Oboe: Auto-tuning Video ABR Algorithms to Network Conditions

Zahaib Akhtar*, Yun Seong Nam*, Ramesh Govindan, Sanjay Rao, Jessica Chen, Ethan Katz-Bassett, Bruno Ribeiro, Jibin Zhan, Hui Zhang

★: Co-primary authors
Internet Video Streaming Today

- Internet video is delivered over:
  - Heterogeneous networks: WiFi, wired, 3G/4G LTE
  - Highly varying or challenging network conditions
Internet Video Streaming Today

- Internet video is delivered over:
  - Heterogeneous networks: WiFi, wired, 3G/4G LTE
  - Highly varying or challenging network conditions

- Quality of experience (QoE) issues are common place

Low quality

Rebuffering
Internet Video Streaming Today

- Internet video is delivered over:
  - Heterogeneous networks: WiFi, wired, 3G/4G LTE
  - Highly varying or challenging network conditions

- Quality of experience (QoE) issues are common place

Low QoE adversely impacts user engagement and revenue
Background: Adaptive Bitrate Streaming

A video clip is encoded with multiple qualities (bitrates)
Background: Adaptive Bitrate Streaming

Each bitrate is split into chunks
Background: Adaptive Bitrate Streaming

Video Client

Network Conditions

Time

Video Server

Bitrates

Time

Request $\text{nth}$ chunk at bitrate $r$
Background: Adaptive Bitrate Streaming

Video Client

Network Conditions vs. Time

Video Server

Bitrates vs. Time

Request $n$th chunk at bitrate $r$
Background: Adaptive Bitrate Streaming

Video Client

Network Conditions

Time

Video Server

Request nth chunk at bitrate r

Bitrates

Time
Background: Adaptive Bitrate Streaming

Adaptive Bitrate Algorithms (ABR)
Background: Adaptive Bitrate Streaming

ABR algorithms

Too conservative

Low quality

Too aggressive

Rebuffering
Background: ABR algorithms

Performance of Designed Adaptation based ABRs critically depends on **configurable parameters**

- **ABR algorithms**
  - Data-Learned Adaptations (e.g., Pensieve\(^1\))
  - Designed Adaptations (e.g., MPC\(^2\), BOLA\(^3\), HYB\(^4\), BB\(^5\))

---

\[1\] Hongzi Mao, et al., SIGCOMM, 2017.
\[2\] Xiaoqi Yin, et al., SIGCOMM, 2015.
\[3\] Kevin Spiteri, et al., INFOCOM, 2016.
\[4\] An ABR algorithm that’s widely used in industry.
\[5\] Te-Yuan Huang, et al., SIGCOMM, 2014.
Parameters are sensitive to network conditions
Parameters are sensitive to network conditions

Network Condition A

Network Condition B
Parameters are sensitive to network conditions

**Network Condition A**

**Network Condition B**

Widely deployed ABR algorithm with parameter $\beta$
Parameters are sensitive to network conditions

![Diagram showing QoE for Condition A and Condition B with different values for β = X and β = Y.]
Parameters of ABRs must be set in a manner sensitive to network conditions.
The problem with ABR algorithms

<table>
<thead>
<tr>
<th>ABR algorithms</th>
<th>Parameter</th>
</tr>
</thead>
<tbody>
<tr>
<td>MPC</td>
<td>Discount factor $d$</td>
</tr>
<tr>
<td>BOLA</td>
<td>Parameter $\gamma$</td>
</tr>
<tr>
<td>HYB</td>
<td>Safety margin $\beta$</td>
</tr>
<tr>
<td>BB</td>
<td>Reservoir $r$</td>
</tr>
</tbody>
</table>

ABR algorithms use **fixed parameter value** or **simple heuristic**
The problem with ABR algorithms

<table>
<thead>
<tr>
<th>ABR algorithms</th>
<th>Parameter</th>
</tr>
</thead>
<tbody>
<tr>
<td>MPC</td>
<td>Discount factor $d$</td>
</tr>
<tr>
<td>BOLA</td>
<td>Parameter $\gamma$</td>
</tr>
<tr>
<td>HYB</td>
<td>Safety margin $\beta$</td>
</tr>
<tr>
<td>BB</td>
<td>Reservoir $r$</td>
</tr>
</tbody>
</table>

ABR algorithms use **fixed parameter value** or **simple heuristic**

Do not perform well across all network conditions
Goal of our work

Design a system to make ABR algorithms work better over a wide range of network conditions
Key Challenges

How to model network conditions?

How to find the best parameter for a given condition?

How to adapt to changes in network conditions?
Contributions

How to model network conditions?
- Leverage *stationarity* of network connections

How to find the best parameter for a given condition?
- **Pre-compute** offline

How to adapt to changes in network conditions?
- **Detect change points online** and adjust parameters
Contributions

How to model network conditions?

- Leveraging stationarity of network connections
- Pre-computing in offline
- Online change point detection and adjusting ABR algorithms in online

Our system, Oboe improves state-of-art ABRs (MPC, BOLA and HYB) upto 38% and outperforms Pensieve by 24%
Key Challenges

How to **model network conditions**?

How to find the best parameter for a given condition?
- Oboe Offline Stage

How to adapt to changes in network conditions?
- Oboe Online Stage
TCP connection throughput can be modeled as a piecewise stationary sequence of network states.
Modeling network conditions

Network state $\mathbf{s} = <\mu_s, \sigma_s>$

where $\mu_s$ is the mean and $\sigma_s$ is the standard deviation of throughput.
Modeling network conditions

Network state \( s = \langle \mu_s, \sigma_s \rangle \)

where \( \mu_s \) is the mean and \( \sigma_s \) is the standard deviation of throughput

Key idea: Use the best parameter for each network state
Key Challenges

How to model network conditions?

