Exploring “Downward Spiral” Effect for Video Streaming Rate Selection.

“Confused, Timid, and Unstable: Picking a Video Streaming Rate is Hard”,
Te-Yuan Huang Nikhil Handigol Brandon Heller Nick McKeown Ramesh Johari

Harsh Trivedi, Malvika Modi, Harshvardhan Agarwal

Why care about rate selection (Video Quality)

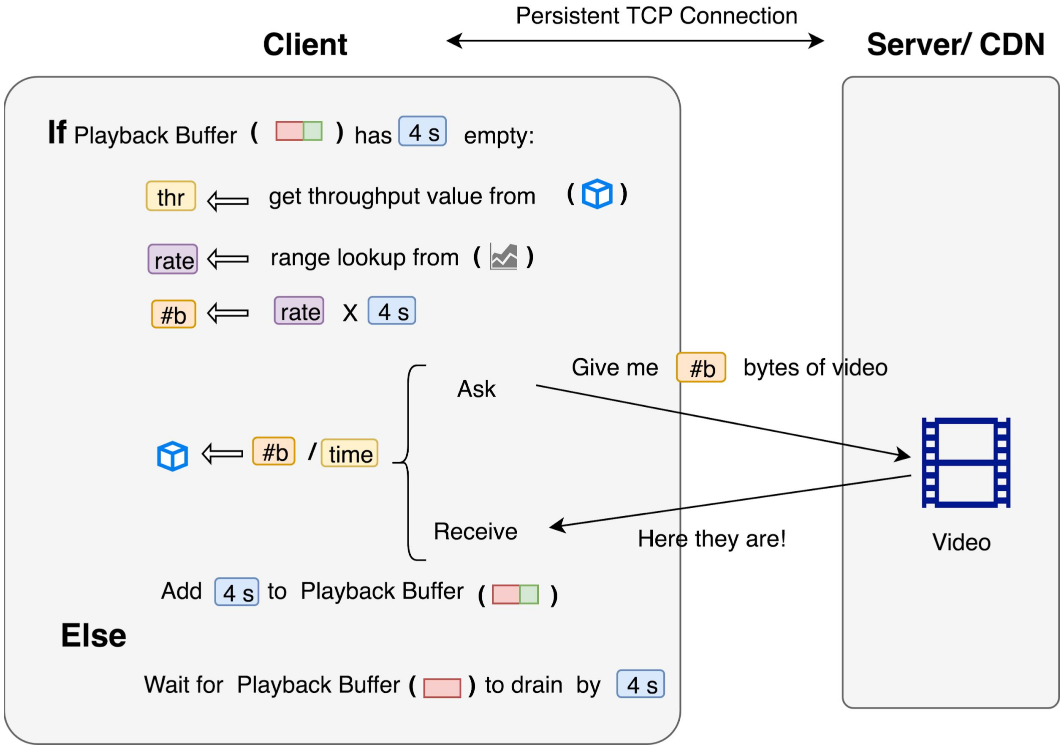
Video Streaming happens in presence of **limited playback buffer**.

- If selected playback rate is too high (High Quality Video):
  - Recurrent rebuffering events. Viewer has to wait!
- If selected playback rate is too low:
  - Viewer sees a low quality video

⇒ Hence, for good viewer experience we want good playback rate selection .

⇒ But doing that for with persistent HTTP is difficult!

Typical (our) HTTP Client for video streaming



Persistent TCP Connection

Client

Server/ CDN

If Playback Buffer (🟥🟩) has 4 s empty:

thr ⟸ get throughput value from (📦)

rate ⟸ range lookup from (📈)

#b ⟸ rate X 4 s

Ask — Give me #b bytes of video

📦 ⟸ #b / time

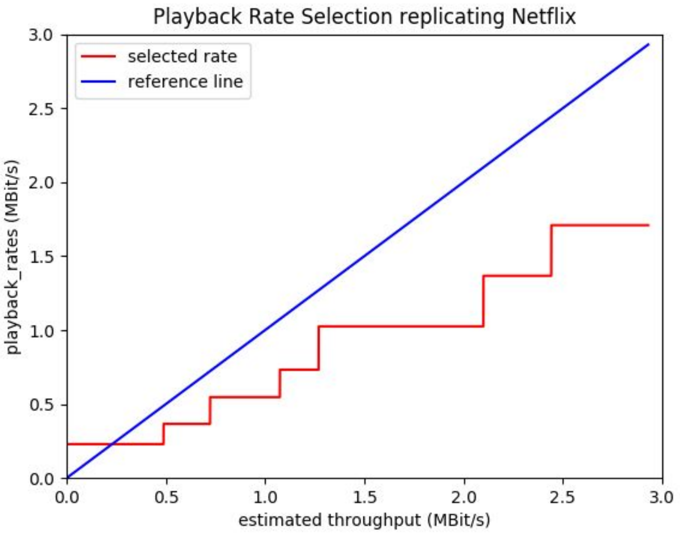
Receive — Here they are!

Video

Add 4 s to Playback Buffer (🟥🟩)

