STALLION: Video Adaptation Algorithm for Low-Latency Video Streaming

Craig Gutterman, Brayn Fridman, Trey Gilliland,
Yusheng Hu, Gil Zussman
Electrical Engineering, Columbia University

COLUMBIA UNIVERSITY
IN THE CITY OF NEW YORK

wim.net
Wireless & Mobile Networking Lab

Grand Challenge on

Adaptation Algorithms for Near-Second Latency

Organized and sponsored by

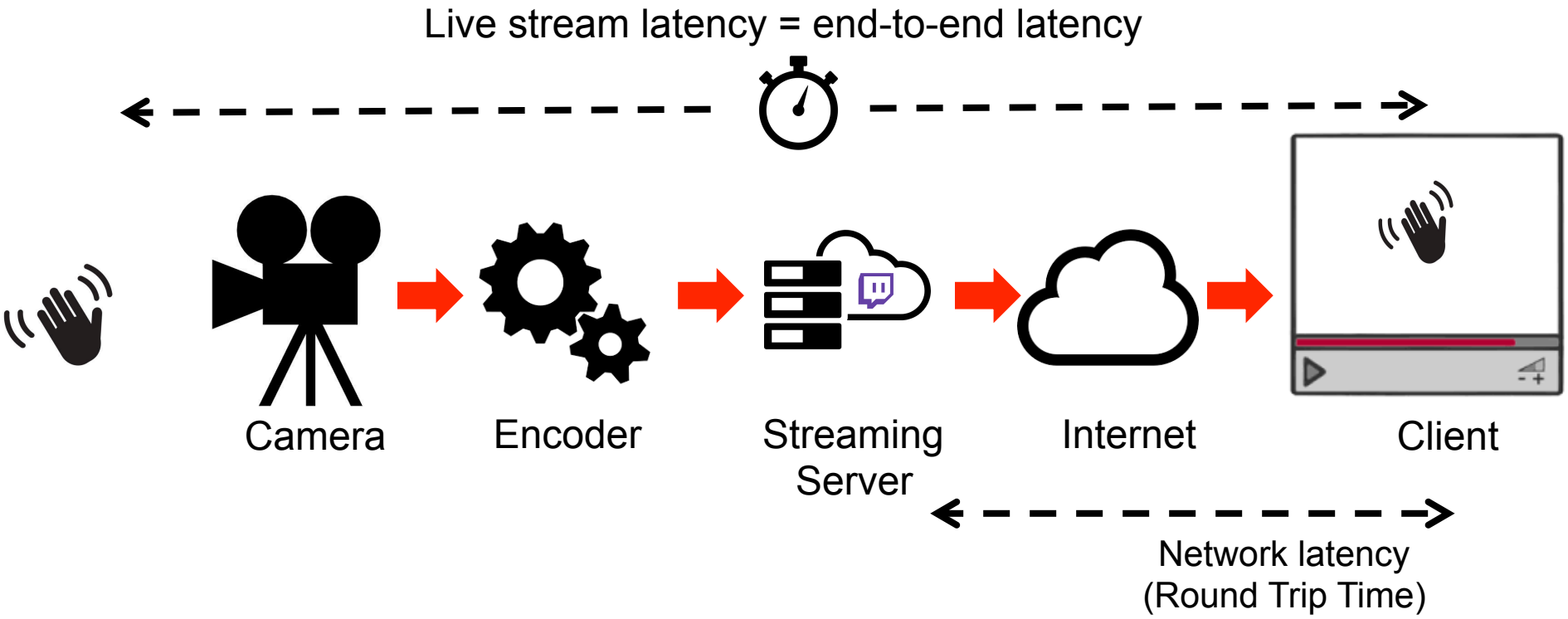
Challenge Description

At Twitch, we have been successful in delivering ultra-low-latency streams to millions of viewers. However, as we drive towards near-second latency, we are finding that existing adaptation algorithms are not able to keep up - smaller player buffers do not provide enough time to respond to changing network conditions. We see this as a key challenge blocking the streaming community from reducing latency at scale.

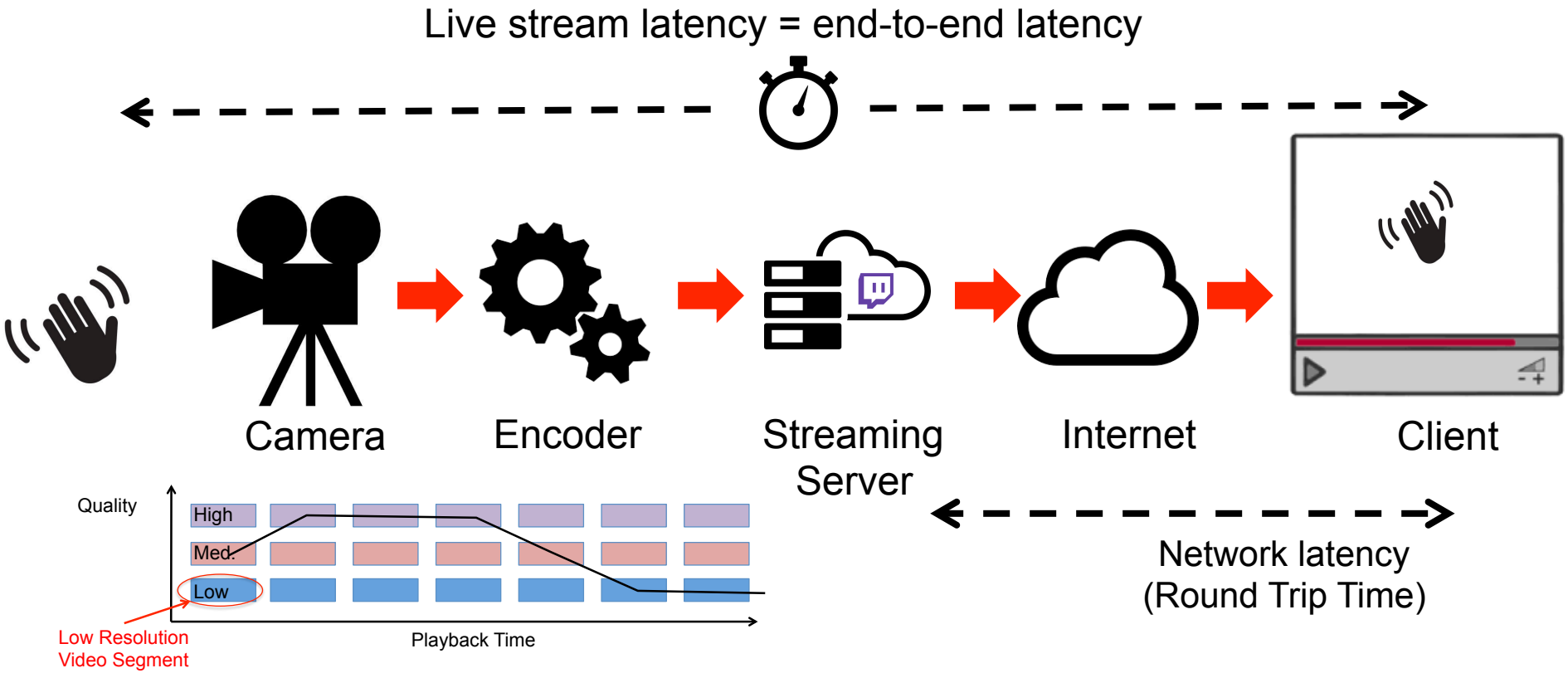
Grand Challenge

- Goal: To build and test a low latency ABR algorithm
- Given
  - Fork of Dash.js modified to pre-request low latency segments
  - Low latency DASH server
- Evaluation
  - 5 network patterns to simulate network conditions
  - Quality of Experience
    - Bitrate
    - Bitrate switches
    - Rebuffering time
    - Live latency
    - Playback speed

