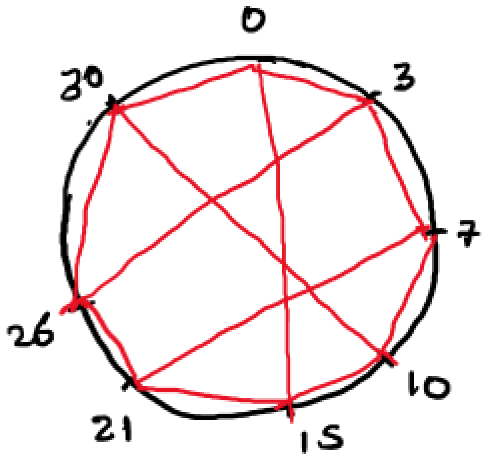
Computer Networks
COL 334/672

Video Streaming

Slides adapted from KR

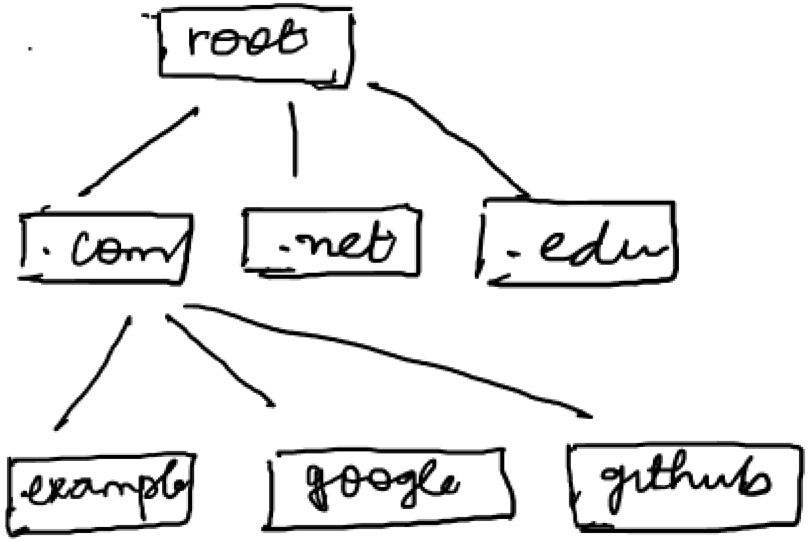
Sem 1, 2025-26

1.



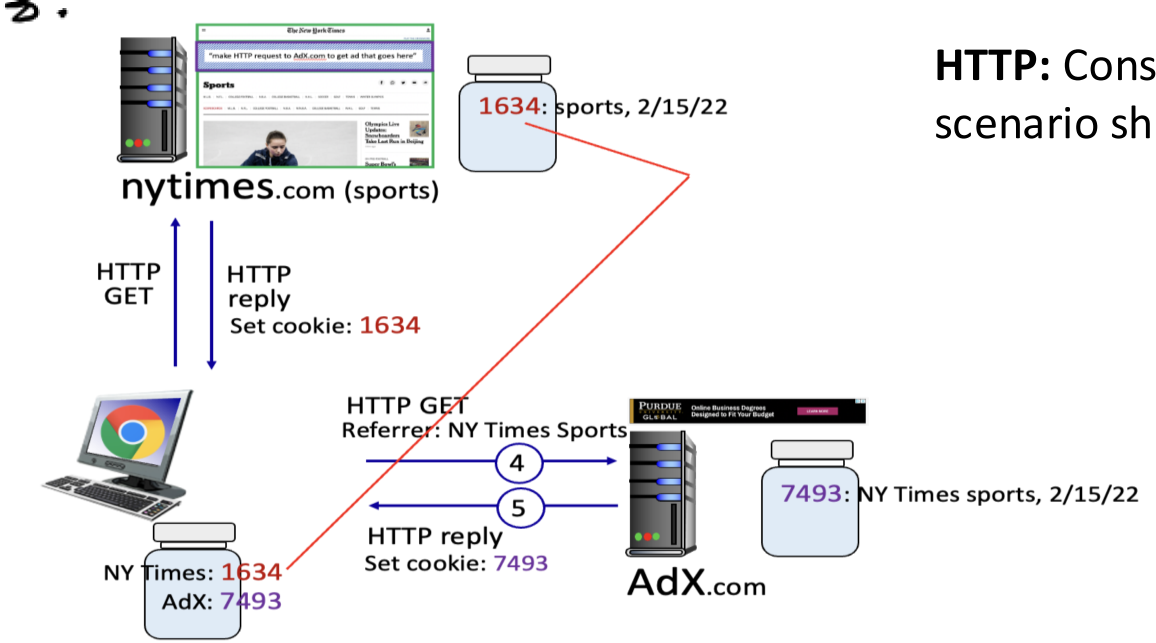
DHT: Consider the following DHT system using a 2^5 virtual space and successors for breaking ties. Assuming node 3 is looking for a filename F with a hash value of 16 and the file is actually present at node 6. The lines represent the connectivity among the peers.

