Advanced Network Programming

Beyond Networking

Lin Wang

Fall 2020, Period 1

Part 2: network infrastructure

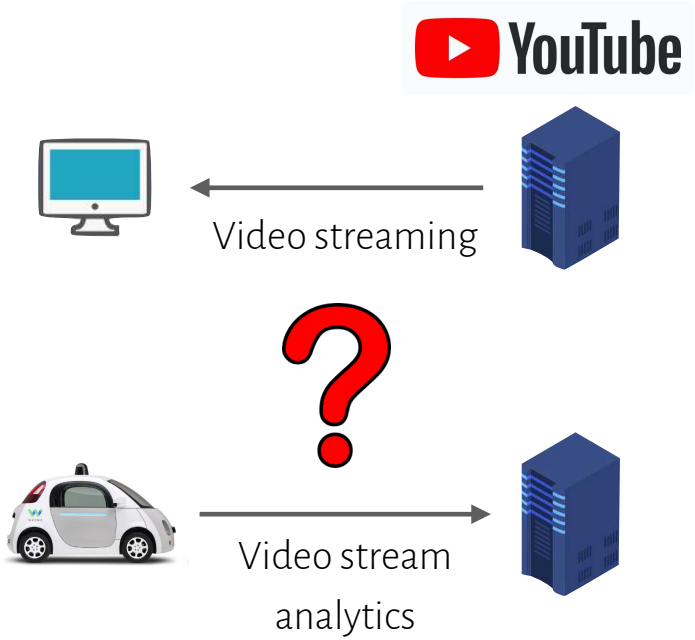
Lecture 7: Network forwarding and routing

Lecture 8: Software defined networking

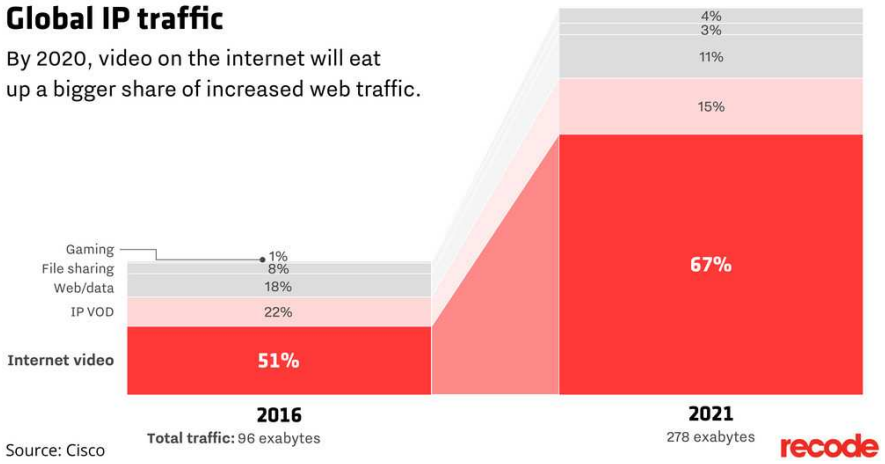
Lecture 9: Programmable data plane

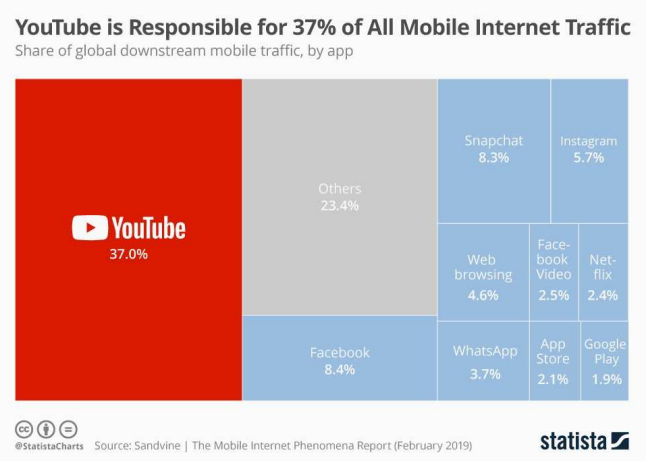
Lecture 10: Cloud networking

Lecture 11: Beyond networking



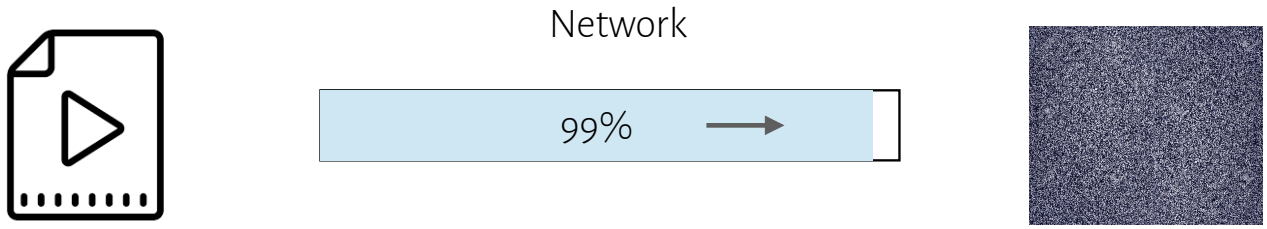
Video content has been dominating the Internet





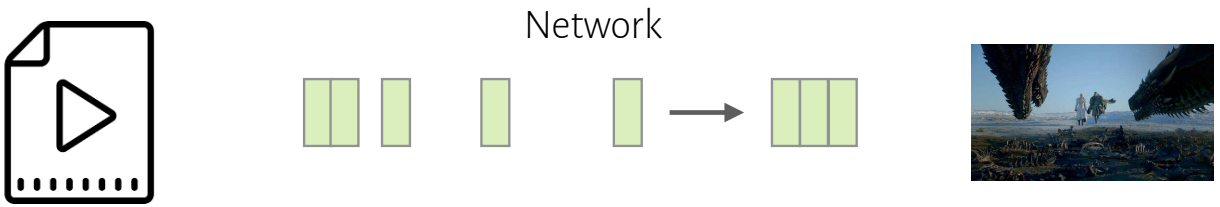
Video traffic is dominant nowadays: by 2021 it would represent more than 67% of the Internet traffic

Pre-streaming era



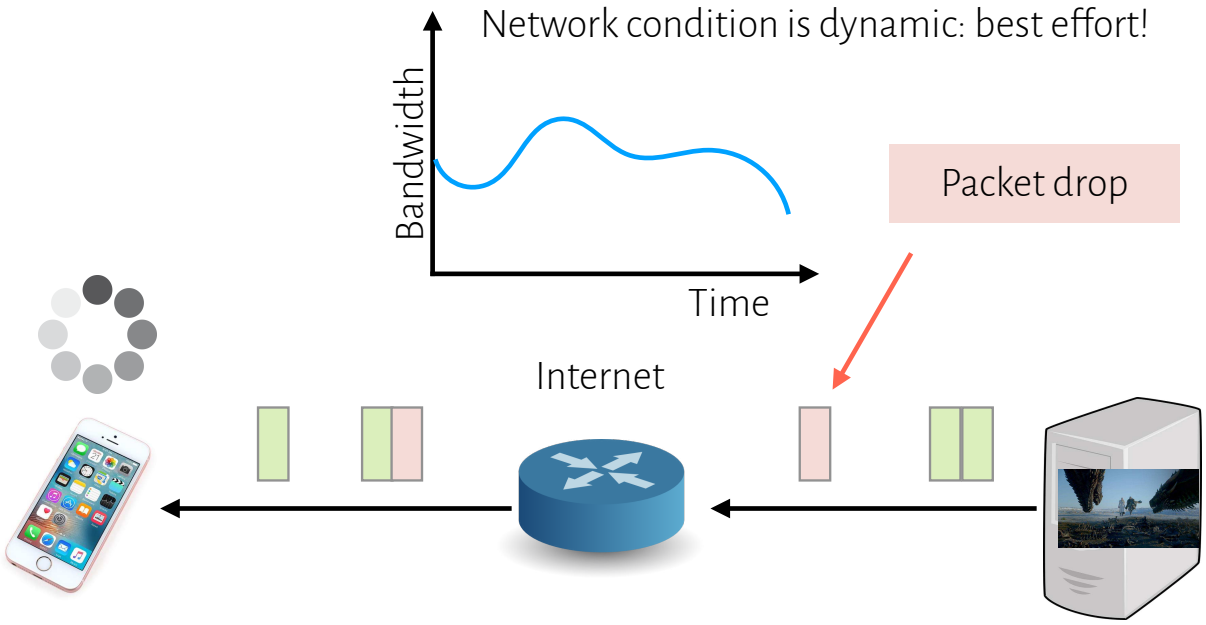
Download the whole video file and play it when the download is finished.

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

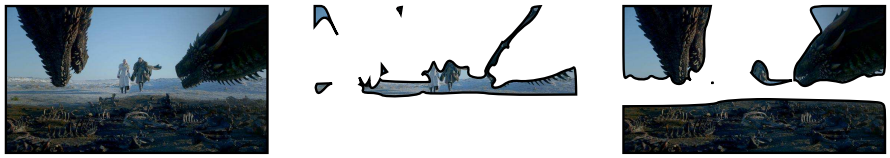
Reduce the amount of data to be transmitted over the network while keeping the video quality

Techniques:

- Frame-level compression: resize/encode the image
- Video-level compression: 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

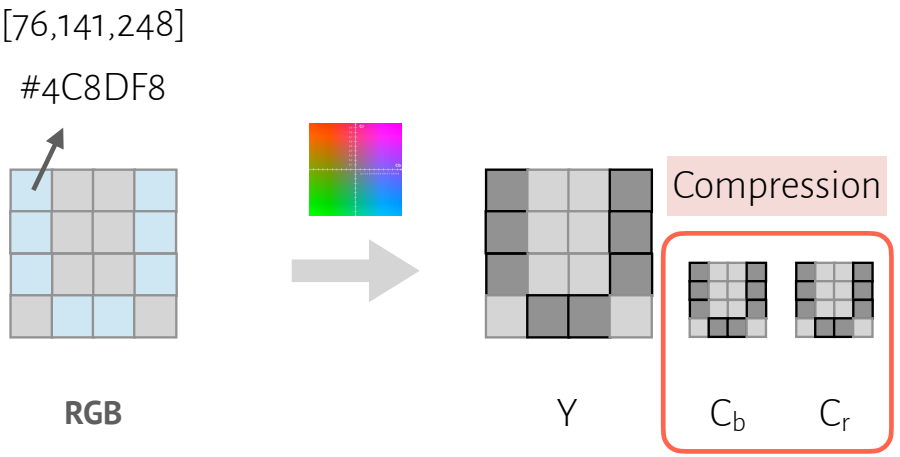
- Changes RGB to YC_bC_r
- Y: luminance, C_bC_r are chrominance

Why this change?

