X_405082

Advanced Computer Networks

Video Streaming

Lin Wang (lin.wang@vu.nl)

Period 2, Fall 2020

VU VRIJE UNIVERSITEIT AMSTERDAM

Course outline

Warm-up

- Fundamentals
- Forwarding and routing
- Network transport

Data centers

- Data center networking
- Data center transport

Programmability

- Software defined networking
- Programmable forwarding

Video

- **Video streaming** 👈
- Video stream analytics

Networking and ML

- Networking for ML
- ML for networking

Mobile computing

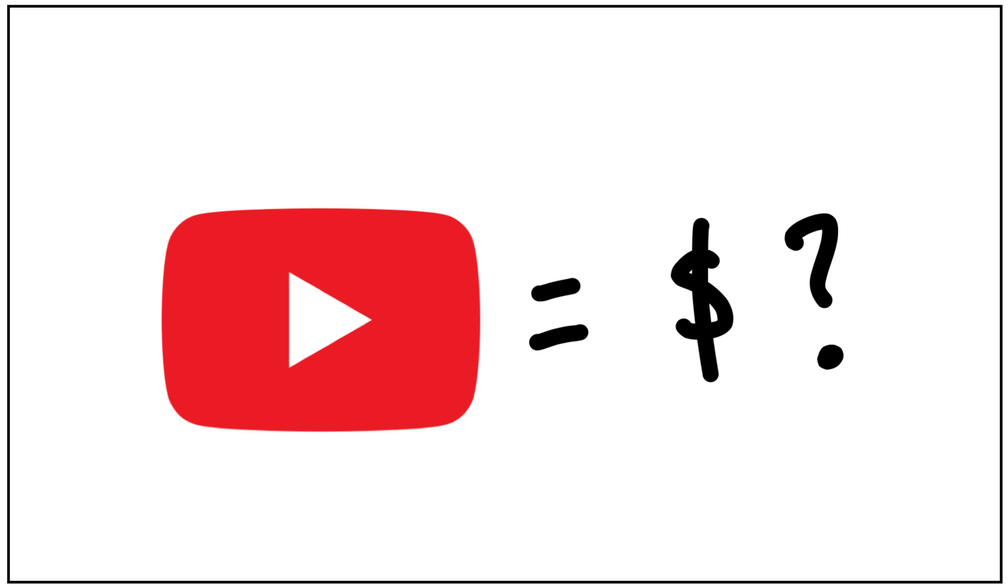
- Wireless and mobile

Learning objectives

How does video get streamed over the Internet?

What are the **challenges** in streaming video over the Internet? How to solve them?

How much is YouTube worth today?

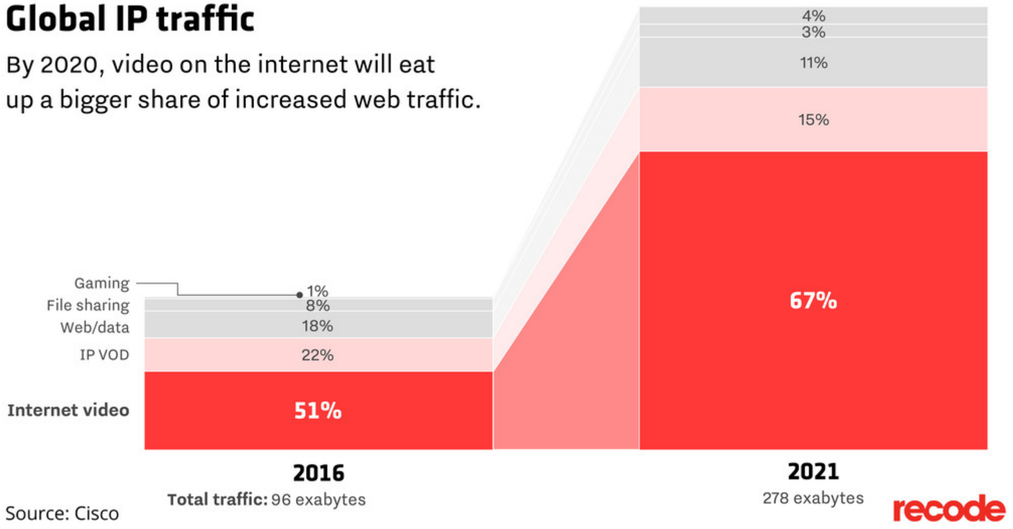


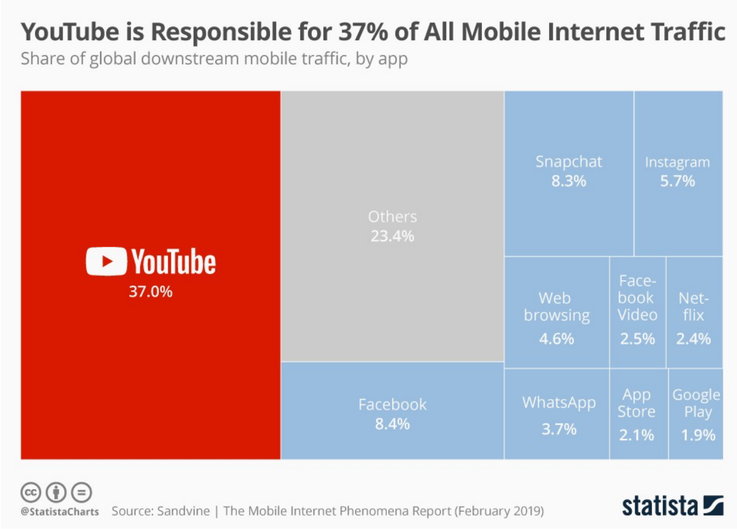
	G = C + F	H	I = G x H	J	J/ G	K/ G	K = I - J	L	M	K - L - M
CY end	Total Revenue ($m)	% GP margin	Gross Profit ($m)	Opex ($m)	Opex as % revenue	% EBIT margin	EBIT ($m)	Tax ($m)	Reinvest (Net capex) ($m)	Unlevered Free Cash Flows ($m)
2019	**16,949**	37%	**6,261**	5,932	35%	2%	**328**	-69	-2751	**-2,491**
2020	**22,094**	37%	**8,156**	7,733	35%	2%	**423**	-89	-2725	**-2,391**
2021	**28,602**	37%	**10,569**	8,581	30%	7%	**1,988**	-418	-3447	**-1,876**
2022	**36,882**	37%	**13,655**	11,065	30%	7%	**2,591**	-544	-4386	**-2,339**
2023	**44,979**	37%	**16,840**	11,245	25%	12%	**5,595**	-1,175	-4288	**132**
2024	**51,690**	38%	**19,642**	12,922	25%	13%	**6,720**	-1,411	-3554	**1,754**
2025	**58,083**	38%	**22,071**	11,617	20%	18%	**10,455**	-2,196	-3386	**4,873**
2026	**63,975**	38%	**24,310**	12,795	20%	18%	**11,515**	-2,418	-3121	**5,976**
2027	**71,032**	38%	**26,992**	14,206	20%	18%	**12,786**	-2,685	-3738	**6,363**
2028	**74,989**	38%	**28,496**	14,998	20%	18%	**13,498**	-2,835	-2096	**8,568**
2029	**79,098**	38%	**30,057**	15,820	20%	18%	**14,238**	-2,990	-2177	**9,071**
2030	**83,368**	38%	**31,680**	16,674	20%	18%	**15,006**	-3,151	-2261	**9,593**
CAGR (%)	**16%**		**16%**	**10%**			**42%**			
Terminal Value										**283,693**
PV (Terminal Value)										144,215
PV of UFCF										23,867
Value of Operating Assets										**168,082**

Valuation of YouTube equals to around US$168 billion.
(Google acquired YouTube for US$1.6 billion in 2006.)

https://mannhowie.com/youtube-valuation

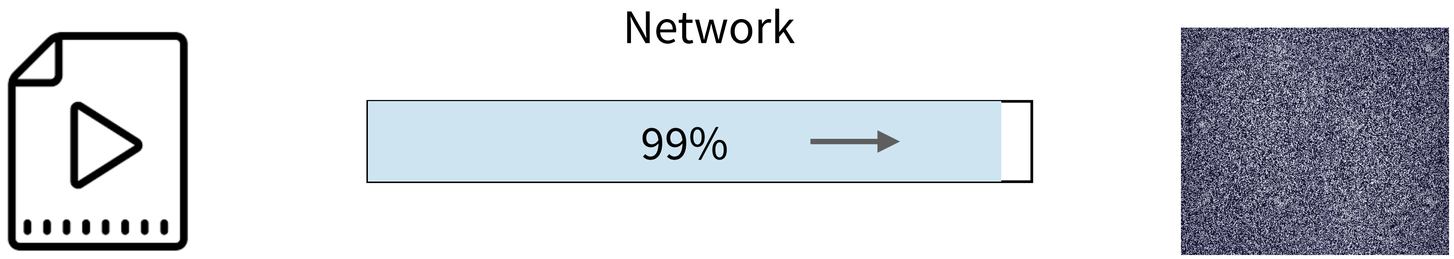
Video content has been dominating the Internet





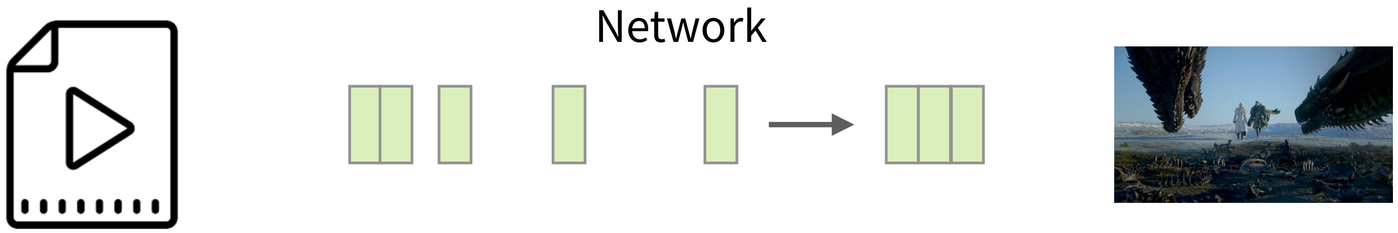
Video traffic is dominant nowadays: by 2021 it would represent more than 67% of the Internet traffic

Pre-streaming era



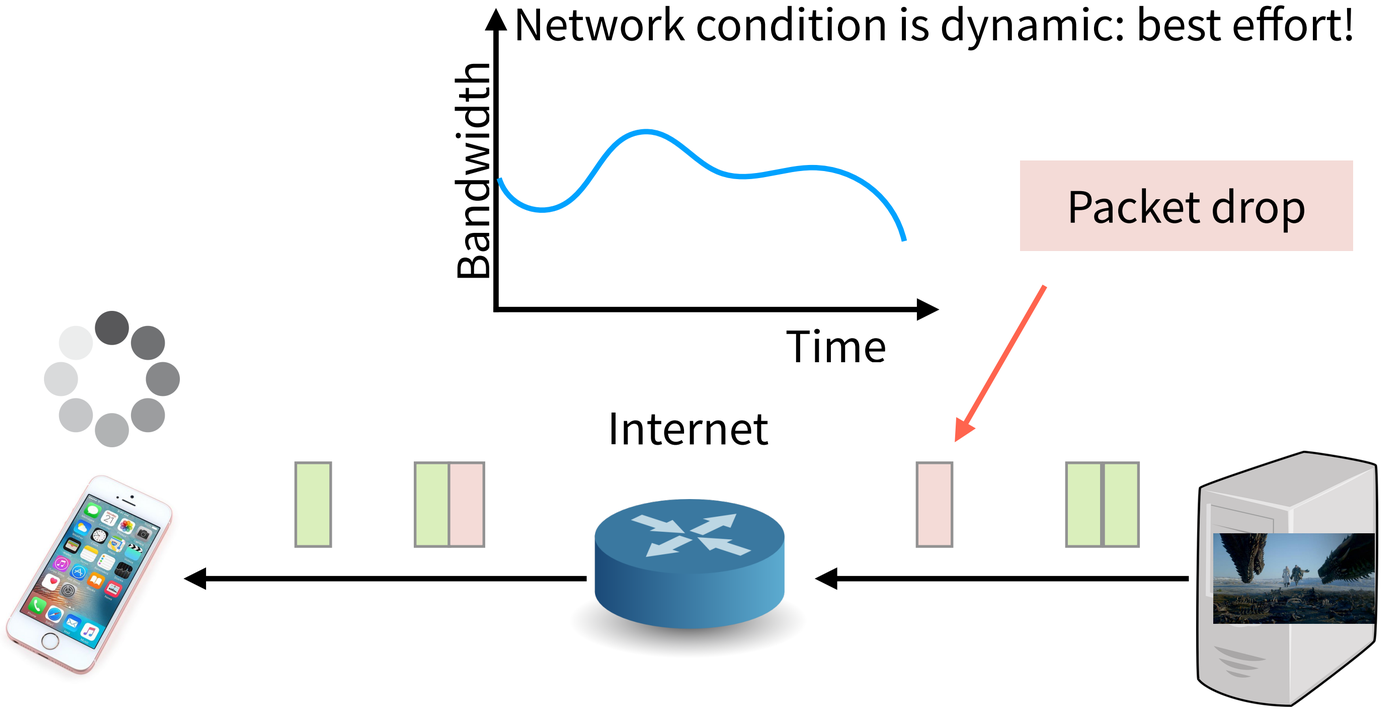
Download the whole video file and play it when the download is finished.

Streaming era



Chunk the video into small segments and stream from any segment.

Challenges in video streaming



How to address/mitigate this issue?

Video compression

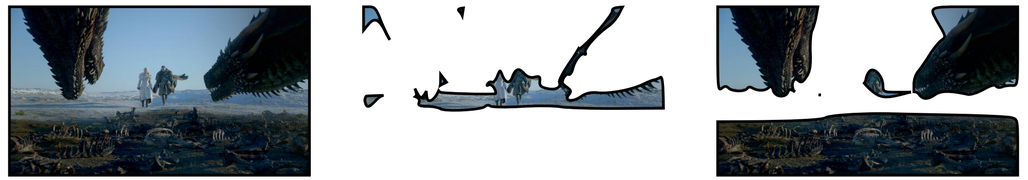
Reduce the amount of data to be transmitted over the network while keeping the video quality

Techniques:

- **Frame-level** compression (spatial): resize/encode the image
- **Video-level** compression (temporal): encode the images across time (calculating deltas)

Frame 1 Frame 1

Frame-level compression



Frame 1 Frame 2 Frame 3

Video-level compression

Frame-level compression

JPEG compression

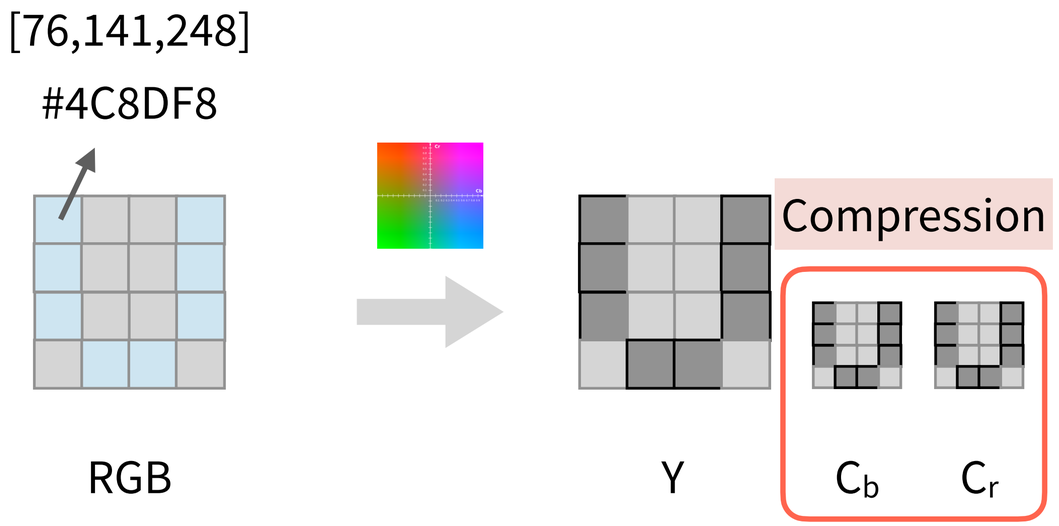
- Changes RGB to YC_bC_r
- Y: luminance, C_bC_r are chrominance

Why this change?