2.



DNS: Consider the scenario on the left, showing the name servers hierarchy in DNS. Suppose the domain *example.com* adds a new page *abc.example.com*.

3.



HTTP: Consider the web browsing scenario shown here.

Quiz on Moodle

Password:

application

How to design a video on-demand streaming service?

Pre-encoded video

① Not live

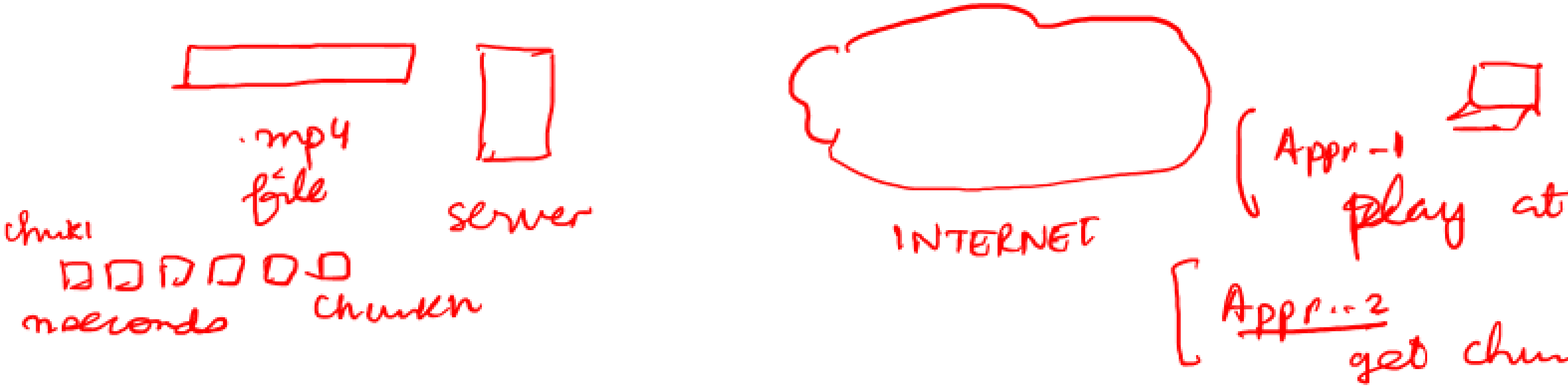
② Not Real-time interactive (Whatsapp / Teams)

Design Goals

#1: No skipping of video content due to packet loss

Reliability : TCP as the transport channels

#2: Implement interactivity (pause, repositioning, fast-forward)



~~Roll~~bar

[Appr-1 play at t= x sec]