- Human eyes are less sensitive to chrominance than to luminance

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

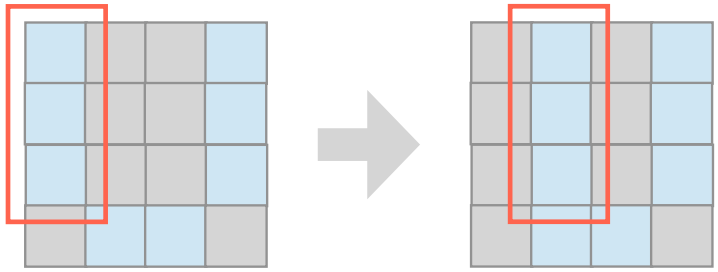
- Total compression rate x2



Video-level compression

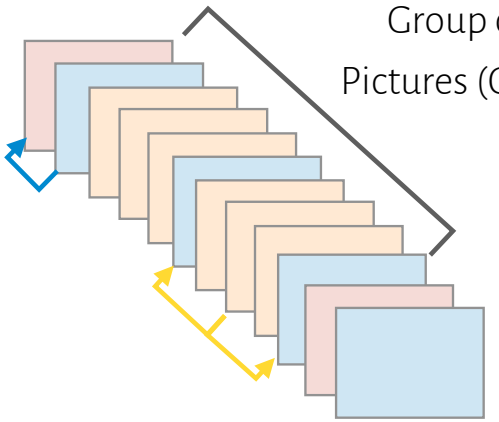
Remove temporal redundancy by keeping track of the relative differences (deltas) between frames

Macro blocks



Compressibility highly depends on the content, why?

H.264



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

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

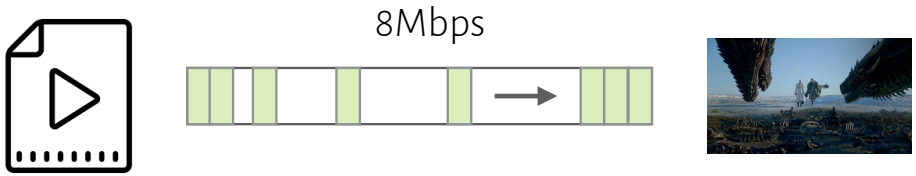
Bitrate

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

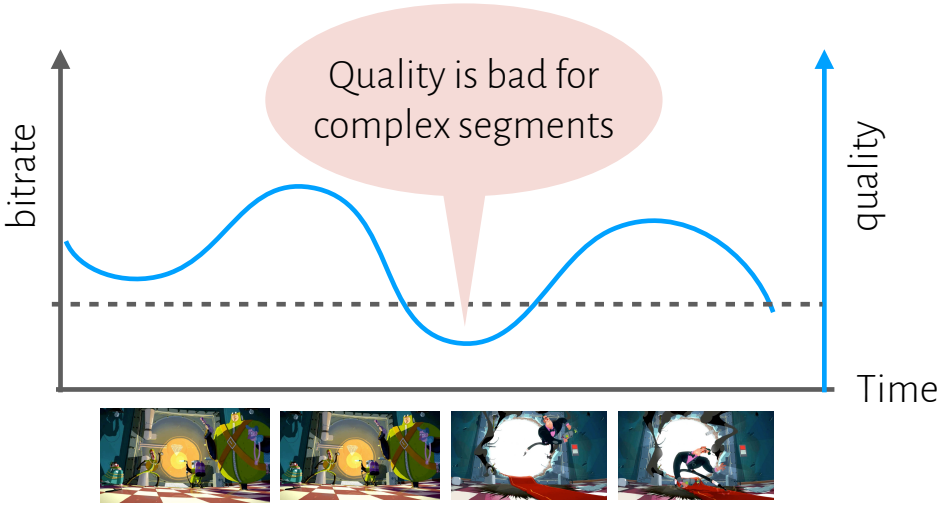


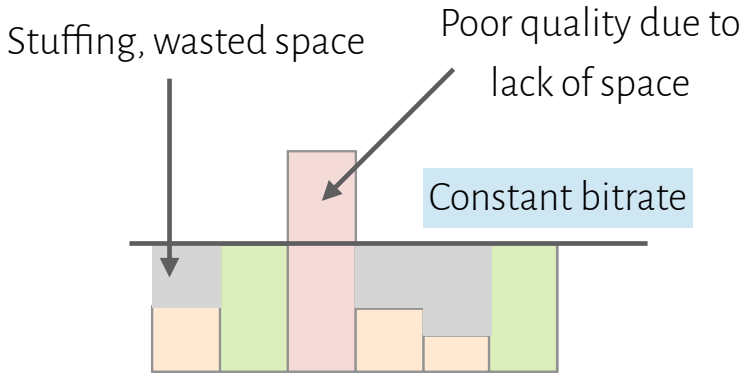
Check here for a live demo: https://reference.dashif.org/dash.js/latest/samples/dash-if-reference-player/index.html

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

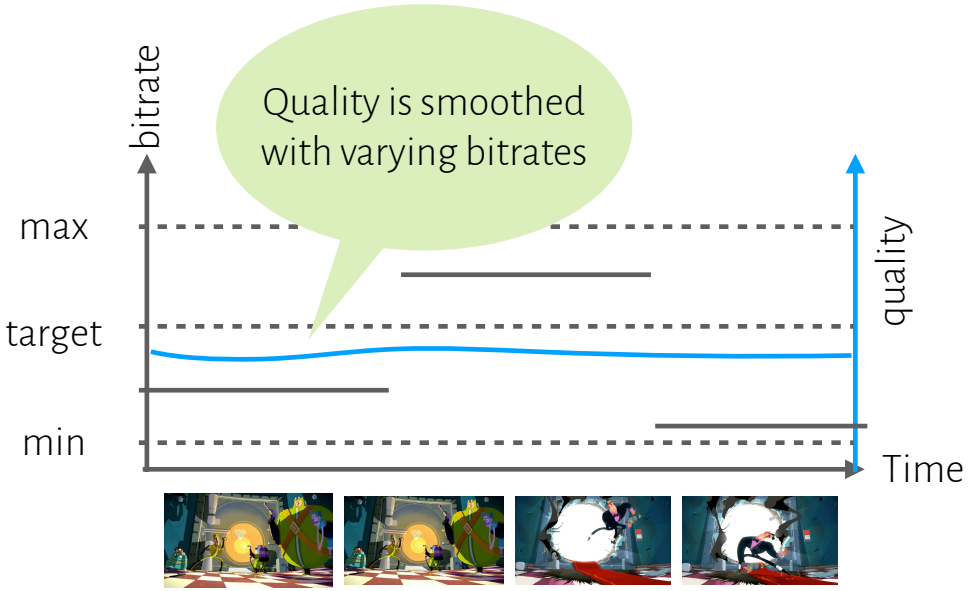




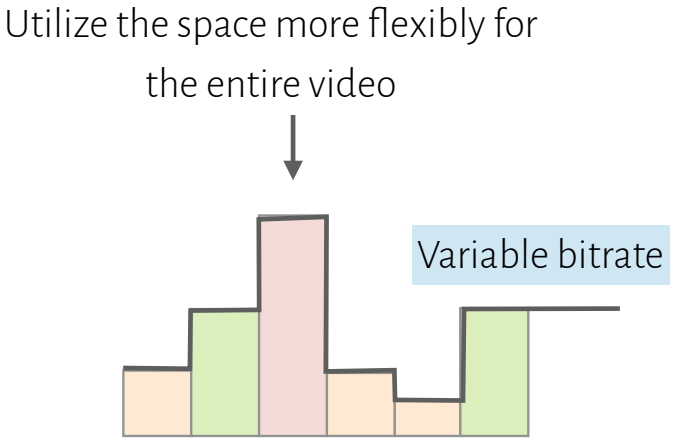
Variable bitrate (VBR)

Encode video with varying bitrates

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

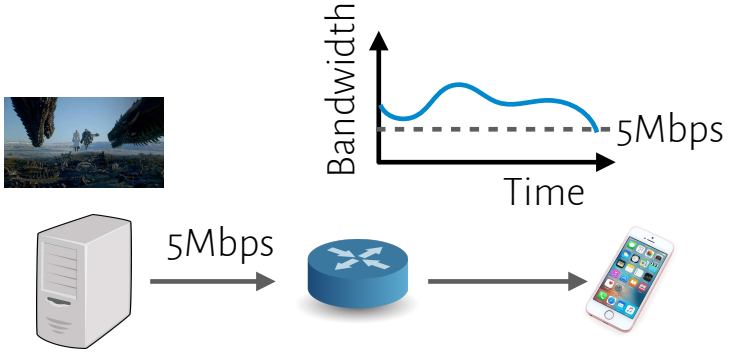


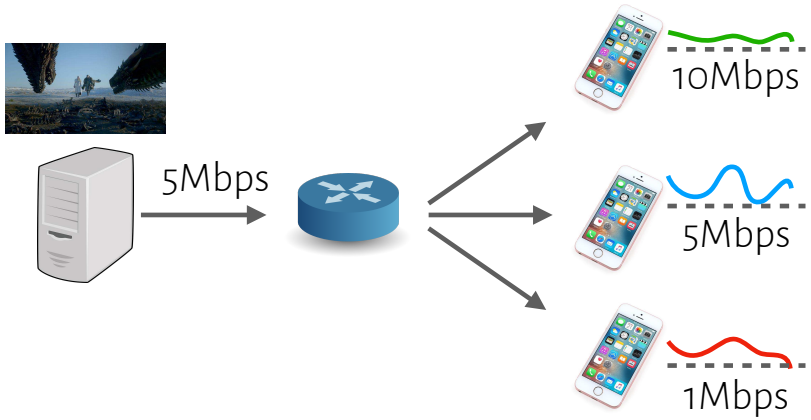
VBR algorithms are more complex and typically require support from the hardware