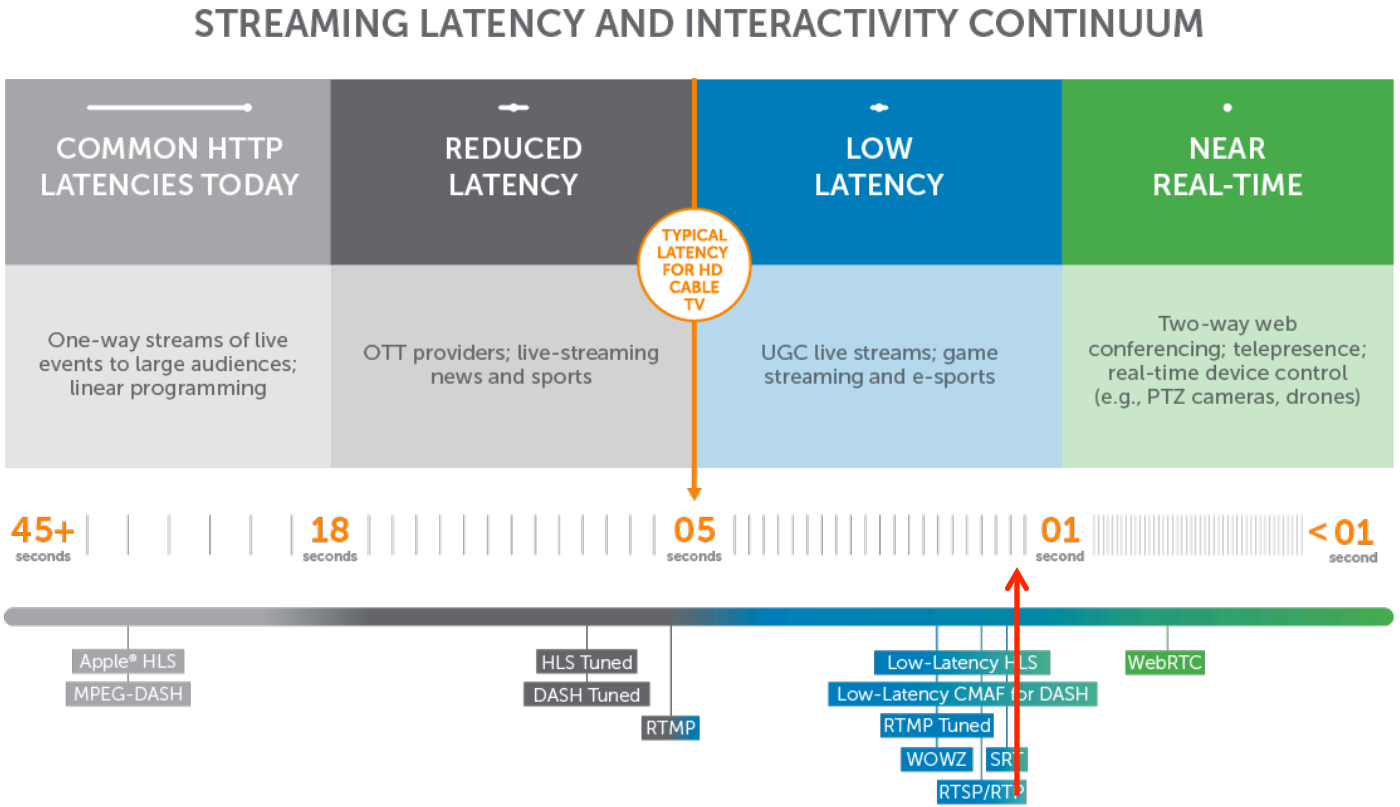
Background


Live stream latency = end-to-end latency
Camera
Encoder
Streaming Server
Internet
Client
Network latency
(Round Trip Time)

Background


Live stream latency = end-to-end latency
Camera
Encoder
Streaming Server
Internet
Client
Quality
High
Med.
Low
Low Resolution Video Segment
Playback Time
Network latency (Round Trip Time)