- Human eyes are less sensitive to chrominance than to luminance

JPEG reduces sizes of C_b and C_r: **quantization**

- Total compression rate x2

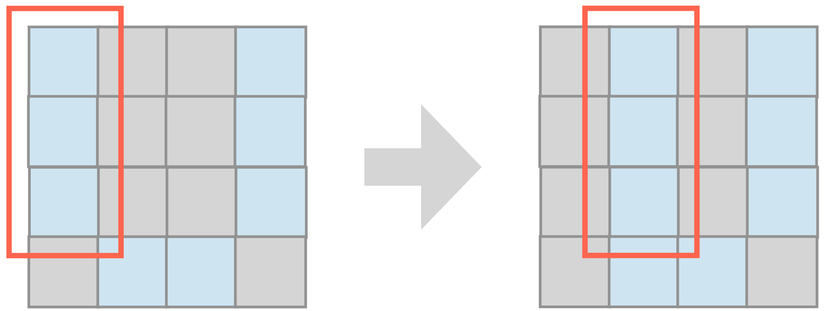


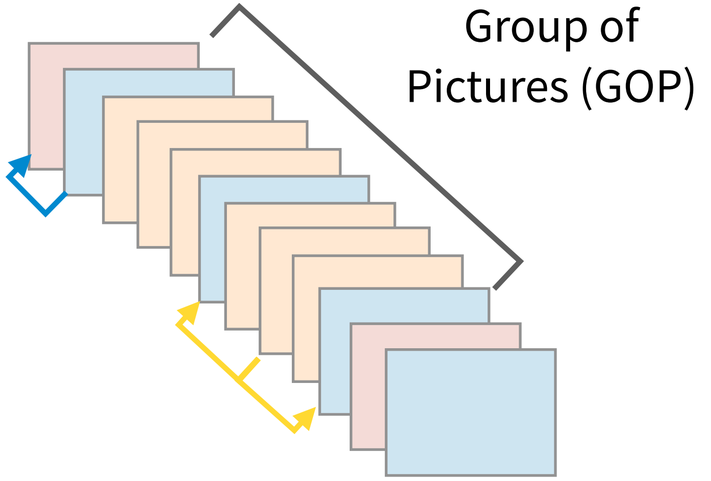
Video-level compression

H.264

Remove temporal redundancy by keeping track of the relative differences (deltas) between frames using motion vectors

Macro blocks





- **I** (intra-coded) frame: self-contained, e.g., JPEG
- **P** (predictive) frame: looks back to I and P frames for prediction
- **B** (bidirectional) frame: looks forward and backward to other frames

Compressibility highly depends on the content, why?

I frames are the largest, P frames are medium-size, and B frames are the smallest.

http://web.engr.oregonstate.edu/~benl/Courses/ece477.sp20/Lectures/4.Video_Compression.pdf

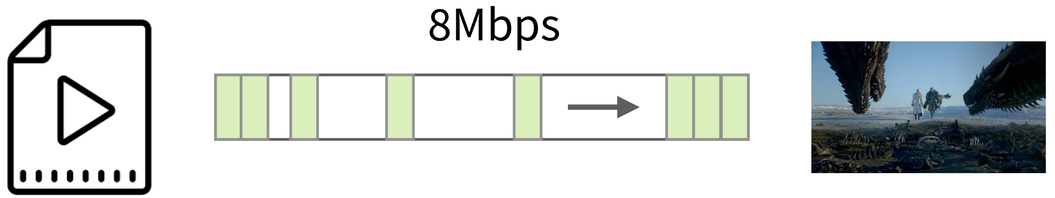
Bitrate

Measures the data size per unit time:

- Amount of data used to encode video (or audio) per second, e.g., Mbps, Kbps

Bitrate affects both the **file size** and the **quality of the video**

- Affect the required bandwidth when streaming the video over the network

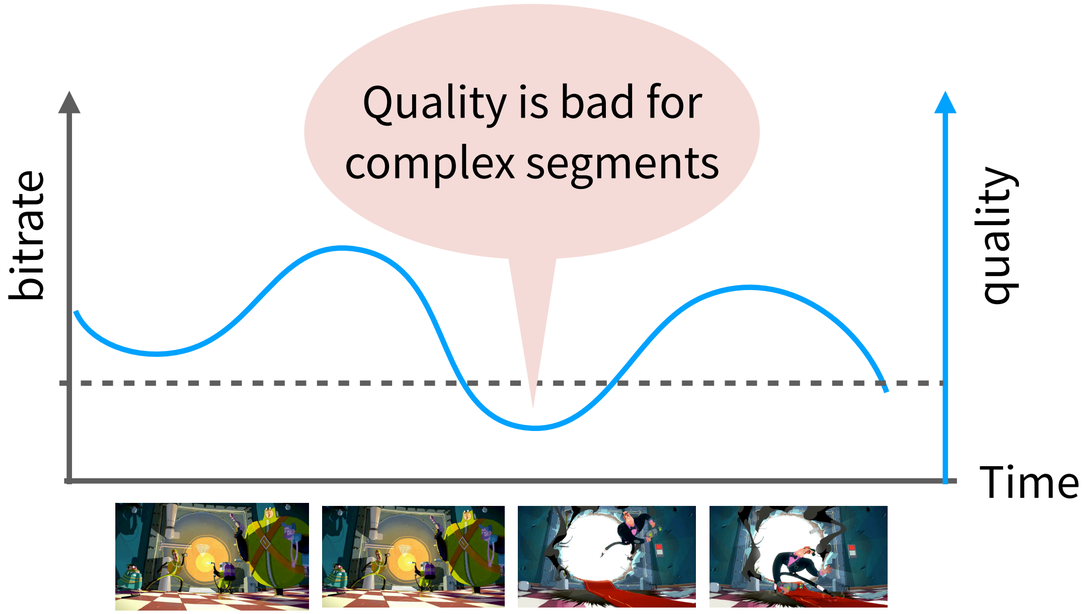


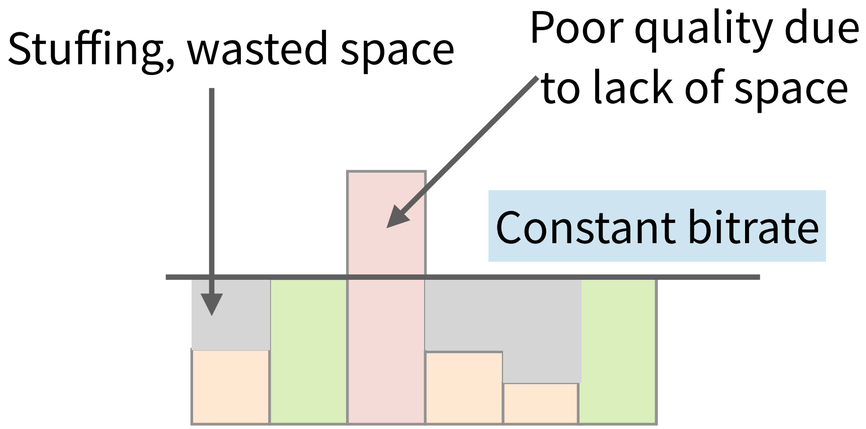
Check here for a live demo: https://reference.dashif.org/dash.js/latest/samples/dash-if-reference-player/index.html

Constant bitrate (CBR)

Compress video with a constant bitrate

- **Constant bitrate** → constant compression ratio → **varying quality**
- In H.264, quality is worse when the motion is higher due to the larger deltas



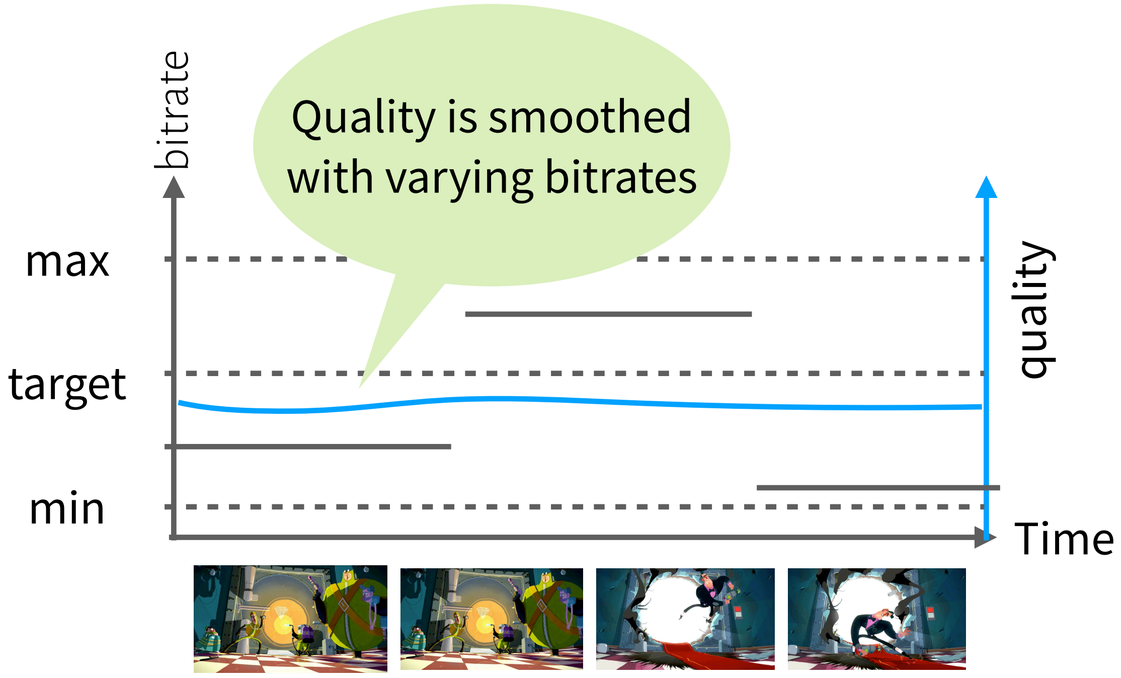


Variable bitrate (VBR)

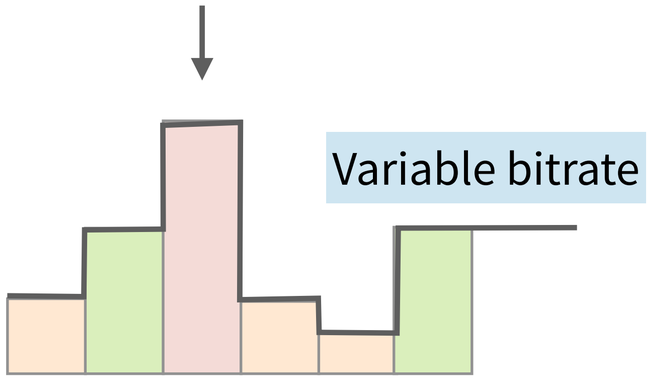
Encode video with varying bitrates

- Higher bitrate for more complex segments
- Smooth out the quality

VBR algorithms are more complex and typically require support from the hardware

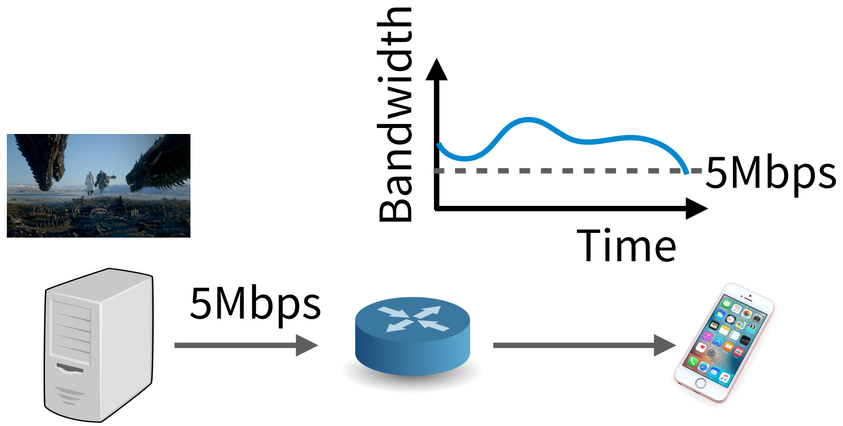


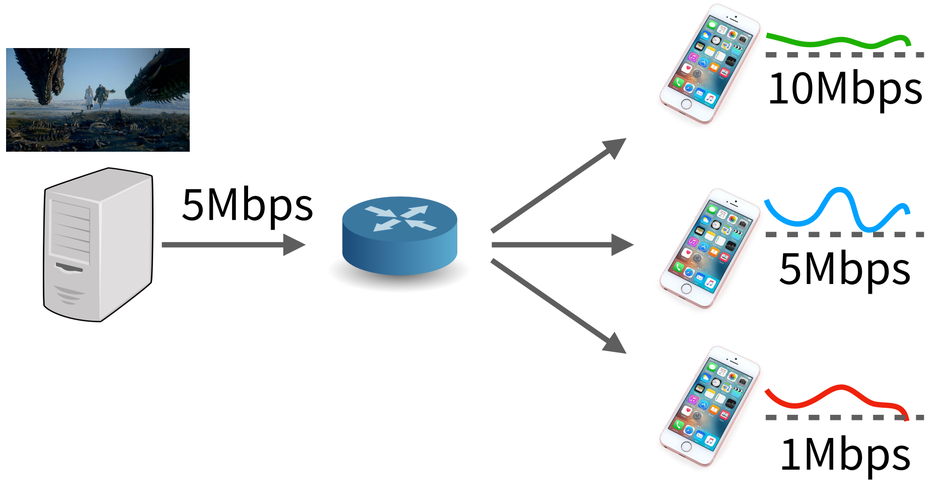
Utilize the space more flexibly for the entire video



Video streaming with CBR

CBR is suitable for video streaming since we know already the required bandwidth which is also constant over time

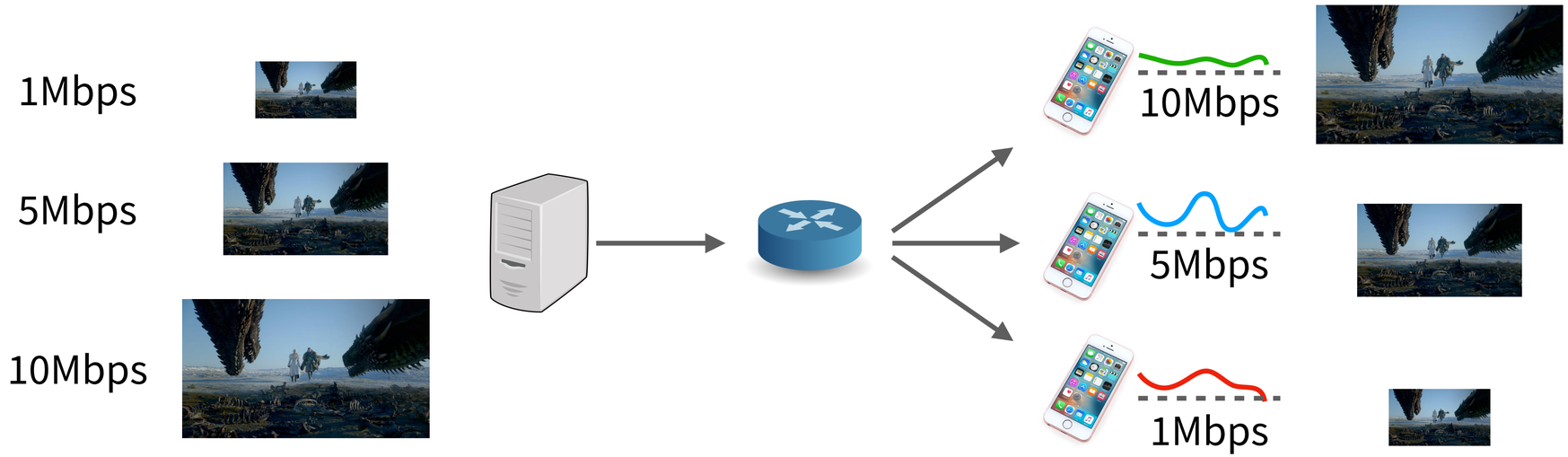




For a single user, CBR is sufficient, though not perfect

CBR is not efficient when multiple users with different bandwidth availabilities are present