Video streaming with CBR

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

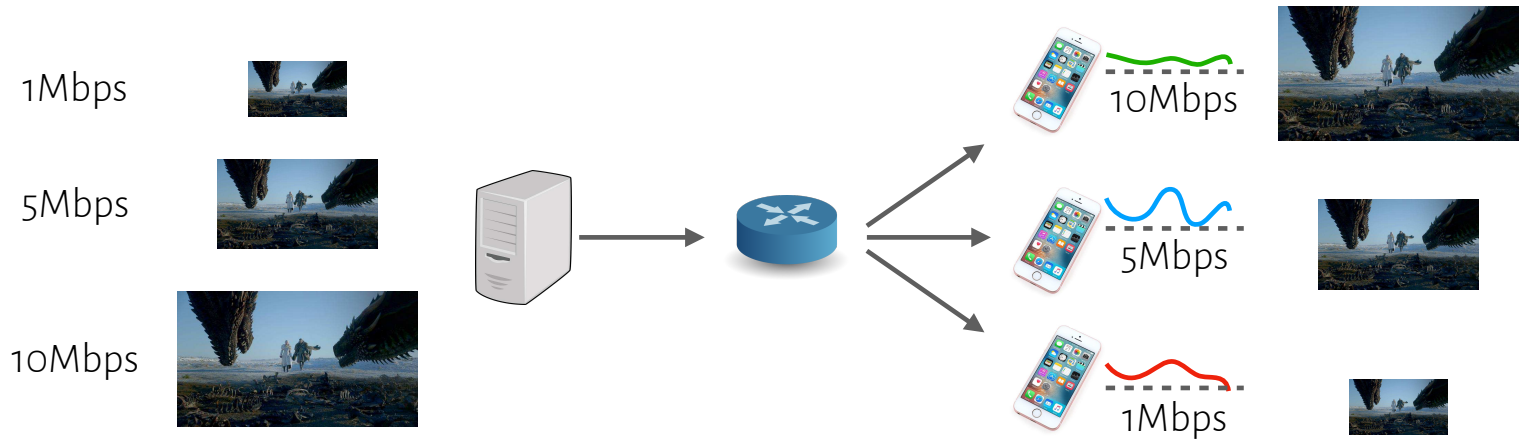




For a single user, CBR is sufficient, though not perfect

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

CBR improvement

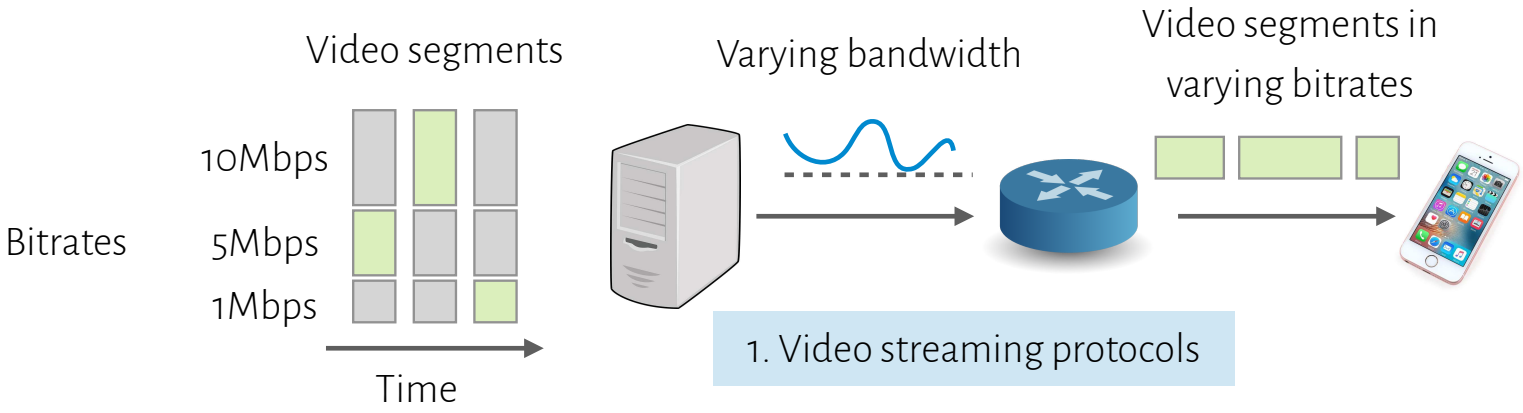


Encode the video with different CBRs at the streaming server and choose a suitable CBR based on the real-time bandwidth availability

Adaptive bitrate (ABR)

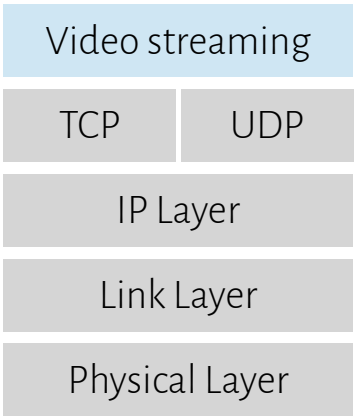
Main idea:

- **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



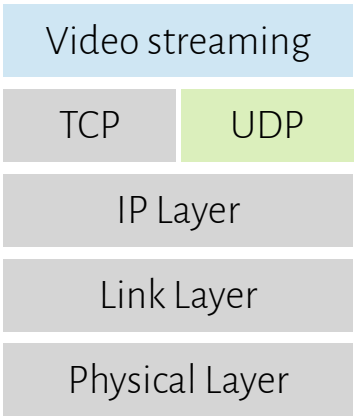
Questions?

Video streaming protocols

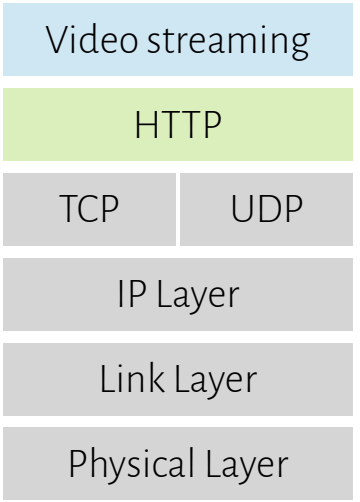


Which transport layer protocol to use?

Video streaming protocols



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



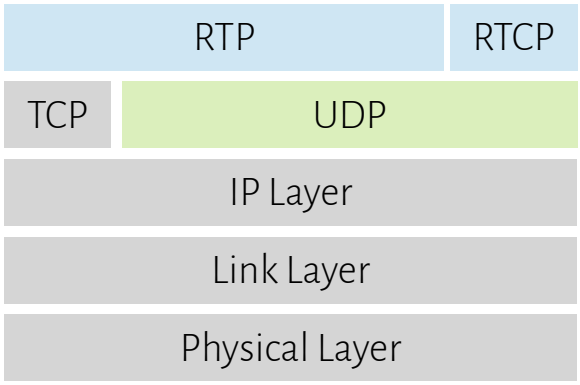
Modern video streaming protocols are based on HTTP

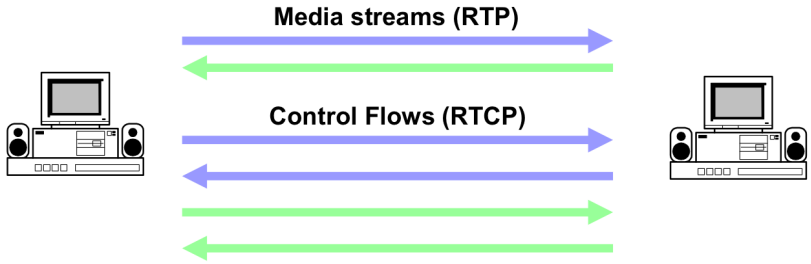
Video streaming protocol: RTP

RFC 1889 RFC 3550

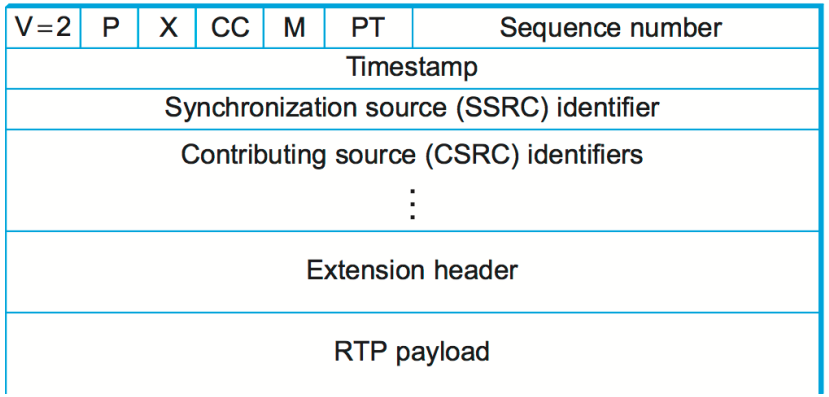
Real-time transport protocol: based on UDP

- 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

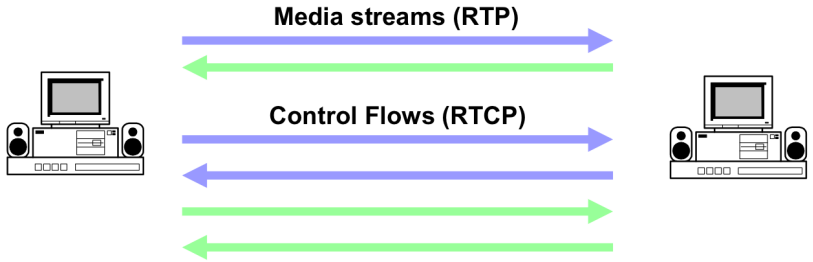
RFC 3550

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!

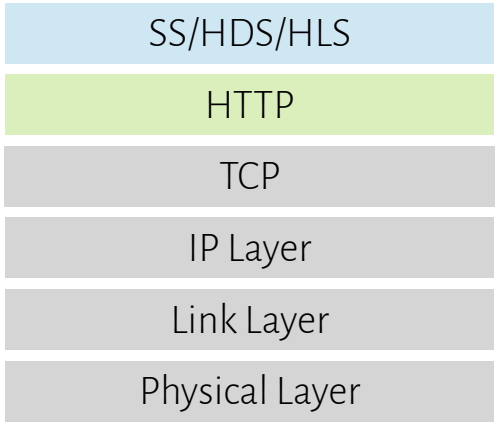
Video streaming protocols based on HTTP

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

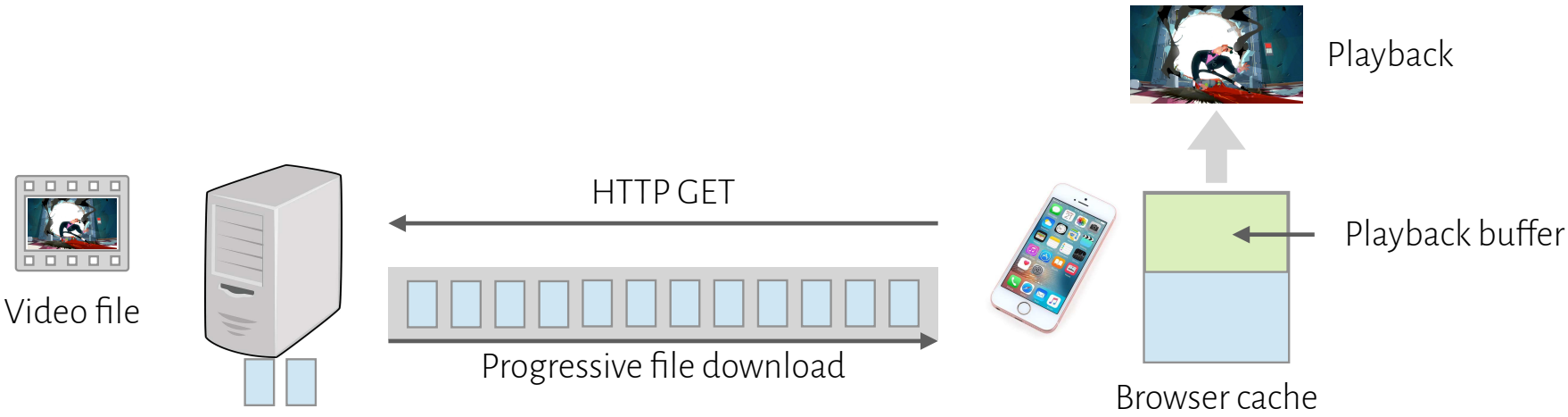
Bad for the industry such as CDN providers like Akamai since every functionality has to be implemented three times