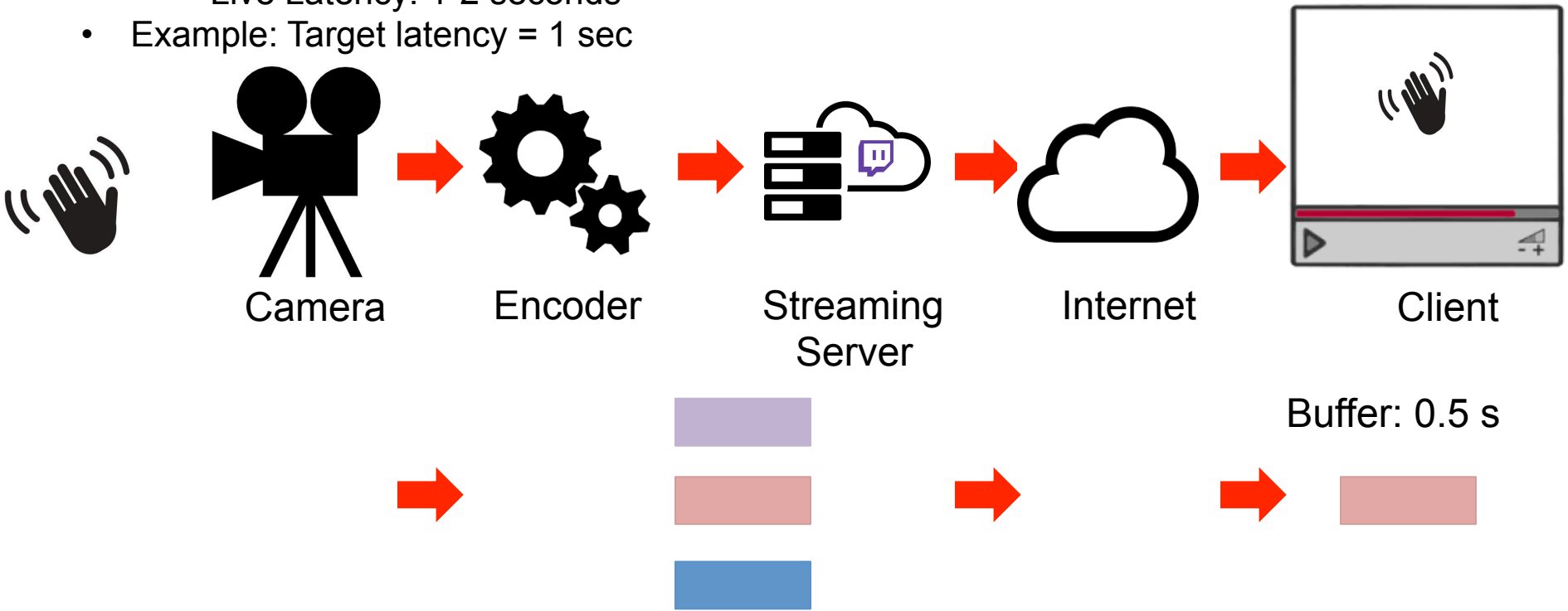
Live Stream Latency



Graphic Credit: Wowza

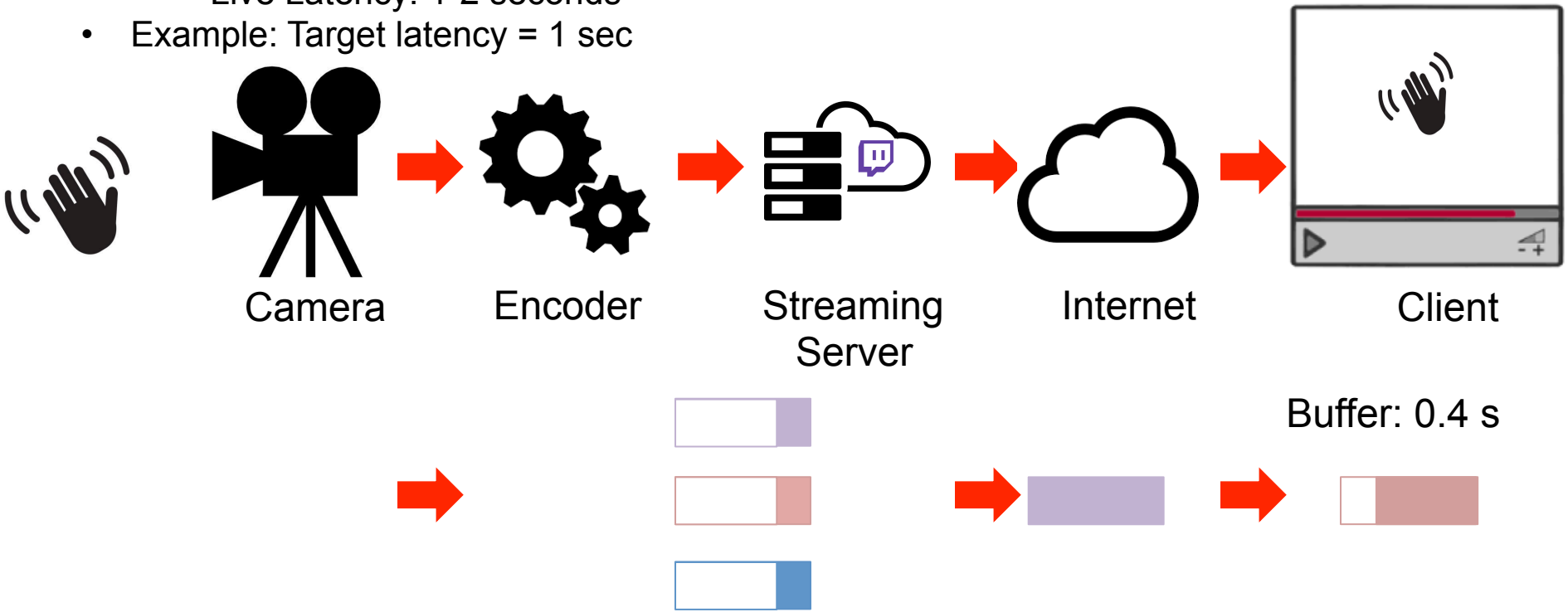
Target Latency

- Challenge Requirements
  - 0.5 second segments
  - Live Latency: 1-2 seconds
- Example: Target latency = 1 sec



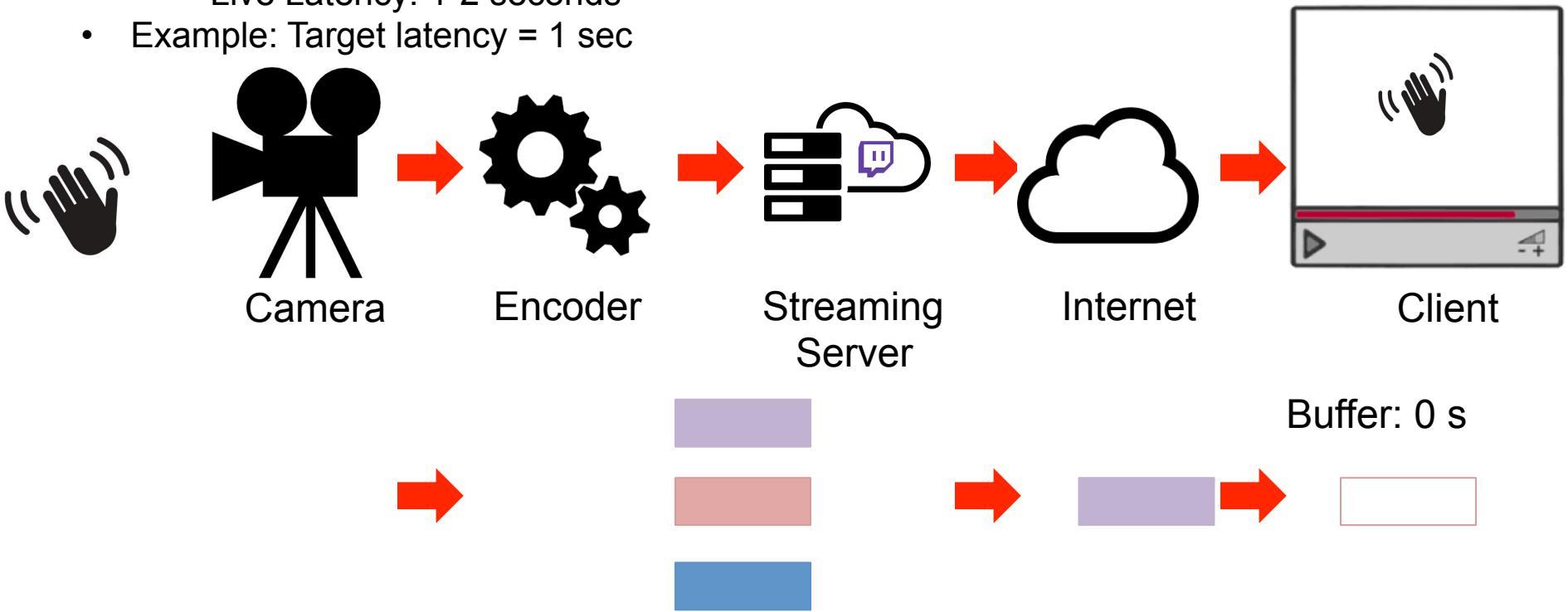
Target Latency

- Challenge Requirements
  - 0.5 second segments
  - Live Latency: 1-2 seconds
- Example: Target latency = 1 sec



Target Latency

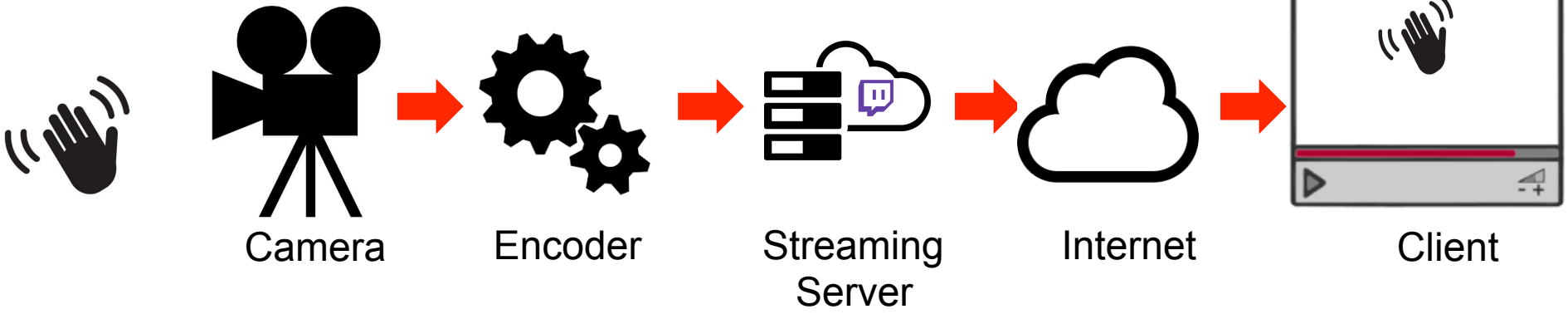
- Challenge Requirements
  - 0.5 second segments
  - Live Latency: 1-2 seconds
- Example: Target latency = 1 sec

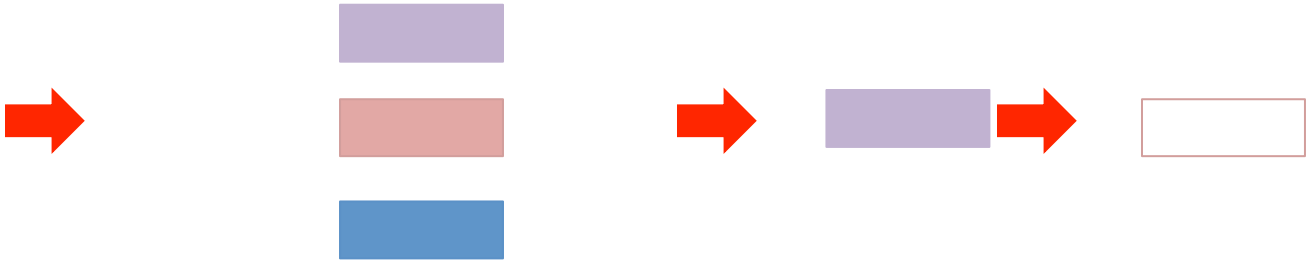


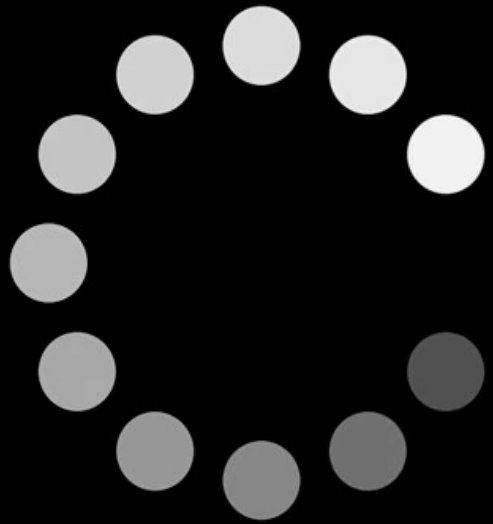
Target Latency

- Challenge Requirements
- Example: Target latency = 1 sec

Run out of video in the buffer!