CBR improvement

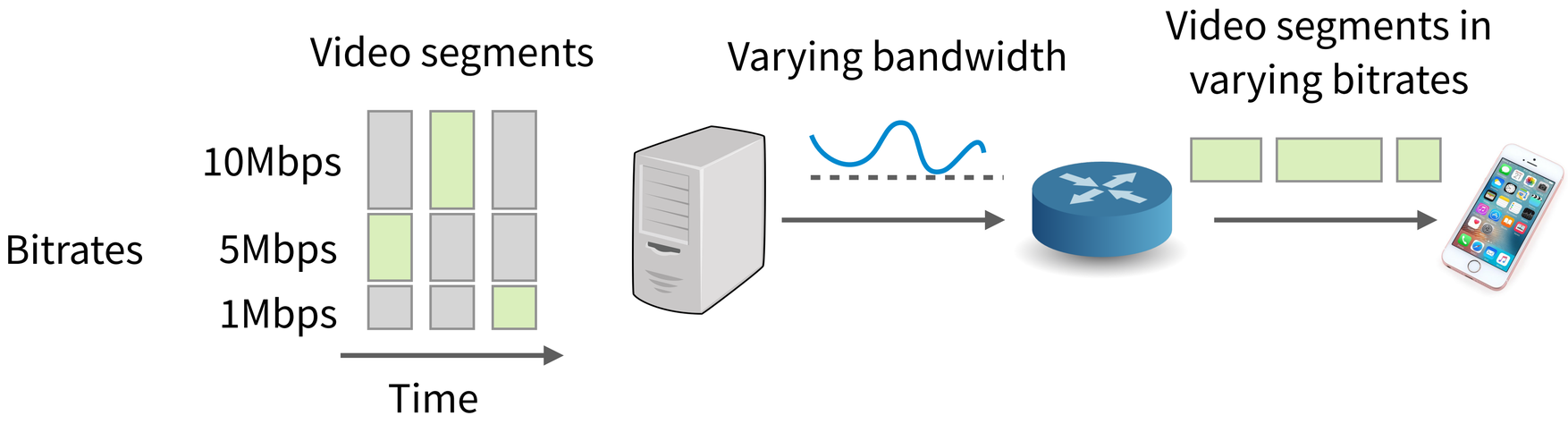


Encode the video with different CBRs at the streaming server and choose a suitable CBR based on the real-time bandwidth availability

Adaptive bitrate (ABR)

Main idea:

- **Chop the video into small segments** (chunks) and encode the segments with different bitrates
- **Adaptively select the bitrate for each segment** in streaming for each user



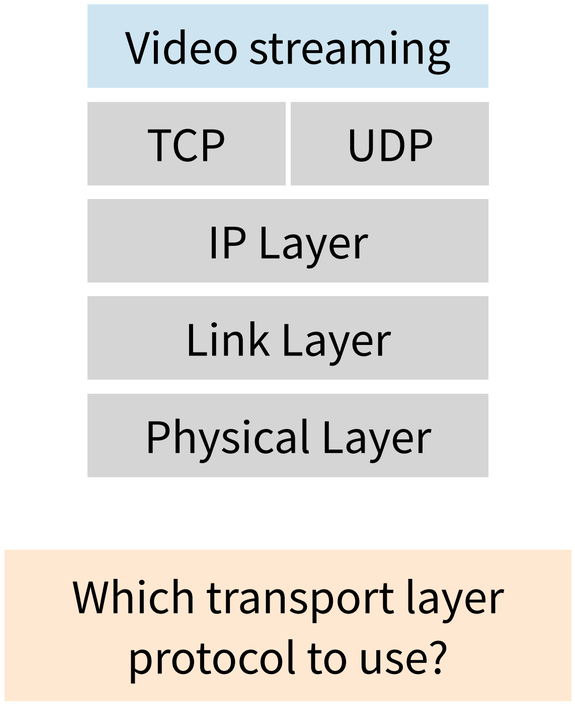
Challenges in video streaming

Streaming protocols

Adaptive bitrate selection algorithms

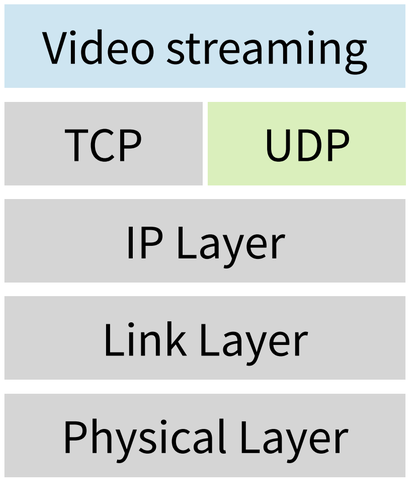
Streaming infrastructure management (e.g., CDN)

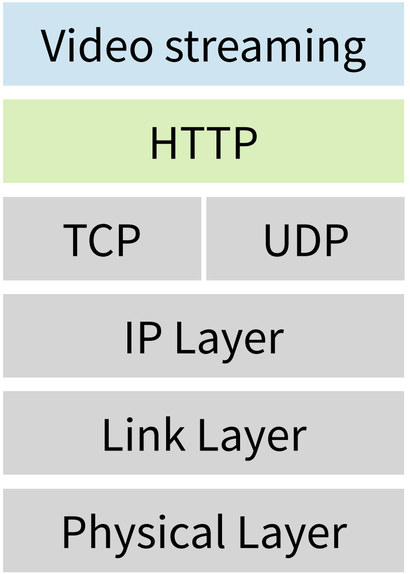
Video streaming protocols



Which transport layer protocol to use?

Video streaming protocols





Majority video streaming protocols are based on UDP in favor of timeliness instead of reliability

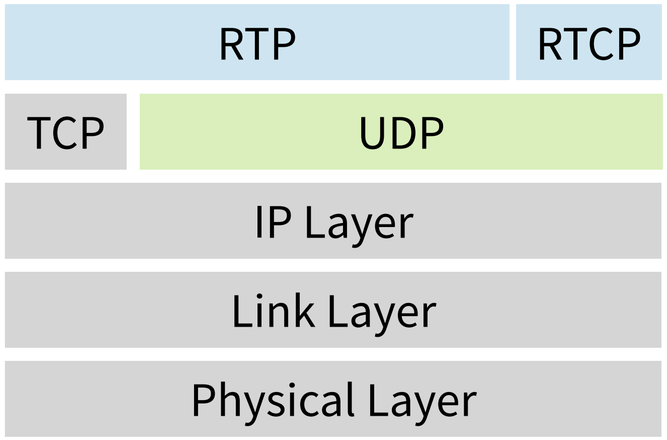
Modern video streaming protocols are based on HTTP

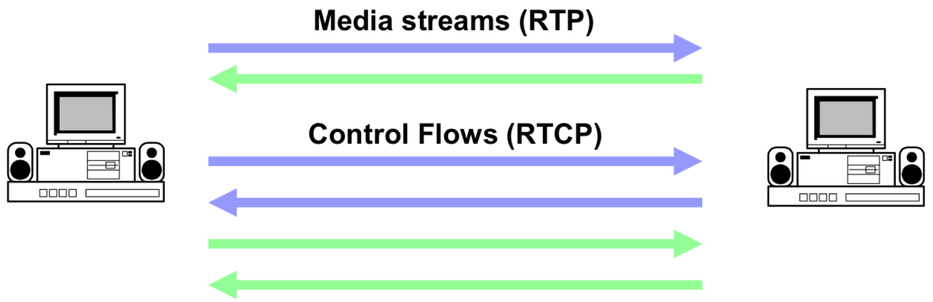
Video streaming protocol: RTP

RFC 1889 RFC 3550

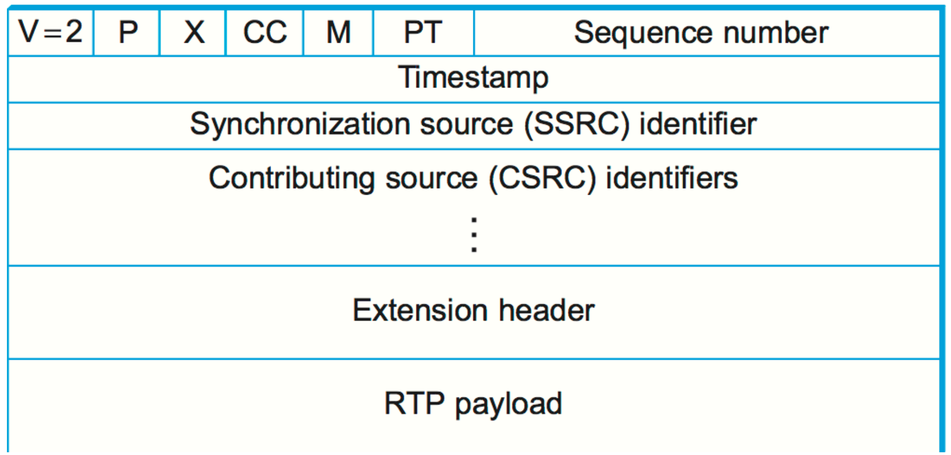
Real-time transport protocol: based on UDP

- Primary standard for audio/video transport in IP networks, widely used for **real-time multimedia applications** such as voice over IP, audio over IP, WebRTC (uses SRTP), and IP television
- Includes **timestamps** for synchronization, **sequence numbers** for packet loss and reordering detection
- Comes with a **control protocol, RTCP**, which is used for QoS feedback and synchronization between media streams, account for around 5% of total bandwidth usage





RTP packet header



- **Sequence number (16 bits):** used for packet loss detection or packet reordering, initially randomized
- **Timestamp (32 bits):** used by the receiver to play back the received samples at appropriate time and interval (e.g., use a clock of 90kHz for a video stream)
- **SSRC (32 bits):** uniquely identify the source of a stream
- **CSRC (32 bits):** enumerate contributing sources to a stream which has been generated from multiple sources

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	10.1.1.1	10.2.2.2	RTP	121	PT=DynamicRTP-Type-111, SSRC=0xC2B13255, Seq=19591, Time=2760404098
2	0.000037	10.2.2.2	10.1.1.1	RTP	121	PT=DynamicRTP-Type-111, SSRC=0xB5770A56, Seq=4305, Time=4131840
3	0.020622	10.1.1.1	10.2.2.2	RTP	131	PT=DynamicRTP-Type-111, SSRC=0xC2B13255, Seq=19592, Time=2760405058
4	0.020653	10.2.2.2	10.1.1.1	RTP	131	PT=DynamicRTP-Type-111, SSRC=0xB5770A56, Seq=4306, Time=4132800
5	0.025986	10.1.1.1	10.2.2.2	RTP	1190	PT=DynamicRTP-Type-96, SSRC=0x69E8BDC, Seq=24102, Time=3068471093
6	0.026109	10.1.1.1	10.2.2.2	RTP	1190	PT=DynamicRTP-Type-96, SSRC=0x69E8BDC, Seq=24103, Time=3068471093
7	0.026153	10.1.1.1	10.2.2.2	RTP	1190	PT=DynamicRTP-Type-96, SSRC=0x69E8BDC, Seq=24104, Time=3068471093
8	0.026290	10.1.1.1	10.2.2.2	RTP	1190	PT=DynamicRTP-Type-96, SSRC=0x69E8BDC, Seq=24105, Time=3068471093

Control in RTP: RTCP

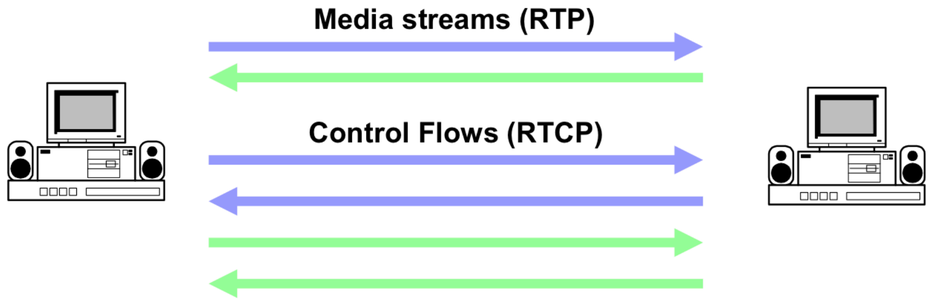
RFC

Receiver constantly measure transmission quality

- Delay, jitter, packet loss, RTT

Regular control information exchange between senders and receivers

- Feedback to sender (receiver report)
- Feed forward to recipients (sender report)



Allow applications to adapt to current QoS

- Limiting a flow or using a different codec

Limited overhead: a small fraction, e.g., 5% max. of total bandwidth per RTP session

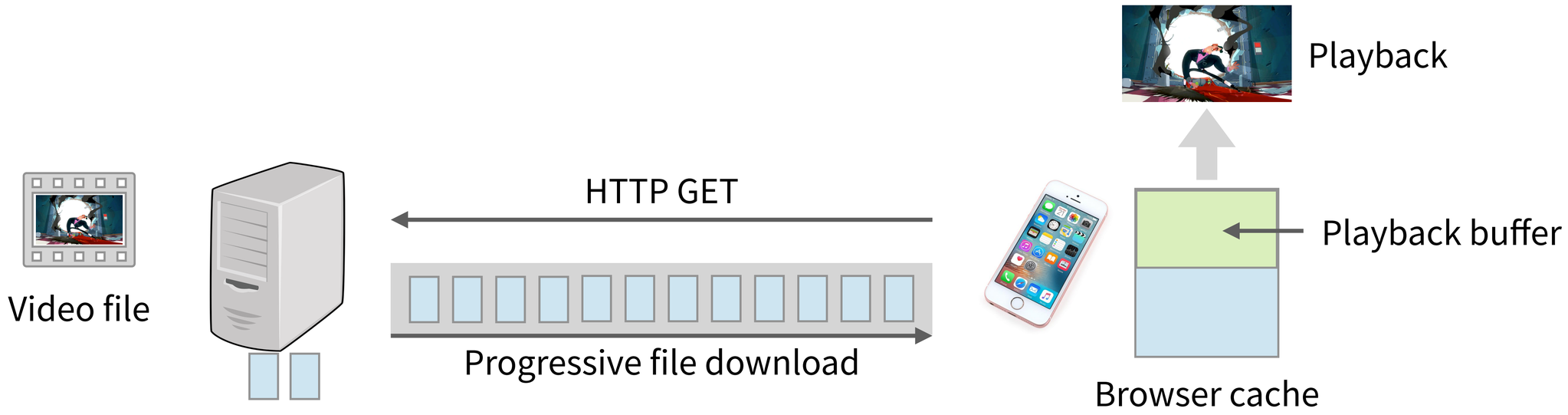
RTP/RTCP has no support for ABR!