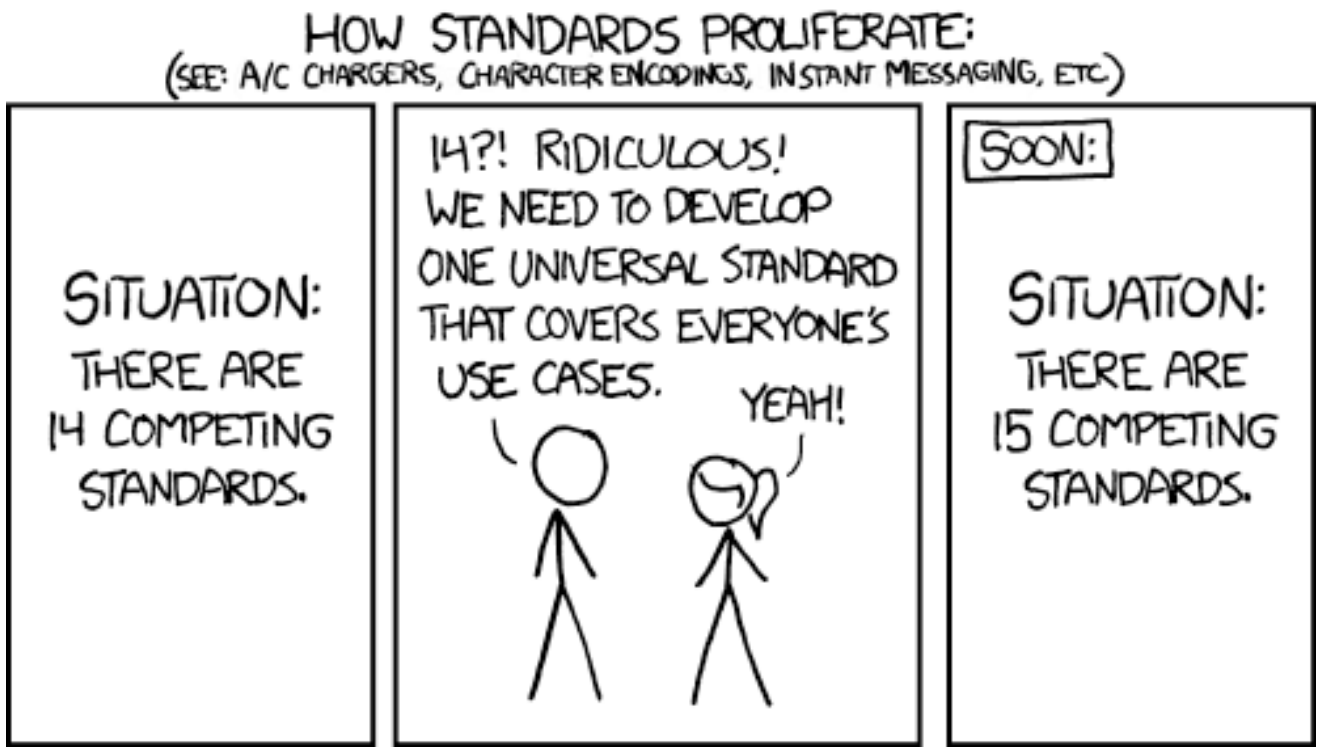


Why HTTP?

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)




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

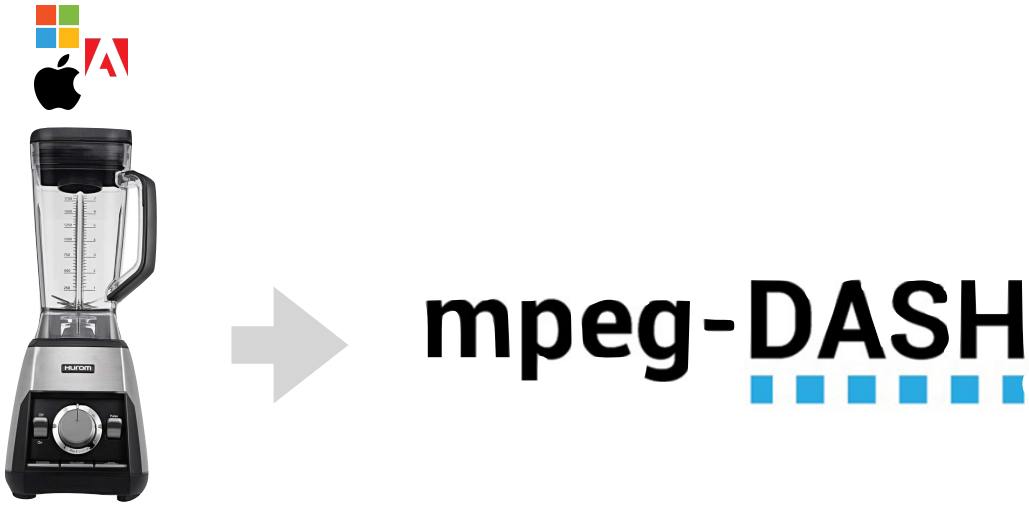
- Blending existing formats into a new format

MPEG (moving pictures experts group)

