in seconds) when the client finishes the reception of chunk $i$
## Rate Selection

### Buffer dynamics

$$b_R[i] = b[i - 1] + T_c - T_c \cdot \frac{r_R[i]}{a[i]}$$

### Rate selection

$$\hat{R}[i] = \arg\min_{R \in \mathcal{R}} |b_R[i] - \beta_{ref}|$$

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$b[i]$</td>
<td>the buffer level (in seconds) when the client finishes the reception of chunk $i$</td>
</tr>
<tr>
<td>$T_c$</td>
<td>the chunk duration (each chunk containing a fixed duration of video)</td>
</tr>
<tr>
<td>$r_R[i]$</td>
<td>the selected video bit rate for chunk $i$ with the nominal bit rate $R$</td>
</tr>
<tr>
<td>$a[i]$</td>
<td>the throughput rate achieved for chunk $i$</td>
</tr>
<tr>
<td>$\hat{R}[i]$</td>
<td>the selected quality level (the nominal bit rate) of the video for chunk $i$</td>
</tr>
<tr>
<td>$\mathcal{R}$</td>
<td>the set of quality levels (nominal bit rates) of the video</td>
</tr>
<tr>
<td>$\beta_{ref}$</td>
<td>the desired buffer level (in seconds)</td>
</tr>
</tbody>
</table>
Outline

✓ Motivation

✓ Rate control for buffer stabilization
  ▪ Buffering delay
  ▪ Modeling buffer dynamics
  ▪ Rate selection

☐ Prototype implementation
  ▪ Server-based streaming architecture (Open-Loop Rate Control, OLAC)
  ▪ Transport protocol configuration

☐ Results
Open-Loop rAte Control (OLAC) [3]

- **Virtual client buffer** simulates client buffer on the server.
- A rate control on the server offers **immediate feedback** from clients.
- **Hybrid** throughput- and buffer-based adaptation balances efficiency and stability.

Transport Protocol Configurations

Our streaming prototype implementation is evaluated with two transport protocol configurations: standard TCP-Cubic and Predictably Reliable Real-time Transport (PRRT).

PRRT [4] provides

• error control under a specific delay constraint (*Predictable Reliability*),
• adaptive proactive and reactive error control (*capacity-approaching*),
• **opportunistic TCP-friendliness** by delay and equation-based congestion control,
• and **accurate throughput estimate** for applications.

Outline

✓ Motivation

✓ Rate control for buffer stabilization
  ▪ Buffering delay
  ▪ Modeling buffer dynamics
  ▪ Rate selection

✓ Prototype implementation
  ▪ Server-based streaming architecture (Open-Loop Rate Control, OLAC)
  ▪ Transport protocol configuration

☑ Results
Performance Comparison

Our benchmark rate controls are

- **DASH** VLC plugin [5] and
- Quality Adaptation Controller (QAC) for adaptive video streaming [6].

We deploy the buffer stabilizer within OLAC streaming architecture on top of

- TCP, referring to Dynamic Adaptive Streaming over TCP (DAST), and
- PRRT, referring to Dynamic Adaptive Streaming over PRRT (DASP).

Therefore, our performance comparison contains four sets of performance results: DASH, QAC, DAST (OLAC over TCP), and DASP (OLAC over PRRT).

Experimental Setup

- Wide area network
- Dynamic video bit rate 1-16 Mbps, chunk duration of 2s, 4s, 6s, and 8s
- Maximum client buffer size is set to the same size of chunk duration
- Entire streaming sessions lasts 180s, competing session appears from 60-120s
Experimental Setup

- Wide area network
- Dynamic video bit rate 1-16 Mbps, chunk duration of 2s, 4s, 6s, and 8s
- Maximum client buffer size is set to the same size of chunk duration
- Three concurrent streaming sessions simultaneously run for 120s

![Diagram showing experimental setup with server, Dummynet Emulator, and clients with bandwidth 16Mbps, RTT 40ms]
Experimental Results

- DASP had zero rebuffering events. DAST reduces the rebuffering ratio by at least 81% and 85%, compared to DASH and QAC, respectively.
- The average bit rate achieved with DAST is increased by 5-19% and 13-78% compared to DASH and QAC, respectively.
Experimental Results

- The average bit rates achieved with DAST and DASP are 17-26% and 27-54% higher compared to DASH and QAC, respectively.
- DASP had zero rebuffering events. DAST achieves with a 68-96% lower rebuffering ratio compared to QAC.
Conclusion

A solution for *low-latency* dynamic video streaming

- effectively stabilizes the buffer at a level **as short as a chunk-duration**,  
- significantly **improves user experience** in low-latency dynamic streaming.
Thank you for your attention!