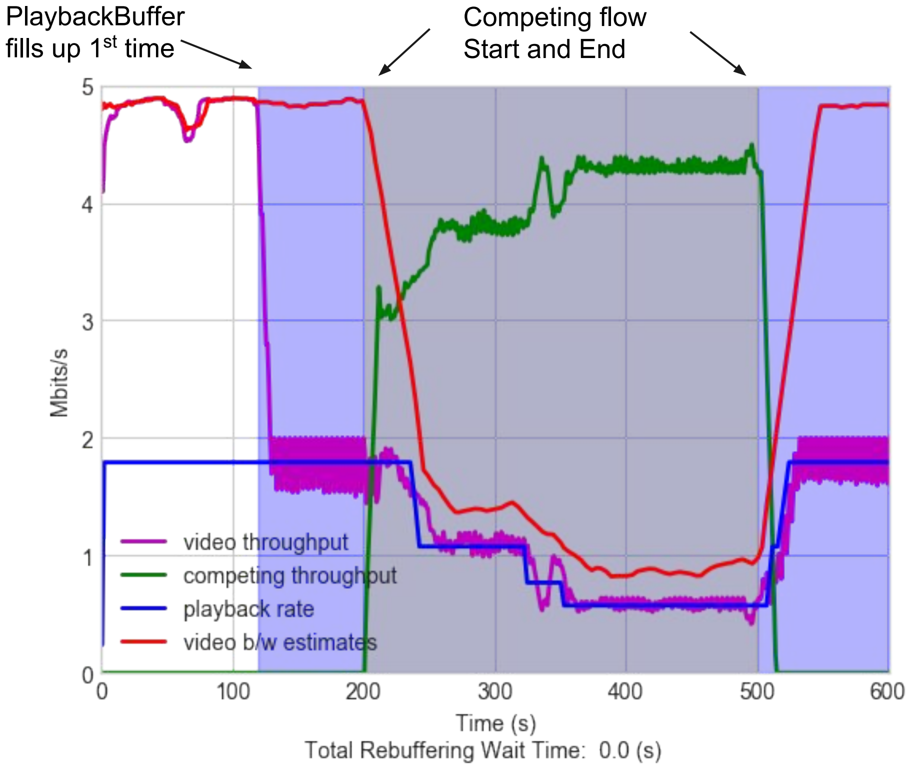
Else

Wait for Playback Buffer (🟥) to drain by 4 s



(📈)

Downward Spiral (for persistent http based video streaming)

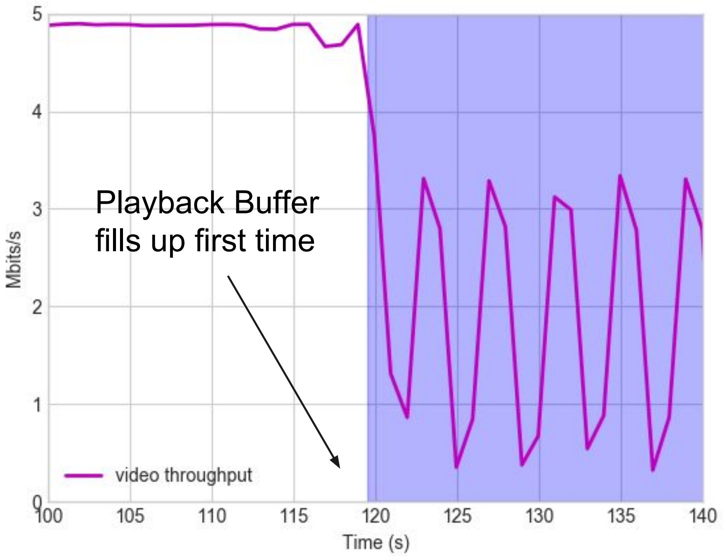


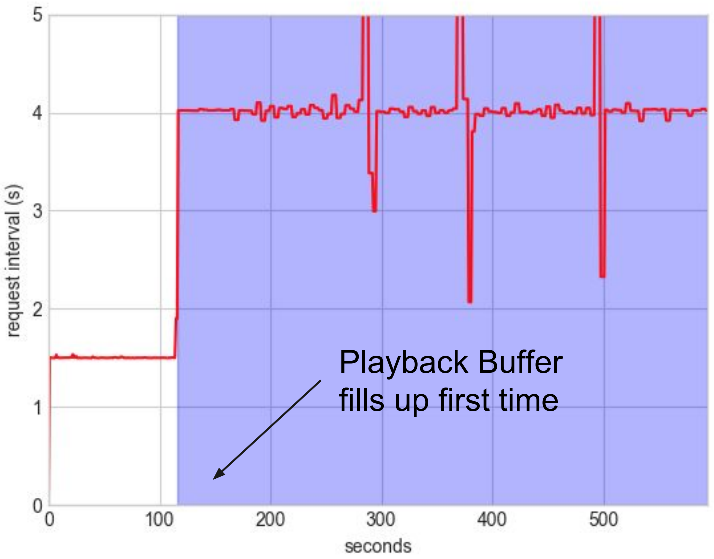
- Videos generally **hosted on CDN.**
- Hence, Rate Selection must be at **Client Side**
- In presence of competing flow, rate selection algorithms have to face weird consequence: **“Downward Sprial”**

⇒ All plots in the presentation are our Emulation / Experiments.

⇒ Many not present in paper.

Effect of Playback Buffer fill up - ON-OFF Seq



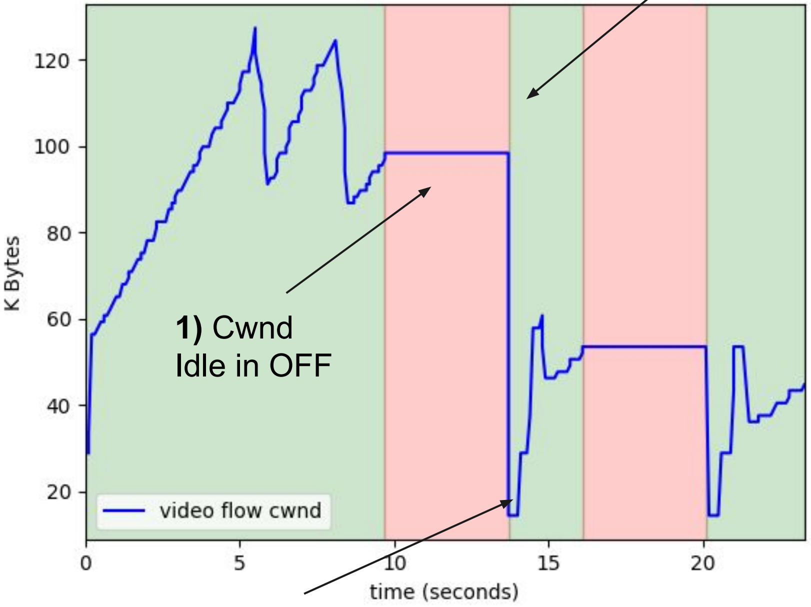


Playback Buffer Fills up first time,
Video Client goes in ON-OFF sequence.
ON ⇒ Request for video segment
OFF ⇒ Wait for buffer to drain to contain next segment.

Playback Buffer Fills up first time,
Intervals between request increases
because now it needs to pause to let buffer drain.