- Standardized MP3, MP4

Standardization work from 2010-2012

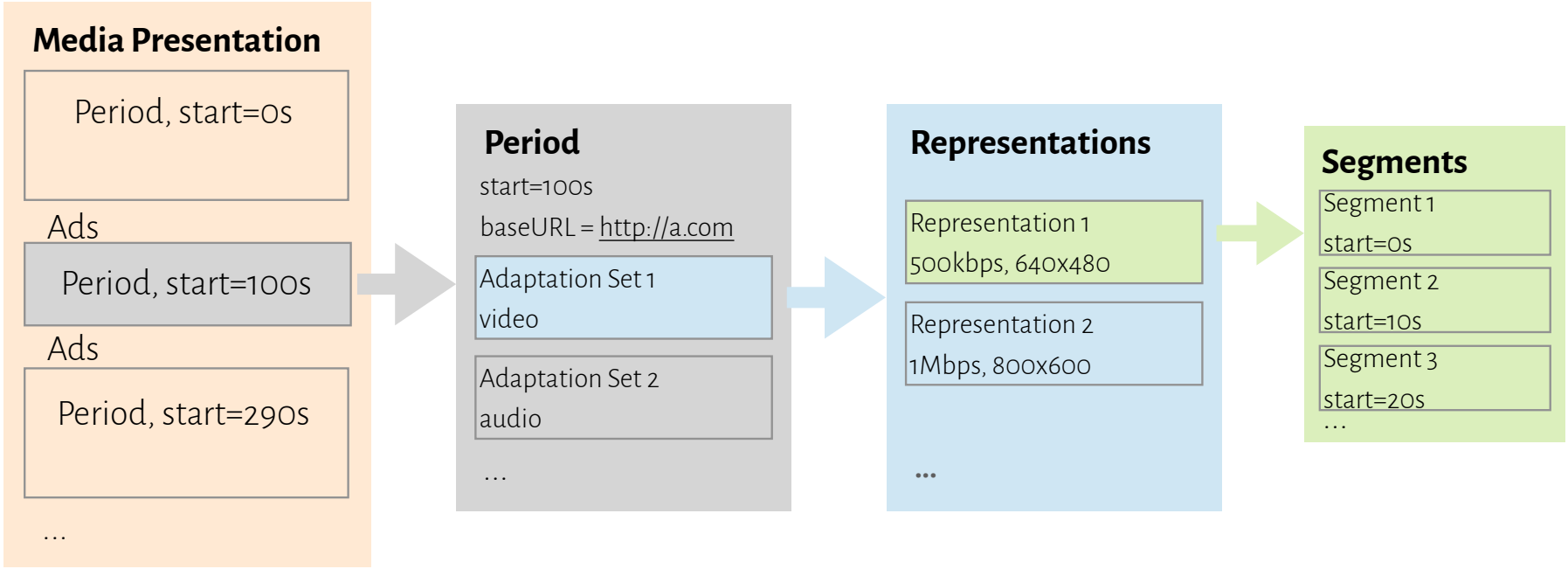
Note: DASH is not a protocol



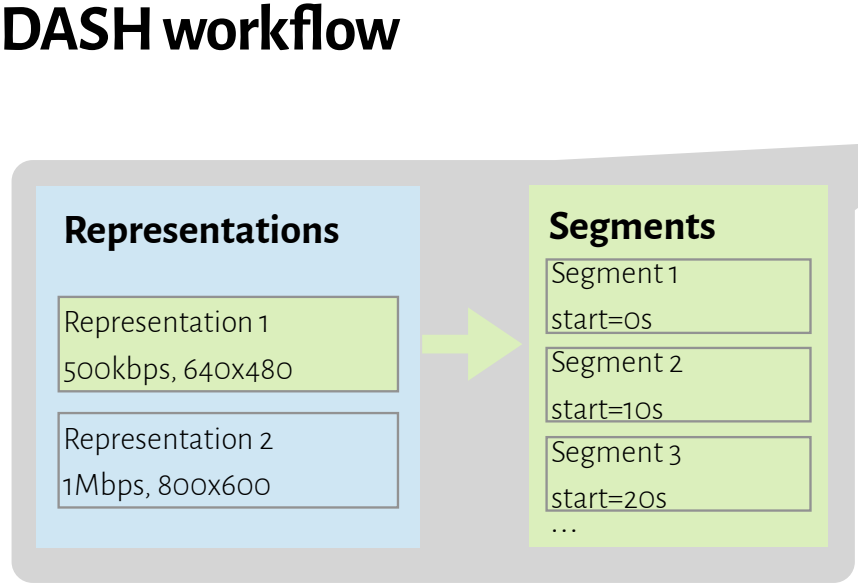
DASH: data model

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

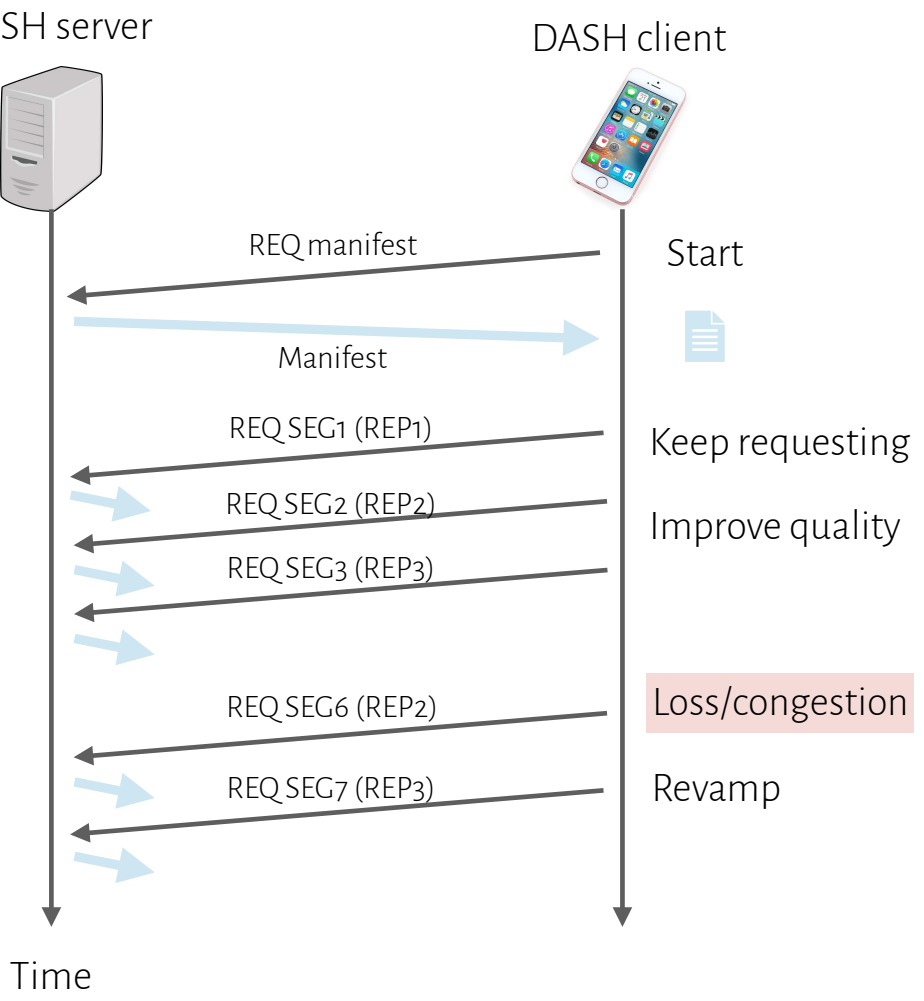
- Ensuring interoperability



DASH workflow

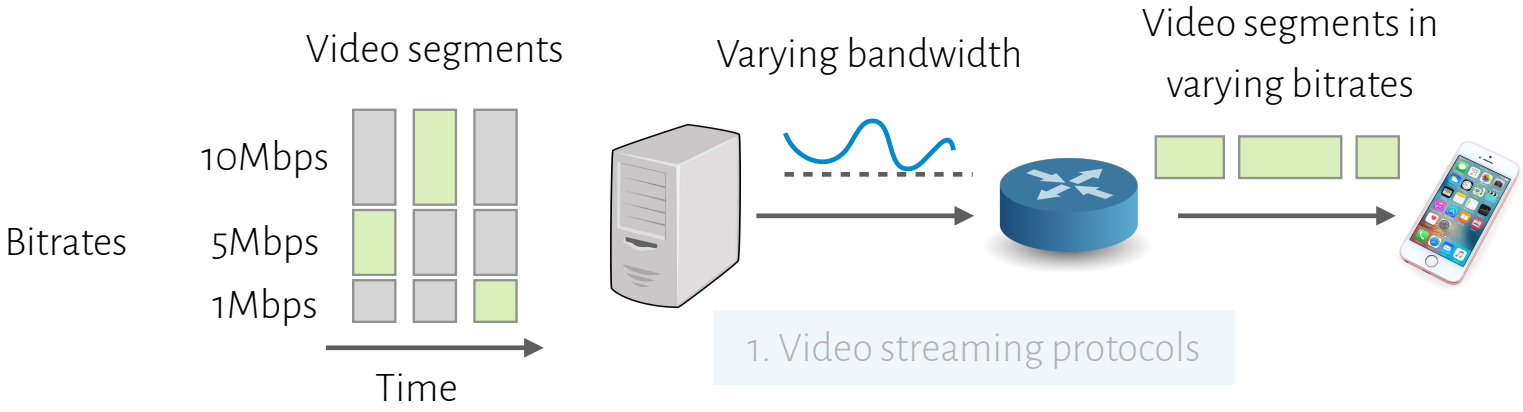


Video file is encoded using the MDP data model described with a **manifest** file



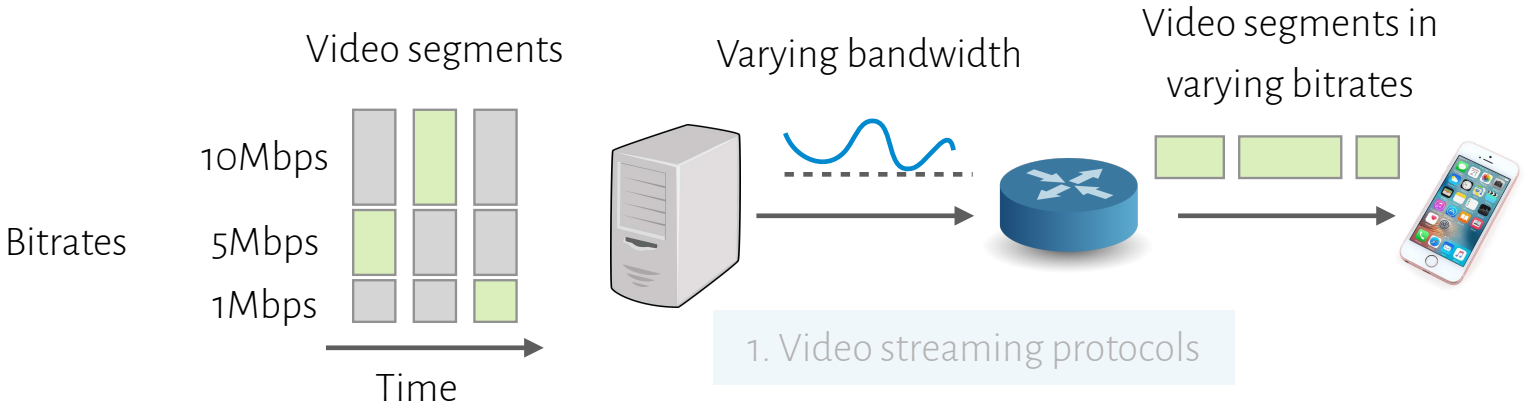
Questions?

Bit rate selection in ABR



How to design such an algorithm? Any ideas?

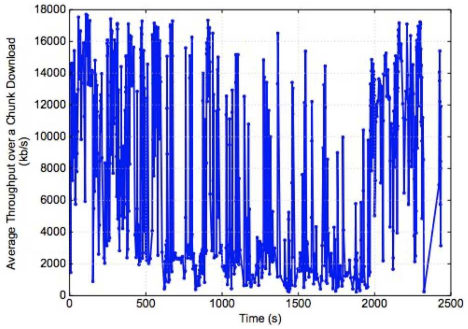
Bit rate selection in ABR



1. Video streaming protocols

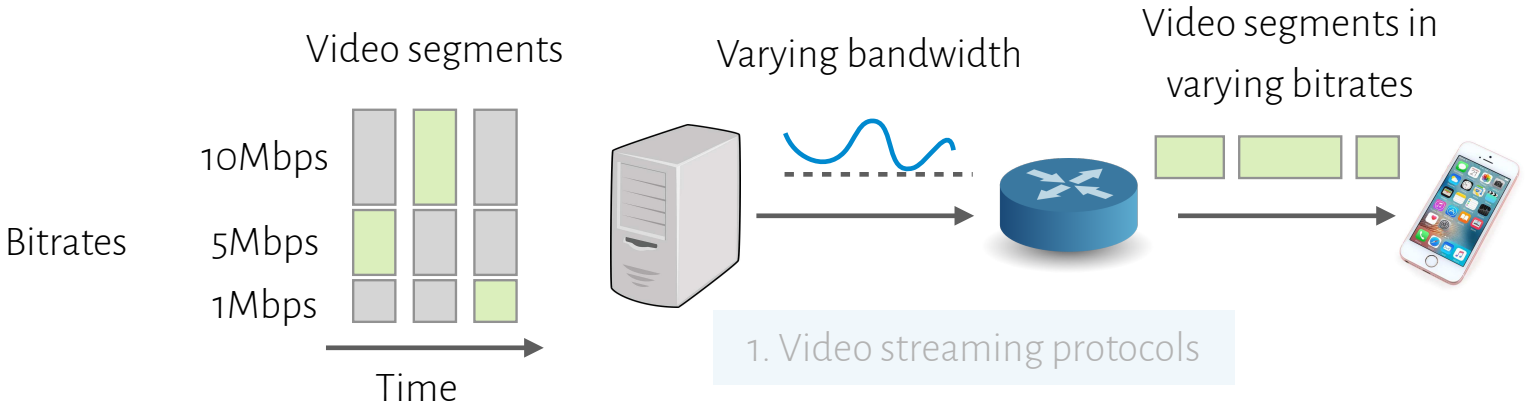
2. Bitrate selection algorithms

The most straightforward approach is to perform bandwidth estimation



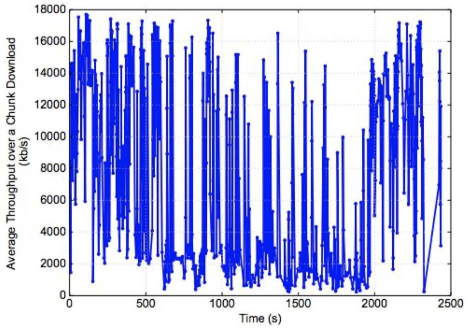
Challenge: bandwidth variation can be very high!