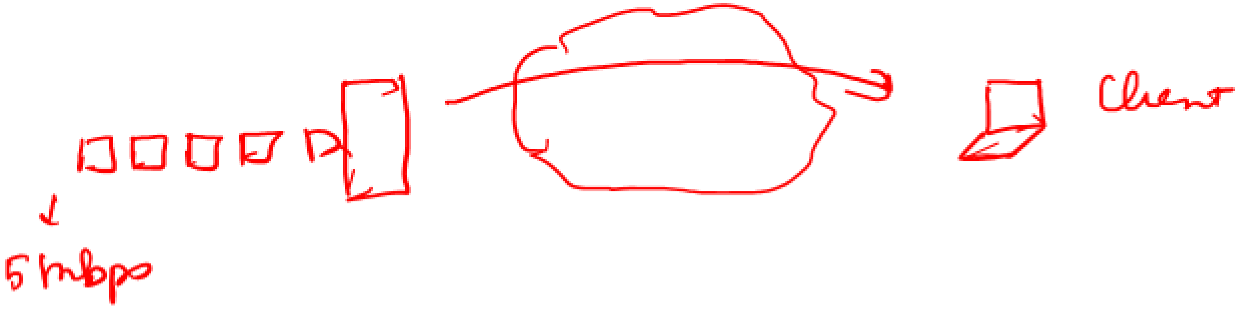
[Appr-2 get chunk k ~~@ 10s~~ (less state in this approach)]

#3: Continuous playout of content, i.e., avoid video freezing, across diverse network condiions

#4: Scale to a million users

Handling diverse network conditions

N/w condition within a session may be dynamic



what rate should it download the data?

↳ download > 5Mbps

Fixed-sized buffer (# bytes

seconds)

↓

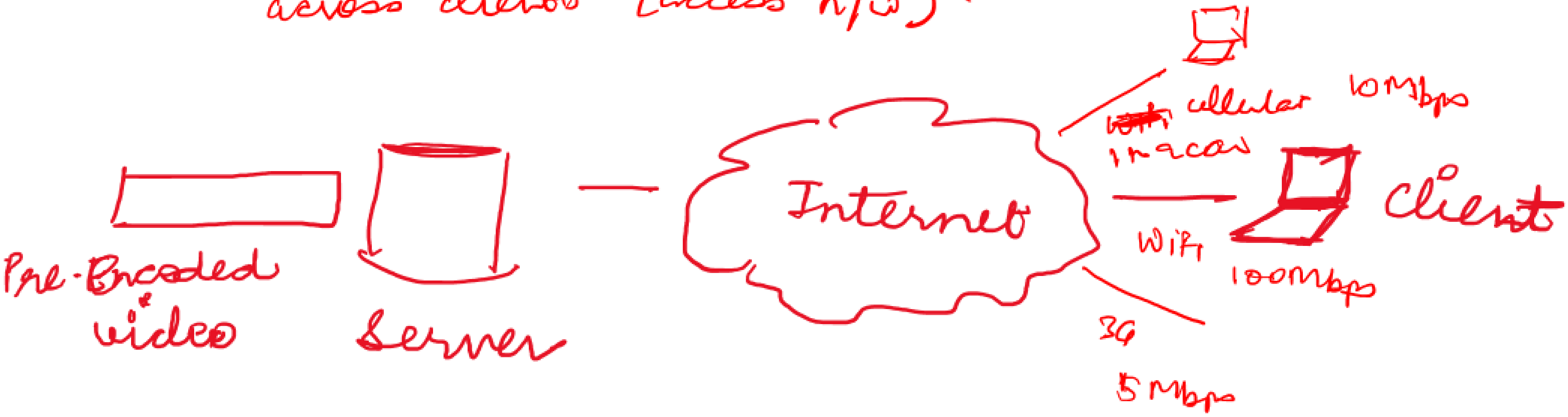
20s

n seconds of extra content

Buffer mitigates some n/w jitter

Handling diverse network conditions

across clients (access n/w).



Q: What bitrate should we encode the video?

Constraints: clients with diverse network conditions

- Real-time encoding [not efficient]
- Pre-encode multiple versions

Bitrate Adaptation

① Store the video at multiple quality levels [Scalable video encoding]

②. Real-time encoding of the video (very compute expensive)

How to achieve scale?

HTTP - Adaptive Streaming (HAS)

DASH: Dynamic Adaptive Streaming over HTTP [illegible].

- Need geographically distributed video servers (special servers?)