Effect of ON-OFF Seq on Congestion Window

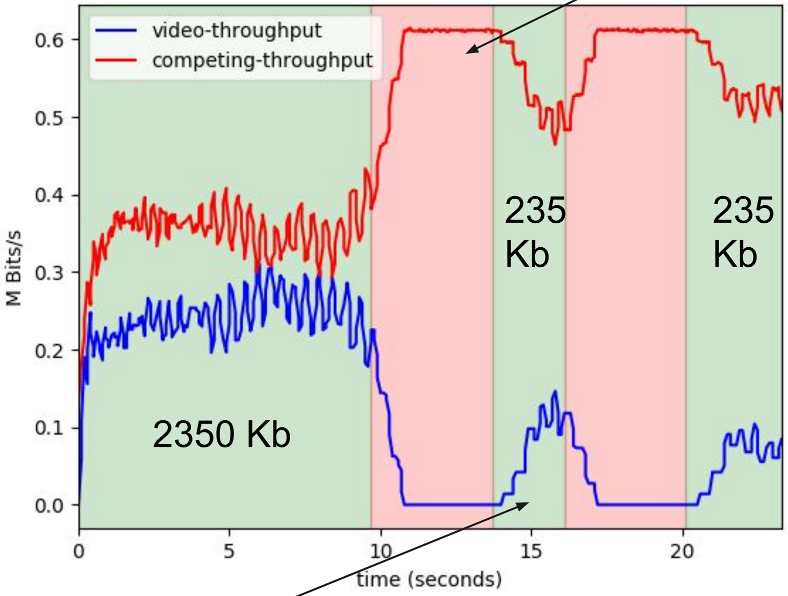
ON **OFF** **ON** **...**



1) Cwnd Idle in OFF

3) Video Client awakes,. Observes heavy loss. Cwnd crashes

2) During OFF,Competing flow occupies full b/w, fills up router buffers



2350 Kb

235 Kb

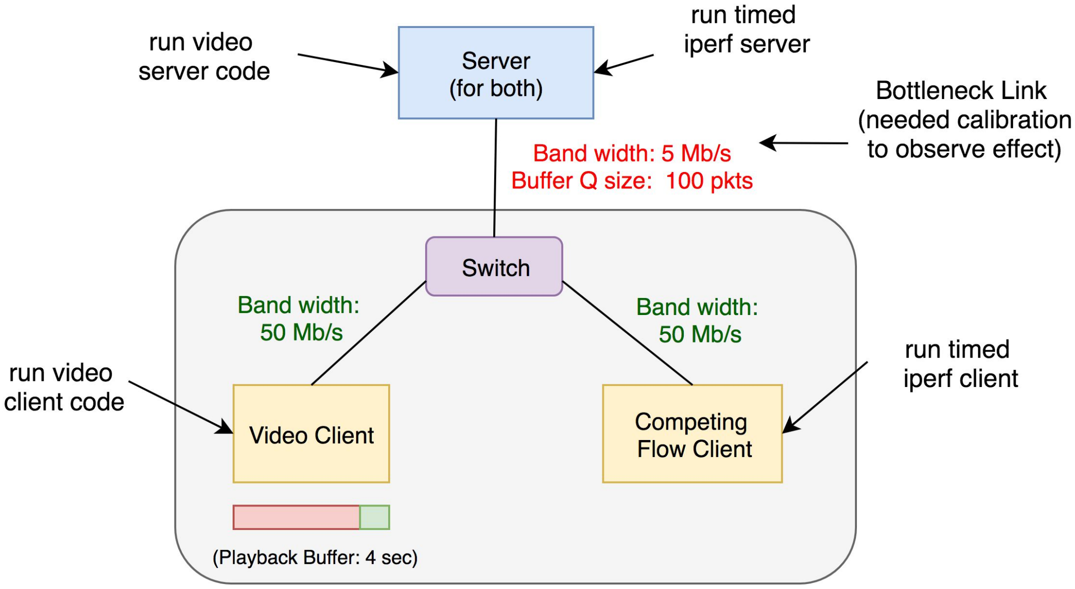
235 Kb

5) For smaller rate (smaller segment), cwnd does not get time to reach fair share. Yet lower perceived throughput. **Yet lower rate!**

4) Client perceives lower throughput. Selects **lower playback rate** next time

Vicious Cycle (1 → 5 → 1)

Emulation using Mininet



All parameters need to set properly to see the effect

1. Playback Buffer
2. Bottleneck Bandwidth
3. Router Buffer Q-size
4. Appropriate Start times

Eg. Buffer Q size should be such that in pause time competing flow should be filling it up.

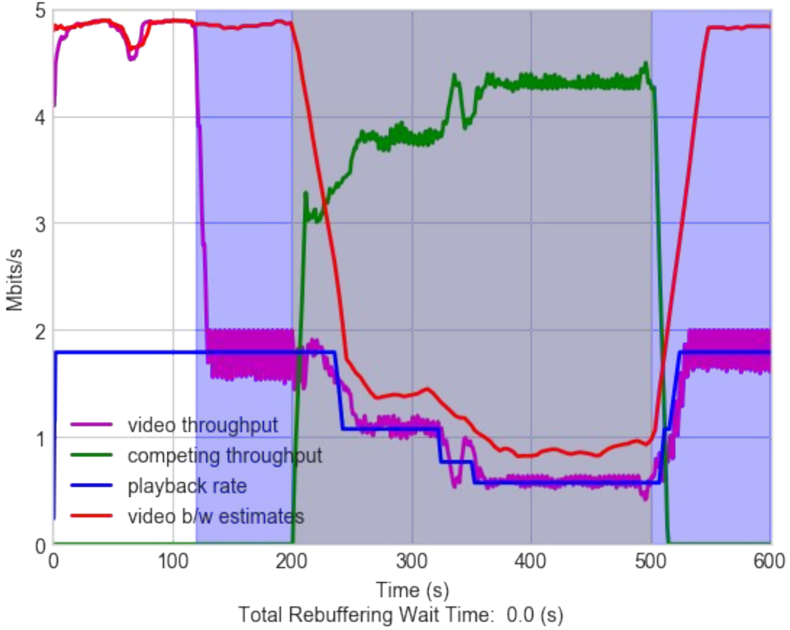
All of this is **Not** possible in
A real environment check.

Even for Emulation it was hard because many things are not given in paper. So we had to do significant trial and error.

