Plan of the module

I. Network coding
   1. Basics
   2. Application to wireless networks
   3. Application to distributed storage systems

II. Video streaming
   1. General overview
   2. Client side
   3. Server side: CDN
   4. Examples of optimization at the client side: the case of mobile clients
   5. Example of optimization at the server side: the cloud resources *(R. Aparicio-Pardo)*

Sources

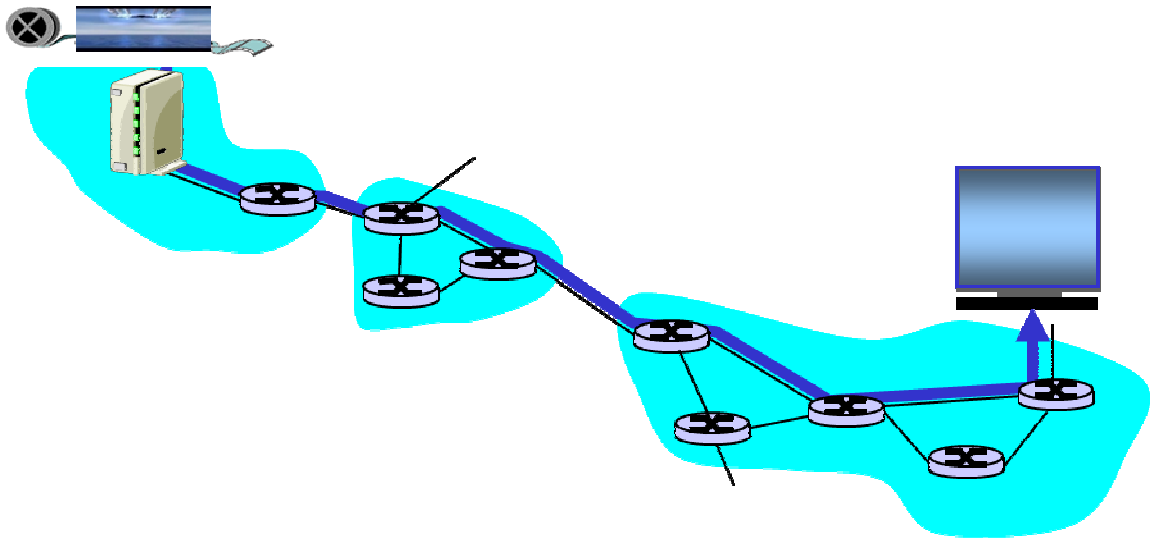
- G. Urvoy-Keller and L. Sassatelli, *Video streaming*, lecture M2 Ubinet, UNS, 2014
- M. Siekkinen, *Video streaming*, lecture Aalto Univ., 2014
- H. Riiser, *Adaptive Bitrate Video Streaming over HTTP in Mobile Wireless Networks*, PhD thesis, Univ. of Oslo, 2013
- Z. Morley Mao, *Multimedia Networking*, lecture, lecture Univ. of Michigan
- *Video Communications and Video Streaming Over Internet: Issues and Solutions*, lecture Sharif Univ. of Technology
- A. C. Begen and T. Stockhammer, *HTTP Adaptive Streaming: Principles, Ongoing Research and Standards*, ICME 2013
- S. Akhshabi, S. Narayanaswamy, A. C. Begen, C. Dovrolis, *An experimental evaluation of rate-adaptive video players over HTTP*, Elsevier SP, Oct. 2011
- And others along the way…

Outline

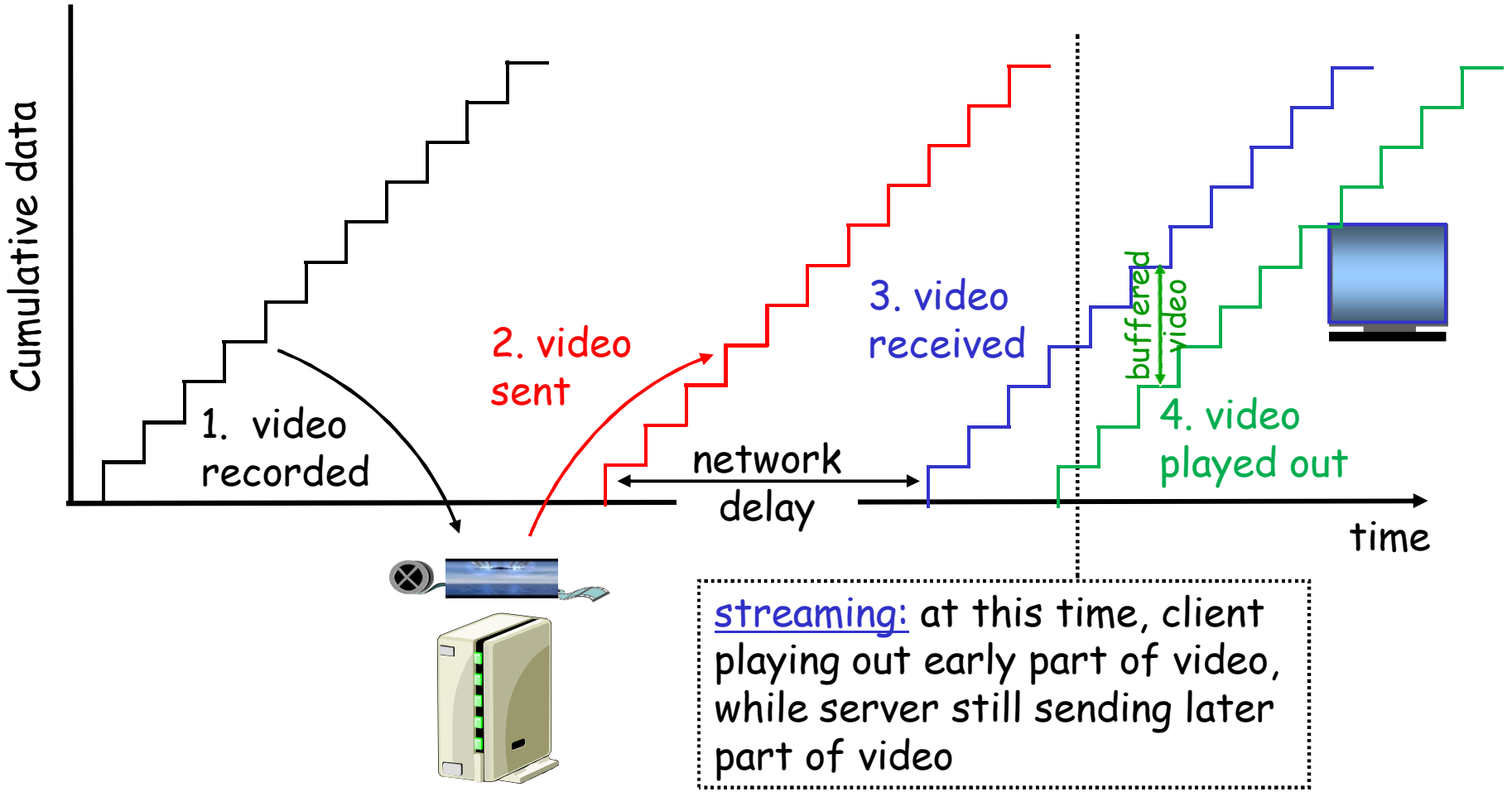
- A general overview of HTTP-based video streaming
  - Principle
  - Current traffic figures
  - Basic architecture
- Client side
  - Quality of Experience: the metrics that matter
  - Various HTTP-based streaming strategies
  - HTTP Adaptive Streaming (HAS)
- Server side
  - Content Distribution Networks (CDNs)
  - Impact of CDN management on QoE

What is streaming?

- **Streaming**: the media stored at a source is transmitted to the client. The client playout starts before all the data has arrived.
- Timing constraint for still-to-be transmitted data: in time for playout
- Notation:
  - "video rate"="bitrate"=the rate at which the played video has been encoded
  - "bandwidth"="downloading rate"="sending rate"= the data rate achieved between the source and destination nodes



Streaming Stored Multimedia: What is it?



Video Streaming Applications

1) Live, interactive video

   Video teleconferencing, video phones, etc.

2) Live, non-interactive video

   Course lectures, news, sporting events, conferences

3) Stored, non-interacive video

   Movies, distance learning, Web videos, etc.

- In this course, we focus on 3): streaming of stored data.