CDNS : Content Distribution Networks

Reuse HTTP servers to download video content

HTTP GET CHUNK1

GET CHUNK-2

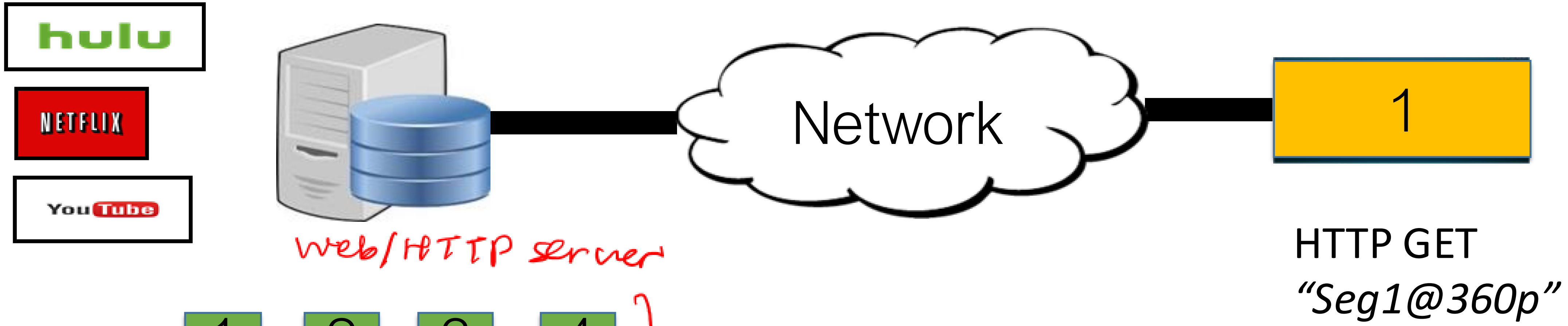
HTML5 : VideoAPI

① Reuse CDN infrastructure

② Firewall friendly

③. Used open-source protocols (HTTP MPEG for encoding)

HTTP Adaptive Streaming (HAS)



web/HTTP server

720p	1	2	3	4
480p	1	2	3	4
360p	1	2	3	4

→ How do I even know about quality level?, → ①. Manifest file

logic to Player side stream

- *"intelligence"* at client: client determines
  - *when* to request chunk (so that buffer starvation, or overflow does not occur) → Buffer size
  - *what encoding rate* to request (higher quality when more bandwidth available)

When to request a new video chunk?

- Client keeps a maximum buffer threshold, i.e., the maximum amount of downloaded but not played video
  - Either expressed as duration or number of bytes

- If the current video buffer occupancy > max buffer threshold, wait for the video buffer to deplete to less than max buffer threshold

- Once video buffer occupancy < max buffer threshold, request a new chunk

At what bitrate?

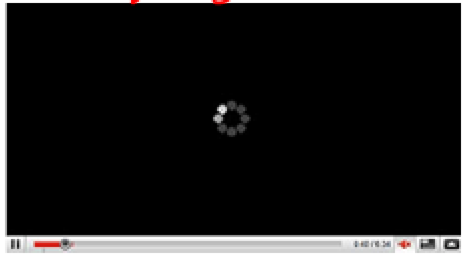
What bitrate to download the chunk (Bitrate adaptation)

Signals available to the player : (Buffer occupancy
Throughput in the past)

Designing a Bitrate Adaptation Algorithm → goal

(good quality / No rebuffering)