Observations -1 (playback fills before competing start)

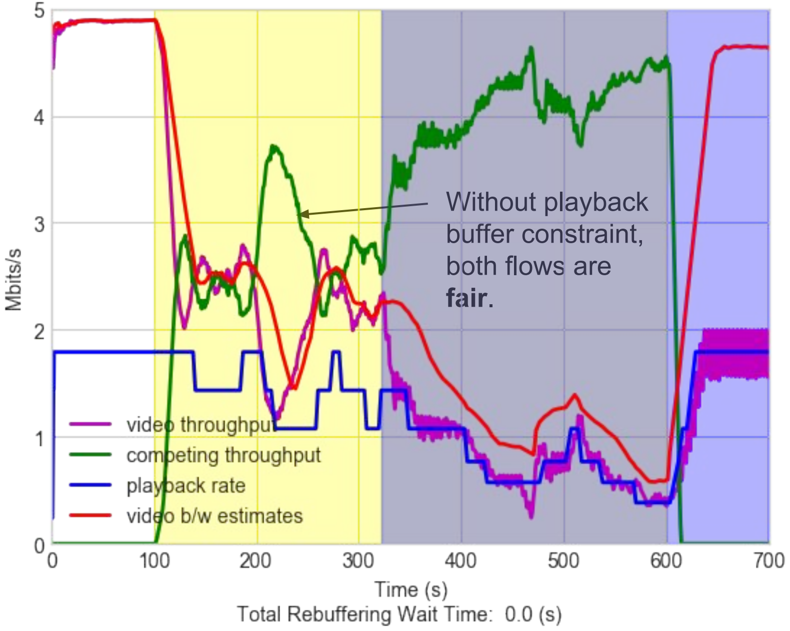
Playback
Buffer fills up

Competing flow
Start and End



Playback
Buffer fills up

Competing flow
Start and End



Without playback buffer constraint, both flows are **fair**.

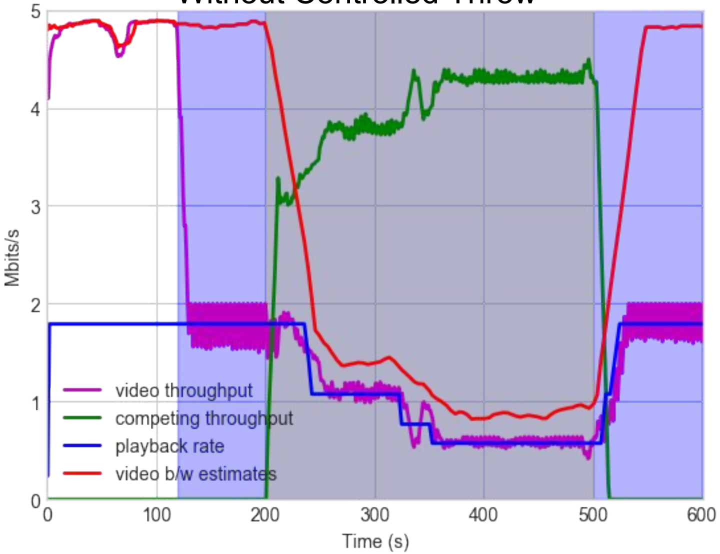
Our Improvement - Motivations

Instead of waiting for buffer to drain, throw away latest packets!

- Main culprit of Downward Spiral is ON-OFF sequence
- Only cause of ON-OFF sequence is Buffer Fillup.
- **Simple Fix**: Instead of waiting for buffer to drain, throw away **latest** packets!
- Very Counterintuitive - but works great!
- ⇒ Completely removes ON-OFF sequence, and so removes Downward Spiral fully!
- ⇒ Doesn’t incur any rebufferring events!

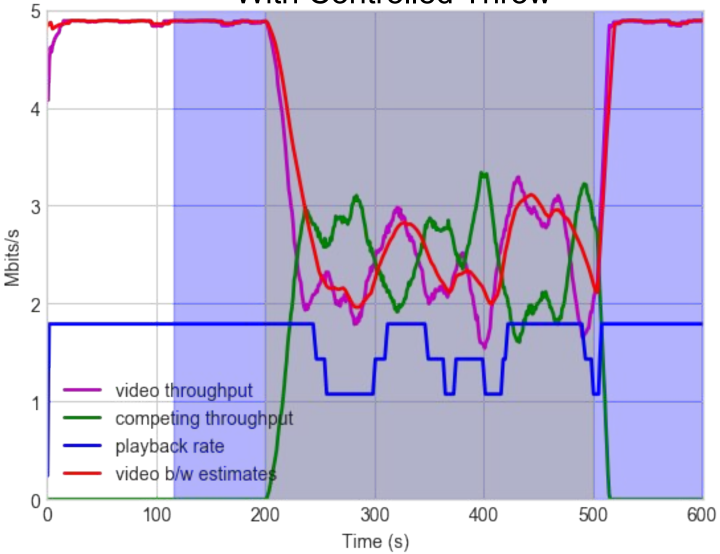
Our Improvement - Result

Without Controlled Throw



Total Rebuffering Wait Time: 0.0 (s)

With Controlled Throw



Total Rebuffering Wait Time: 0.0 (s)

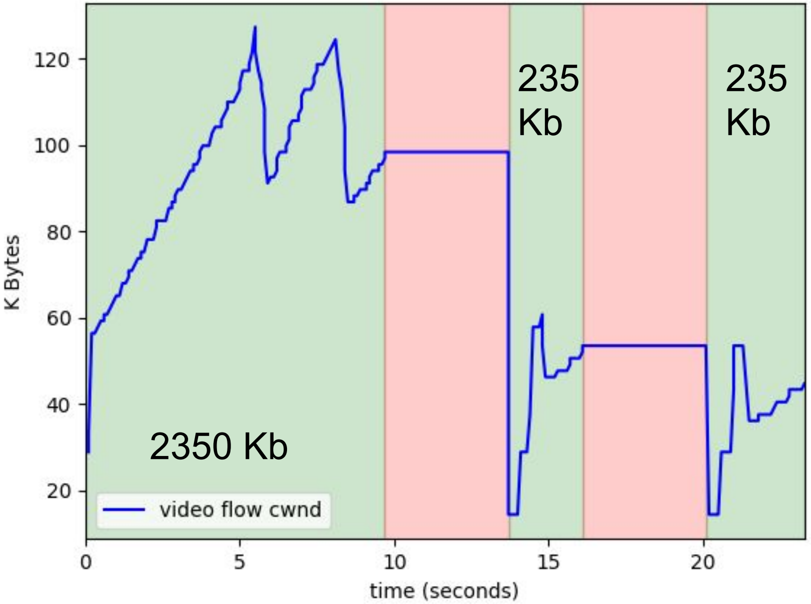
It **completely removes** the “Downward Spiral” without incurring any re-buffering events!