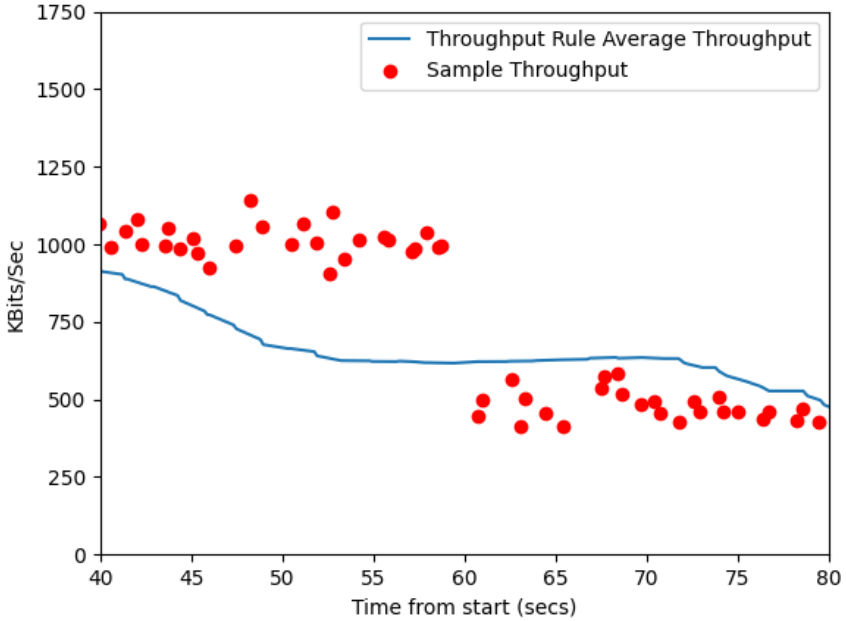
ABR Algorithms

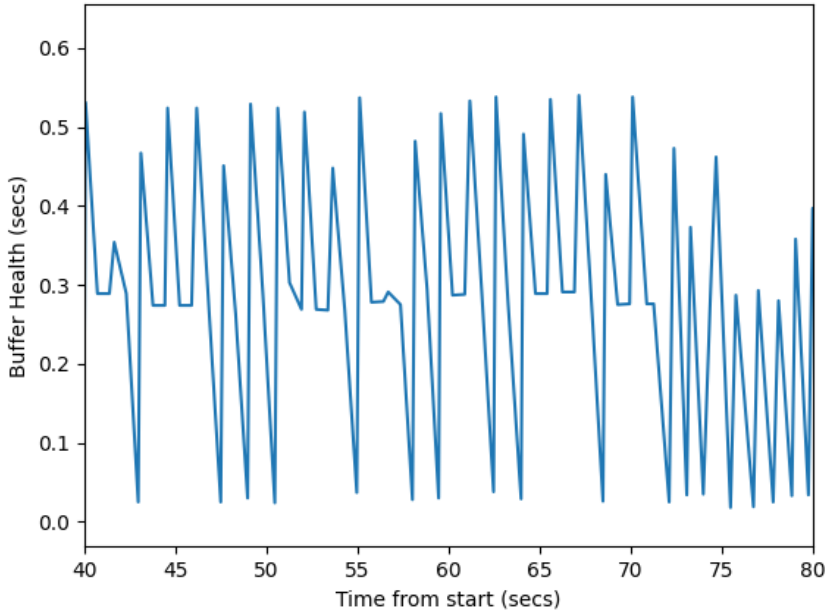
- Throughput-based algorithms
  - Estimate network throughput available between the client and the server
  - Averaging of segment burst times
    - Sliding window
    - Expected weighted moving average (EWMA)
  - Examples: Festive, PANDA, and Squad
- Buffer-based algorithms
  - Use buffer level to decide on the bitrate of the next segment
  - Examples BBA, BOLA
- Hybrid algorithms
  - Use both through prediction and buffer level to decide on the bitrate of the next segment
  - Examples ELASTIC, MPC, DYNAMIC
  - DYNAMIC is the default ABR algorithm provided by the standard reference player dash.js

[ABD11] [JSZ12] [CCP13] [LZG14][YJS15] [HJM15] [XYJ16][SSS18]

Throughput Measurements

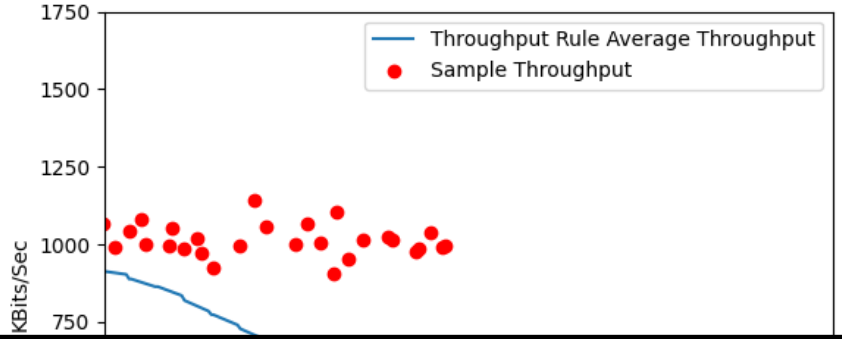
- Average Throughput determined by DYNAMIC's Throughput Rule
- $$segment\ throughput = \frac{size_{segment}}{end_{segment} - start_{segment}}$$

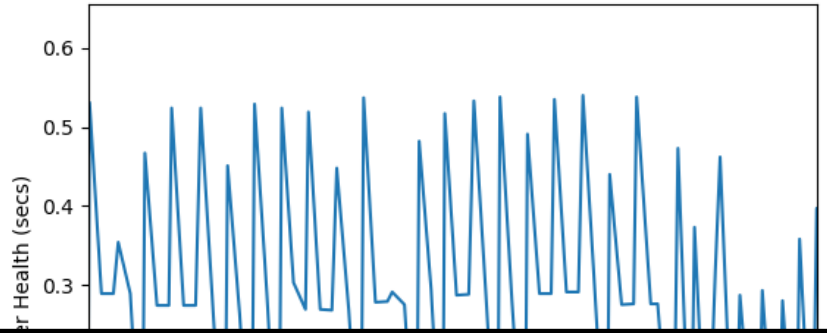




Throughput Measurements

- Average Throughput determined by DYNAMIC's Throughput Rule
- $$segment\ throughput = \frac{size_{segment}}{end_{segment} - start_{segment}}$$





System does not react fast enough to network variations, causing stalls and variable resolutions

STALLION ABR

- Unstable network conditions result in client bandwidth fluctuations during playback
- Stallion uses a sliding window to measure the mean and standard deviation of both the bandwidth and latency.