Modern video streaming protocols are based on HTTP

HTTP 1.1+ supports **progressive download**

- Prevalent form of web-based media delivery for video share sites
- Progressive = playback begins while download is in progress (byte range request)

HTTP is friendly with the middleboxes (e.g., firewalls) on the network



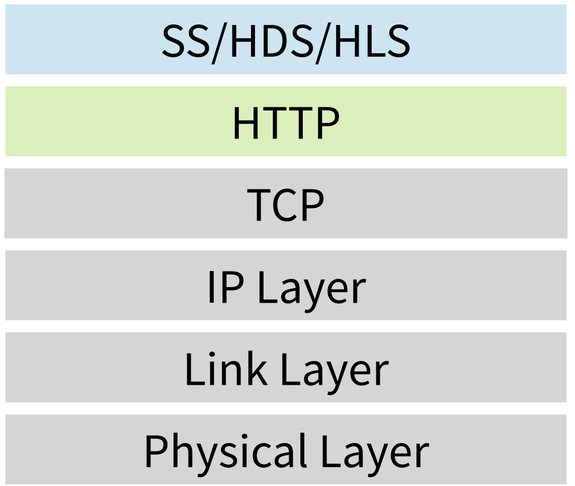
Video streaming protocols based on HTTP

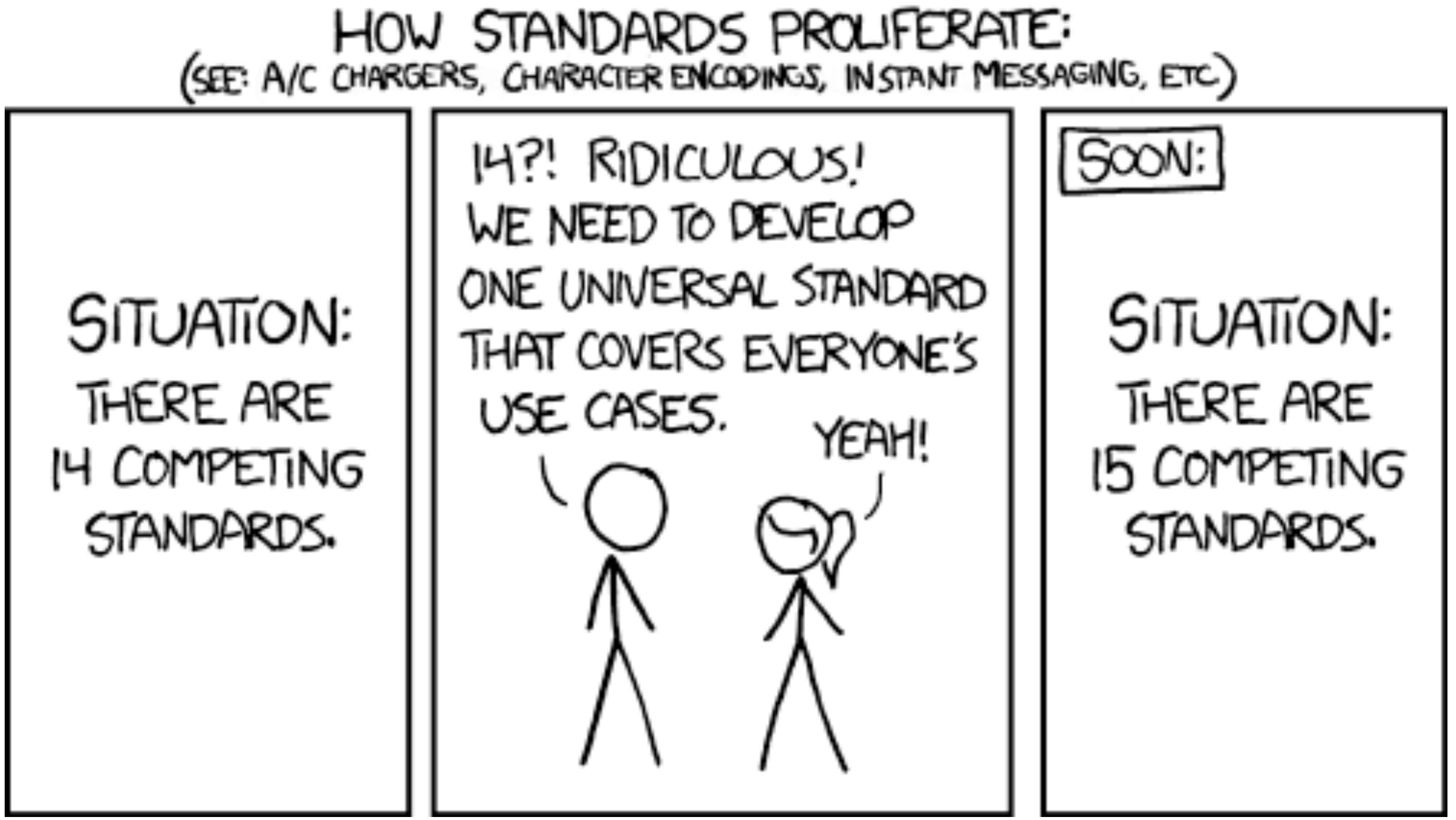
Three major players

- Microsoft Smooth Streaming
- Adobe HTTP Dynamic Streaming (HDS)
- Apple HTTP Live Streaming (HLS)

Each has a **proprietary format** and its own ecosystem

Bad for the industry such as CDN providers like Akamai since every functionality has to be implemented three times




HOW STANDARDS PROLIFERATE:
(SEE: A/C CHARGERS, CHARACTER ENCODINGS, INSTANT MESSAGING, ETC)
SITUATION: THERE ARE 14 COMPETING STANDARDS.
14?! RIDICULOUS! WE NEED TO DEVELOP ONE UNIVERSAL STANDARD THAT COVERS EVERYONE'S USE CASES.
YEAH!
SOON:
SITUATION: THERE ARE 15 COMPETING STANDARDS.

Yet another standard: MPEG-DASH

Dynamic adaptive streaming over HTTP (DASH) is an ISO standard for the adaptive delivery of segmented content

- Blending existing formats into a new format

MPEG (moving pictures experts group)

- Standardized MP3, MP4

Standardization work from 2010-2012

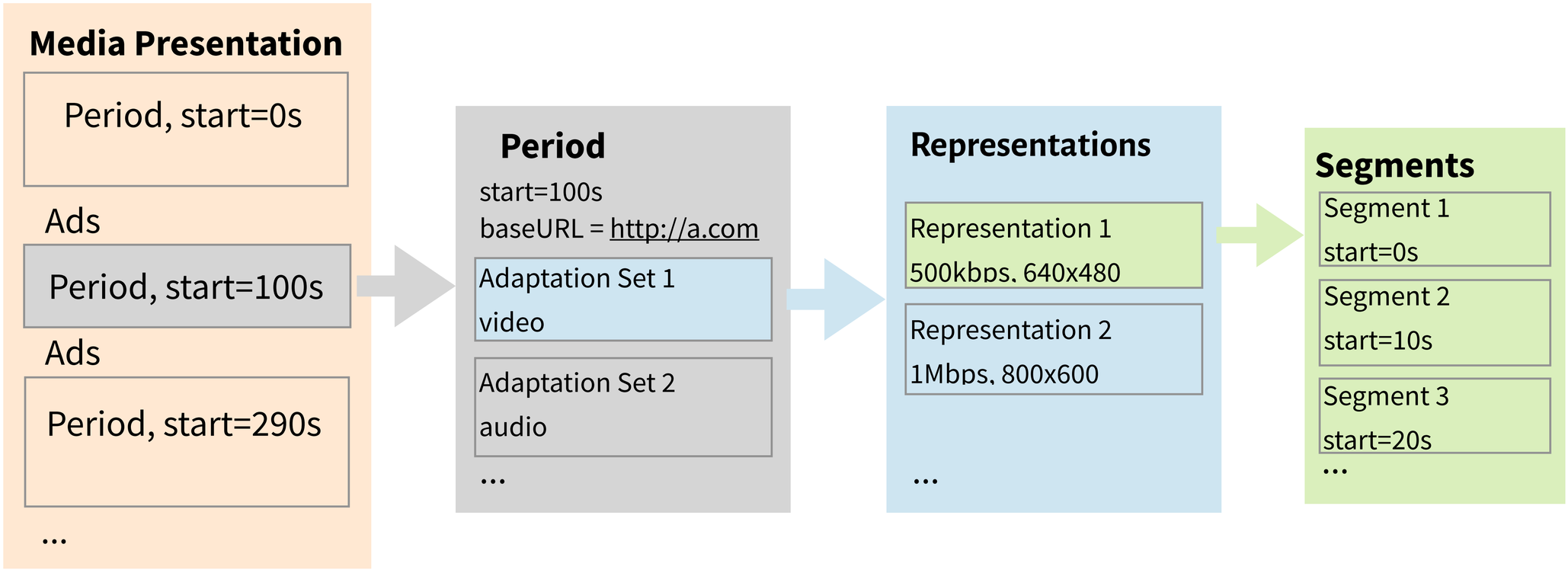
Note: DASH itself is not a protocol

mpeg-DASH

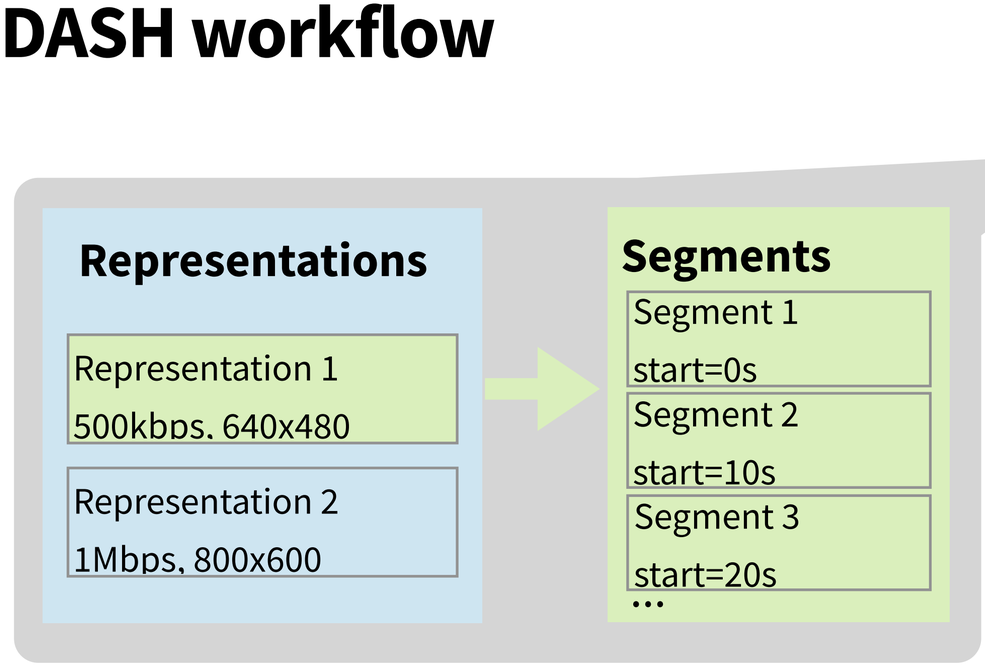
DASH: data model

MDP (media presentation description) describes accessible segments and corresponding timing

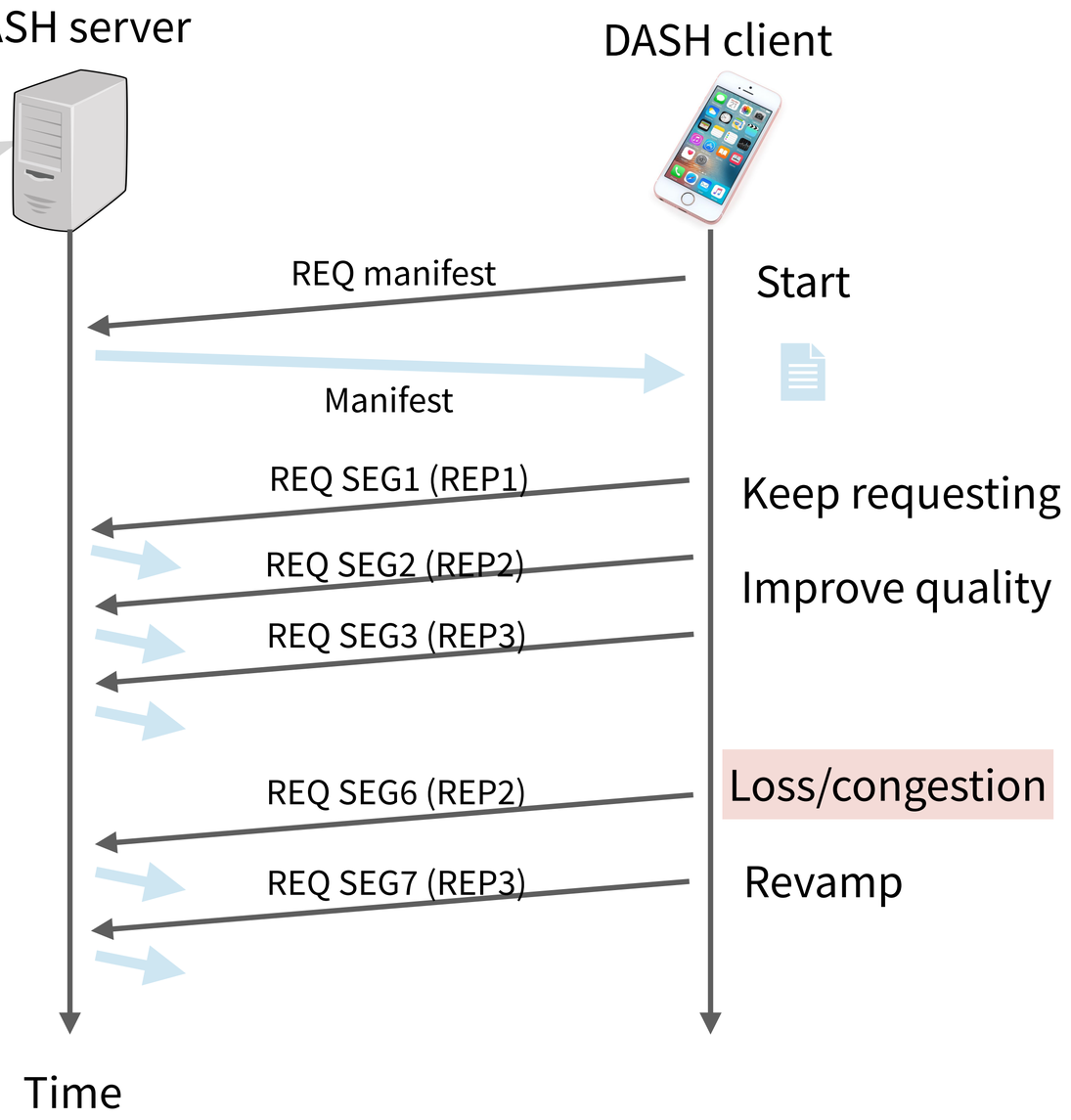
- Ensuring interoperability



DASH workflow



Video file is encoded using the MDP data model described with a **manifest** file