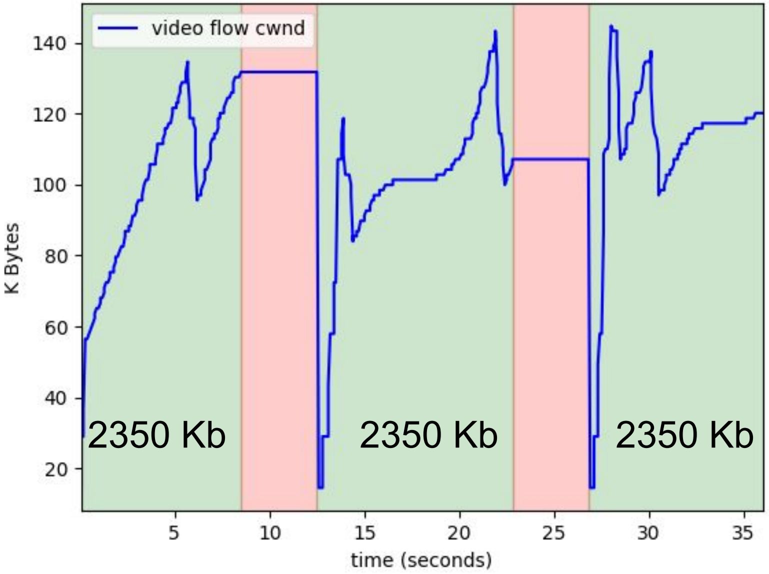
Authors Suggested Improvements

1. Bigger Segments:
2. Optimistic Client
3. Wider Moving Average

Improvement (1) - Bigger Segments - Motivation

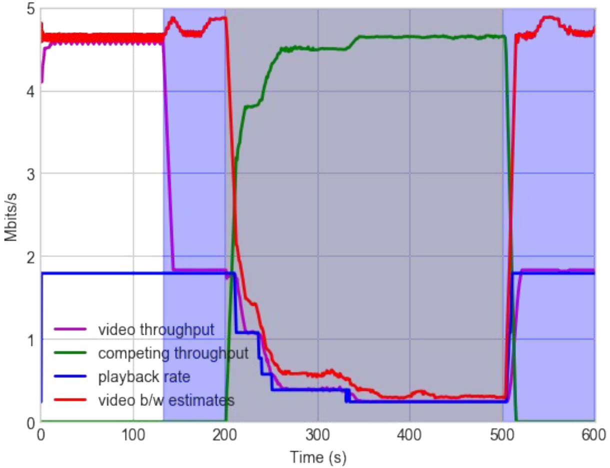
Bigger segments allow congestion window to raise to fair share.





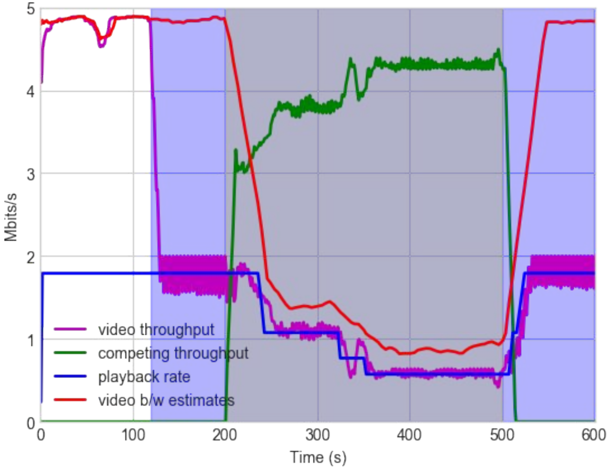
Improvement (1) - Bigger Segments - Results

1 second segment



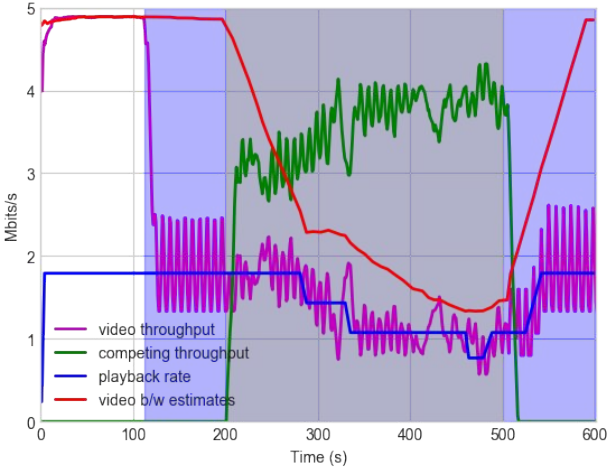
Total Rebuffering Wait Time: 0.0 (s)

4 second segment



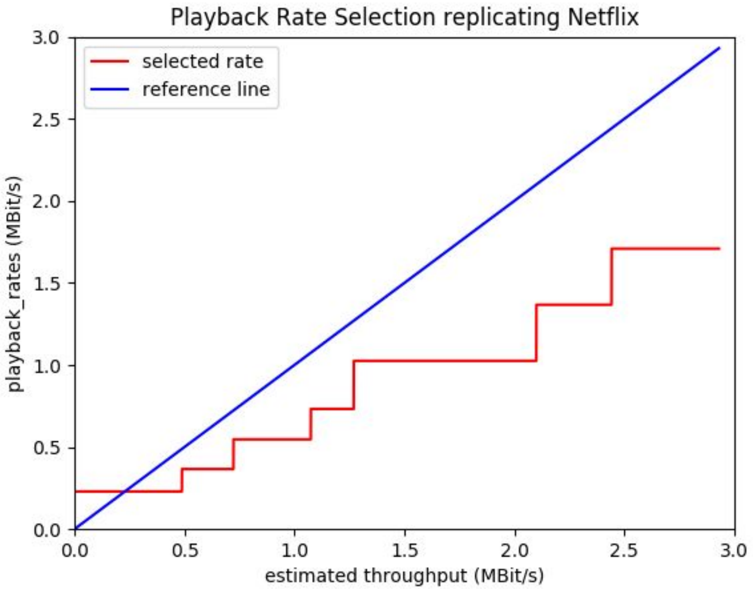
Total Rebuffering Wait Time: 0.0 (s)