Bit rate selection in ABR



1. Video streaming protocols

2. Bitrate selection algorithms

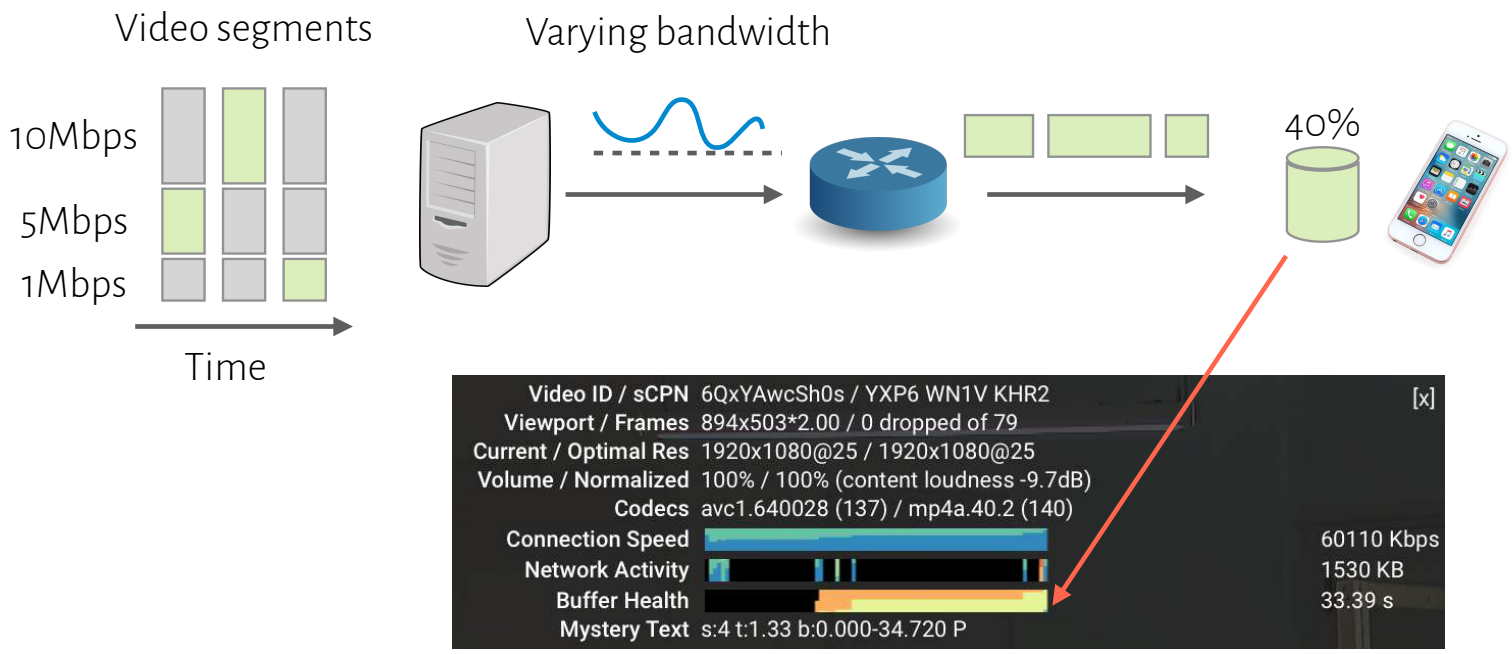


Other ideas?

The most straightforward approach is to perform bandwidth estimation

Challenge: bandwidth variation can be very high!

ABR algorithm: buffer-based



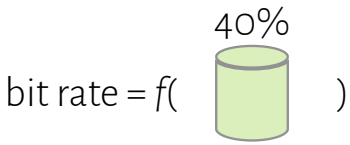
Make ABR decisions based on the buffer occupancy at the client

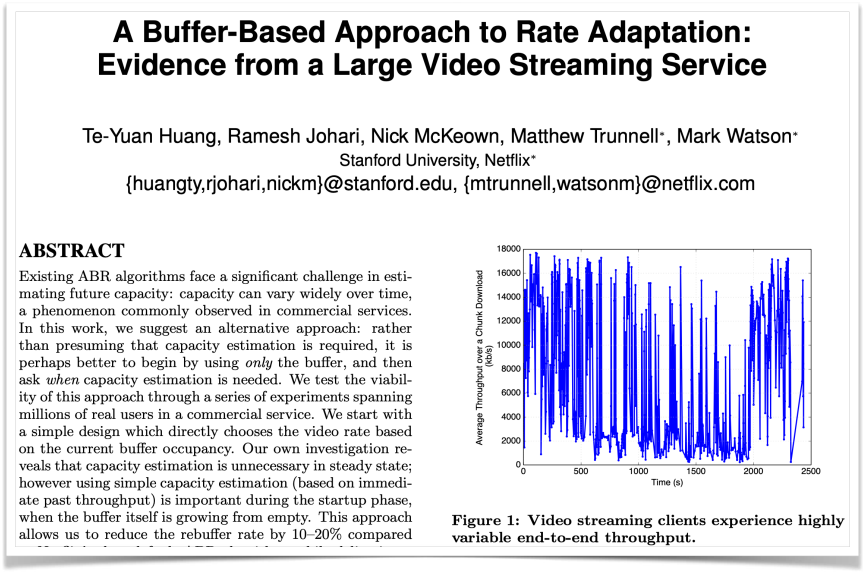
ABR algorithm: buffer-based

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

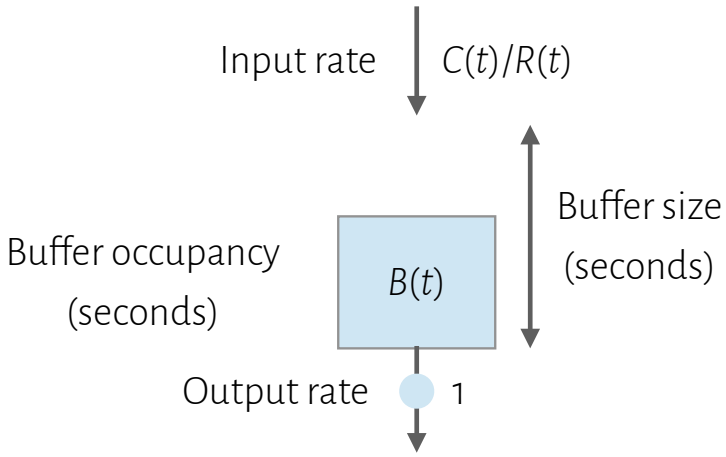
ABSTRACT

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

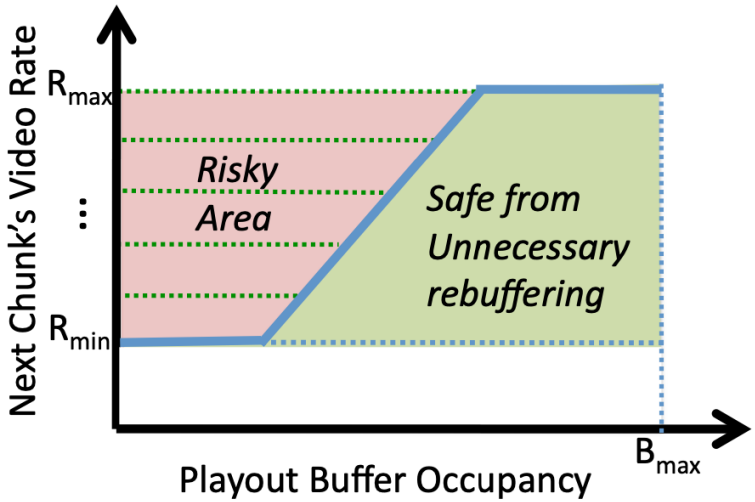
- At a certain point, it is safe to increase $R(t)$ to improve the streaming quality

$C(t)/R(t) < 1$: 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

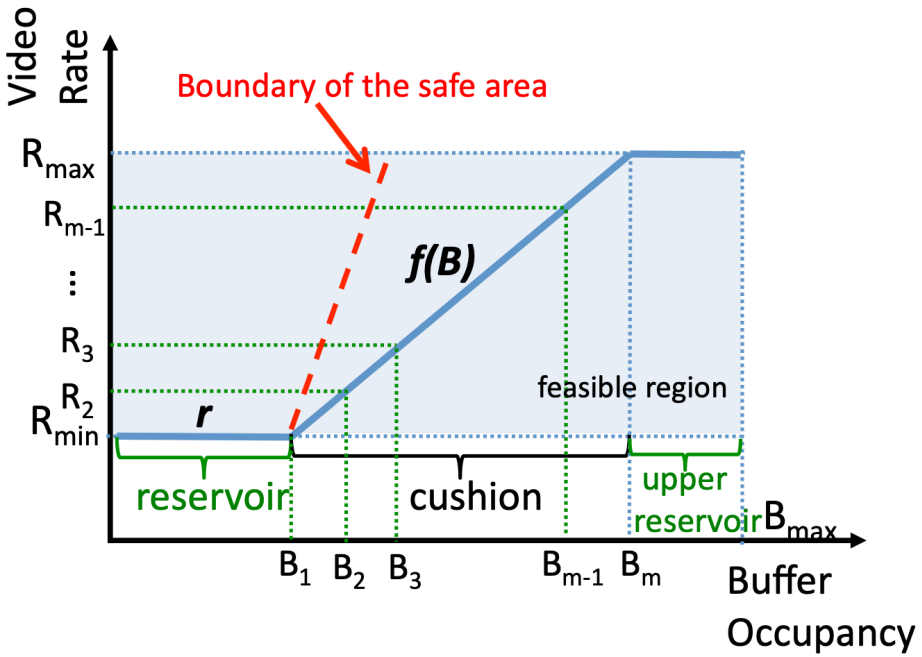
- As long as $C(t) > R_{min}$ for all t and we adapt $f(B) \rightarrow R_{min}$ as $B \rightarrow 0$, we will never unnecessarily rebuffer because the buffer will start to grow before it runs dry

Goal 2: average video rate maximization

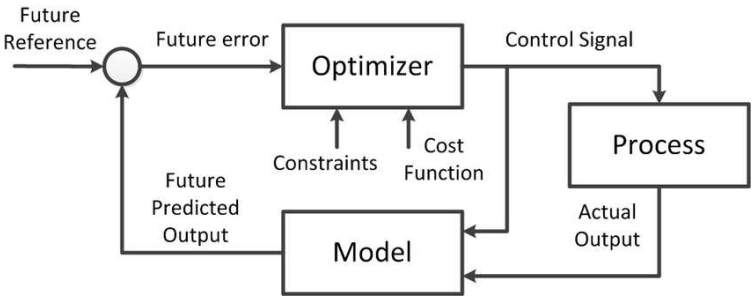
- As long as $f(B)$ is increasing and eventually reaches R_{max}, the average video rate will match the average capacity when $R_{min} < C(t) < R_{max}$ for all $t > 0$

BBA in practice

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



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-

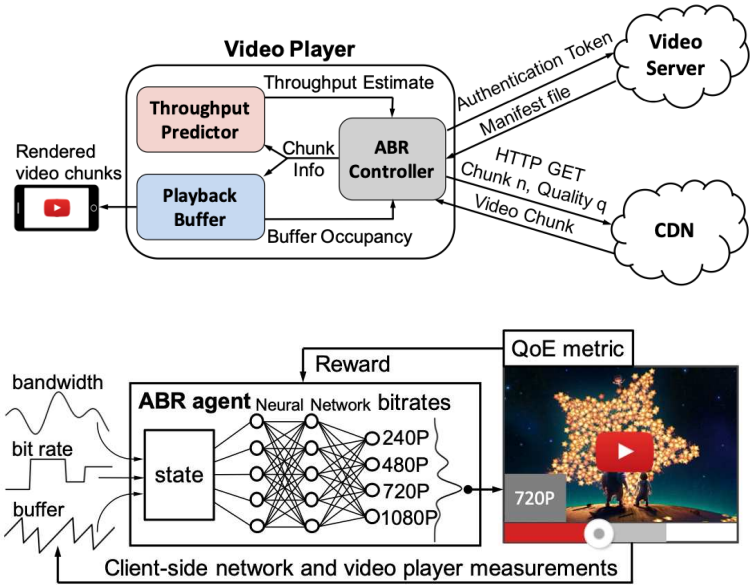
1 Introduction

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.