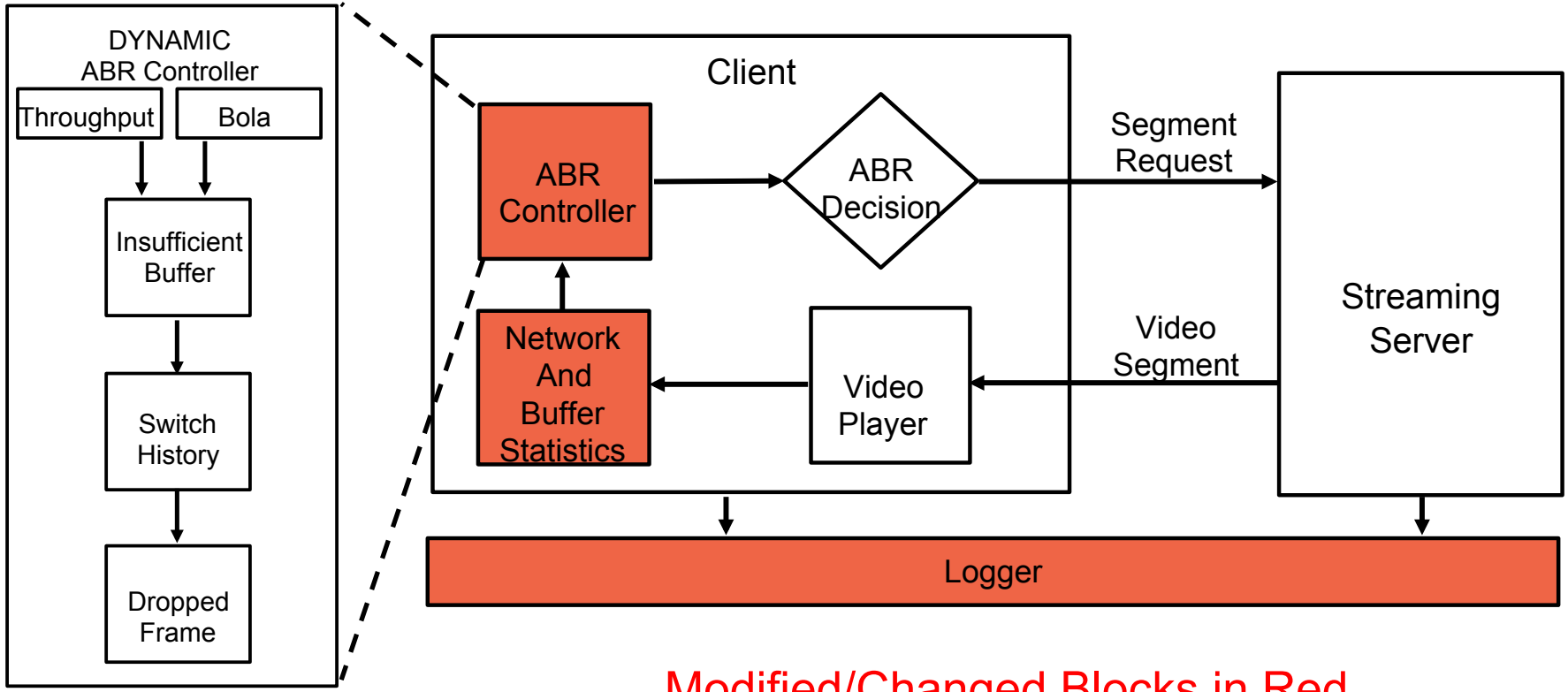
Notation	Meaning
C	Throughput
l	Latency
R	Segment bitrate
z	Safety factor
σ	Sample standard deviation
d	Segment Length

$$C_{safe} = \hat{C} - z_C \sigma_C$$

$$l_{safe} = \hat{l} + z_l \sigma_l$$

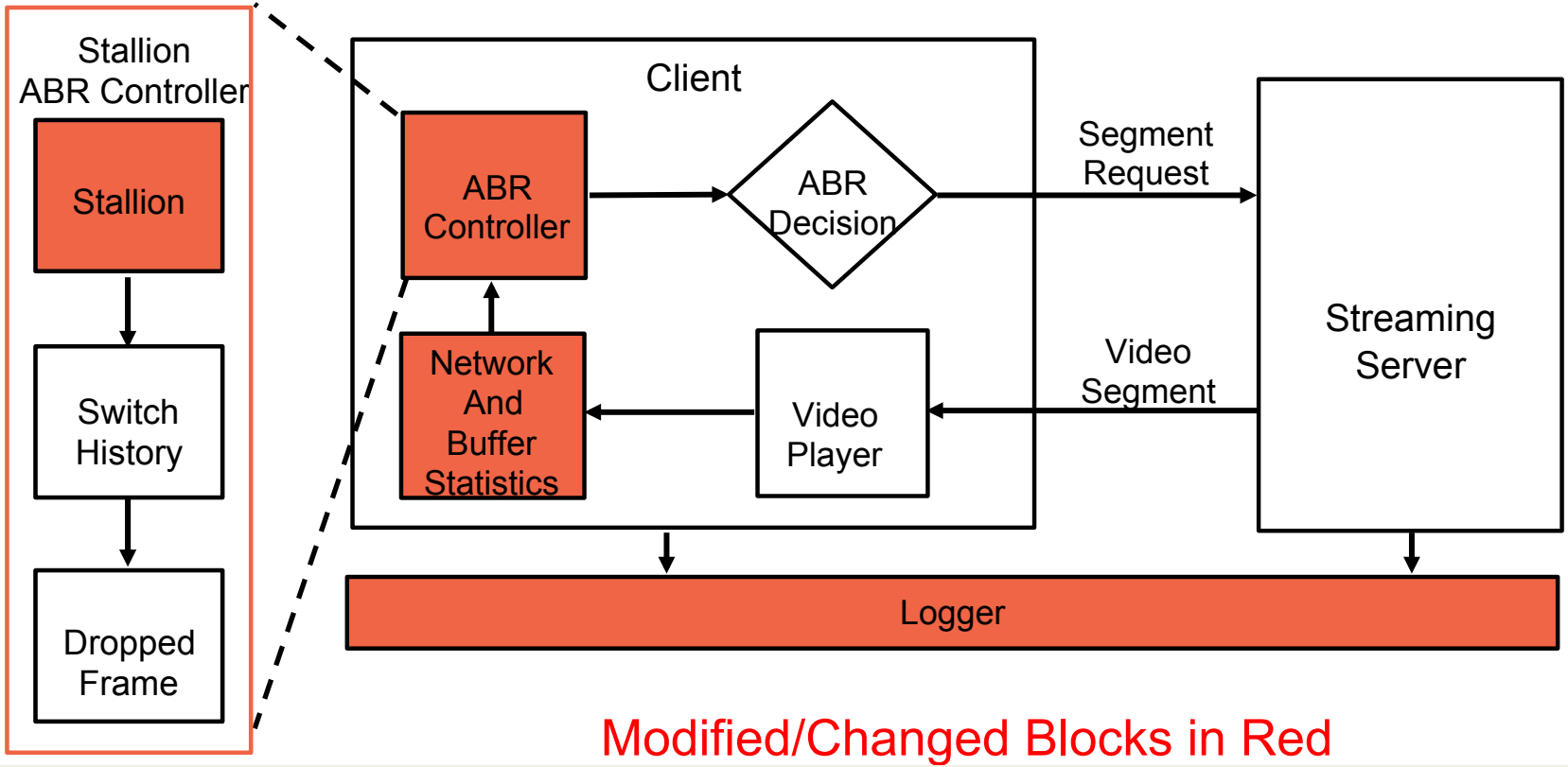
$$R_{max} = C_{safe} * (1 - \frac{l_{safe}}{d})$$