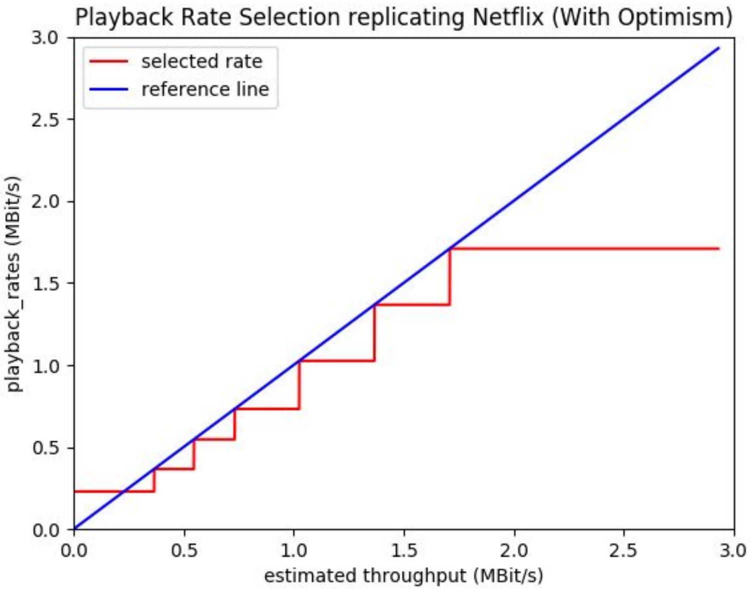
8 second segment



Total Rebuffering Wait Time: 0.0 (s)

Improvement (2) - Optimistic Client - Motivation

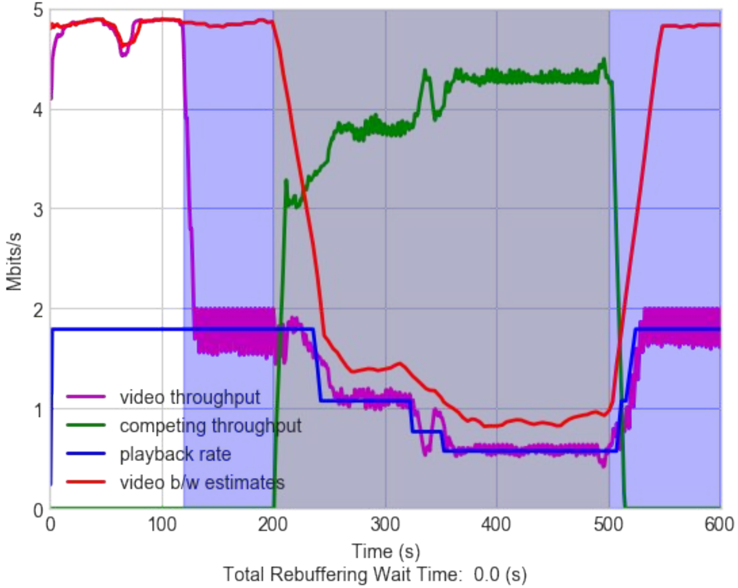




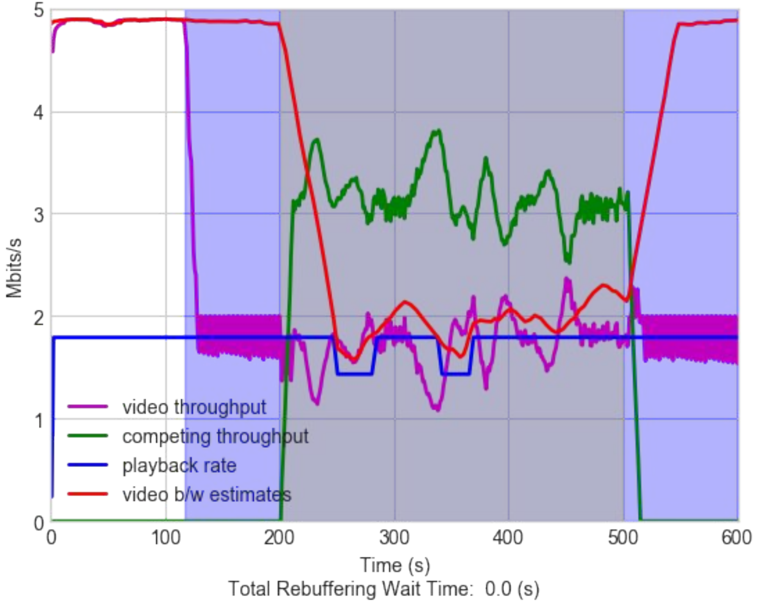
Given the estimated throughput, choose the highest playback rate.
Since we know lower rate leads to even lower rate, it’s better to be optimistic in choose playback rate.