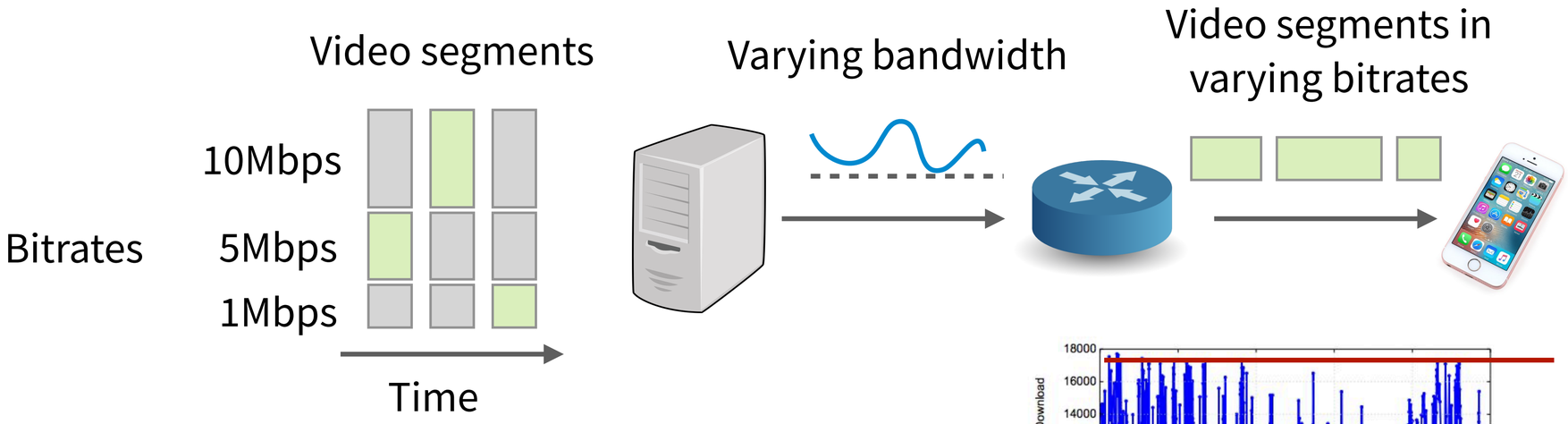
Challenges in video streaming

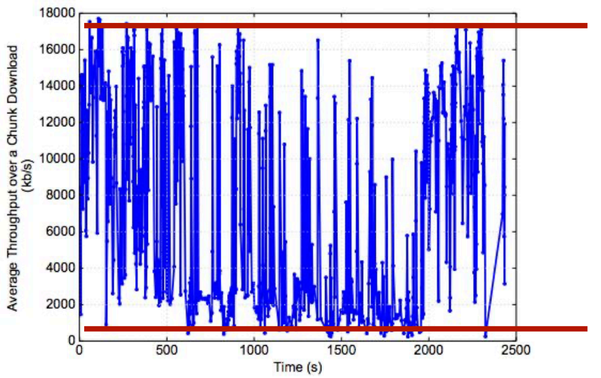
Streaming protocols

Adaptive bitrate selection algorithms

Streaming infrastructure management (e.g., CDN)

Bitrate selection in ABR



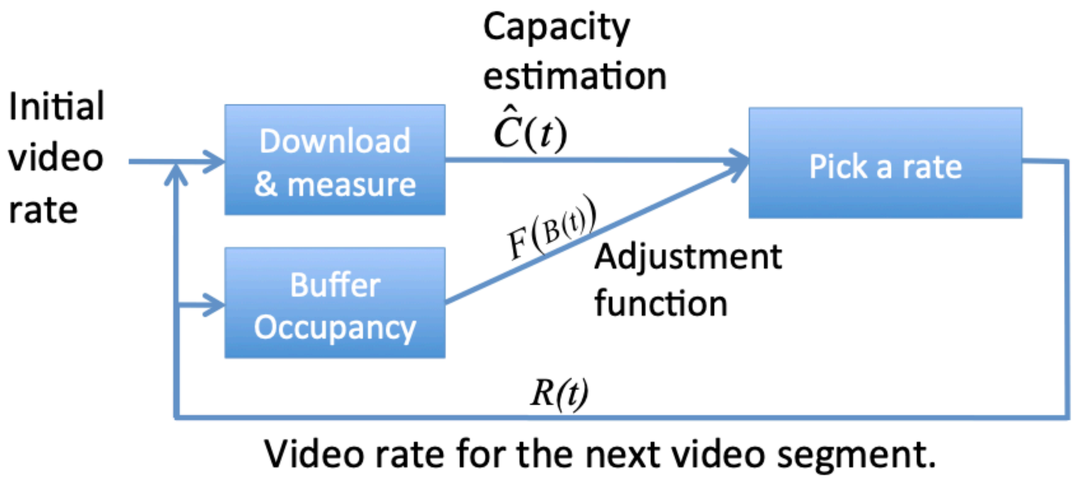


ABR algorithm: maximize the video quality and minimize rebuffing events

The most straightforward approach is to perform bandwidth estimation

Challenge: bandwidth variation can be very high! (10% of all sessions experience such variation and 22% experience at least half of such variation)

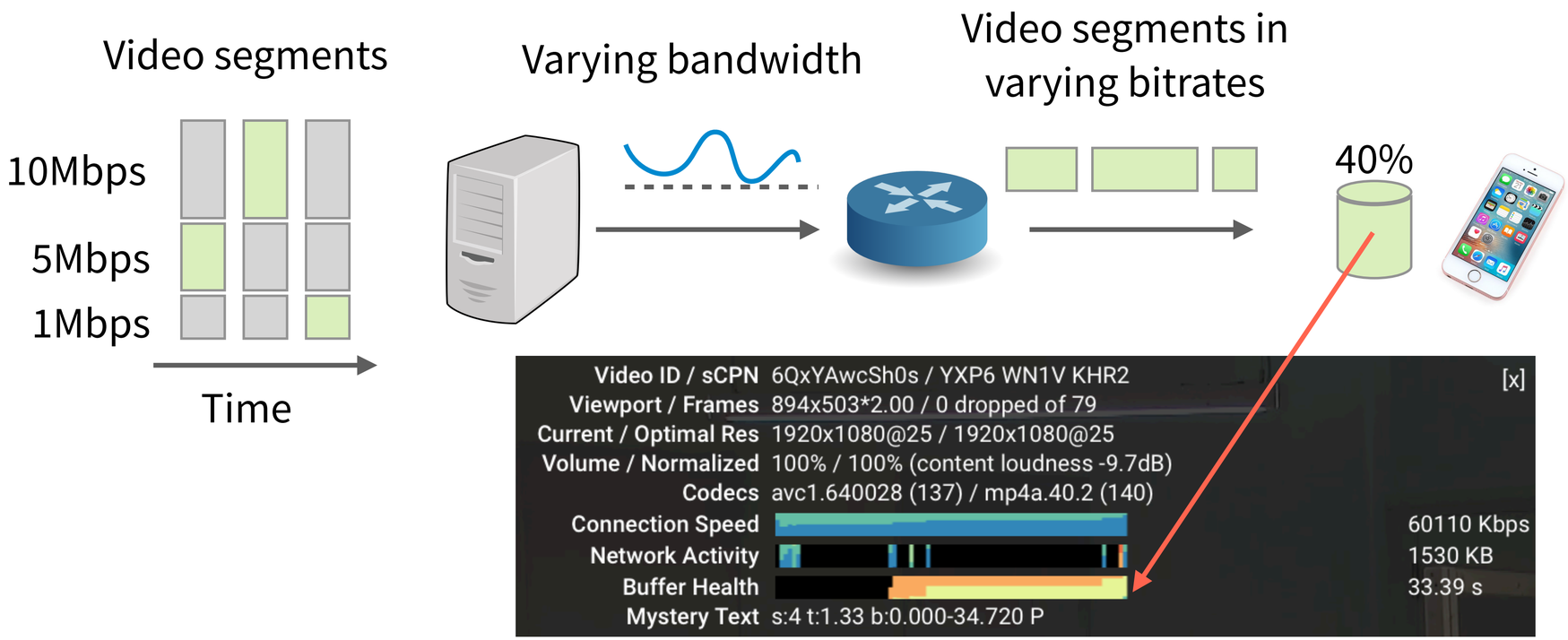
Existing rate-based approaches



Existing rate-based approaches are mainly based on bandwidth estimation, with an "adjustment" based on the current level of the playback buffer. However, deciding the adjustment function is challenging.

However, using buffer occupancy is suggestive: the buffer is the **exact state variable** an ABR algorithm tries to control.

ABR algorithm: buffer-based



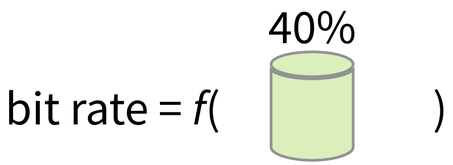
Can we make ABR decisions solely based on the buffer occupancy at the client?

ABR algorithm: buffer-based

Main **motivation**

- Avoid bandwidth estimation
- Buffer occupancy contains implicit information about the bandwidth

BBA (buffer-based algorithm): pick the bitrate based on a function of buffer occupancy



A Buffer-Based Approach to Rate Adaptation: Evidence from a Large Video Streaming Service

Te-Yuan Huang, Ramesh Johari, Nick McKeown, Matthew Trunnell*, Mark Watson*
Stanford University, Netflix*
{huangty,rjohari,nickm}@stanford.edu, {mtrunnell,watsonm}@netflix.com

ABSTRACT

Existing ABR algorithms face a significant challenge in estimating future capacity: capacity can vary widely over time, a phenomenon commonly observed in commercial services. In this work, we suggest an alternative approach: rather than presuming that capacity estimation is required, it is perhaps better to begin by using *only* the buffer, and then ask *when* capacity estimation is needed. We test the viability of this approach through a series of experiments spanning millions of real users in a commercial service. We start with a simple design which directly chooses the video rate based on the current buffer occupancy. Our own investigation reveals that capacity estimation is unnecessary in steady state; however using simple capacity estimation (based on immediate past throughput) is important during the startup phase, when the buffer itself is growing from empty. This approach allows us to reduce the rebuffer rate by 10–20% compared

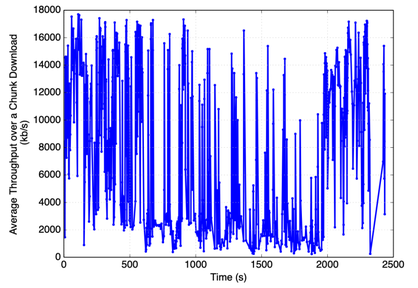
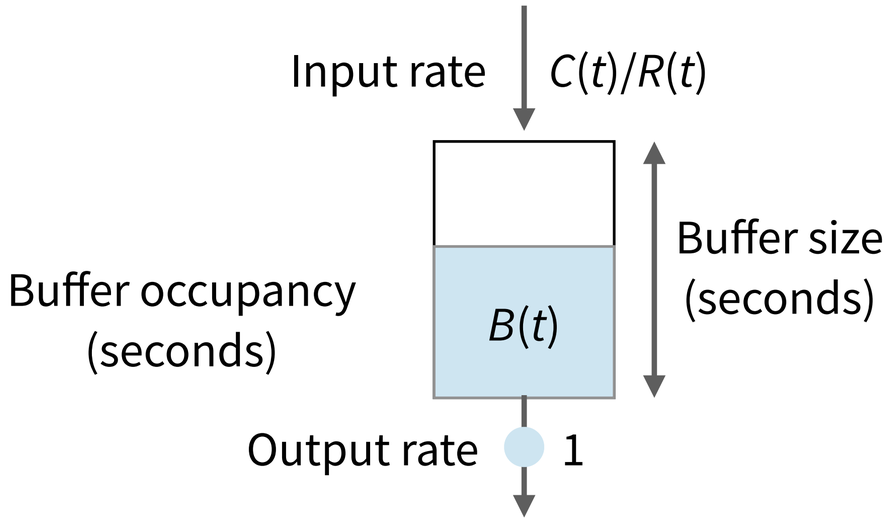


Figure 1: Video streaming clients experience highly variable end-to-end throughput.

ACM SIGCOMM 2014

BBA: system model



We use the **unit of video seconds:** representing how many seconds of video we can fetch/buffer

System dynamics

$C(t)/R(t) > 1$: buffer $B(t)$ grows

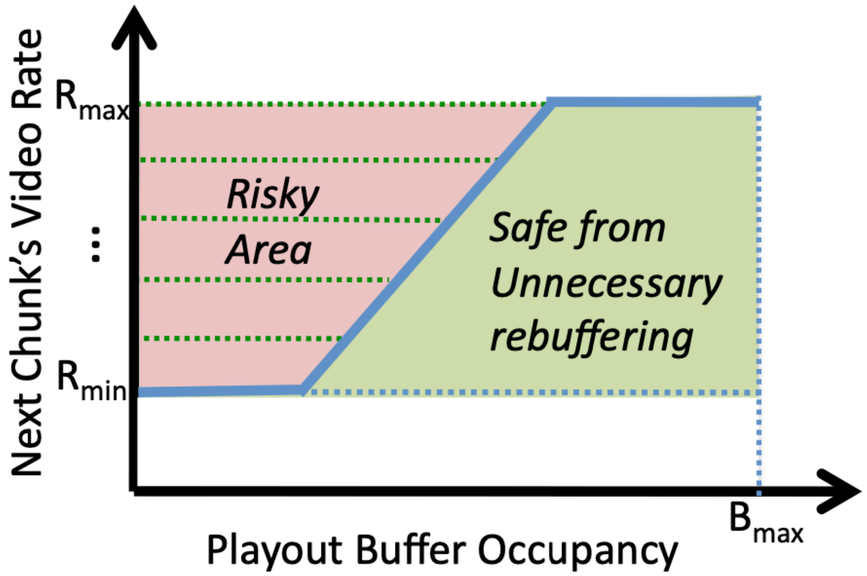
- At a certain point, it is safe to increase $R(t)$ to improve the streaming quality

$C(t)/R(t) < 1$: buffer $B(t)$ drains

- Arrival rate is smaller than 1 second of video
- The chosen rate $R(t)$ is **too high**
- Buffer will be depleted and “rebuffering” happens

Question: find a good function $R(t) = f(B(t))$

BBA: theoretical analysis



Assumptions: infinitesimal segment size, continuous bit rate, videos are CBR coded, videos are infinitely long

Goal 1: no unnecessary rebuffering

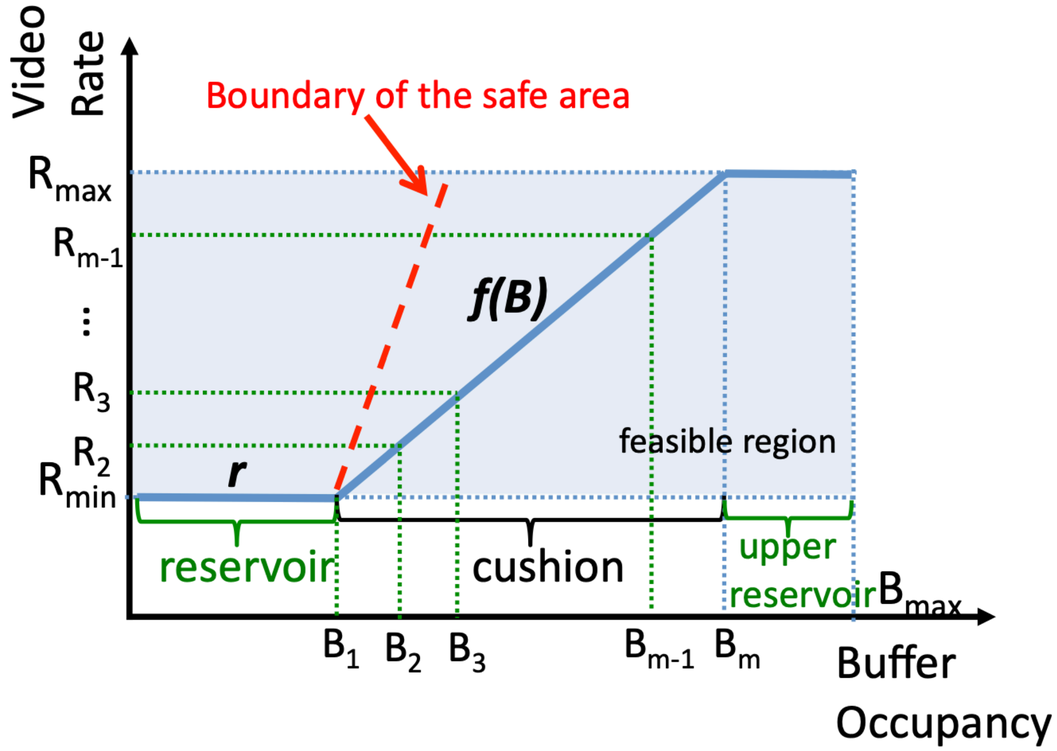
- As long as $C(t) > R_{min}$ for all t and we adapt $f(B) \rightarrow R_{min}$ as $B \rightarrow 0$, we will never unnecessarily rebuffer because the buffer will start to grow before it runs dry