Implementation



Modified/Changed Blocks in Red

Stallion Implementation

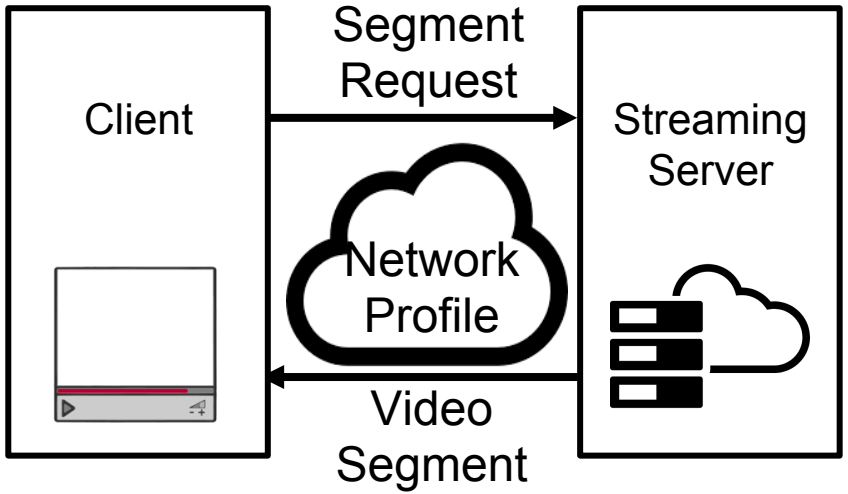


Modified/Changed Blocks in Red

- https://github.com/Wimnet/twitch_challenge

Performance Evaluation

- Network profiles used to emulate varying network conditions



Notation	Time Duration (s)	Data Rate (Kbps)
Cascade	30,30,30,30,30	1200,800,400, 800,1200
Intra Cascade	15,15,15,15,15,15,15 ,15,15	1000,800,600,400 ,200,400,600,800, 1000
Spike	10,10,10	1200,300,500
Slow Jitters	5,5,5,5,5,5	500,1200,500, 1200,500,1200
Fast Jitters	0.25,5,0.1,1,0.25,5	500,1200,500, 1200,500,1200

Performance Evaluation

- Server and dash player run on Apple Macbook Pro
- Video (Big Buck Bunny) encoded in 3 bitrates (200 Kbps, 600 Kbps, 1000 Kbps)
- Each ABR algorithm was run 20 times on each network profile
- Compare vs. DYNAMIC, the default ABR algorithm provided by the standard reference player dash.js

Quality of Experience

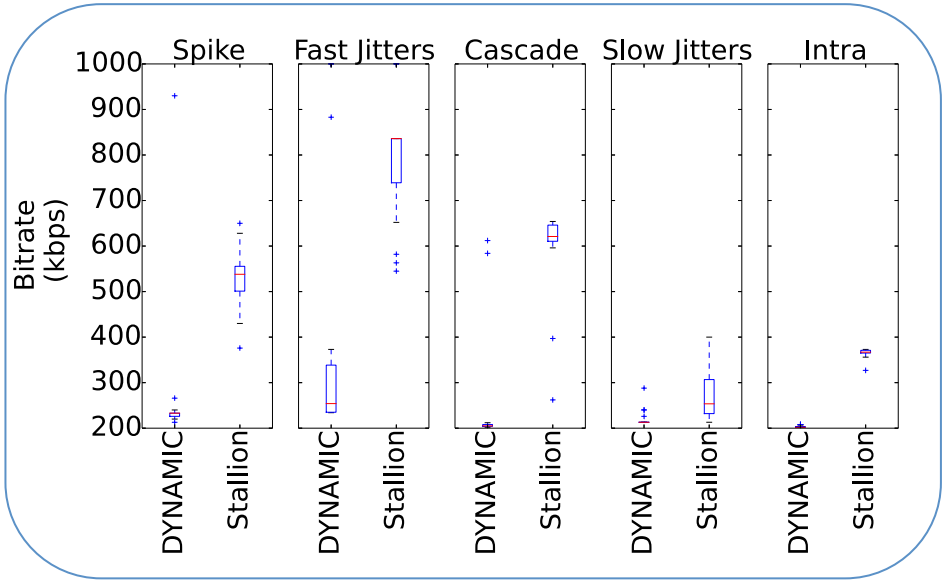
- Selected Bitrate
- Stall duration
- Live latency
- Playback Speed
- Number of bitrate switches
- QoE metric provided by the Challenge:

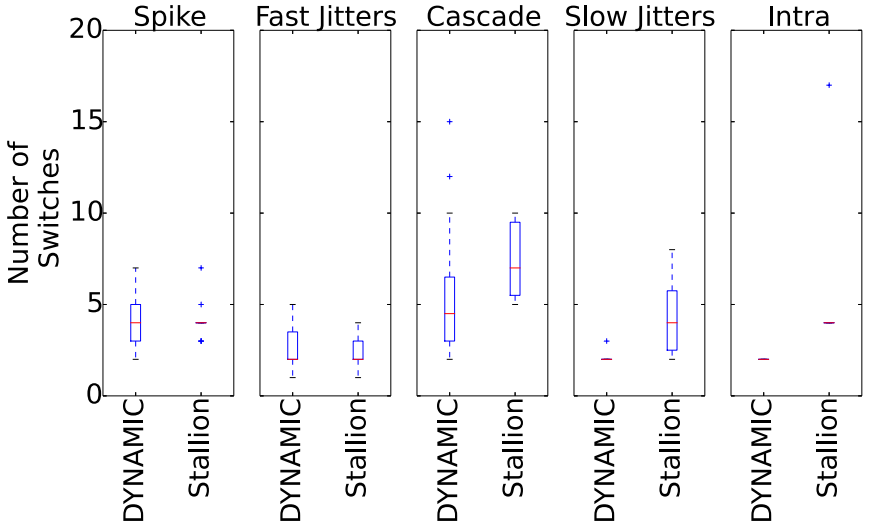
$$QoE = \sum_{s=1}^{S} (\alpha R_s - \beta E_s - \gamma L_s - \sigma |1 - P_s|) - \sum_{s=1}^{S-1} \mu |R_{s+1} - R_s|$$

- https://github.com/twitchtv/acm-mmsys-2020-grand-challenge/blob/master/NQoE.pdf

Results

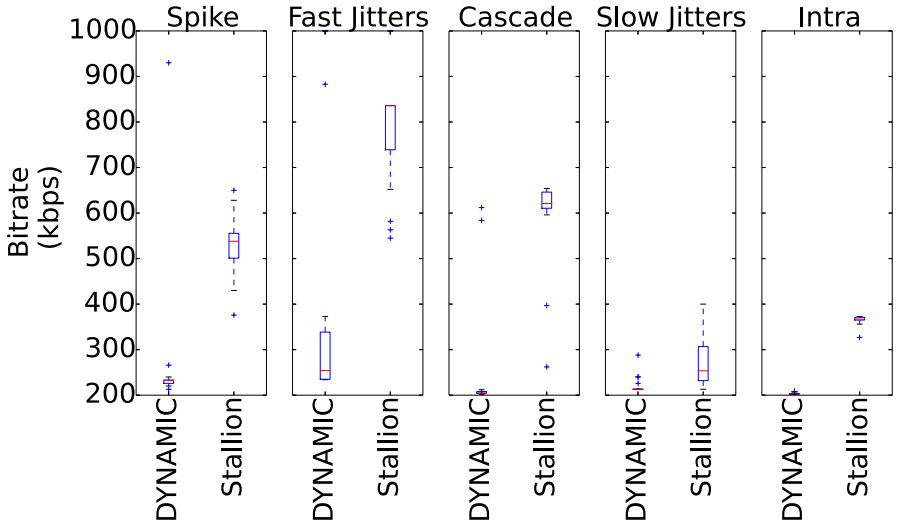
$$QoE = \sum_{s=1}^{S} (\alpha R_s - \beta E_s - \gamma L_s - \sigma |1 - P_s|) - \sum_{s=1}^{S-1} \mu |R_{s+1} - R_s|$$