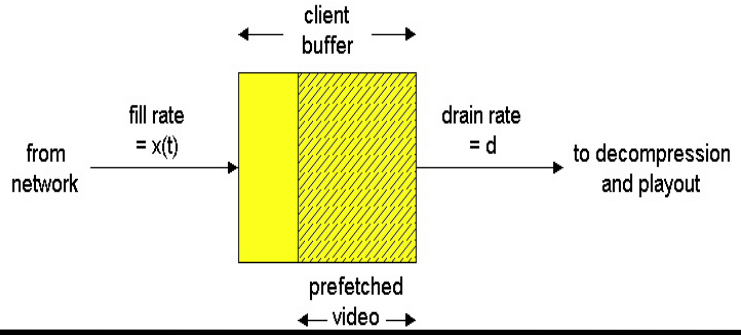
QoS constraints for video streaming

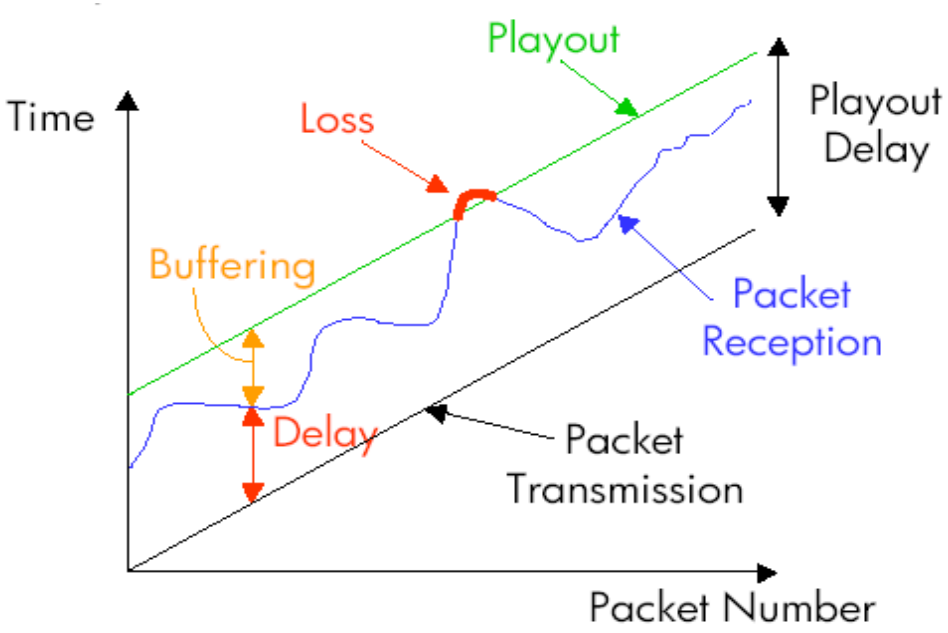
- Problem: Internet only offers best-effort service
- No guarantees on:
  - Bandwidth
    - Can not reserve bandwidth in Internet today
    - Available bandwidth is dynamic
    - If transmit faster than available bandwidth, then congestion occurs, packet loss, and drop in video quality
    - If transmit slower than available bandwidth, then sub-optimal video quality
    - Goal: Match video bit rate with available bandwidth
  - Loss rates
  - Delay jitter
- Specifically, these characteristics are **unknown** and **dynamic**
- Goal: Design a system to reliably deliver high-quality video over the Internet

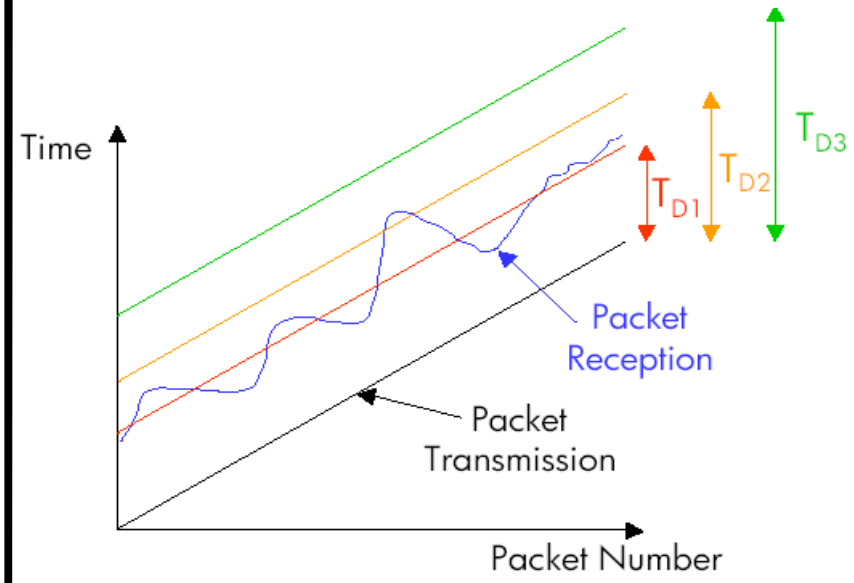
Overcoming Internet best-effort quality: Playout buffer

- Meant to compensate for bandwidth variations, delay jitter and enables retransmission of lost packets.
- Corresponds to adding an offset to the playout time of each packet
  - If (packet delay < offset) then OK
  - Buffer packet until its playout time
  - If (packet delay > offset) then problem
- Playout buffer typically 30 secs to several minutes of playback

Overcoming Internet best-effort quality: Playout buffer







Effect of different buffering delays

Outline

- A general overview of HTTP-based video streaming
  - Principle
  - Current traffic figures
  - Basic architecture
- Client side
  - Quality of Experience: the metrics that matter
  - Various HTTP-based streaming strategies
  - HTTP Adaptive Streaming (HAS)
- Server side
  - Content Distribution Networks (CDNs)
  - Impact of CDN management on QoE

Sandvine traffic reports

- Following info is extracted from Sandvine reports
  - Semiannual reports on Internet evolution in terms of traffic and usage too
- Sandvine is a Canadian company
  - Customers are tier-1 and tier-2
  - Products to monitor traffic, optimize content delivery
- Idea is to get an understanding of the role of video delivery in the Internet

Fixed Access: traffic breakdown

	Upstream		Downstream		Aggregate	
Rank	Application	Share	Application	Share	Application	Share
1	BitTorrent	24.53%	Netflix	34.21%	Netflix	31.09%
2	HTTP	14.27%	YouTube	13.19%	YouTube	12.28%
3	SSL	6.54%	HTTP	11.65%	HTTP	11.84%
4	Netflix	6.44%	iTunes	3.64%	BitTorrent	5.96%
5	YouTube	5.52%	SSL	3.42%	SSL	3.80%
6	Skype	2.23%	BitTorrent	3.40%	iTunes	3.33%
7	Facebook	2.17%	MPEG	2.85%	MPEG	2.62%
8	FaceTime	1.50%	Facebook	1.99%	Facebook	1.83%
9	Dropbox	1.20%	Amazon Video	1.90%	Amazon Video	1.82%
10	iTunes	1.15%	Hulu	1.74%	Hulu	1.58%
		64.40%		76.24%		74.58%

sandvine

Table 2 - Top 10 Peak Period Applications - North America, Fixed Access

	Upstream		Downstream		Aggregate	
Rank	Application	Share	Application	Share	Application	Share
1	BitTorrent	33.20%	YouTube	19.27%	YouTube	17.38%
2	HTTP	10.07%	HTTP	17.46%	HTTP	16.26%
3	YouTube	7.67%	BitTorrent	11.10%	BitTorrent	14.71%
4	SSL	5.63%	SSL	6.19%	SSL	6.10%
5	Skype	4.54%	Facebook	3.88%	Facebook	3.95%
6	Facebook	4.29%	RTMP	3.66%	RTMP	3.27%
7	eDonkey	3.64%	MPEG	3.54%	MPEG	3.21%
8	Dropbox	2.11%	Netflix	3.23%	Netflix	2.98%
9	MPEG	1.51%	Flash Video	2.37%	Flash Video	2.17%
10	iTunes	1.30%	iTunes	2.23%	iTunes	2.08%
		72.66%		70.69%		70.01%

sandvine

Table 6 - Top 10 Peak Period Applications - Europe, Fixed Access

- Video streaming dominates almost everywhere
  - Netflix clearly the biggest (31%) in North America followed by YouTube (12%)
  - YouTube is the largest in Europe (17%) as well as in Latin America (26%)
    - Netflix gaining ground rapidly
  - BitTorrent traffic still largest (27%) in Asia-Pacific
- P2P has declined sharply (except in Asia-Pacific)
  - Now less than 10% of total traffic, was 31% in 2008

Streaming user behavior

- Deeper analysis of fixed access usage in the US
- Classified into three types of streaming service users
  - 15 % of users contribute 54 % of traffic
  - 15 % less active contribute 0.5 % of traffic

Top 15th percentile

15th – 85th percentile

Bottom 15th percentile

	"Cord Cutter"	Typical Subscriber	"Non-Streamer"
Mean Monthly Usage	212 GB	29 GB	4.5 GB
Mean Real-Time Entertainment Usage	153 GB	13 GB	40 MB
Streaming Share	72%	45%	1%
Average Hours of Streaming	100 !!	9	<1
Share of Total Traffic	53.9%	45.7%	0.5%

Mobile streaming traffic

	Upstream		Downstream		Aggregate	
Rank	Application	Share	Application	Share	Application	Share
1	Facebook	26.95%	YouTube	17.61%	YouTube	17.26%
2	SSL	12.49%	Facebook	14.03%	Facebook	14.76%
3	HTTP	11.80%	HTTP	12.70%	HTTP	12.59%
4	YouTube	3.77%	MPEG	8.64%	MPEG	7.77%
5	Instagram	3.47%	SSL	6.52%	SSL	7.25%
6	BitTorrent	2.09%	Google Market	5.27%	Google Market	4.78%
7	MPEG	1.70%	Pandora Radio	5.15%	Pandora Radio	4.72%
8	Pandora Radio	1.61%	Netflix	5.05%	Netflix	4.55%
9	Gmail	1.61%	Instagram	3.49%	Instagram	3.49%
10	iCloud	1.56%	iTunes	3.10%	iTunes	2.84%
		65.50%		78.46%		77.17%

sandvine

Table 4 - Top 10 Peak Period Applications - North America, Mobile Access

- Real-time entertainment (read: streaming) is clearly the most dominant traffic category (over 30%) in North America, Europe, and Asia-Pacific (over 40%)
  - Comes second in South America behind Social Networking
  - Very small share of traffic in Africa

Role of video streaming in the Internet

- Overall, it is dominant and growing fast
- Forecasts: Globally, IP video will represent 80% of all traffic by 2019, up from 67% in 2014. [Cisco visual index 2014-2019]
- Mobile streaming growth will be accelerated by
  - continuing 4G deployments
  - evolution towards 5G (1000x capacity)
- The importance is huge to ISPs (both fixed and mobile)
- Lots of money involved in the business
  - Video and CDN providers

A few words about video compression

Examples:

- MPEG 1 (CD-ROM) 1.5 Mbps
- MPEG2 (DVD) 3-6 Mbps
- MPEG4 (often used in the Internet, < 1 Mbps)

New trends:

- **Ultra High Definition, a.k.a. 4K** (because it packs 4 times as many pixels as 1080p): new identified trend in video distribution at the 2014 CES.
- Main content provider like Netflix and YouTube plan to offer it, and the demand for streaming is expected to increase because such a resolution does not fit onto a Blu-ray anymore.
- Required bandwidth for 4K videos: **15Mbps** -> the residential access is expected to quickly grow -> **4K is foreseen to put a heavy strain on ISPs**

Extensions:

- Layered (scalable) video
  - adapt layers to available bandwidth
  - Multiple Description Codes (MDC)
  - Scalable Video Coding (SVC): Annex G extension of the H.264/MPEG-4 AVC video compression standard

Outline

- A general overview of HTTP-based video streaming
  - Principle
  - Current traffic figures
  - Basic architecture
- Client side
  - Quality of Experience: the metrics that matter
  - Various HTTP-based streaming strategies
  - HTTP Adaptive Streaming (HAS)
- Server side
  - Content Distribution Networks (CDNs)
  - Impact of CDN management on QoE

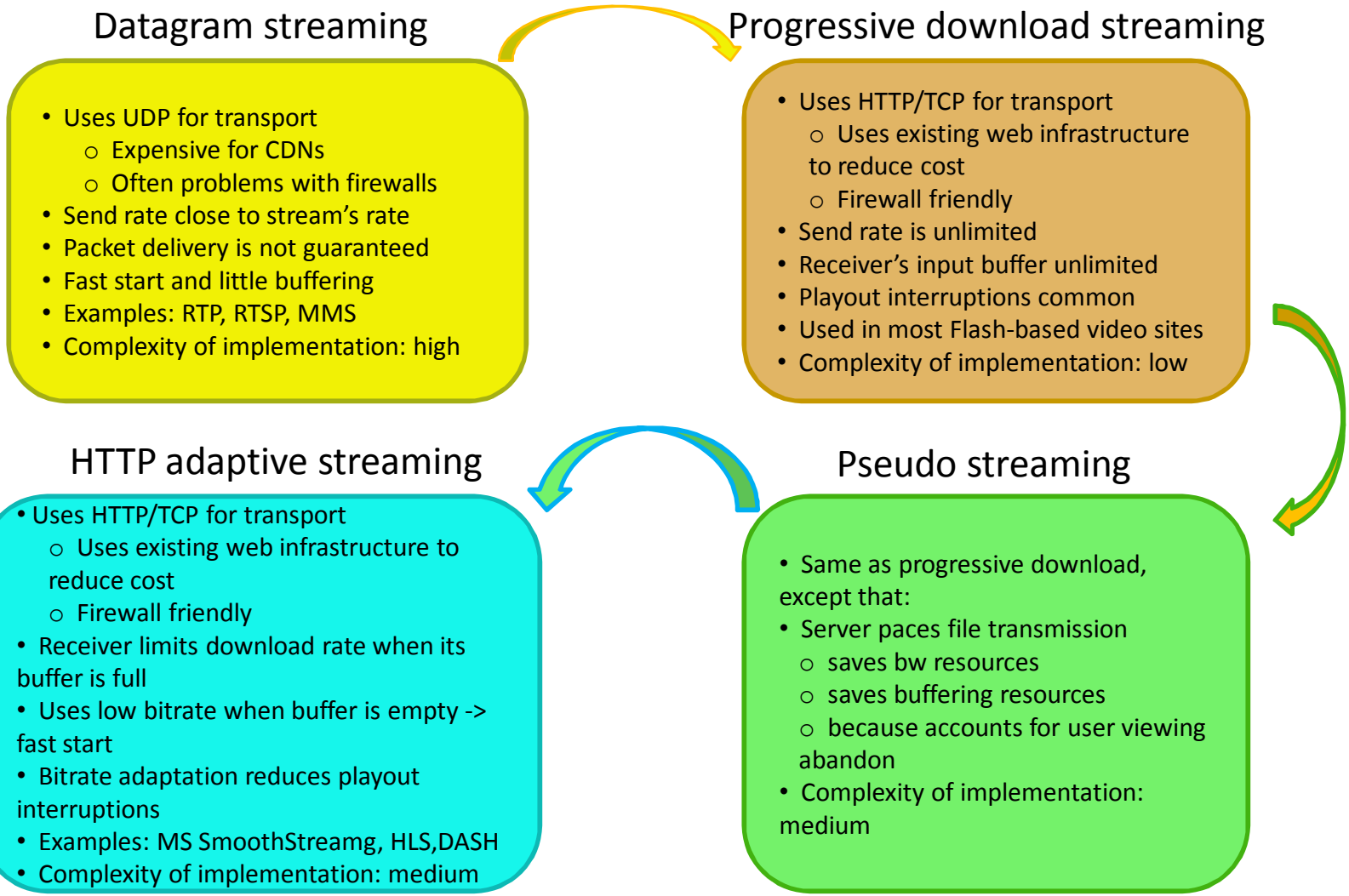
A brief history of video streaming

- Used to be admitted that real-time video over a best-effort network like the Internet would have to be streamed using a datagram protocol: giving the streaming application packet-level control
- -> this meant in practice that it should be carried by UDP, not TCP:
  - RTP streaming systems can have full control over packet retransmission -> can trade packet loss for delay (e.g., audio and video conferencing). Problem with RTP: new media codecs cannot easily be supported.
  - RTP requires out-of-band signaling (RTSP-Real Time Streaming Protocol , and SIP)
- In addition to this, protocols based on datagram streaming suffer from:
  - Packet-level control -> very complicated implementation as needs to deal with flow and congestion control, packet loss, out-of-order delivery, etc.
  - Firewalls and NAT routers frequently cause problems with UDP
  - Higher cost of the infrastructure as Content Delivery Networks (CDNs) require specialized solutions for caching and load balancing (almost all deployed infrastructure optimizations target HTTP because of its massive popularity)
- => most of the industry adopted progressive download streaming using HTTP
- Its simplicity has made it the streaming protocol most commonly used today (e.g., by YouTube).

Whereby HTTP-based streaming

- With this approach, the client simply downloads a media stream as a file in a normal media container format such as MP4, and plays back the video while it is downloading.
- Benefits:
  - firewall traversing
  - all CDNs support it (can automatically take advantage of transparent web caching)
- Downsides:
  - playout interruptions are more likely to occur
  - larger buffer is required (limiting progressive streaming's suitability for real-time communication)
  - multicast is not an option (no longer considered a big loss, since there is no widely available multicast infrastructure)
- -> The biggest arguments for datagram protocols for non-interactive streaming is now mostly irrelevant

Evolution from datagram streaming to adaptive HTTP streaming



Outline

- A general overview of HTTP-based video streaming
  - Principle
  - Current traffic figures
  - Basic architecture
- Client side
  - Quality of Experience: the metrics that matter
  - Various HTTP-based streaming strategies
  - HTTP Adaptive Streaming (HAS)
- Server side
  - Content Distribution Networks (CDNs)
  - Impact of CDN management on QoE

Quality of Experience in video streaming

- QoE: this is the quality the client **perceives**
- -> subjective, and may be given by complex functions of the QoS metrics
  - **Video quality, ratio of rebufferings, startup delay, …**
- Different stakeholders
  - ISP: The one providing client's access to Internet
    - often the one customers blame for poor QoE
  - Video service provider: YouTube, Netflix, …
  - CDN operator: YouTube, Akamai, Limelight networks, Level 3, …
  - Client: you, the one who matters
  - Device manufacturer: hardware performance
  - Client software provider: OS and player software
- What determines QoE, i.e. who is in control?
  - Jointly the behavior of all the above stakeholders
  - Outcome of a rather complex process

QoE metrics for video streaming

- Buffering events
  - If playback buffer runs dry, playback pauses -> annoys the user
  - Happens if download rate is smaller than encoding rate for too long
  - Initial buffering attempts to avoid such events
  - Studies suggest this is the most important reason for abandoning viewing
- Initial startup delay (a.k.a. joining time)
  - Time it takes for playback to begin after clicking play
  - Not as fatal as buffering events for audience retention

QoE metrics for video streaming

- More “traditional” quality metrics
  - Video resolution, frame rate, target encoding rate (quality)
  - PSNR
  - Subjective metrics (e.g. MOS)
  - ...
- Adaptive streaming metrics
  - Rate and amplitude of quality switches
  - Avg quality