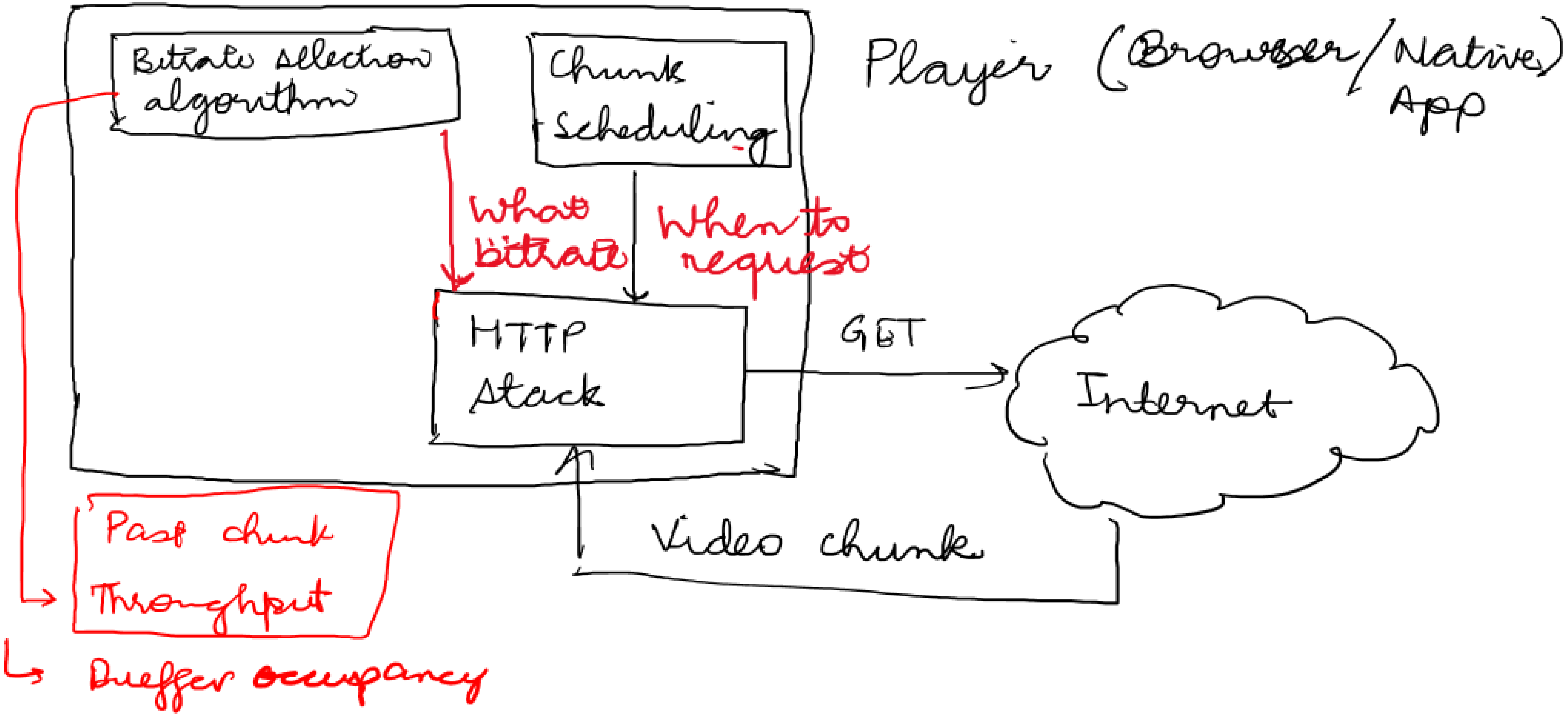
Minimize stall duration
freezes / rebuffering



Maximize
average bitrate → quality

bitrate switches Minimize

Bitrate Adaptation



- **Q**: What are the signals available to the player for bitrate adaptation?

Bitrate Adaptation: Algo #1

Idea:

Estimate network bandwidth based on the past download rate.

Download chunk at a bitrate just less than the estimated network bandwidth

How do you estimate b/w

↳ Average across latest K chunks

1 chunk : Too many fluctuation

↓

Too many bitrate oscillation

Bitrate adaptation: Algo #1

- **Idea**:
  - Estimate network bandwidth based on the past download rate.
  - Download chunk at a bitrate just less than the estimated network bandwidth

- **Algorithm**

1. Estimation: Take into account historical values, not just the last chunk throughput
2. Smoothing: Apply a smoothing filter such as average, harmonic mean or EWMA
3. Quantization: Select bitrate from the discrete set of bitrates based on estimated throughput