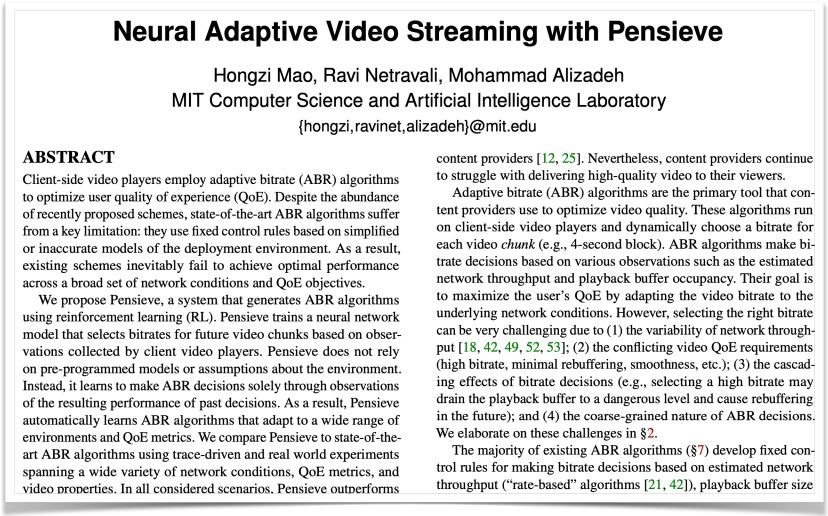
Given the complex Internet video delivery ecosystem and presence of diverse bottlenecks, the *bitrate adaptation logic* in the client-side video player becomes critical to optimize user experience [16]. In the HTTP-based delivery model that predominates today [44], videos are typically chunked and encoded 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: deep reinforcement learning based



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.

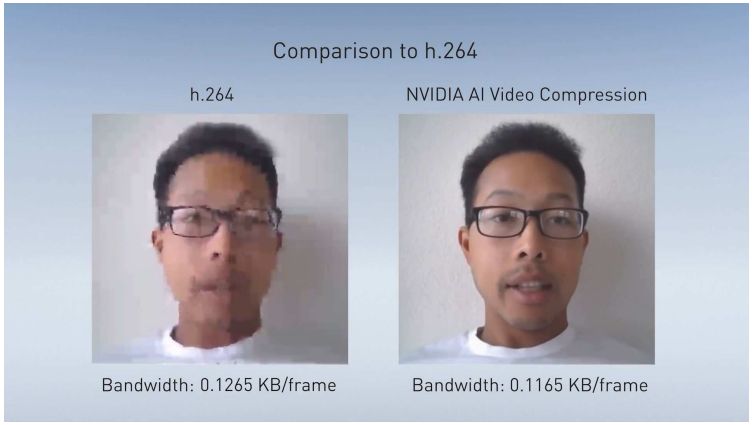
Adaptive bitrate (ABR) algorithms are the primary tool that content providers use to optimize video quality. These algorithms run on client-side video players and dynamically choose a bitrate for each video *chunk* (e.g., 4-second block). ABR algorithms make bitrate decisions based on various observations 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.

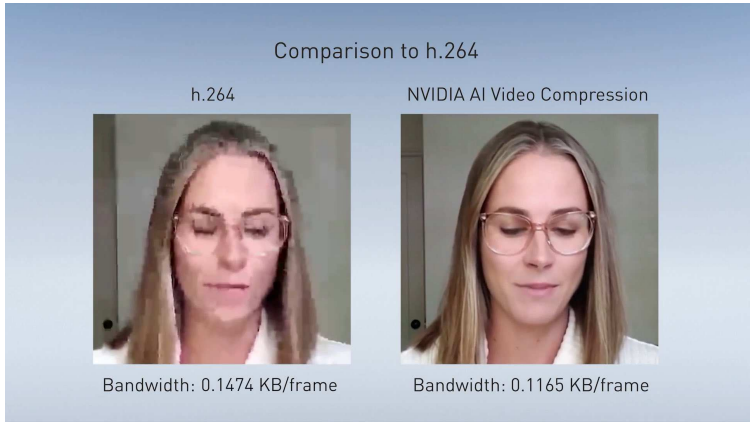
The majority of existing ABR algorithms (§7) develop fixed control rules for making bitrate decisions based on estimated network throughput ("rate-based" algorithms [21, 42]), playback buffer size

ACM SIGCOMM 2017

Replacing video codecs with machine learning

Very little bandwidth consumption for super-high quality real-time video streaming



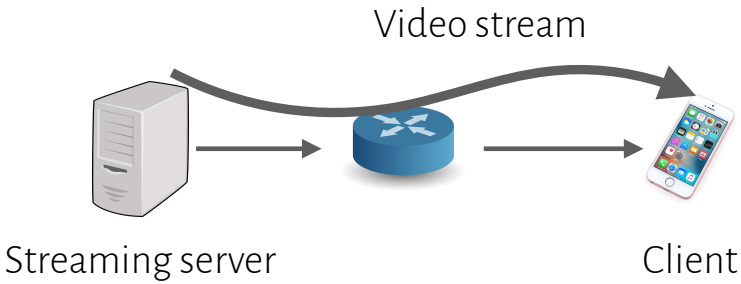


https://www.youtube.com/watch?v=NqmMnjJ6GEg

What are the down-sides of this cool technology?

Questions?

Video streaming vs. video stream analytics



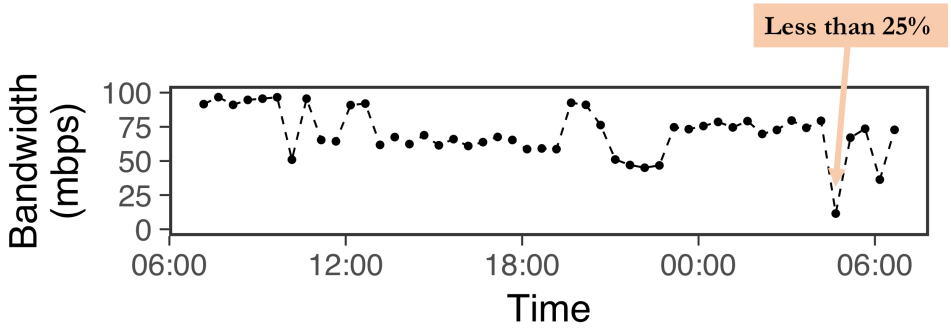
Video streaming

Video stream analytics: still an active research field

Challenges in video stream analytics

Large volume of traffic needs to be sent across the wide area network (WAN)

WAN has **scarce, expensive, and variable bandwidth**

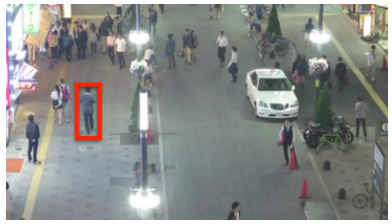


Applications have **quality of service requirements** which are complex to optimize

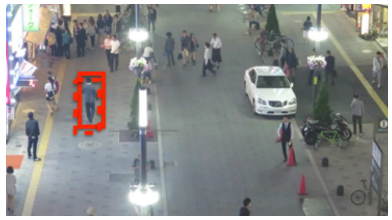
- Unlike video streaming where quality of experience is well-defined
- Video analytics rely on **deep learning** models and the analytics accuracy has a nonlinear relationship with the quality metrics (resolution, frame rate, latency)

Application-specific optimization

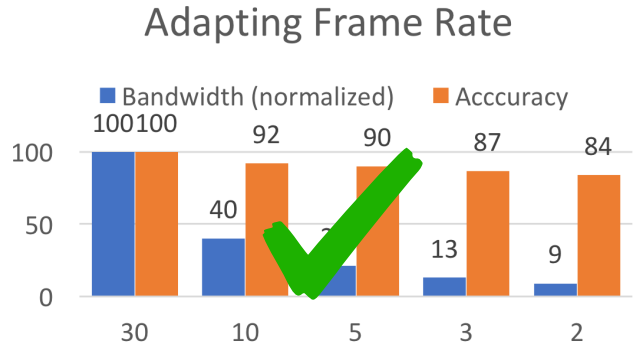
Scenario 1: a surveillance application that detects pedestrians on a busy street

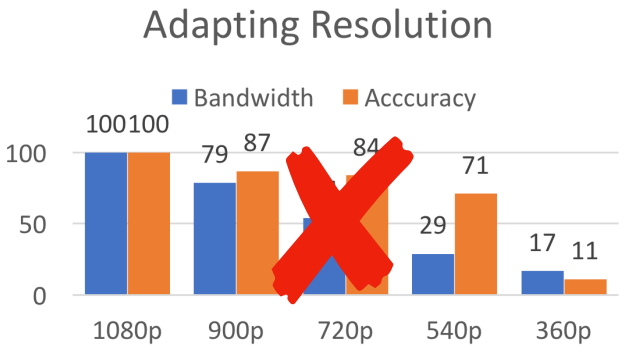


t=0s, small target in far-field views



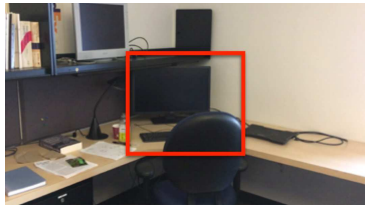
t=1s, small difference



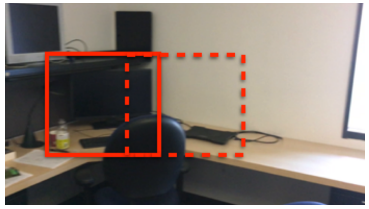


Application-specific optimization

Scenario 2: an application that detects objects on a mobile phone



t=0s, nearby and large target

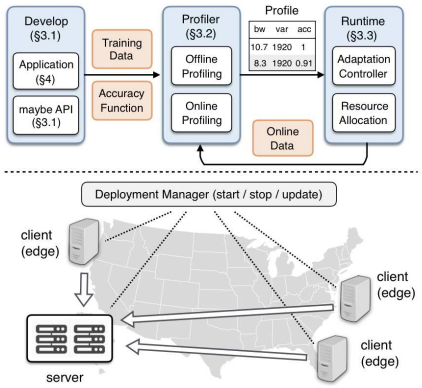


t=1s, large difference due to camera movement

A general framework: AWStream

Systematic and quantitative adaptation

- New **programming abstractions** to express adaptation
- **Automatic** data-driven **profiling**
- **Runtime adaptation** balancing the different goals



AWStream: Adaptive Wide-Area Streaming Analytics

Ben Zhang
UC Berkeley

Xin Jin
Johns Hopkins University

Sylvia Ratnasamy
UC Berkeley

John Wawrzynek
UC Berkeley

Edward A. Lee
UC Berkeley

ABSTRACT

The emerging class of wide-area streaming analytics faces the challenge of scarce and variable WAN bandwidth. Non-adaptive applications built with TCP or UDP suffer from increased latency or degraded accuracy. State-of-the-art approaches that adapt to network changes require developer writing sub-optimal manual policies or are limited to application-specific optimizations.