Goal 2: average video rate maximization

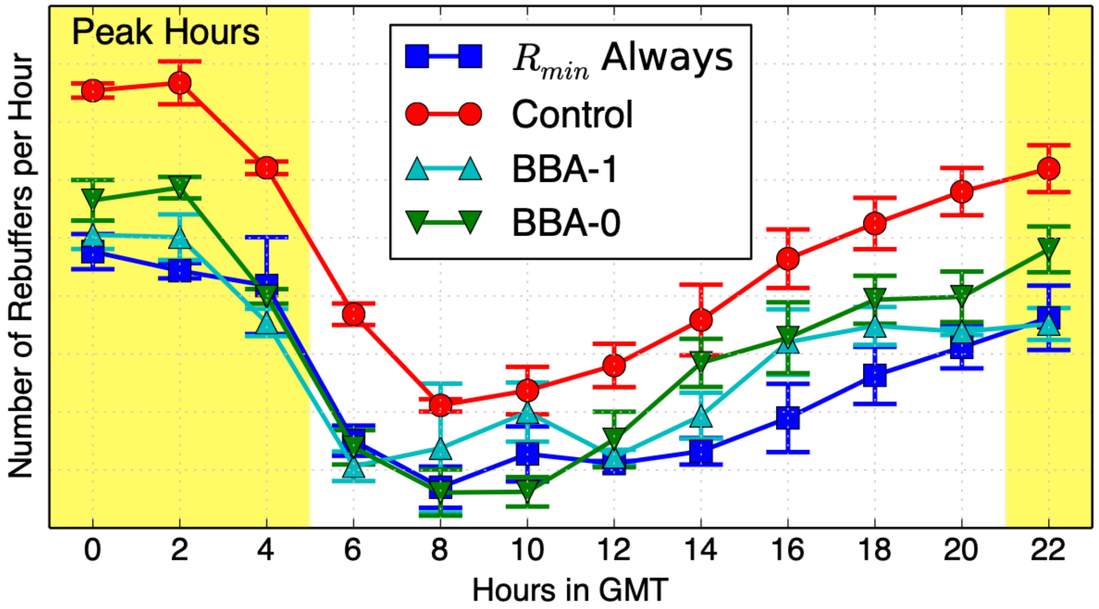
- As long as $f(B)$ is increasing and eventually reaches R_{max}, the average video rate will match the average capacity when $R_{min} < C(t) < R_{max}$ for all $t > 0$

BBA in practice

Assumptions do not always hold in practice, we need to be **more conservative**

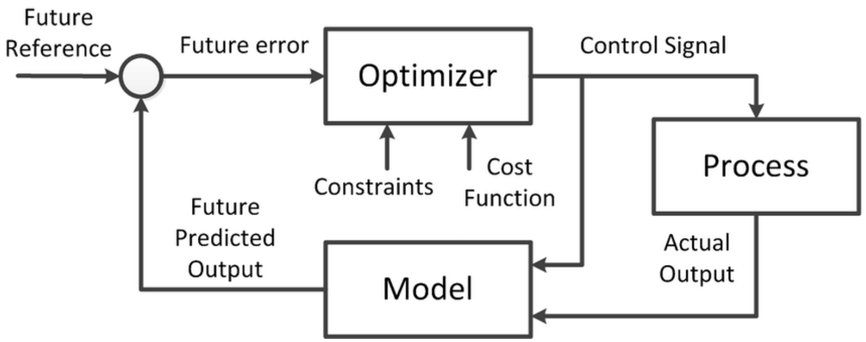


BBA: results



BBA achieves much less rebuffering rate than alternatives.

ABR algorithm: control theory based



Model the ABR control problem as Markov processes and apply control theory

A Control-Theoretic Approach for Dynamic Adaptive Video Streaming over HTTP

Xiaoqi Yin, Abhishek Jindal, Vyas Sekar, Bruno Sinopoli
Carnegie Mellon University
{yinxiaoqi522, abhishekjindal93}@gmail.com, {vsekar,brunos}@andrew.cmu.edu

ABSTRACT

User-perceived quality-of-experience (QoE) is critical in Internet video applications as it impacts revenues for content providers and delivery systems. Given that there is little support in the network for optimizing such measures, bottlenecks could occur anywhere in the delivery system. Consequently, a robust bitrate adaptation algorithm in client-side players is critical to ensure good user experience. Previous studies have shown key limitations of state-of-art commercial solutions and proposed a range of heuristic fixes. Despite the emergence of several proposals, there is still a distinct lack of consensus on: (1) How best to design this client-side bitrate adaptation logic (e.g., use rate estimates vs. buffer occupancy); (2) How well specific classes of approaches will perform under diverse operating regimes (e.g., high throughput variability); or (3) How do they actually balance different QoE objectives (e.g., startup delay vs. re-

1 Introduction

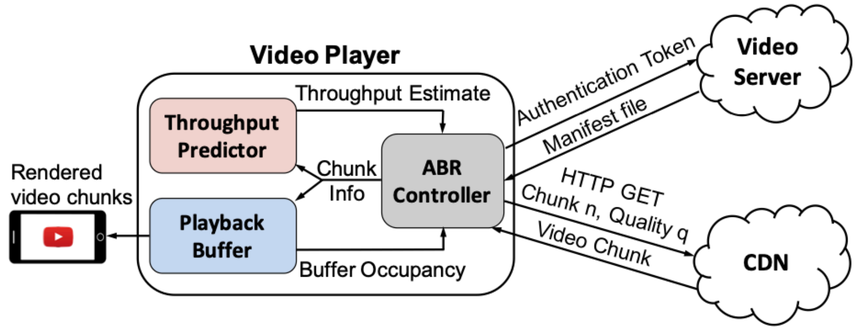
Many recent studies have highlighted the critical role that user-perceived quality-of-experience (QoE) plays in Internet video applications, as it ultimately affects revenue streams for content providers [24, 35]. Specifically, metrics such as the duration of rebuffering (i.e., the player's playout buffer does not have content to render), startup delay (i.e., the lag between the user clicking vs. the time to begin rendering), the average playback bitrate, and the variability of the bitrate delivered have emerged as key factors.

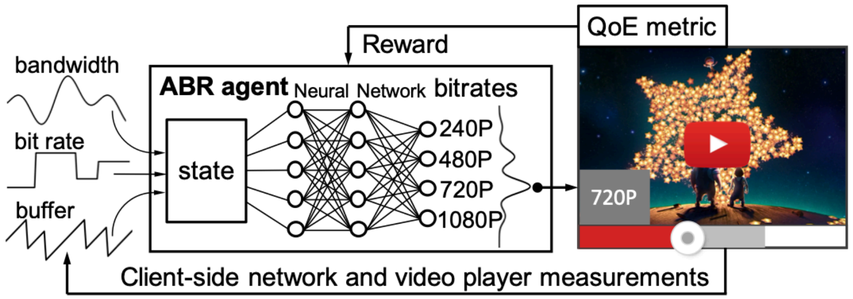
Given the complex Internet video delivery ecosystem and presence of diverse bottlenecks, the *bitrate adaptation logic* in the client-side video player becomes critical to optimize user experience [16]. In the HTTP-based delivery model that predominates today [44], videos are typically chunked and encoded at different bitrate levels. The goal of an adaptive video player is to choose the bitrate level for future chunks

ACM SIGCOMM 2015

ABR algorithm: learning based

We will discuss it next week.





Model the ABR control problem as a Markov Decision Process and apply deep reinforcement learning

Neural Adaptive Video Streaming with Pensieve

Hongzi Mao, Ravi Netravali, Mohammad Alizadeh
MIT Computer Science and Artificial Intelligence Laboratory
{hongzi,ravinet,alizadeh}@mit.edu

ABSTRACT

Client-side video players employ adaptive bitrate (ABR) algorithms to optimize user quality of experience (QoE). Despite the abundance of recently proposed schemes, state-of-the-art ABR algorithms suffer from a key limitation: they use fixed control rules based on simplified or inaccurate models of the deployment environment. As a result, existing schemes inevitably fail to achieve optimal performance across a broad set of network conditions and QoE objectives.

We propose Pensieve, a system that generates ABR algorithms using reinforcement learning (RL). Pensieve trains a neural network model that selects bitrates for future video chunks based on observations collected by client video players. Pensieve does not rely on pre-programmed models or assumptions about the environment. Instead, it learns to make ABR decisions solely through observations of the resulting performance of past decisions. As a result, Pensieve automatically learns ABR algorithms that adapt to a wide range of environments and QoE metrics. We compare Pensieve to state-of-the-art ABR algorithms using trace-driven and real world experiments spanning a wide variety of network conditions, QoE metrics, and video properties. In all considered scenarios, Pensieve outperforms content providers [12, 25]. Nevertheless, content providers continue to struggle with delivering high-quality video to their viewers.

Adaptive bitrate (ABR) algorithms are the primary tool that content providers use to optimize video quality. These algorithms run on client-side video players and dynamically choose a bitrate for each video *chunk* (e.g., 4-second block). ABR algorithms make bitrate decisions based on various observations such as the estimated network throughput and playback buffer occupancy. Their goal is to maximize the user's QoE by adapting the video bitrate to the underlying network conditions. However, selecting the right bitrate can be very challenging due to (1) the variability of network throughput [18, 42, 49, 52, 53]; (2) the conflicting video QoE requirements (high bitrate, minimal rebuffering, smoothness, etc.); (3) the cascading effects of bitrate decisions (e.g., selecting a high bitrate may drain the playback buffer to a dangerous level and cause rebuffering in the future); and (4) the coarse-grained nature of ABR decisions. We elaborate on these challenges in §2.

The majority of existing ABR algorithms (§7) develop fixed control rules for making bitrate decisions based on estimated network throughput ("rate-based" algorithms [21, 42]), playback buffer size

ACM SIGCOMM 2017

Challenges in video streaming

Streaming protocols

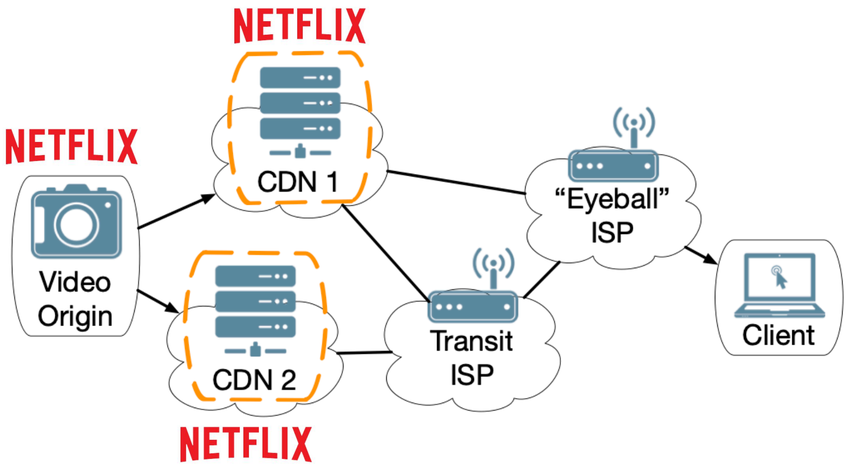
Adaptive bitrate selection algorithms

Streaming infrastructure management (e.g., CDN)

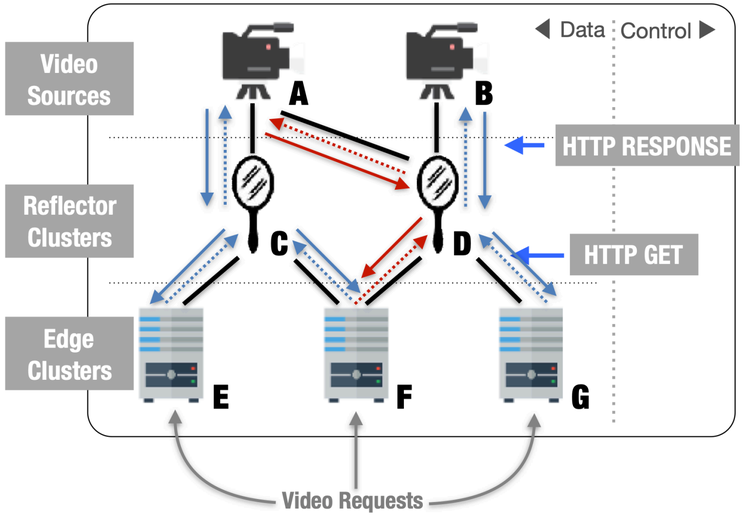
How to deliver video on the network?

Modern infrastructures use content delivery networks (CDNs) to localize traffic

- Caching only possible for VoD (e.g., Netflix, YouTube)
- No caching for live video delivery (e.g., Twitch), live video mega events (e.g., World Cup)

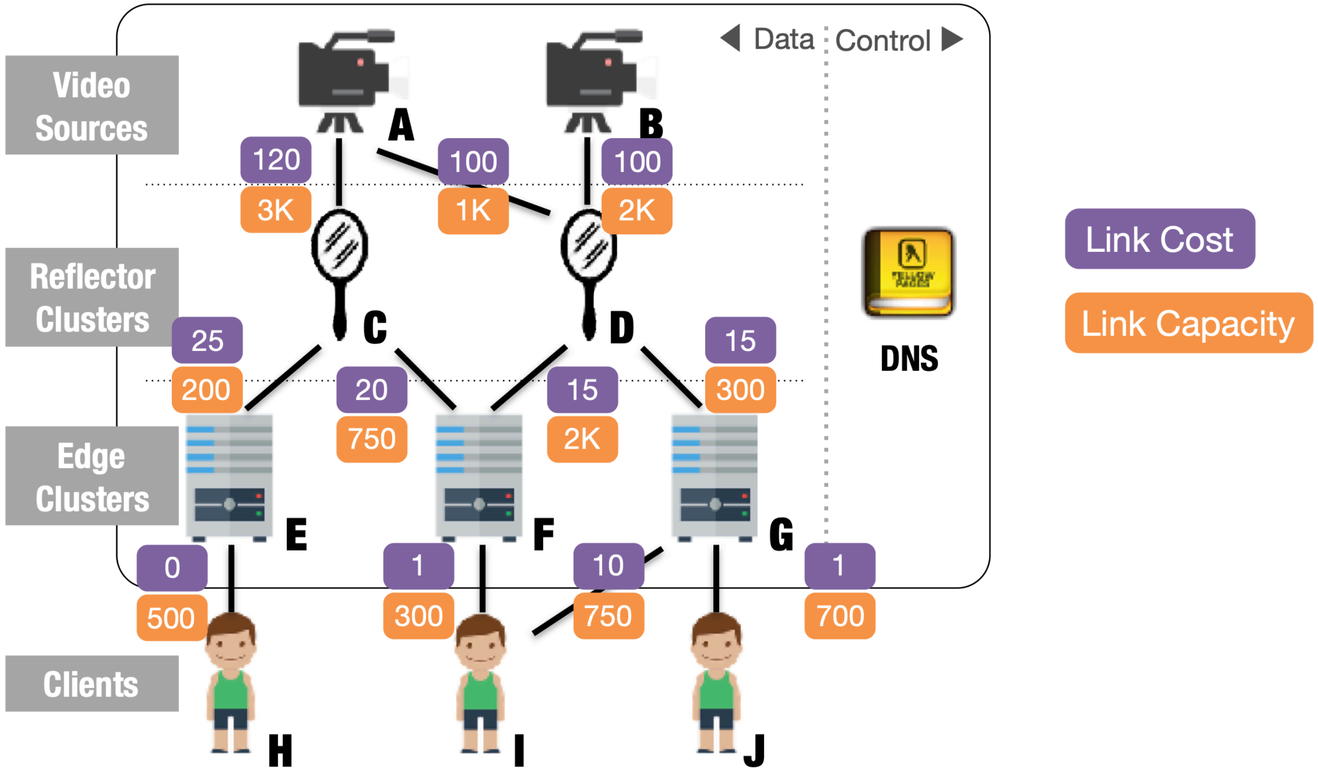


CDN caching for VoD



No caching for live video delivery

Clients to video sources assignments



Two general goals: (1) maximize **service quality** (bitrate per client), and (2) minimize **delivery cost**