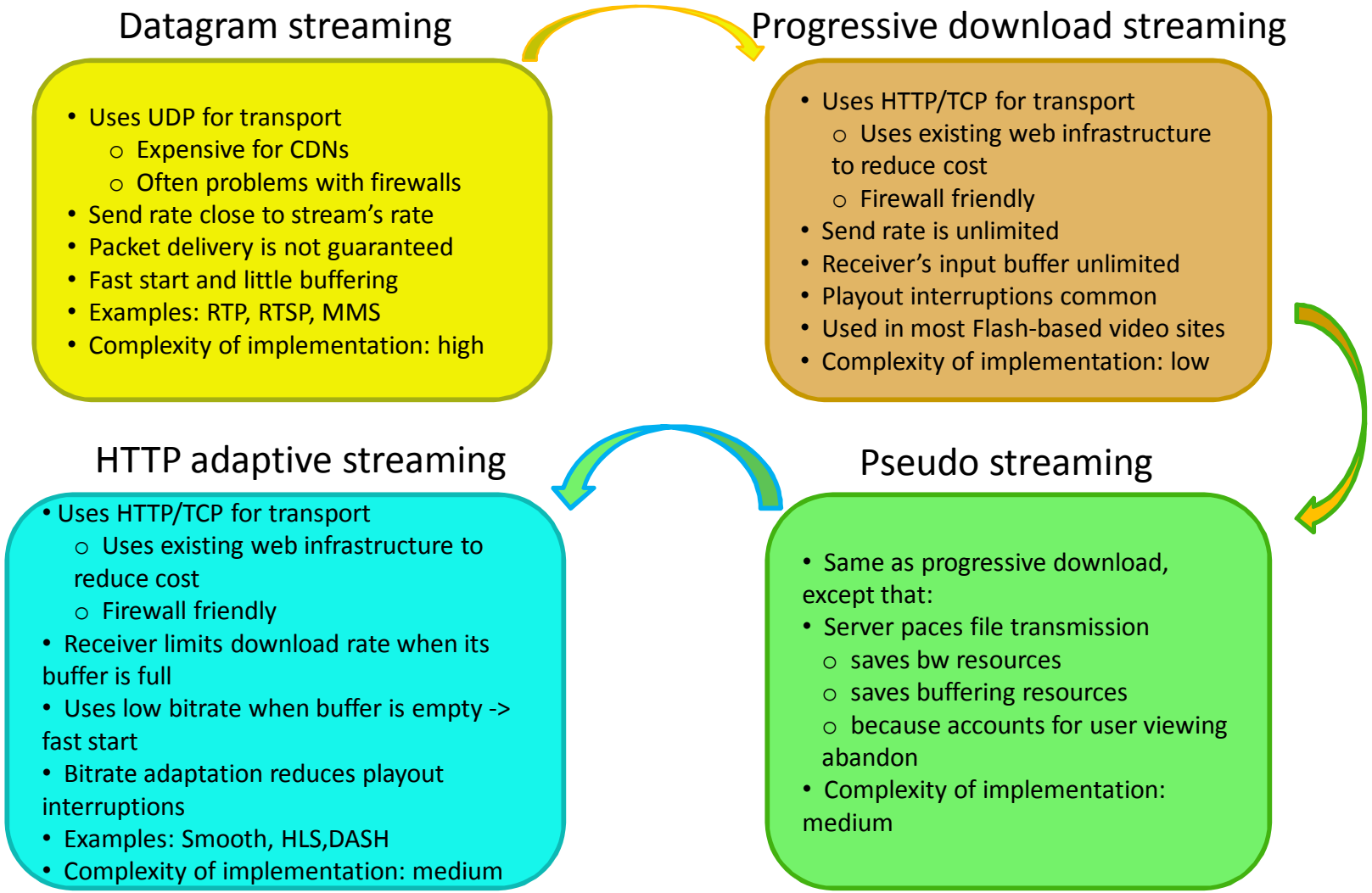
Delivery related QoE metrics

- Several factors influence the probability of experiencing buffering event
  - Amount of initially buffered content
  - Instantaneous TCP bulk transfer capacity from server to client
  - Content download strategy, i.e. streaming strategy

Outline

- A general overview of HTTP-based video streaming
  - Principle
  - Current traffic figures
  - Basic architecture
- Client side
  - Quality of Experience: the metrics that matter
  - Various HTTP-based streaming strategies
  - HTTP Adaptive Streaming (HAS)
- Server side
  - Content Distribution Networks (CDNs)
  - Impact of CDN management on QoE

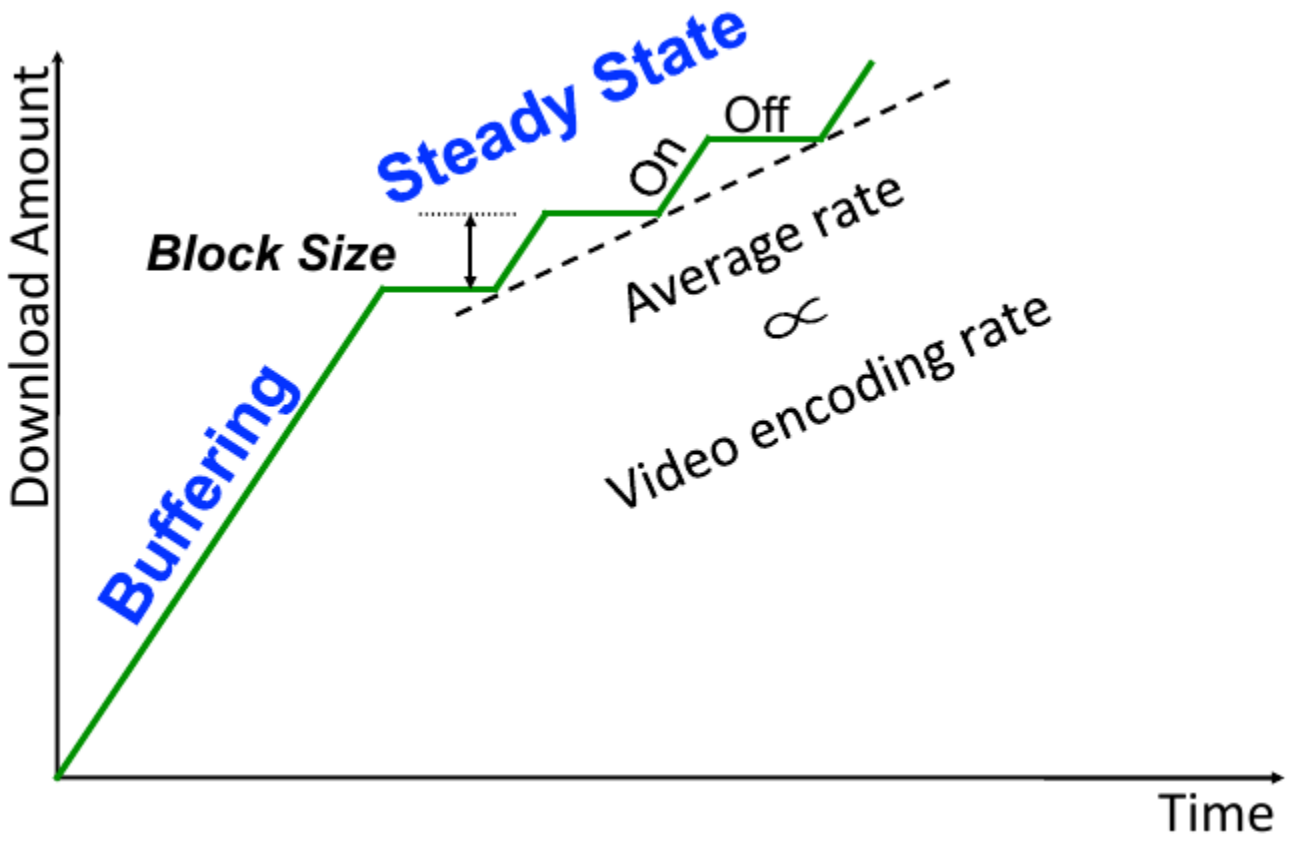
Evolution from datagram streaming to adaptive HTTP streaming



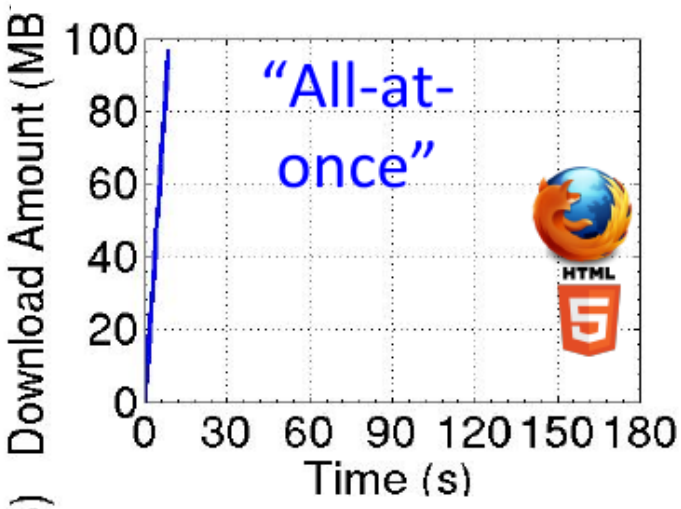
Streaming strategy

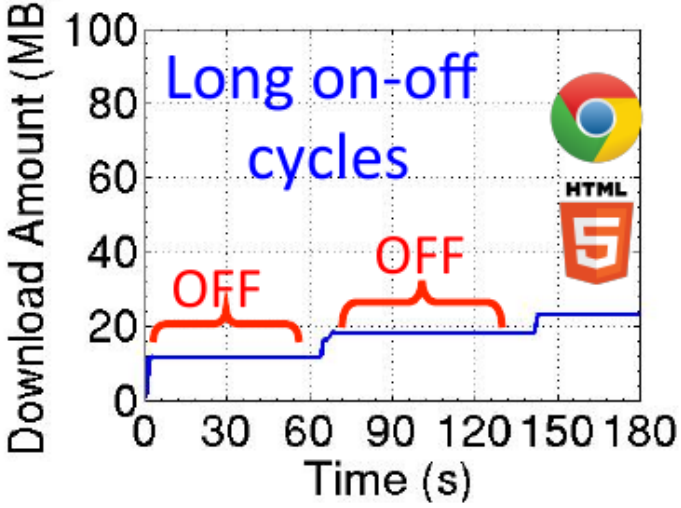
- Turns out that a number of different strategies are used
  - Interplay between server and client application
- We'll borrow some results from earlier studies
  - *A. Rao et al: Network Characteristics of Video Streaming Traffic. CoNEXT 2011.*
  - *M. Hoque et al: Mobile Multimedia Streaming Techniques: QoE and Energy Saving Perspective. Pervasive and Mobile Computing 2014.*