Example

Available bitrates

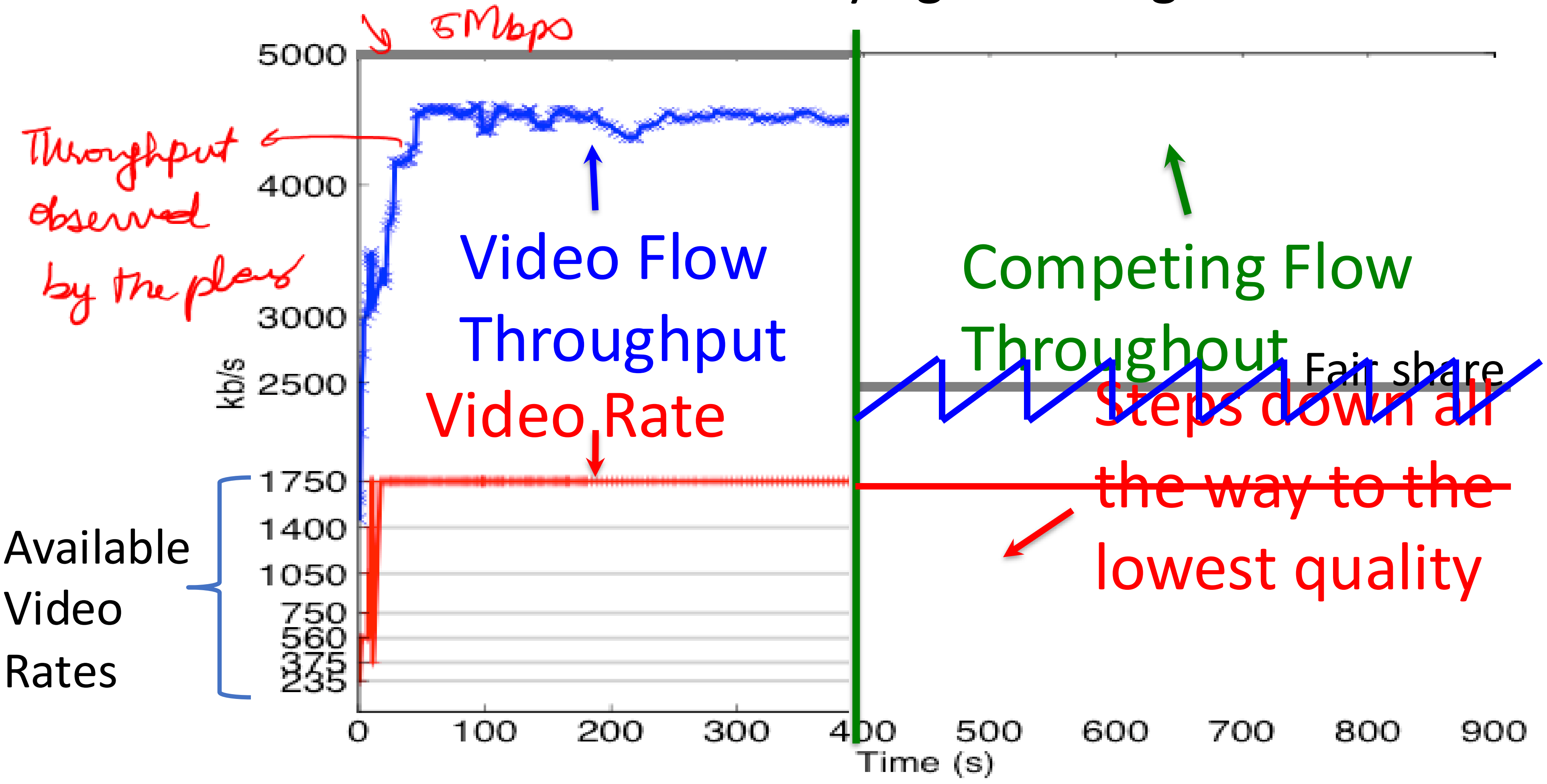
= {200, 400, 800, 1600} Kbps

Chunk throughputs

{700, 900, 1000, 700}

Issue with Rate-based Adaptation

- Poor interaction with the underlying TCP congestion control



Video Flow Throughput

Video Rate

Competing Flow Throughput

Fair share

Steps down all the way to the lowest quality

Available Video Rates

Source: Confused, Timid and Unstable: Picking a Video Rate is Hard

TCP Throughput of the Video Flow

- TCP sender resets its congestion window during OFF period
- Throughput will be affected especially with a competing flow
- Experience packet loss during slow start
- 50% of the segments get < 1.8Mb/s

Source: Confused, Timid and Unstable: Picking a Video Rate is Hard