Problems with CDNs today

Low service quality

- A big service quality gap between the current CDN and the optimal

High delivery cost

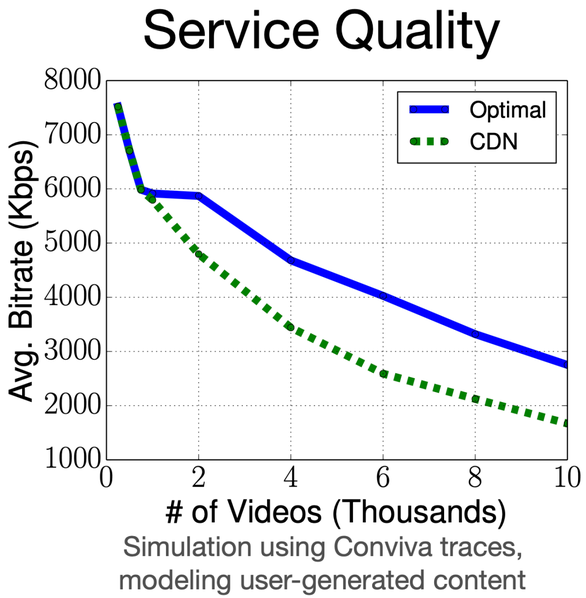
- Current CDNs use almost 2x the cost per request than the optimal cost

Control is not fine-grained

- Control is done by aggregating videos into large groups

Slow DNS updates

- Cannot "push" updates, DNS entries get cached



Simulation using Conviva traces, modeling user-generated content

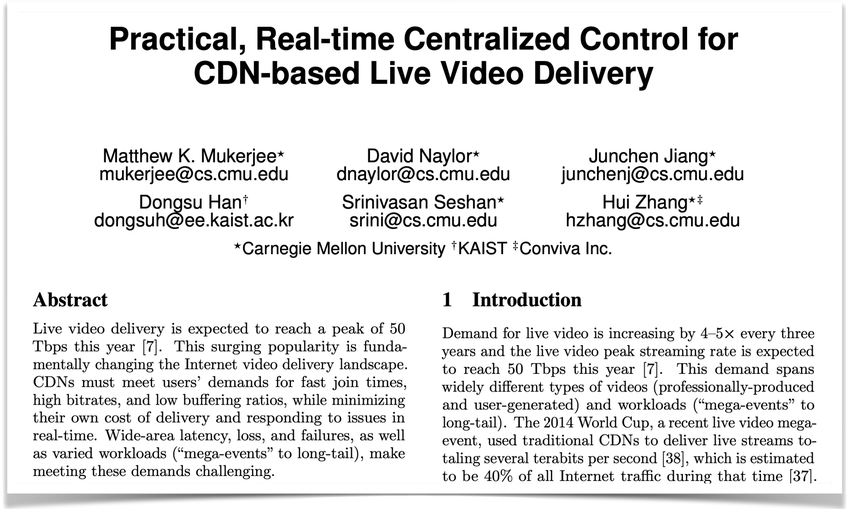
Motivation for a new design

Goals of Video Delivery Network (**VDN**)

- Achieve optimal service quality
- Achieve optimal delivery cost
- Fine-grained control: per-video control
- Real-time response: sub-second response to failures and joins

How to achieve these goals?

- **Centralization optimization** with **distributed control**
- Some insights are already shedded in a previous study: Liu et al., A case for coordinated Internet video control plane, SIGCOMM 2012.



Practical, Real-time Centralized Control for CDN-based Live Video Delivery

Matthew K. Mukerjee* mukerjee@cs.cmu.edu
David Naylor* dnaylor@cs.cmu.edu
Junchen Jiang* junchenj@cs.cmu.edu
Dongsu Han† dongsuh@ee.kaist.ac.kr
Srinivasan Seshan* srini@cs.cmu.edu
Hui Zhang*‡ hzhang@cs.cmu.edu
*Carnegie Mellon University †KAIST ‡Conviva Inc.

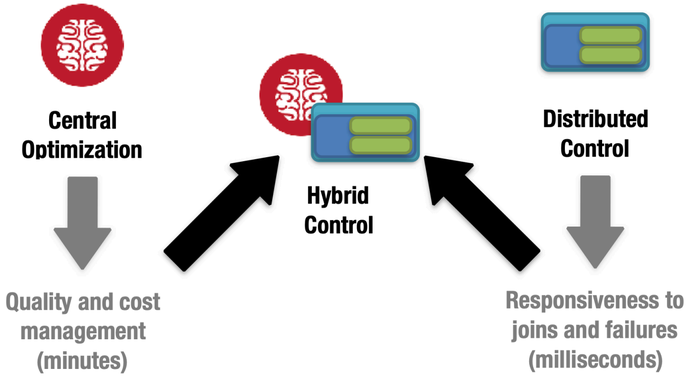
Abstract

Live video delivery is expected to reach a peak of 50 Tbps this year [7]. This surging popularity is fundamentally changing the Internet video delivery landscape. CDNs must meet users' demands for fast join times, high bitrates, and low buffering ratios, while minimizing their own cost of delivery and responding to issues in real-time. Wide-area latency, loss, and failures, as well as varied workloads ("mega-events" to long-tail), make meeting these demands challenging.

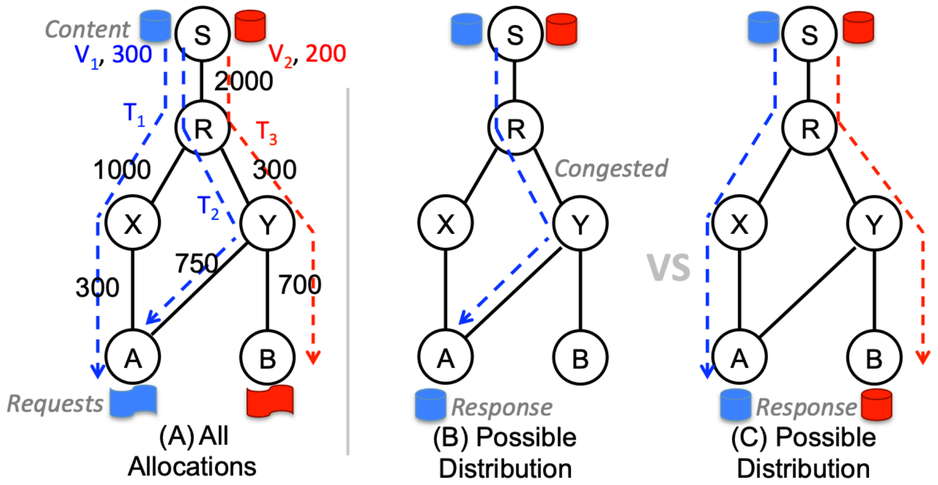
1 Introduction

Demand for live video is increasing by 4–5× every three years and the live video peak streaming rate is expected to reach 50 Tbps this year [7]. This demand spans widely different types of videos (professionally-produced and user-generated) and workloads ("mega-events" to long-tail). The 2014 World Cup, a recent live video mega-event, used traditional CDNs to deliver live streams totaling several terabits per second [38], which is estimated to be 40% of all Internet traffic during that time [37].

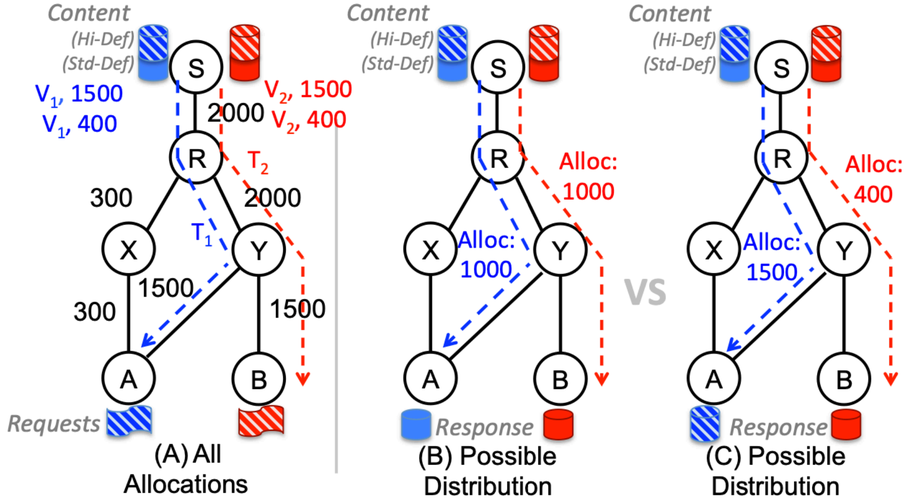
ACM SIGCOMM 2015



Why we need centralized optimization?



Local decision making leads to congested network and less satisfied client requests



Local decision making leads to poor bitrate for all clients in the name of fairness

Centralized optimization

MAXIMIZE **SERVICE QUALITY**

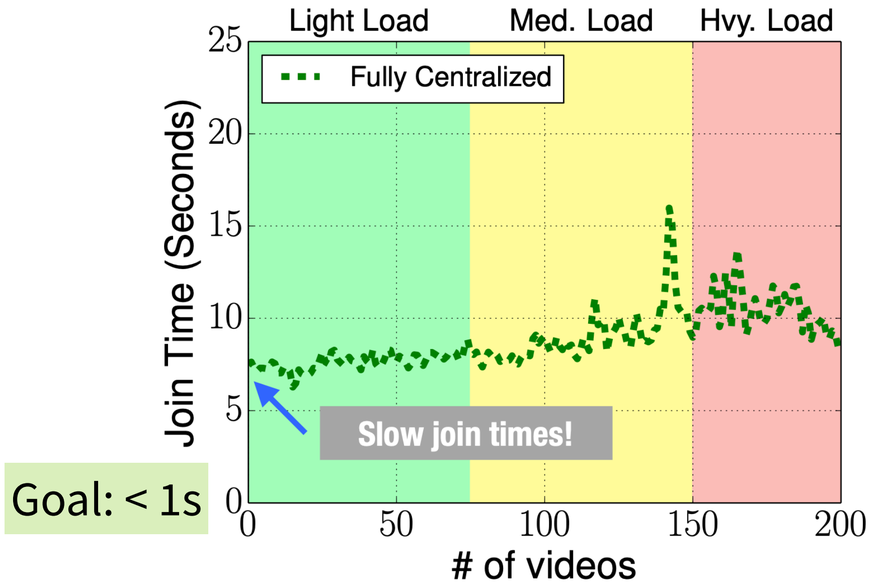
MINIMIZE **DELIVERY COST**

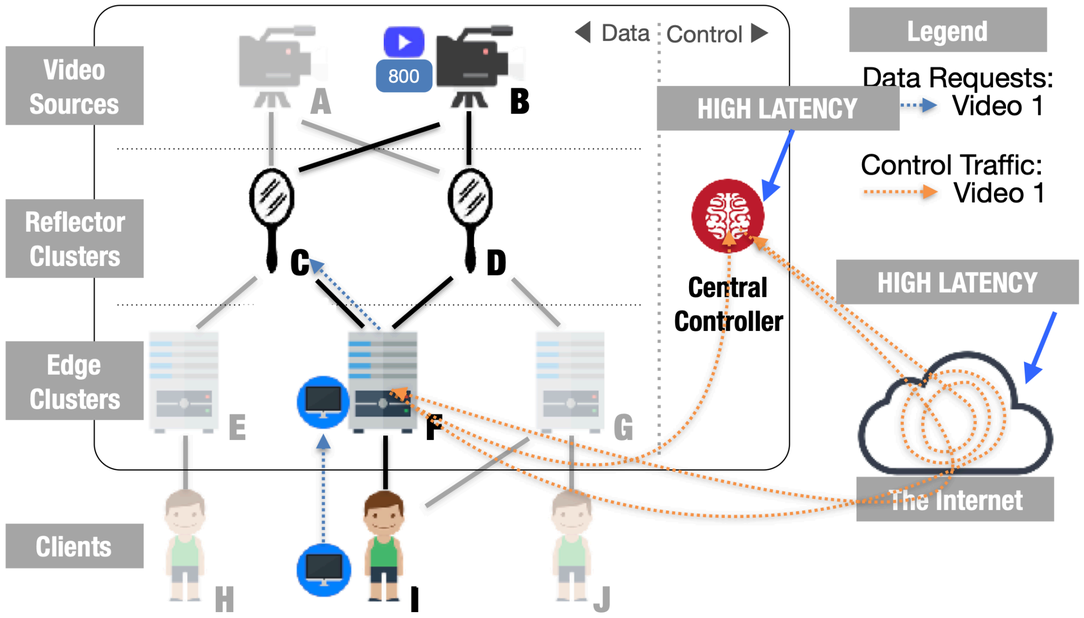
SUBJECT TO **DON'T EXCEED LINK CAPACITY**

SENDER MUST HAVE RECEIVED VIDEO

Formulate an **Integer Linear Program** and solve it to obtain the optimal decisions

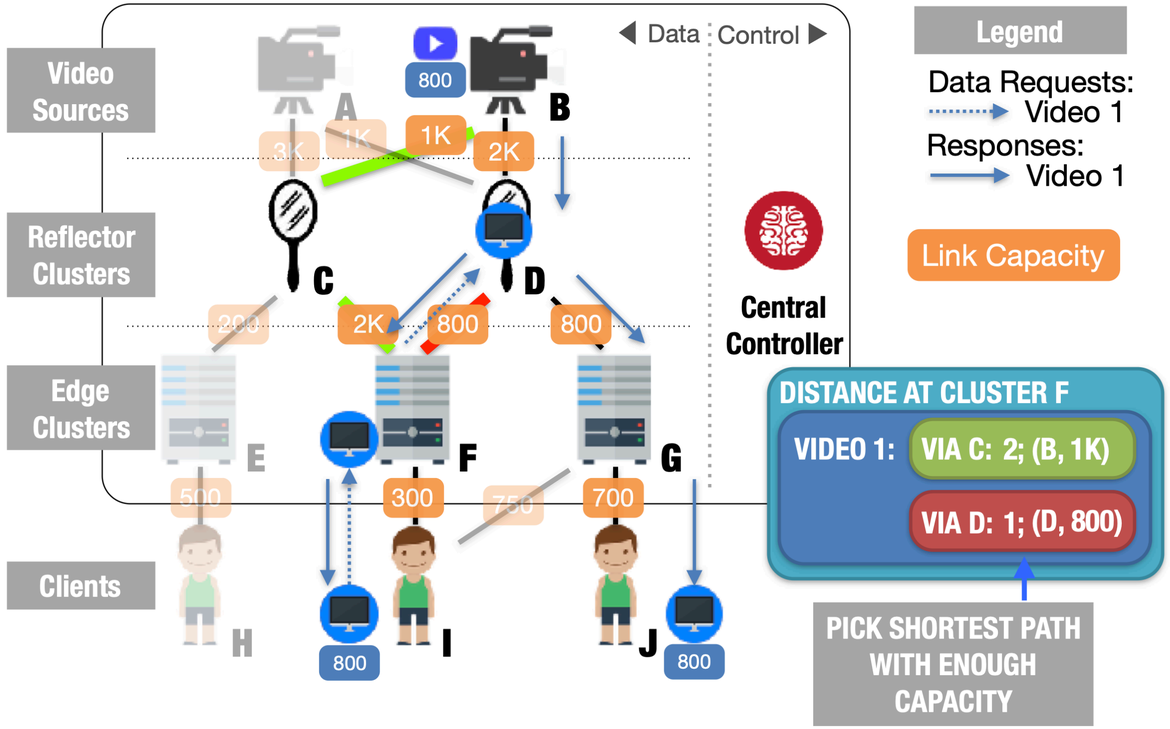
Problems with centralization: slow join time