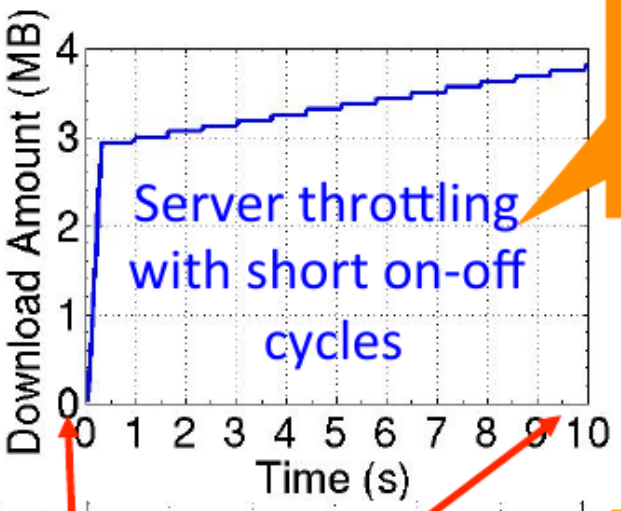
Generic behavior: « pseudo streaming »



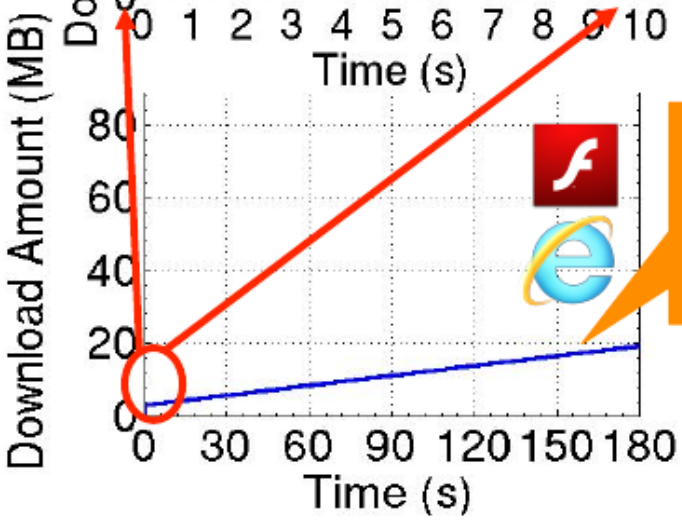
Identified streaming strategies







Server enforces a download rate control (e.g: 1.5x or 2x encoding rate)

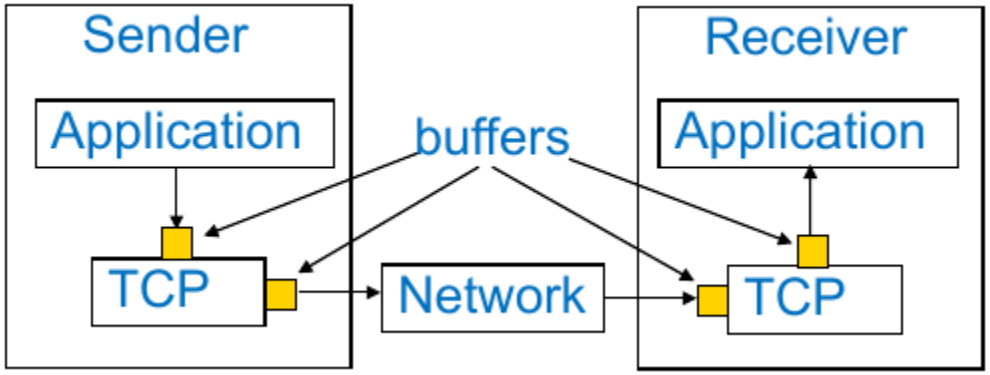


Also *encoding rate* streaming possible: dl at encoding rate in steady state phase

Streaming strategies vastly different

Who chooses the strategy?

- Let's quickly review the interplay between applications and TCP



Who chooses the strategy?

- All-at-once and server throttling are **enforced by server**
  - But all-at-once may be explicitly requested by client application (HTTP request parameter)
- Long ON-OFF cycles are (usually) **caused by client application**
  - Client application periodically stops reading from TCP socket
  - TCP receive buffer fills up -> TCP flow control activates and forbids server side TCP to send more packets
  - Transfer paused until application reads again from socket
- Encoding rate streaming seems to be **unintentional**
  - Client's application buffer fills up -> receive TCP buffer fills up -> TCP flow control activates
  - Client application reads data from socket at encoding rate -> new data transferred at this rate

Streaming strategies in use

Service	YouTube			Netflix
Container	Flash	HD (Flash)	HTML5	Silverlight
IE 9	Short	No	Short	Short
Firefox	Short	No	No	Short
Chrome	Short	No	Long	Short
iOS (native)	-	-	Based on encoding rate	Short
Android (native)	-	-	Long	Long

Streaming strategy differs with application type and container
Clients do not throttle the rate of transfer in wired and wifi!

Strategies used by mobile clients

Table 3
Streaming techniques for popular video streaming services to mobile phones of three major platforms. The selection of a streaming technique does not depend on the wireless interface being used for, rather depends on the player, video quality, device and the video service provider.

	iPhone4S (iOS 5.0)	iPhone5 (iOS 7.0)	Galaxy S3/Galaxy S3 LTE (Android-4.0.4)		Lumia825 (WP8)
YouTube					
Streaming	*(App)* Throttling Factor = 2.0	*(App)* Throttling Factor = 1.25	*(Flash)* Encoding rate(HD), Throttling(<HD) Factor = 1.25	*(App& HTML5)* ON–OFF-M	(App) Fast Caching
Quality	LD(240p), SD(360p), HD(720p)	LD(240p), SD(360p), HD(720p)	LD(240p), SD(360, 480p), HD(720,1080p)	LD(240p), SD(360, 480p), HD(720p)	SD(270p), HD(720p)
Container	MP4(360,720p)	MP4(360,720p) 3GPP(240p)	XFLV	MP4(>240p) WebM(>240p) 3GPP(270p)	MP4(720p) 3GPP(270p)
Vimeo					
Streaming	*(App)* HLS Chunk Size = 10s	*(App)* ON–OFF-M	*(App)* ON–OFF-S		*(App)* Fast Caching
Quality	240–720p	SD(270, 480p), HD(720p)	SD(270p), HD(720p)		HD(720p)
Container	MP4	MP4	MP4		MP4
Dailymotion					
Streaming	*(App)* Throttling Factor = 1.25	*(App)* HSL Chunk Size = 10s	*(App)* Fast Caching(288p), ON–OFF-S(>288p)		*(App)* Throttling Factor = 1.25
Quality	LD(240)	240–720p	SD(288, 480p), HD(720p)		SD(288p)
Container	MP4	MP4	MP4		MP4
Netflix					
Streaming	*(App)* HLS Chunk Size = 10s	*(App)* HLS Chunk Size = 10s	*(App)* ON–OFF-S		*(App)* MSS Chunk Size = 4s
Quality	240–720p	240–720p	HD(720p)		240–720p
Container	isma, ismv	isma, ismv	MP4		isma, ismv

Impact of streaming strategy on QoE

- Amount of buffered data directly relates to the probability of a buffering event
  - Level of protection against transient network issues
- But providers and clients need to consider other issues too
  - Unnecessarily served content upon abandoning: **half videos are abandoned before half viewing**
  - **->** Bandwidth waste
  - Capped data plans
  - Energy efficiency

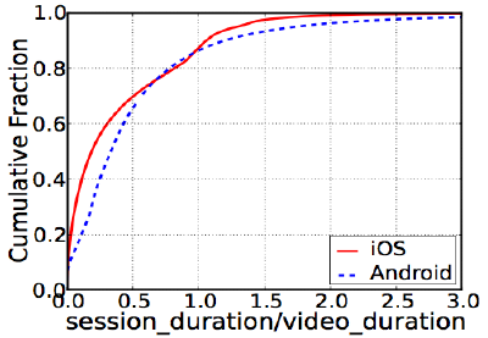
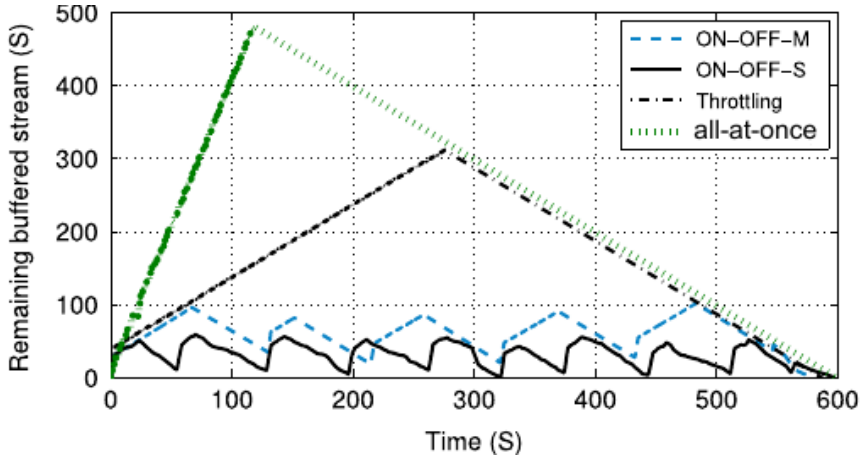


Fig. 1: Ratio Between Session Duration and Video Duration (CDF)



Strategy	No ON-OFF	Long ON-OFF	Short ON-OFF
Engineering Complexity	Not required	Explicit support at Application Layer	
Receive buffer occupancy	Large	Moderate	Small
Unused bytes on user interruption	Large amount	Moderate amount	Small amount

Y. Liu, F. Li, L. Guo, B. Shen and S. Chen, "A Comparative Study of Android and iOS for Accessing Internet Streaming Services," in PAM, Hong Kong, 2013.