We present AWStream, a stream processing system that simultaneously achieves low latency and high accuracy in the wide area, requiring minimal developer efforts. To realize this, AWStream uses three ideas: (*i*) it integrates application adaptation as a first-class programming abstraction in the stream processing model; (*ii*) with a combination of offline and online profiling, it automatically learns an accurate profile that models accuracy and bandwidth trade-off; and (*iii*) at runtime, it carefully adjusts the application data rate to match the avail-

Analytics. In *SIGCOMM '18: ACM SIGCOMM 2018 Conference, August 20–25, 2018, Budapest, Hungary.* ACM, New York, NY, USA, 17 pages. https://doi.org/10.1145/3230543.3230554

1 INTRODUCTION

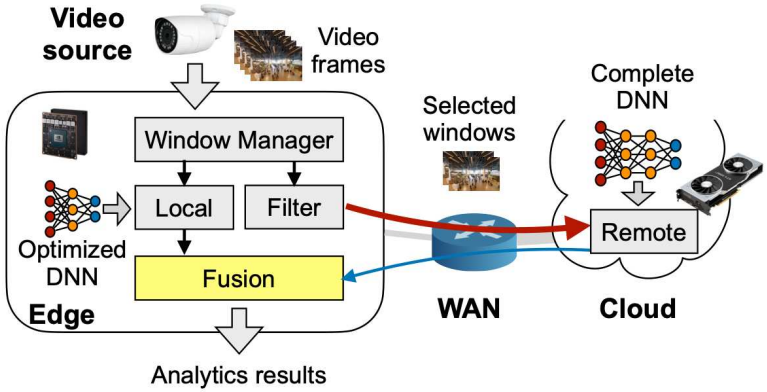
Wide-area streaming analytics are becoming pervasive, especially with emerging Internet of Things (IoT) applications. Large cities such as London and Beijing have deployed millions of cameras for surveillance and traffic control [46, 85]. Buildings are increasingly equipped with a wide variety of sensors to improve energy efficiency and occupant comfort [44]. Geo-distributed infrastructure, such as content delivery networks (CDNs), analyze requests from machine logs across the globe [54]. These applications all transport, distill, and process streams of data across the wide area, in real time.

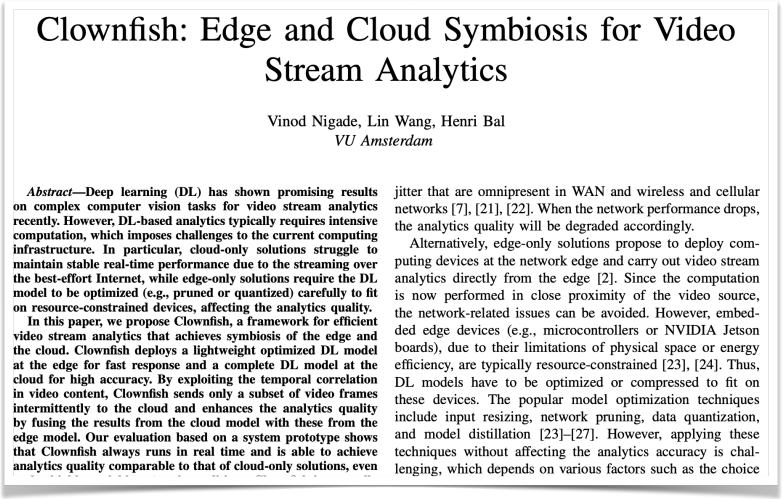
A key challenge that the above applications face is dealing

ACM SIGCOMM 2018

Clownfish: real-time video stream analytics

Combine a **local fast** processing and a **remote accurate** processing





Clownfish: Edge and Cloud Symbiosis for Video Stream Analytics

Vinod Nigade, Lin Wang, Henri Bal
VU Amsterdam

***Abstract*—Deep learning (DL) has shown promising results on complex computer vision tasks for video stream analytics recently. However, DL-based analytics typically requires intensive computation, which imposes challenges to the current computing infrastructure. In particular, cloud-only solutions struggle to maintain stable real-time performance due to the streaming over the best-effort Internet, while edge-only solutions require the DL model to be optimized (e.g., pruned or quantized) carefully to fit on resource-constrained devices, affecting the analytics quality.**

In this paper, we propose Clownfish, a framework for efficient video stream analytics that achieves symbiosis of the edge and the cloud. Clownfish deploys a lightweight optimized DL model at the edge for fast response and a complete DL model at the cloud for high accuracy. By exploiting the temporal correlation in video content, Clownfish sends only a subset of video frames intermittently to the cloud and enhances the analytics quality by fusing the results from the cloud model with these from the edge model. Our evaluation based on a system prototype shows that Clownfish always runs in real time and is able to achieve analytics quality comparable to that of cloud-only solutions, even jitter that are omnipresent in WAN and wireless and cellular networks [7], [21], [22]. When the network performance drops, the analytics quality will be degraded accordingly.

Alternatively, edge-only solutions propose to deploy computing devices at the network edge and carry out video stream analytics directly from the edge [2]. Since the computation is now performed in close proximity of the video source, the network-related issues can be avoided. However, embedded edge devices (e.g., microcontrollers or NVIDIA Jetson boards), due to their limitations of physical space or energy efficiency, are typically resource-constrained [23], [24]. Thus, DL models have to be optimized or compressed to fit on these devices. The popular model optimization techniques include input resizing, network pruning, data quantization, and model distillation [23]–[27]. However, applying these techniques without affecting the analytics accuracy is challenging, which depends on various factors such as the choice

ACM/IEEE SEC 2020

Open research projects / thesis topics

1. Latency control for edge-based mobile Augmented Reality

- Mobile AR, WebRTC, video streaming, deep learning

2. Edge-based deep learning management framework

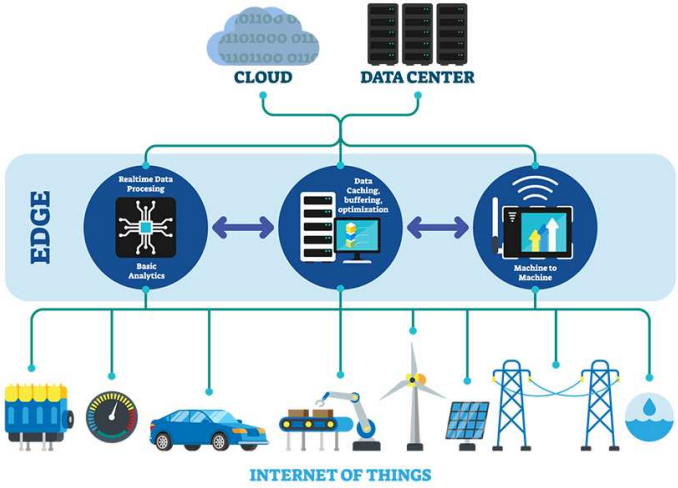
- Deep learning, RPC, monitoring, programming model, scheduling

3. Machine learning on switches

- P4, machine learning

4. Intermittent edge computing

- Microcontrollers, battery-free computing/communication



Source: mc.ai

If you are interested, please contact me at: lin.wang@vu.nl

More topics

1. Empirically evaluating the multicore scalability of Linux networking stack (or how good or bad is it at delivering full performance scalability) on DAS

2. Understanding performance of DPDK on DAS by benchmarking it thoroughly on DAS

3. How low can we go -- answering what is the lowest round trip time we can achieve between a pair of machines on DAS and why. You need to benchmark and explain every last nanosecond time spent.

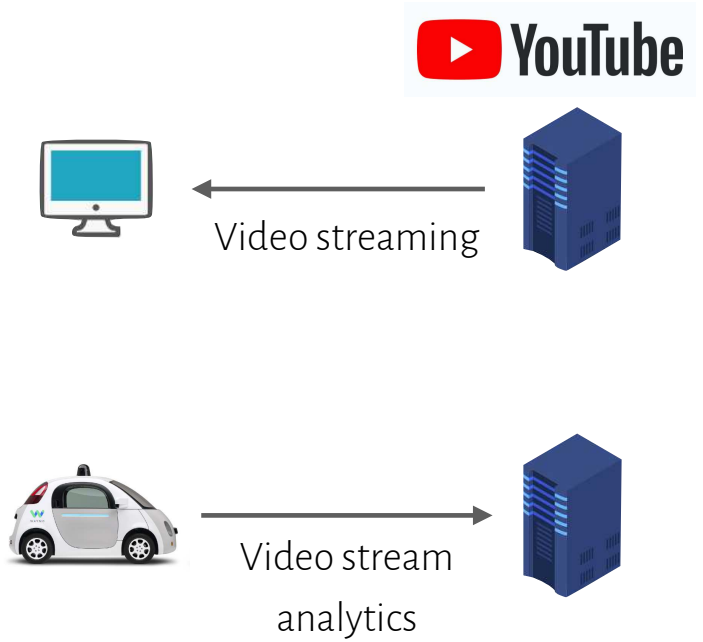
4. Designing and building a reliable “message” oriented protocol (instead of byte oriented TCP, a message oriented protocol would send a complete “message” and receive a complete “message”) - we will use the same ANP infrastructure for development and testing

More will be announced on Canvas!

Summary

Lecture 11: Beyond networking

- Video streaming
- Compression methods
- Video streaming protocol: RTP
- DASH
- ABR algorithms: BBR
- Video stream analytics
- Open research projects



Course summary

Project
ANP networking stack

Part I
2. Networking concepts
3. Linux networking internals
4. Multicore scalability
5. User space networking
6. RDMA

Part II
7. Forwarding and routing
8. Software defined networking
9. Programmable data plane
10. Data center networking
11. Beyond network: video streaming