How to find the best parameter for each network state?
  ● Oboe Offline Stage

How to adapt to changes in network state?
  ● Oboe Online Stage
Finding the best parameter: Oboe Offline Step 1

Generate synthetic stationary traces for each network state.
Finding the best parameter: Oboe Offline Step 2

Explore parameter space for each state and get QoE vectors
Finding the best parameter: Oboe Offline Step 2

Explore parameter space for each state and get QoE vectors
Finding the best parameter: Oboe Offline Step 2

Explore parameter space for each state and get QoE vectors
Finding the best parameter: Oboe Offline Step 3

Find the best parameter by vector dominance for each state
Oboe Offline Stage: Design Questions

- Use real or synthetic traces?
- How to quantize network state space?
- How to reduce the cost of parameter space exploration?
- How to decouple ABR algorithms from Virtual Player?
- How to take publisher preferences into account?
Key Challenges

How to model network conditions?

How to find the best parameter for each network state?
  ● Oboe Offline Stage

How to adapt to changes in network state?
  ● Oboe Online Stage
Adapting to network state changes

Online change point detection\cite{11} algorithm identifies network throughput distribution changes in real time

Adapting to network state changes: Online Step 1

Online change point detection\textsuperscript{[11]} algorithm identifies network throughput distribution changes in real time.

Adapting to network state changes: Online Step 2

Find the best parameter from the mapping for a new state.
Adapting to network state changes: Online Step 3

Reconfigure ABR algorithm parameter
Oboe Online Design Questions

- Why not a simple moving average?
- How many throughput samples to detect changes?
- Are computational overheads acceptable in real time?
Evaluation methodology

Comparison

- Existing ABRs vs. Existing ABRs + Oboe

QoE Metrics

- Average Bitrate
- Rebuffering Ratio
- Bitrate change magnitude
- QoE-lin (Linear combination of three metrics)
Evaluation methodology

TestBed Setup

<table>
<thead>
<tr>
<th>Chrome Browser Video Client</th>
<th>Apache Video Server</th>
</tr>
</thead>
<tbody>
<tr>
<td>Video player</td>
<td>Video</td>
</tr>
<tr>
<td>- Dash.js</td>
<td>- 3 min long</td>
</tr>
<tr>
<td>- Chrome DevTool API</td>
<td>- Encoded with 6 bitrates</td>
</tr>
</tbody>
</table>

Dataset

- 600 throughput traces from real users
- Real users used a desktop or a mobile
- Upto 6 Mbps
## Various Evaluations

<table>
<thead>
<tr>
<th>Method</th>
<th>Detail</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABRs performance</td>
<td>MPC, BOLA, BB, HYB and Pensieve</td>
</tr>
<tr>
<td>Public datasets</td>
<td>HSDPA(^{[11]}) and FCC(^{[12]})</td>
</tr>
<tr>
<td>Various settings</td>
<td>Live setting and different videos</td>
</tr>
<tr>
<td>Alternative predictors</td>
<td>Ideal predictors on MPC</td>
</tr>
<tr>
<td>Publisher specification</td>
<td>Different rebuffering tolerance</td>
</tr>
<tr>
<td>Pilot Deployment</td>
<td>Partial deployment on AWS</td>
</tr>
</tbody>
</table>

\(^{[11]}\) Haakon Riiser, et. al. MMSys, 2013  
\(^{[12]}\) Federal Communications Commission. Raw Data 2016
Solves an optimization problem to choose bitrates using predicted throughputs and player buffer occupancy

**Discount factor $d$**

- Reduces a predicted throughput by $d$ to compensate prediction errors
- Simple heuristic based on previous prediction errors
RobustMPC vs MPC+Oboe

- Improves QoE-lin for 71% of sessions
- For 19% of the sessions, more than 20% benefit
RobustMPC vs MPC+Oboe

- Improves QoE-lin for 71% of sessions
- For 19% of the sessions, more than 20% benefit
RobustMPC vs MPC+Oboe

- Improves QoE-lin for 71% of sessions
- For 19% of the sessions, more than 20% benefit
Where does benefit come from over RobustMPC

Similar average bitrates

Reduces the # of sessions with rebuffering from 33% to 5%

Improves the median per chunk change magnitude by 38%
Where does benefit come from over RobustMPC

Similar average bitrates

Reduces the # of sessions with rebuffering from 33% to 5%

Improves the median per chunk change magnitude by 38%
MPC+Oboe vs Pensieve

- Improves QoE-lin for 80% of sessions
- 24% better in average QoE-lin
Where does benefit come from over Pensieve?

- **Hypothesis**: Pensieve performs better when it is trained with a constrained throughput range

<table>
<thead>
<tr>
<th>Model</th>
<th>Trained</th>
<th>Tested</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pen-Specialized</td>
<td>0 - 3 Mbps</td>
<td>0 - 3 Mbps</td>
<td>Better</td>
</tr>
<tr>
<td>Original Pensieve</td>
<td>0 - 6 Mbps</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Pensieve unable to specialize to network conditions
- **Oboe** specializes parameters for every network state
Summary

Oboe is a system to make ABR algorithms work better in a wide range of network conditions
- by auto-tuning parameters to current network state

Oboe is general
- Can be applied to many existing ABRs
- Improves existing ABRs upto 38% in QoE metrics
- Outperforms Pensieve by 24% in average QoE
Live demo tomorrow in the demo session!

https://github.com/USC-NSL/Oboe

Thanks!