Improvement (2) - Optimistic Client - Results

Rate Lookup without Optimism

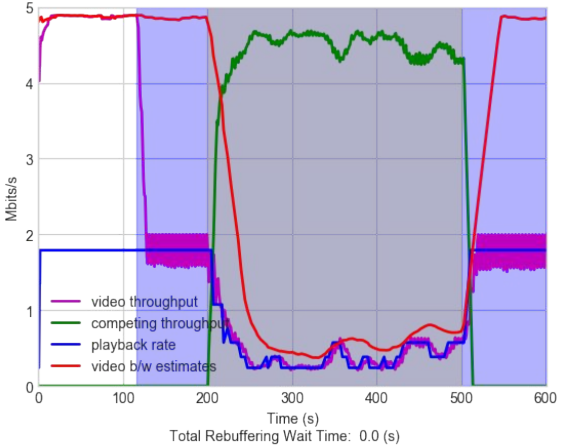


Rate Lookup with Optimism

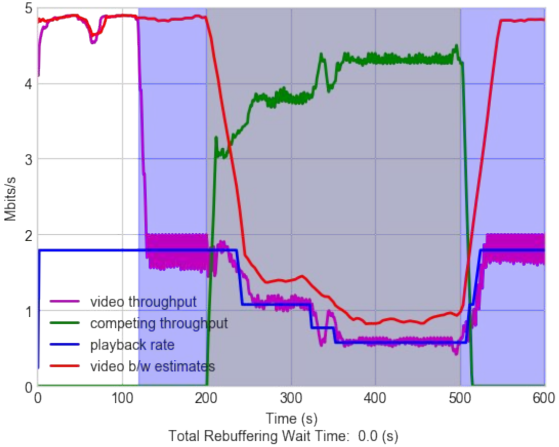


Improvement (3) - Wider Averaging - Results

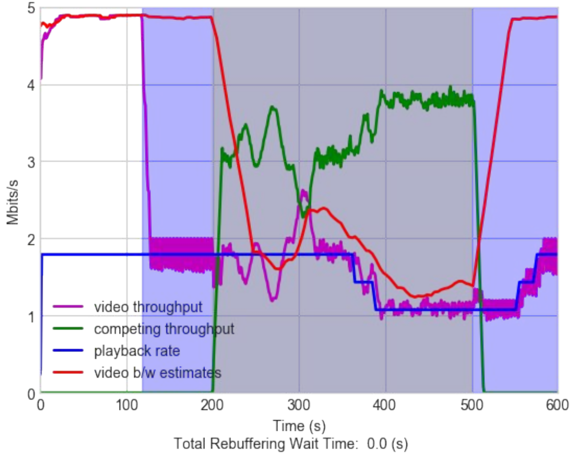
No Moving Average



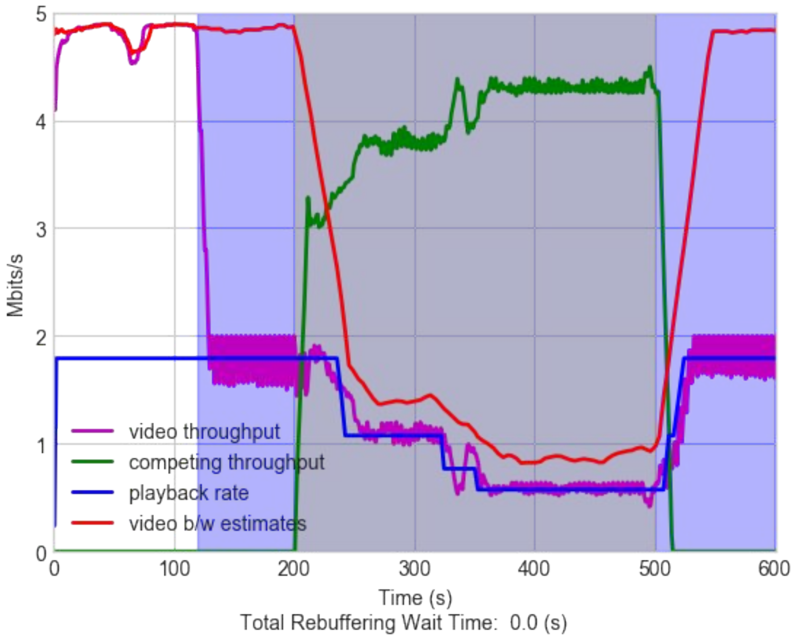
Moving Average of 10



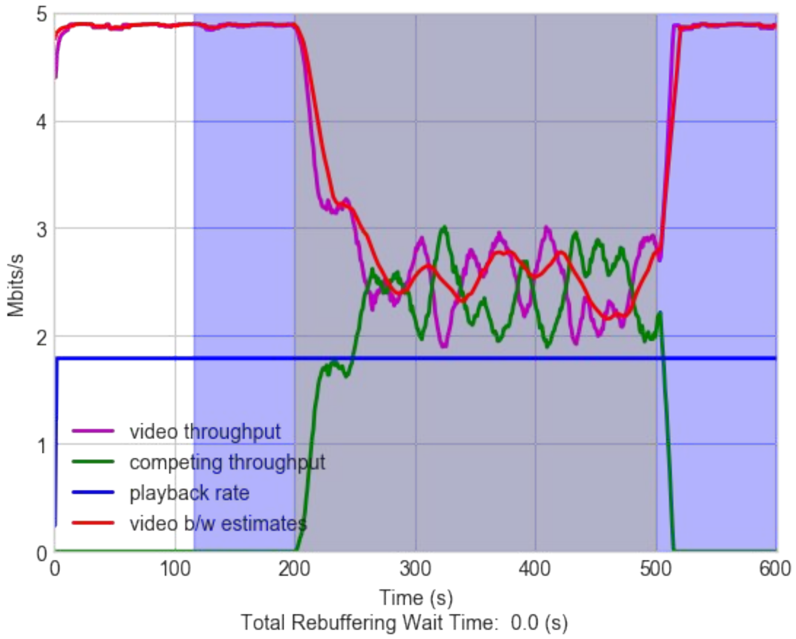
Moving Average of 50



Before and After (Optimistic + Our Improvement)