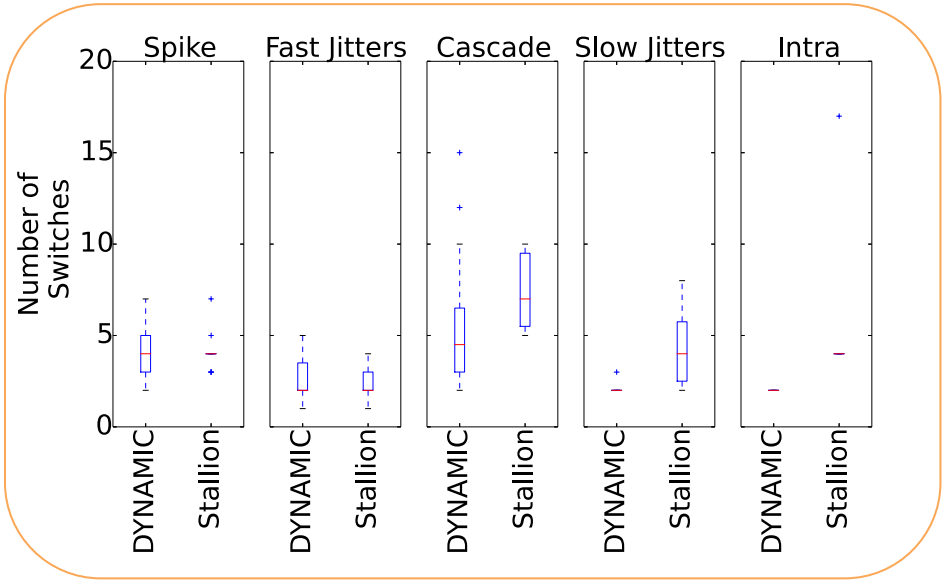




Results

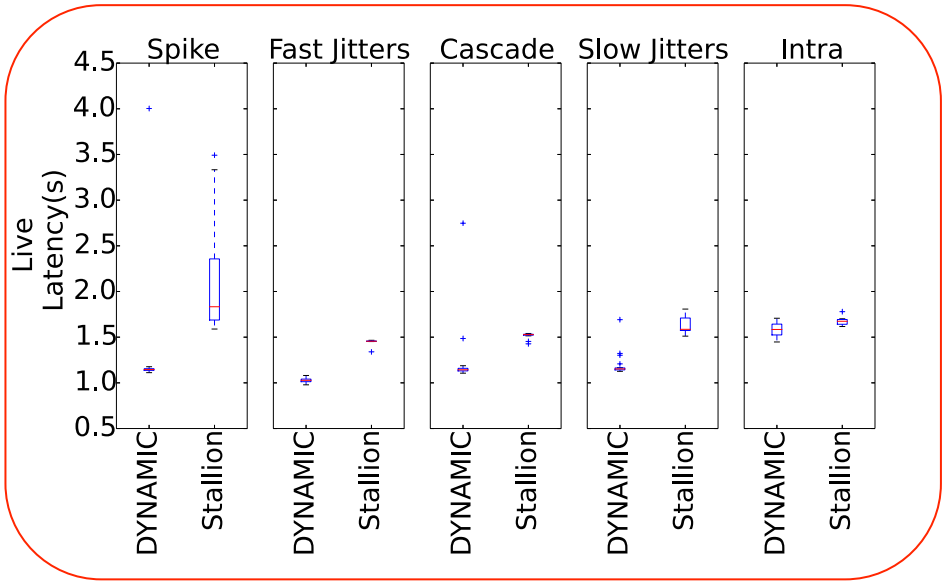
$$QoE = \sum_{s=1}^{S} (\alpha R_s - \beta E_s - \gamma L_s - \sigma |1 - P_s|) - \sum_{s=1}^{S-1} \mu |R_{s+1} - R_s|$$

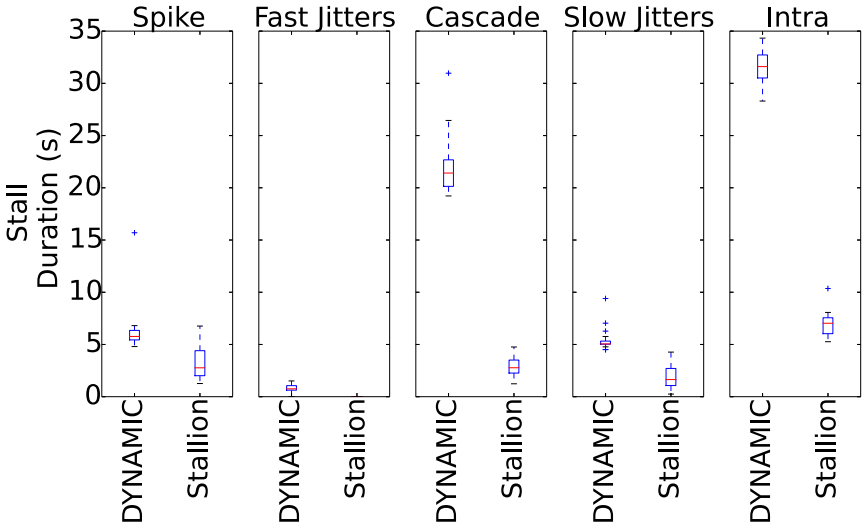




Results

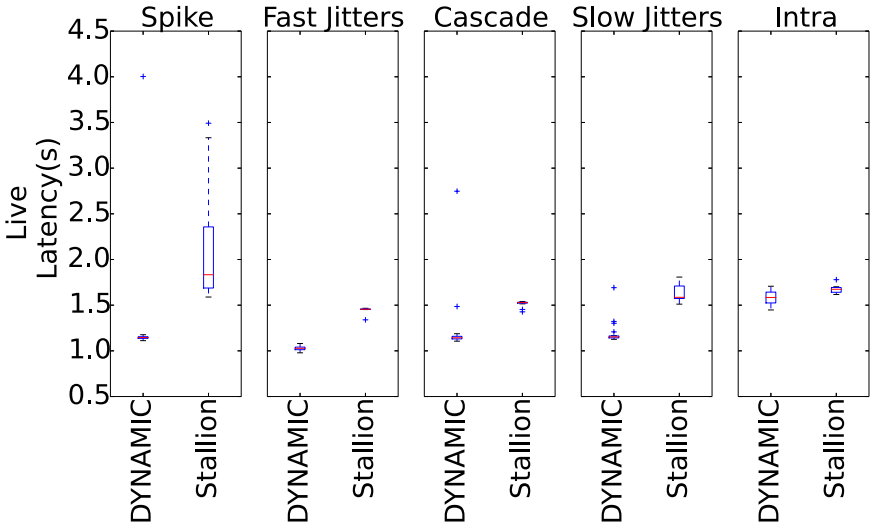
$$QoE = \sum_{s=1}^{S} (\alpha R_s - \beta E_s - \gamma L_s - \sigma |1 - P_s|) - \sum_{s=1}^{S-1} \mu |R_{s+1} - R_s|$$