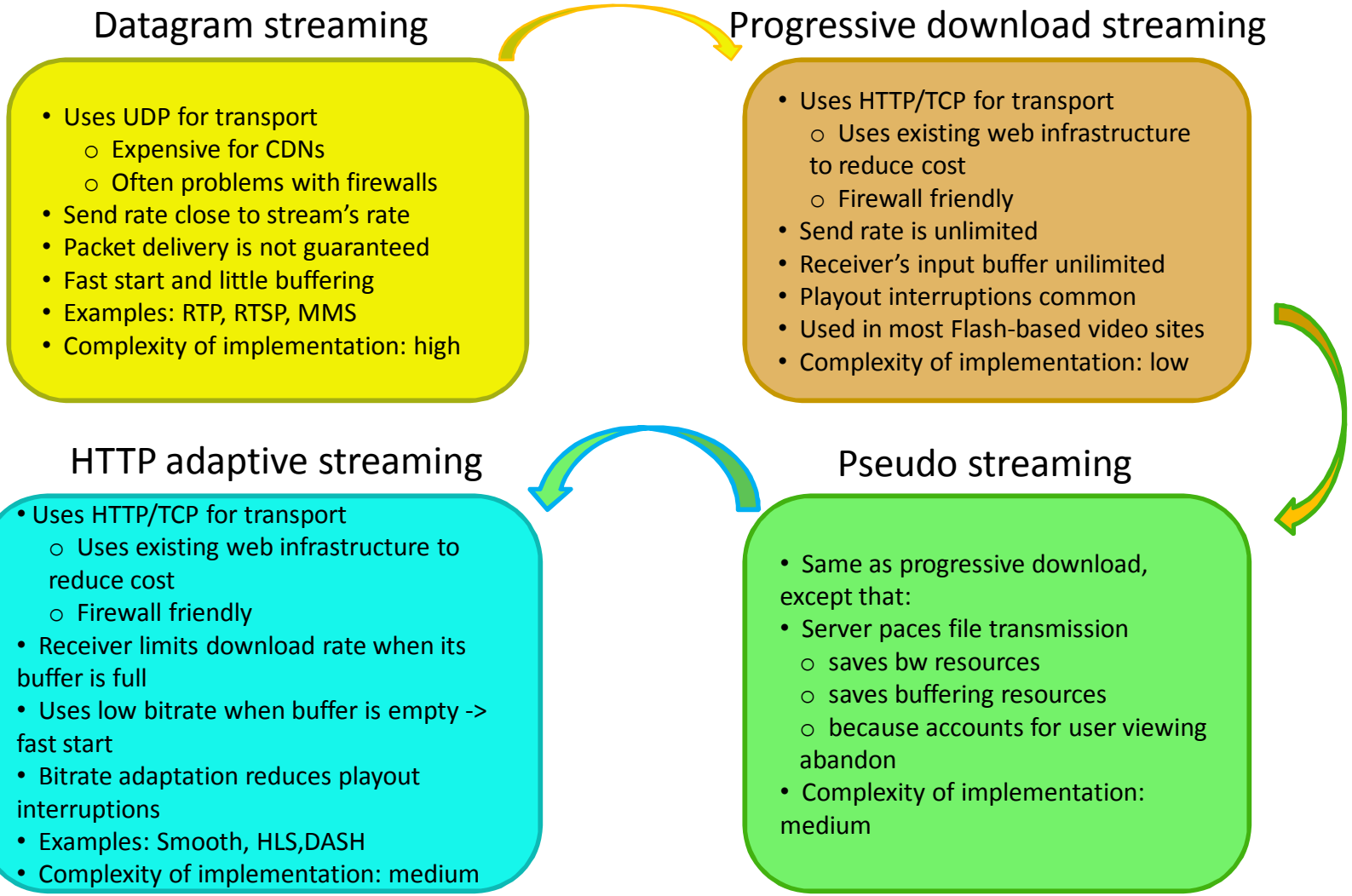
QoE summary

- Many metrics to capture quality
  - "Traditional" – QoS - metrics being replaced by video delivery based metrics - QoE
- QoE is crucial for video providers and telcos:
  - Direct impact on user engagement -> revenue (through ads etc.)
  - Currently lots of efforts to model and improve it
- Many factors influence QoE
  - Different stakeholders
  - Video CDN operations are a crucial one
  - Streaming strategy is another important one
- There is still room for improvement

Outline

- A general overview of HTTP-based video streaming
  - Principle
  - Current traffic figures
  - Basic architecture
- Client side
  - Quality of Experience: the metrics that matter
  - Various HTTP-based streaming strategies
  - HTTP Adaptive Streaming (HAS)
- Server side
  - Content Distribution Networks (CDNs)
  - Impact of CDN management on QoE

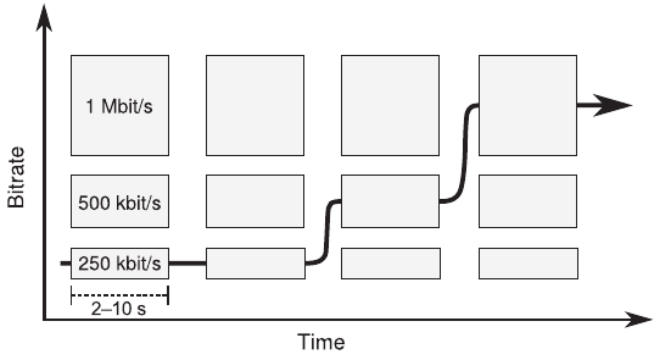
Evolution from datagram streaming to adaptive HTTP streaming



HTTP Adaptive Streaming: Motivation

- Idea: make the requested bitrate of the video fit the (possibly varying) network ressources
- For strained networks: wireless (4G/5G), and wired networks (4K exacerbates) – shared networks like HFC or DSL backhaul are likely to face bandwidth issues

- “available resources”: usually network bandwidth (can also possibly CPU load, battery capacity, and screen size)
- client-side adaptation by far the most popular in recent systems: all the information that is relevant when choosing which quality to use is available to the client
- HAS offers a solution for the most significant problem with streaming over HTTP: fluctuating bandwidth
- -> Being able to switch seamlessly to a stream with a lower bitrate whenever the bw or buffer is too low makes HTTP much more usable for video streaming, especially on mobile devices.
- -> HAS bound to generalize



HAS: principle

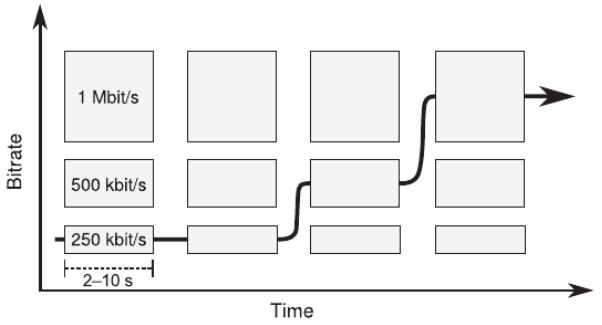
- A stream is split into a sequence of segments downloaded individually, instead of performing one large file download per stream.
- Each segment is typically 2–10 seconds of the stream
- The segment data can be either a multiplexing container format that mixes data from several tracks (audio, video, subtitles, etc.), or it can contain data from just a single track, requiring the receiver to download and process segments from several tracks in parallel.
- The video track is available in multiple different bitrates, each representing a different quality level. The quality can only change on segment boundaries.

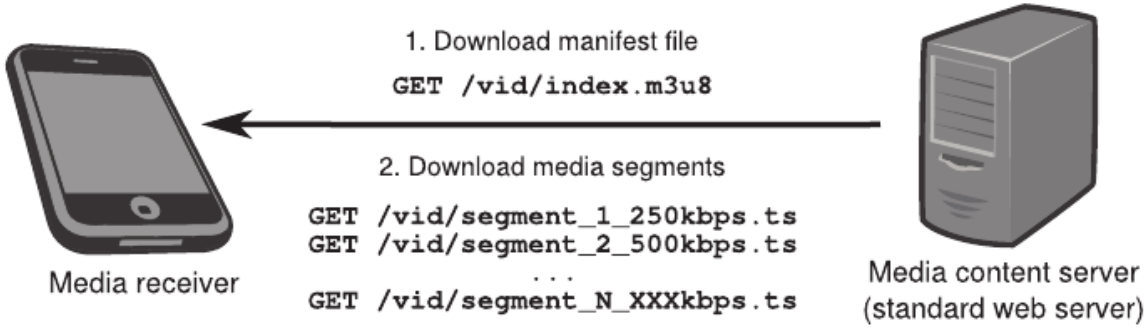
- HAS flavors: DASH, Microsoft Smooth Streaming, HLS,…
- -> DASH and HLS now
- How the client chooses the bitrate to request: up to its software, no need to make it uniform.