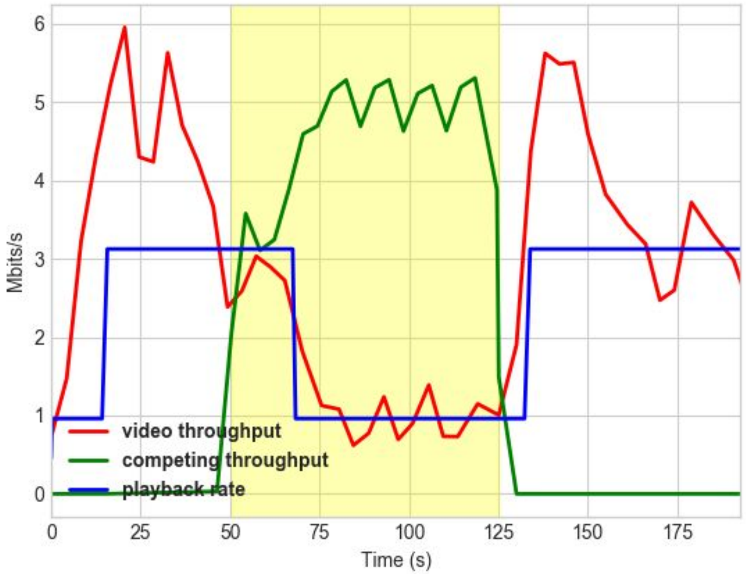


Before



After

Real Environment : Reproducing Downward Spiral



Downward Spiral reproduced on **Vimeo** service

Real World Setup : Reproducing Downward Spiral

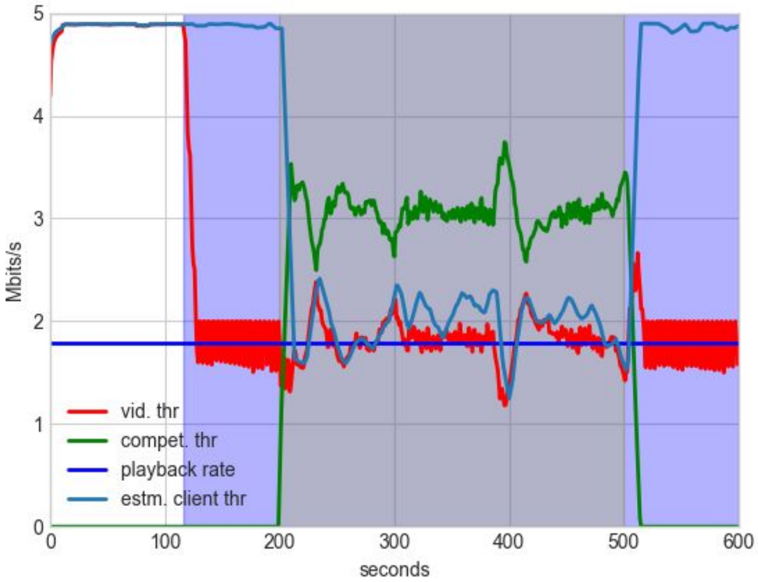
- ➔ Set up a Bottleneck Link
- ➔ Download DASH manifest file prior to streaming.
- ➔ Video Client streams from Vimeo.
- ➔ Introduce Competing Flow - Download the same video.
- ➔ Kill Competing Flow
- ➔ Log HTTP Requests made while client flow is active.
- ➔ Track Throughput of Client
- ➔ Track Throughput of Competing Flow
- ➔ Calculate Playback Rate

Conclusion of our work

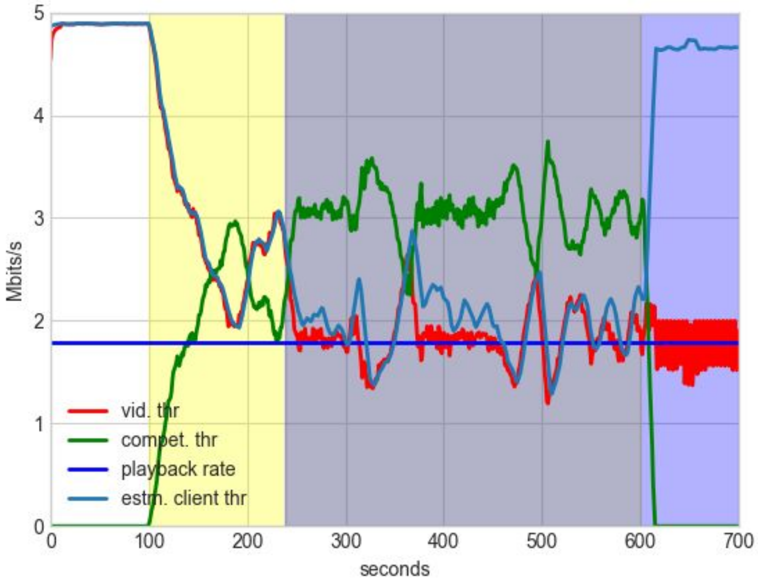
1. Emulation on "downward spiral" mininet on all experiments of paper with our own calibrated video streaming client.
2. Provided a our simple solution that seems to completely remove downward spiral.
3. Showed 3 improvements on emulation with author suggested changes.
4. Show real environment experiment with youtube-dl, mahimahi on vimeo

Extra Slide-1:

Observations -2 (turn off client-side rate selection: fix at 1.7 Mbps)



Rebufferring events: **0 (s)**

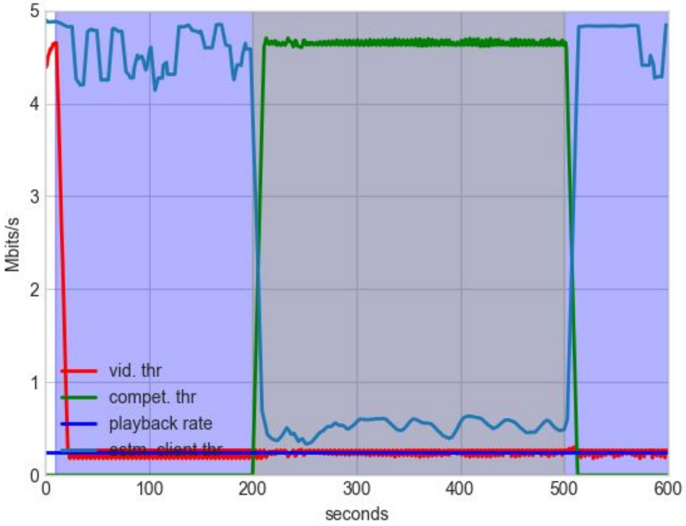


Rebufferring events: **0 (s)**

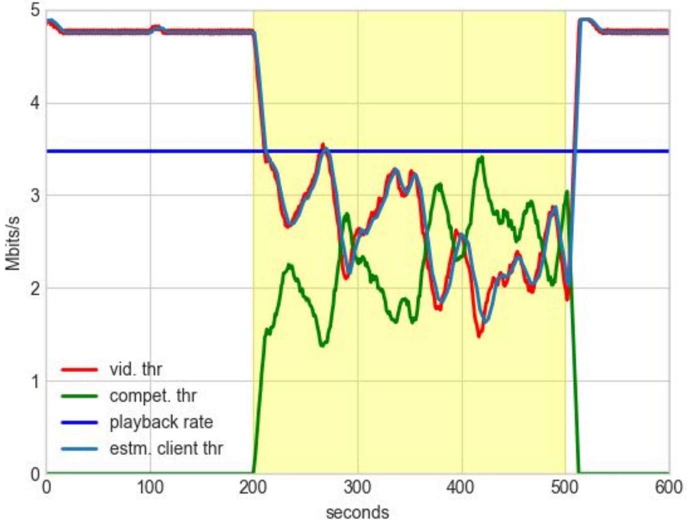
The client is able to sustain the maximum playback rate without incurring any rebufferring events, and bandwidth is available. So clearly the problem is in client side rate selection algorithm

Extra Slide-2:

Observations - 3 [Trade off] (high vs low fixed playback rate)



Rebufferring events: **0 (s)**



Rebufferring events: **33 (s)**

References

- Huang, Te-Yuan, et al. "Confused, timid, and unstable: picking a video streaming rate is hard." *Proceedings of the 2012 Internet Measurement Conference*. ACM, 2012.