Slow join time due to (1) time-consuming **global optimization** and (2) large **network latency**

Distributed control

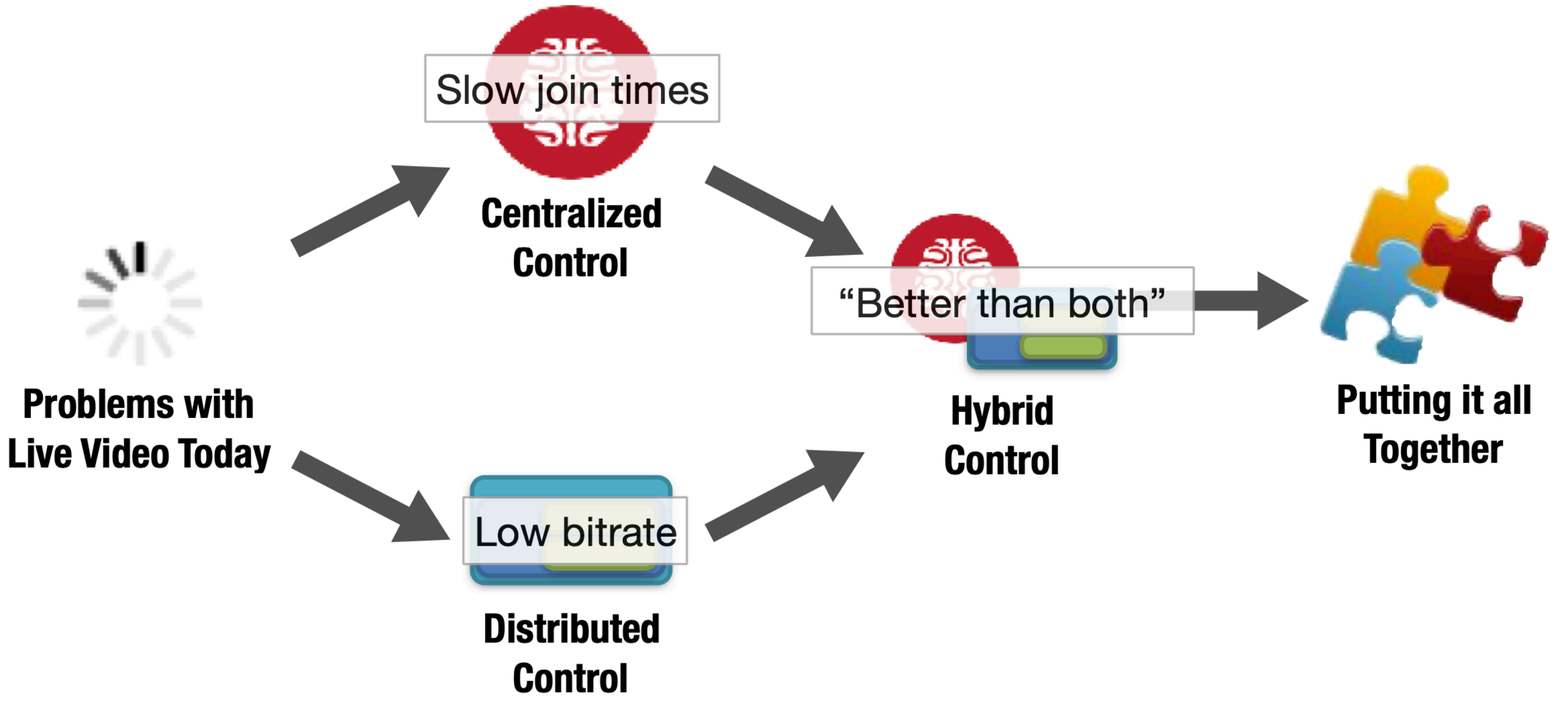


#hops to reach the video; bottleneck link capacity

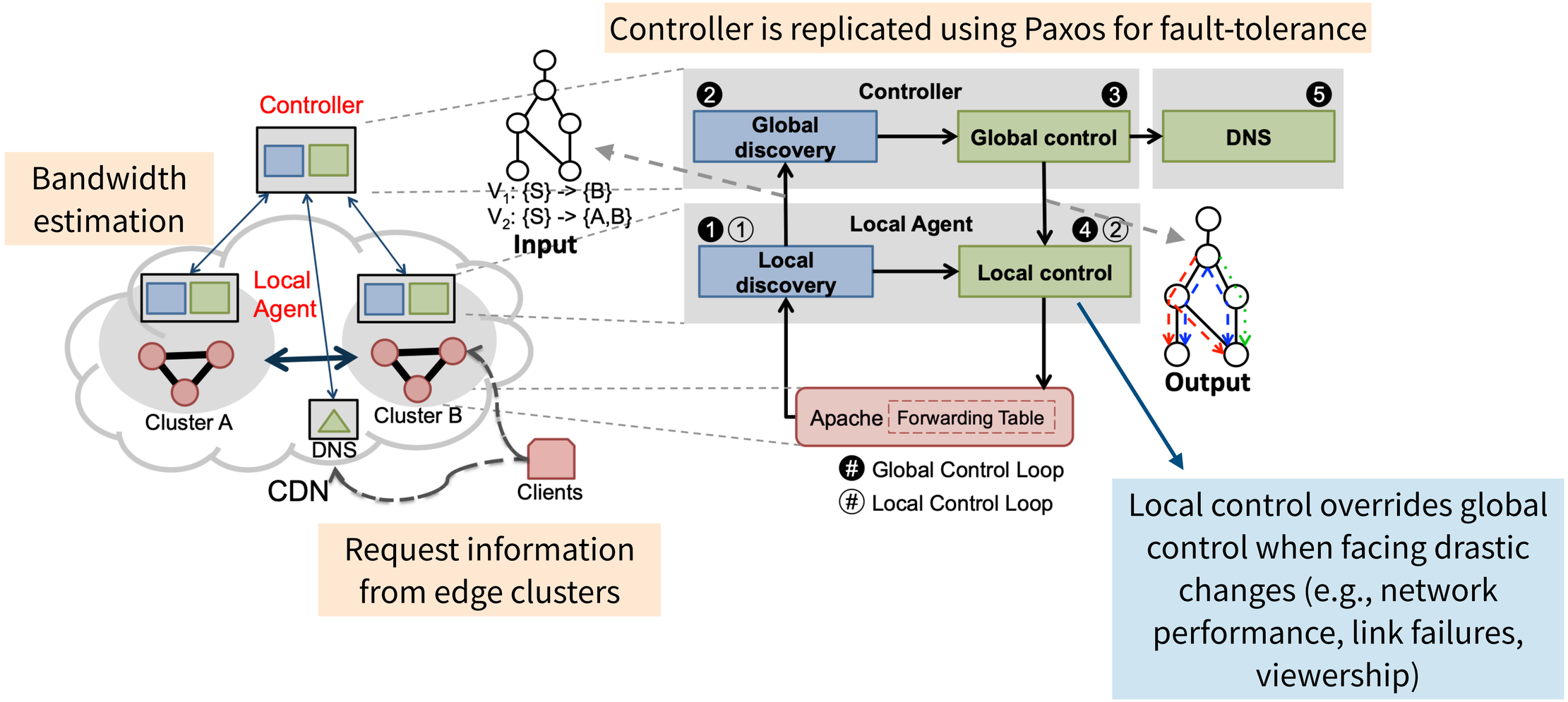
Similar to a distance vector routing protocol

Decision making can be **fast** (ms), but the decisions could be **suboptimal**

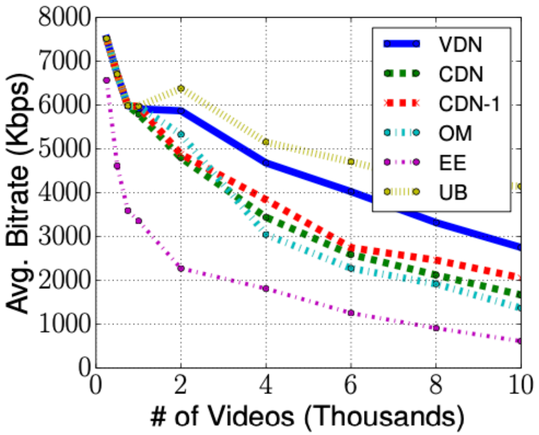
Hybrid control



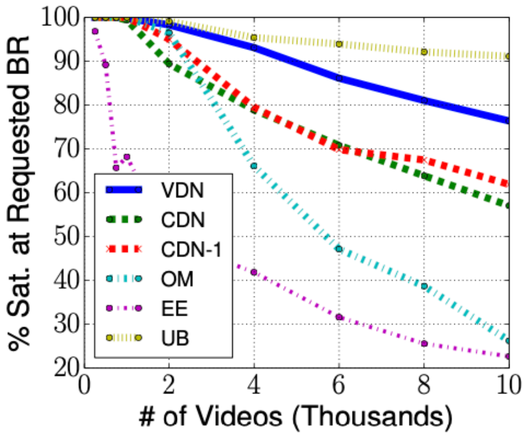
VDN: hybrid control



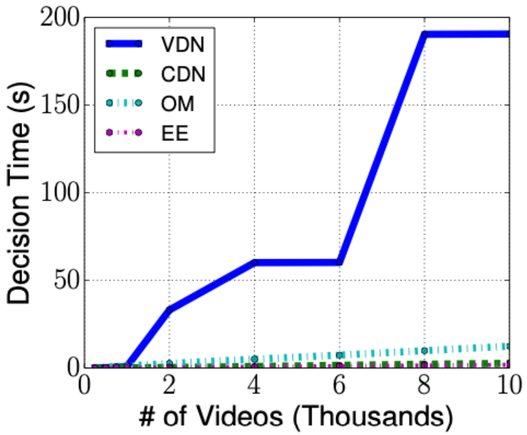
VDN benefits: service quality



(a) Avg. client bitrate.



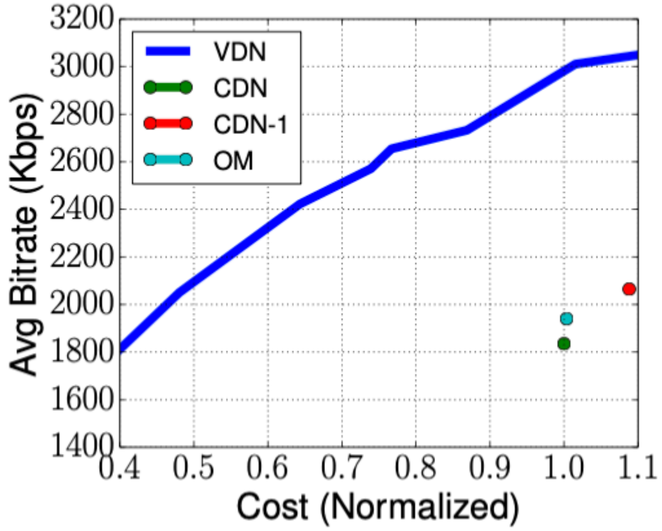
(b) % at requested bitrate.

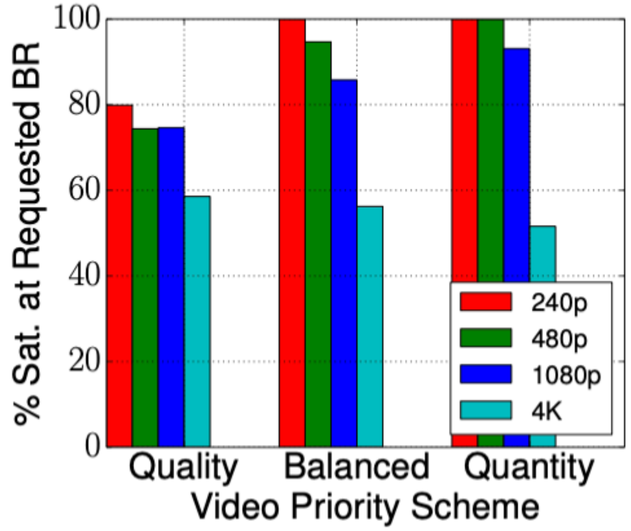


(c) Processing time.

VDN is able to **improve the average bitrate** of client requests and the **percentage of satisfaction with requested bitrates**. More importantly, VDN **scales to thousands of videos**.

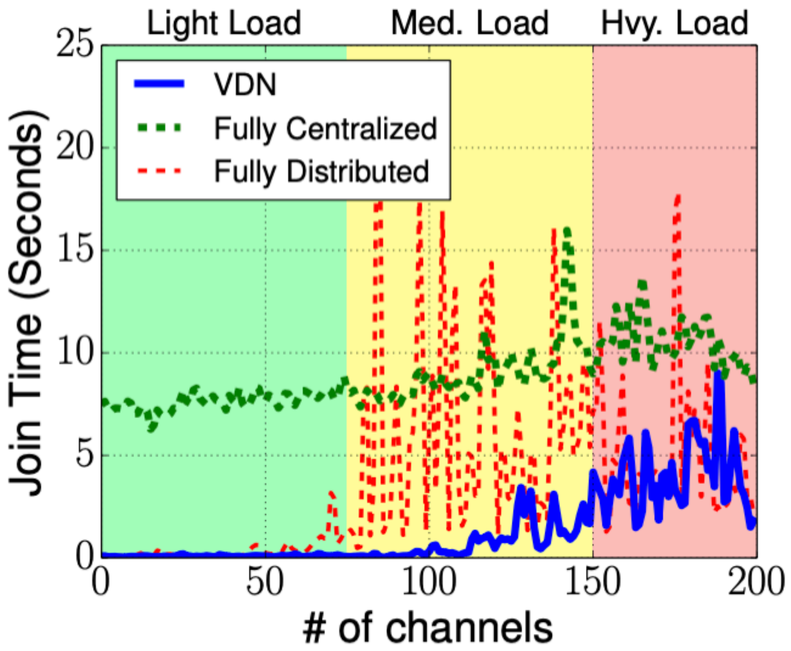
VDN benefits: quality vs. cost





VDN significantly **outperforms traditional CDNs** in terms of cost under the same service quality and allows for **flexible tradeoffs** between quality and cost

VDN benefits: join time



VDN is able to provide **low and stable** join times.
(The fully distributed solution also provides low join times, but it seems massive spikes as the system get loaded due to the lack of coordination.)

Summary

How does video get streamed over the Internet?

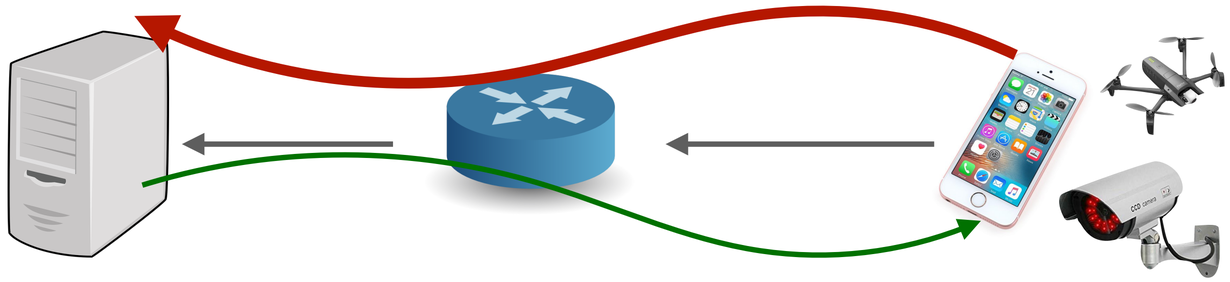
- Video is split into chunks and get streamed chunk by chunk
- Video compression: frame-level, video-level
- Constant bitrate, variable bitrate, adaptive bitrate

Challenges in video streaming

- Video streaming protocols: STP, HTTP-based, DASH
- Adaptive bitrate selection algorithms: rate-based, buffer-based, others
- Video streaming infrastructure management: VDN

Next time: video stream analytics

Video stream



Cloud server

Client