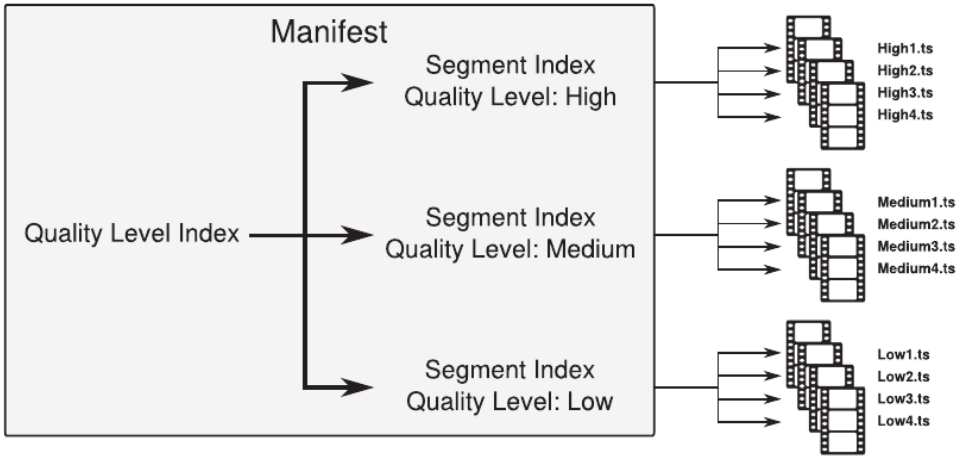
HAS: manifest file

- Each segment is separately downloadable using its own URL.
- Standard HTTP GET requests are sent for every segment, and the URLs for these requests contain what is needed to uniquely identify the segment to be downloaded (timestamp of the first video frame in the segment, bitrate values, …).
- To reduce the entire stream to a single URL, most adaptive formats use a manifest file to describe the stream's structure. The manifest file includes:
  - General info (total stream duration, encryption info, if it is VOD or Live,…)
  - Which types of streams are available (e.g., audio, video, subtitles, etc.)
  - URLs for each media segment, and info about the segments' durations and start times
  - Quality levels are available for each stream
- -> a media player needs only the URL to the manifest file to start playing video, because all the segment URLs will be known after it has downloaded and parsed the manifest

Workflow of HAS and manifest file







Performance of HAS

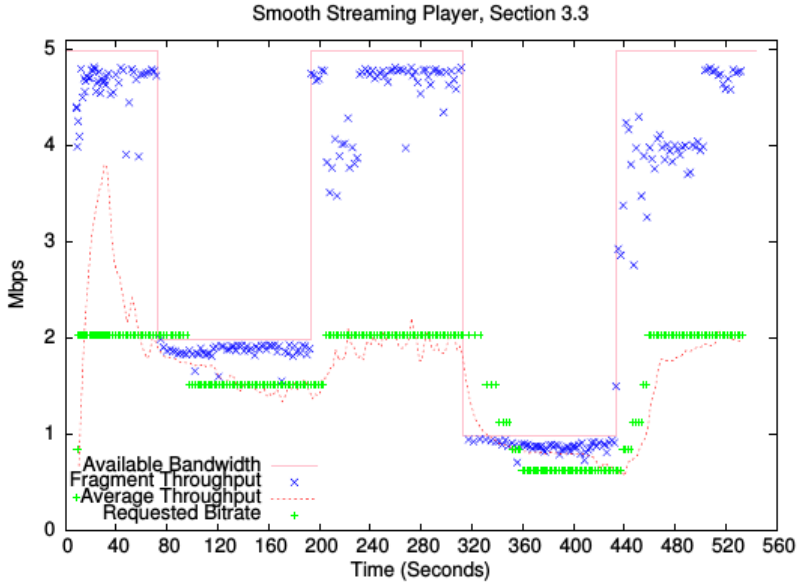


Figure 3: Per-fragment throughput, average TCP throughput and the requested bitrate for the video traffic under persistent avail-bw variations. Playback starts at around t=10 s, almost 3 s after the user clicked PLAY.

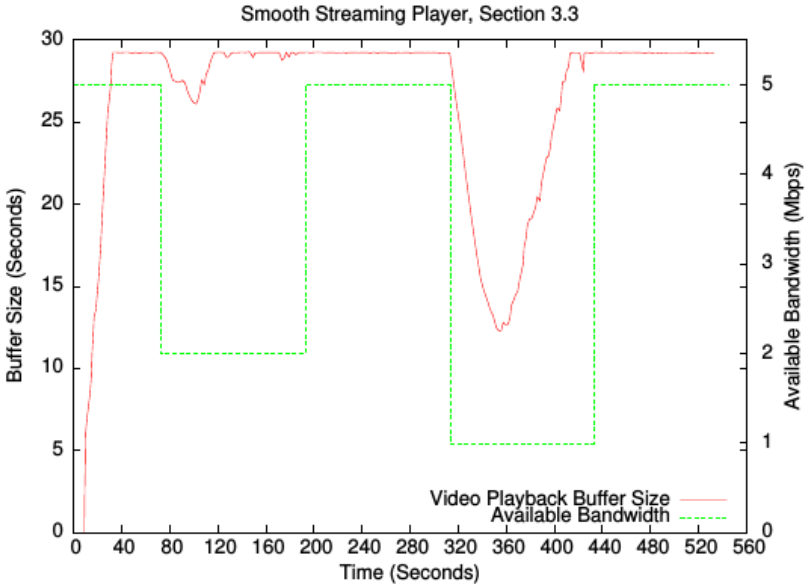


Figure 4: Video playback buffer size in seconds under persistent avail-bw variations.

Performance of HAS

- 1) 1st first player uses the highest profile (P2.75)
- 2) The 2 players could have shared the 4 Mbps bottleneck by switching to P1.52, however, they do not: player 2 oscillates between lower profiles.
- 3) The oscillations continue for both players.
- 4) Stable because lowest rate
- 5) The two players start oscillating in a synchronized manner.

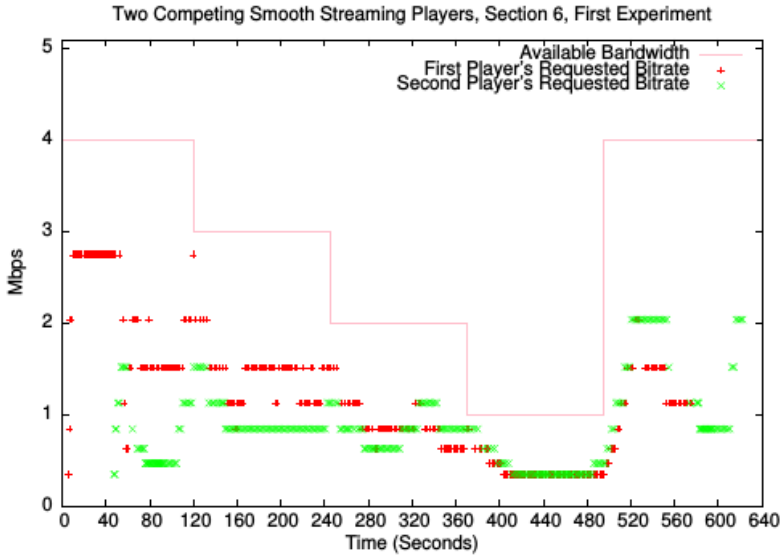
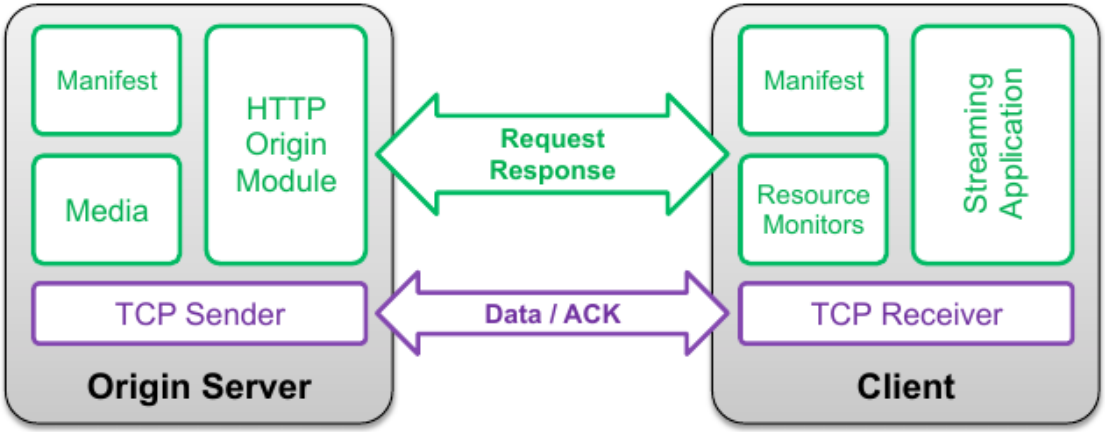


Figure 16: Two Smooth Streaming players compete for avail-bw. The players start the playback at around t=7 s and t=57 s, respectively.

Performance of HAS



- The interplay between TCP loop control and HAS loop control is intricate, specifically when the RDA bases its decision solely on the measured dl rate.
- -> Can generate inefficiency and unfairness
- New players (Netflix) are increasingly using the buffer state info.

Plan of the module

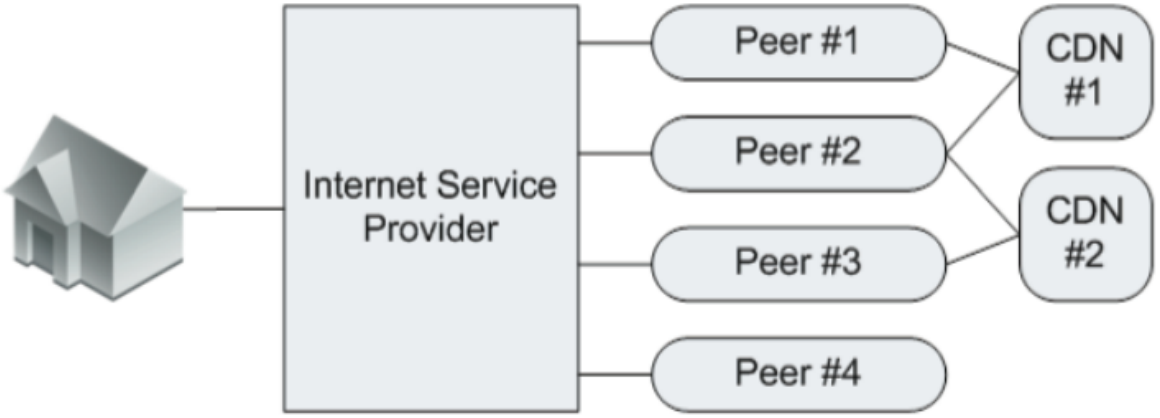
I. Network coding
   1. Basics
   2. Application to wireless networks
   3. Application to distributed storage systems

II. Video streaming
   1. General overview
   2. Client side
   3. Server side: CDN
   4. Examples of optimization at the client side: the case of mobile clients
   5. Example of optimization at the server side: the cloud resources *(R. Aparicio-Pardo)*

How is Internet video delivered?