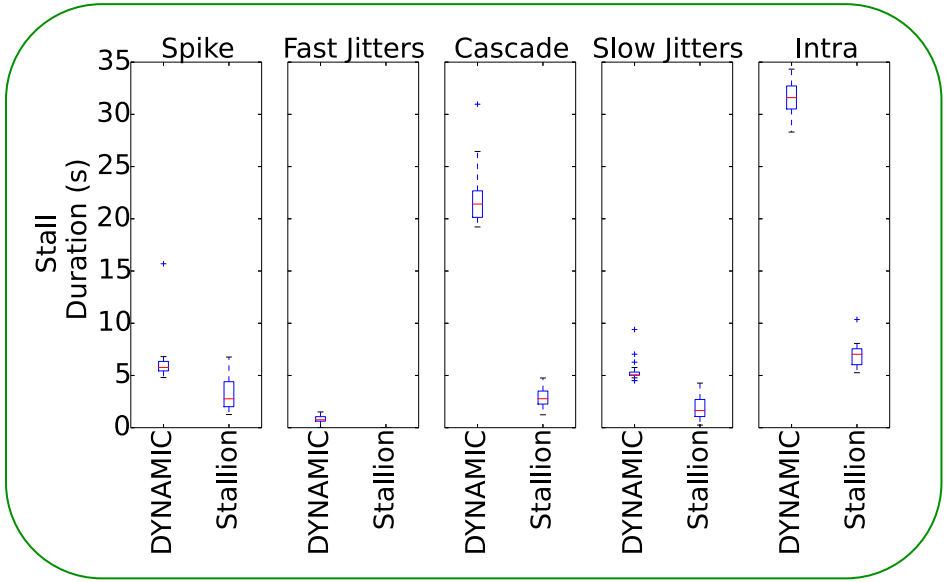




Results

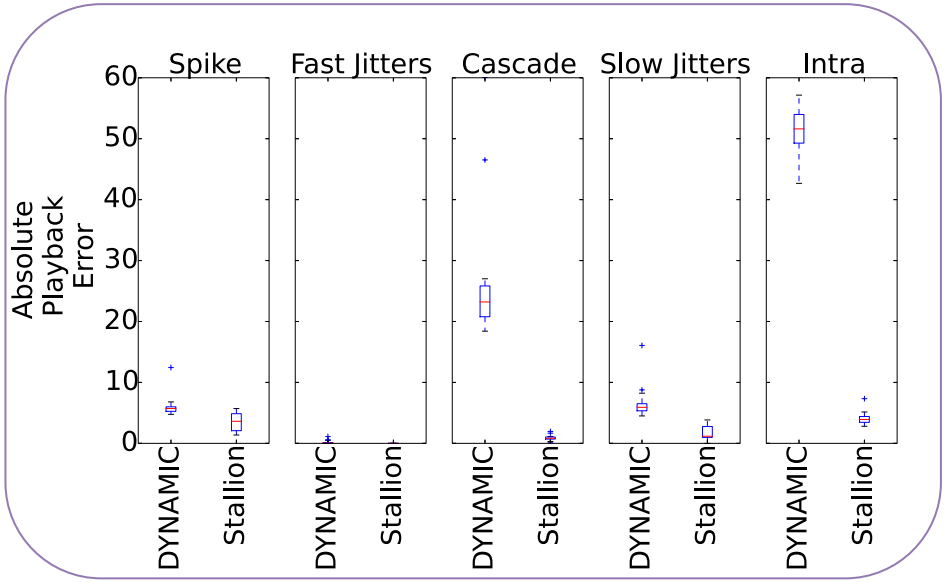
$$QoE = \sum_{s=1}^{S} (\alpha R_s - \beta E_s - \gamma L_s - \sigma |1 - P_s|) - \sum_{s=1}^{S-1} \mu |R_{s+1} - R_s|$$

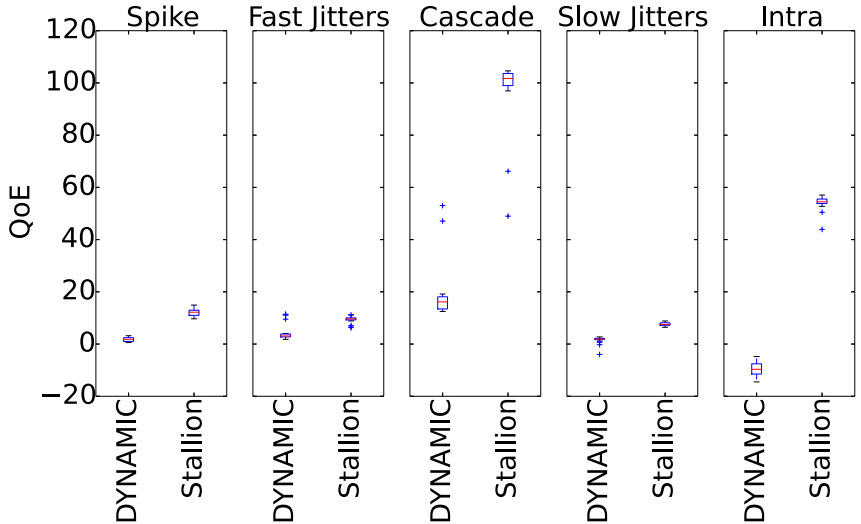




Results

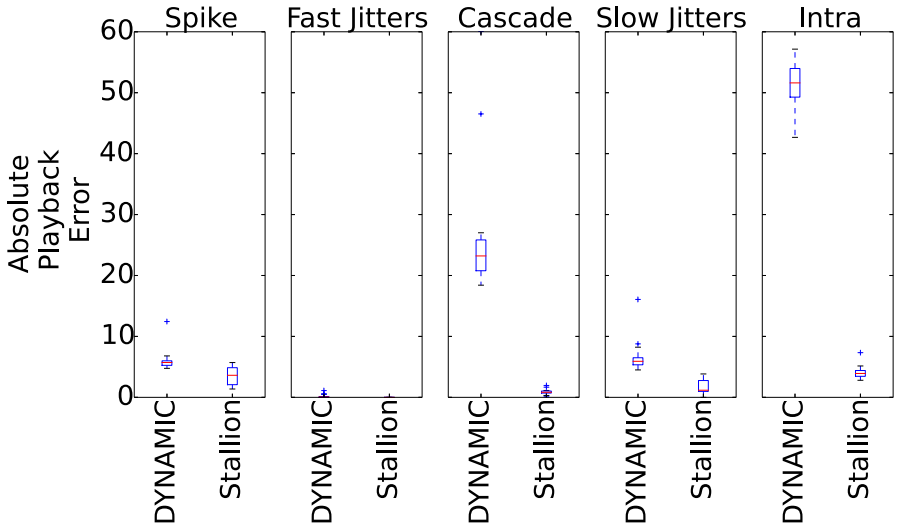
$$QoE = \sum_{s=1}^{S} (\alpha R_s - \beta E_s - \gamma L_s - \sigma |1 - P_s|) - \sum_{s=1}^{S-1} \mu |R_{s+1} - R_s|$$

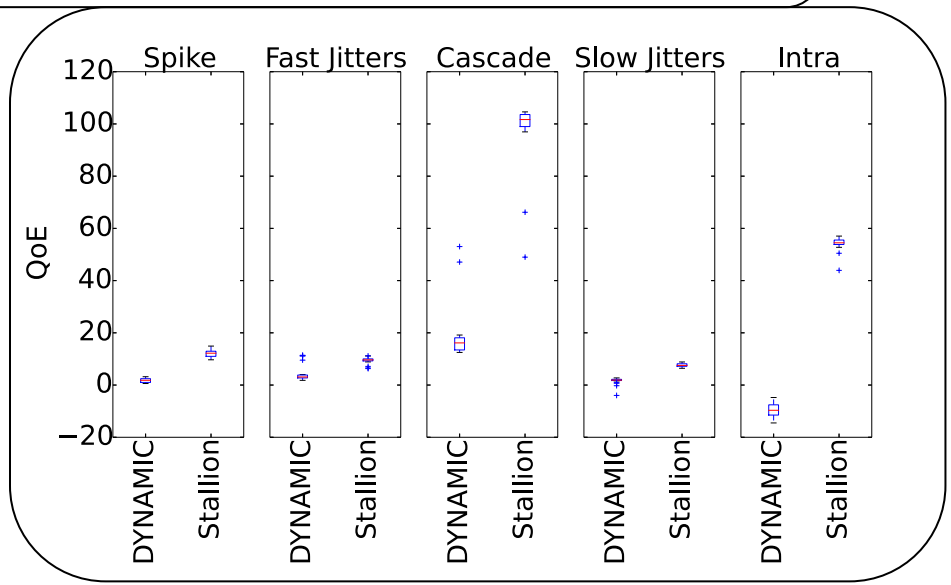




Results

$$QoE = \sum_{s=1}^{S} (\alpha R_s - \beta E_s - \gamma L_s - \sigma |1 - P_s|) - \sum_{s=1}^{S-1} \mu |R_{s+1} - R_s|$$





Summary

- Challenge Goal: To develop and test a low latency ABR algorithm
- Developed Stallion which relies on using the standard deviation of measured throughput to give a safe request for the segment bitrate
- Stallion shows 1.8x increase in bitrate, and 4.3x reduction in the number of stalls compared to DYNAMIC
- Beyond the challenge
  - Additional network profiles
  - Improve the safe throughput estimate
  - Add a machine learning system to auto tune Stallion parameters

Thank You!

STALLION: Video Adaptation Algorithm for Low-Latency Video Streaming

Craig Gutterman, Brayn Fridman, Trey Gilliland, Yusheng Hu, Gil Zussman
Columbia University

clg2168@columbia.edu

COLUMBIA UNIVERSITY
IN THE CITY OF NEW YORK

wim.net
Wireless & Mobile Networking Lab