- Typically it is not
  - A small set of centralized servers
  - Clients connecting to those servers
- Why not?
  - Inefficient: long distance between server and client -> delay and bandwidth issues
  - Expensive: requires high capacity servers
- Use of a (large) set of servers distributed near clients
  - Under the control of a company...
  - … that makes agreement with local ISP
  - CDN: Content Distribution Networks
  - E.g.: Akamai, Limelight but also Google and Netflix

Modern video delivery architecture



- Your ISP peers with other ISPs which connect to CDNs
- Several choices available
  - Several CDNs serving same content may exist
  - CDN possibly multihomed (servers connected by several of the peering ISPs)

How to choose from alternatives

- Q: Who and based on which criteria…
  - ...chooses the CDN?
  - ...chooses the ISP-level path?
- Partly you by using specific hardware
  - Apple TV, iPad, Android tablet, etc.
  - Visible to service provider through the user-agent string in HTTP
- Partly content provider
  - Can be a function of load, time, type of device,…
  - Example: Netflix serving the same episode of House of Cards at the same time through the same Internet connection
    - Apple TV -> Open Connect (Netflix's own CDN)
    - iPad -> Limelight CDN at off peak times, Open Connect at peak hours
- Partly CDN provider
  - Through which (peering) ISPs content is delivered
- Partly even your ISP
  - Makes an agreement with CDN provider

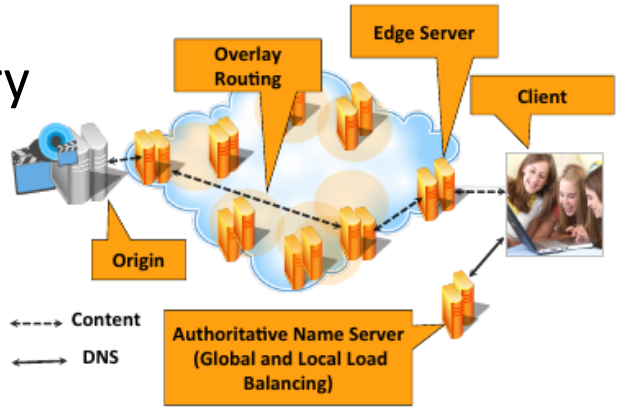
Bruce M. Maggs and Ramesh K. Sitaraman. Algorithmic Nuggets in Content Delivery. SIGCOMM Comput. Commun. Rev. 45, 3 (July 2015).

ALGORITHMIC NUGGETS IN CONTENT DELIVERY

Content Delivery Network (CDN)

- Top-3 objectives:
  - high reliability,
  - fast and consistent performance,
  - low operating cost.
- We present some of the main steps from the instant that a browser or other application makes a request for content until that content is delivered.

1st step: DNS query



- Request for content => starts with a DNS query
- Query forwarded to the CDN's authoritative name server.
- It decides which of the CDN's clusters to serve the content from: decision with a variant of the ***stable marriage***
- Then a second set of name servers decide which particular web server or servers within the cluster will serve the content.
  - Within the cluster, load is managed using a **consistent hashing** algo
  - the client's application can issue the request to the web server
- The web servers that serve content to clients are called edge servers
- Akamai's CDN currently has over 170,000 edge servers located in over 1300 networks in 102 countries and serves 15-30% of all Web traffic.

2nd step: HTTP request

- When an edge server receives an HTTP request, it checks to see if the requested object is already present in the server's cache. If not, the server begins to query other servers in order:
  - other servers in the same cluster, who share a LAN.
  - If none of these servers have the object, it may ask a "parent" cluster.
  - If all else fails, the server may issue a request for the object to an origin server operated by the content provider, who is a customer of the CDN.
- As new objects are brought into the CDN's edge server, it may be necessary to evict older items from the cache: ***Bloom filters*** are used to decide which objects to cache and which not to
- Using the CDN servers as relays in an "***overlay routing network***". The idea is that exploring multiple overlay paths between the origin and the edge servers and picking the best ones can provide paths with lower latency and higher throughput.
- For fault tolerance, compute the solution independently on several machines, and then to apply a ***leader-election algorithm*** to determine which machine should distribute its solution to other CDN components

Choosing the Cluster

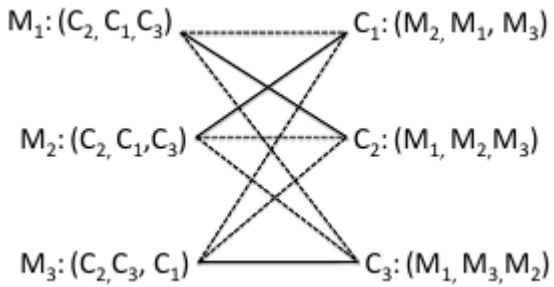
- Global load balancing is the process of mapping clients to the server clusters of the CDN.
- Rather than making load balancing decisions individually for the billions of clients, cluster assignments at the granularity of map units.
- A map unit: < IP address prefix, traffic class >
  - Tens of traffic classes such as video, web content, applications, software downloads, etc., each of which has distinctive properties.
  - Tens of millions of map units to capture all of the clients and traffic classes of the trillions of requests per day served by Akamai.

Choosing the Cluster: Global Load Balancing

- The goal of global load balancing is to assign each map unit M_i, $1 \leq i \leq M$, to a server cluster C_j , $1 \leq j \leq N$.
- For each map unit, the candidate clusters are ordered in descending order of preference
- Likewise, each server cluster C_j has preferences regarding which map units it would like to serve.
  - For example, a cluster deployed in an upstream provider ISP may prefer to serve clients in the ISP's downstream customer ISPs.
- The goal of global load balancing is to assign map units to clusters such that preferences are accounted for and capacity constraints are met.

Choosing the Cluster: Stable Allocations

- Classical algorithmic paradigm for assigning map units to clusters in global load balancing.
- Finding a stable marriage is achieved by a simple and distributed algorithm called the Gale-Shapley algo, works in rounds
- The solution provides each map unit with the most preferred cluster possible in any stable marriage
  - fits the CDN's mission of maximizing performance for clients.



A stable marriage (marked in bold) is a matching of map units to clusters such that no unmatched pair prefer each other over their matched partners.

Choosing the Cluster: Stable Allocations

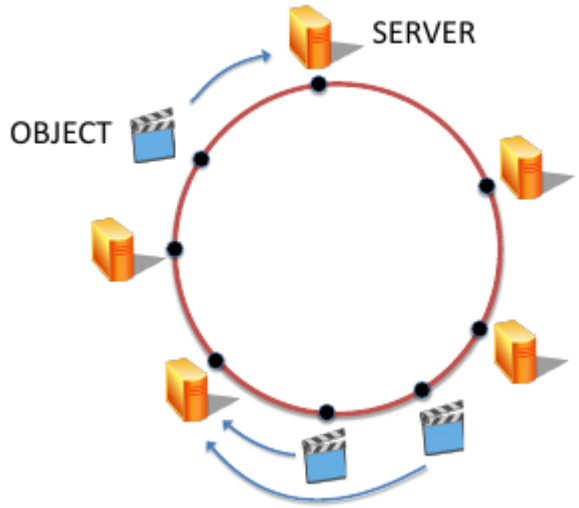
- Extension for CDN:
  - Unequal number of map units and clusters
    - Tens of millions of map units versus thousands of clusters
  - Partial preference lists: unnecessarily expensive to measure and rank every cluster for each map unit. It suffices to rank only those clusters that are close.
  - Modeling integral demands and capacities: canonical stable marriage problem considers unit value demands and capacity, it can be extended to the case in which capacity and demand are expressed as arbitrary integers. For each map unit, we estimate the content access traffic generated by its clients and represent that as a demand value. Likewise, for each cluster, we can estimate its capacity, which is the amount of demand it can serve.
- Challenges:
  - Complexity and scale: load balancing across tens of millions of map units and thousands of clusters for over a dozen traffic classes
  - Time to solve: the map unit assignment must be recomputed every 10 to 30 seconds, as network performance and client demand change on short time scales.
  - Demand and capacity estimation: tight feedback loop to estimate demand and capacity based on past history
  - Incremental and persistent allocation.

Load balancing within a single cluster: from basic hashing…

- Hashing: remember the basics
- When a hash table is used in a sequential computer program, an arbitrary subset $S \subset U$ of $O(n)$ objects of interest are inserted into the buckets of B: each element $x \in S$ is inserted into bucket $h(x)$.
- An array of linked lists can be used to represent the buckets: insertion, removal and testing can be implemented in a straightforward manner so that the expected time per operation is constant.
- In the CDN setting, an object is a file such as a JPEG image or HTML page, and a bucket is the cache of a distinct web server.
- But: a server (bucket) may fail. The class of hash functions like $h(x)$ above does not deal well with that:
  - Simply remapping all of the objects in the lost bucket to another bucket is not ideal because then one bucket stores double the expected load.
  - Balancing the load by renumbering the existing buckets and rehashing the elements using a new hash function has the disadvantage that many objects will have to be transferred between buckets.

Load balancing within a single cluster: … to Consistent hashing

- Consistent hashing solves this problem in a clever way. Consistent hashing first maps both objects and buckets (servers) to the unit circle. An object is then mapped to the next server that appears on the circle in clockwise order.
- If a bucket fails: the objects that were formerly mapped to the bucket are instead mapped to the next bucket that appears in clockwise order on the unit circle.
- If a new bucket is added: the only objects that have to be moved are those that are mapped to the new bucket.



Load balancing within a single cluster: Consistent hashing

- Popular Objects: a single object may be so popular that it is not possible for a single server within a cluster to satisfy all of the requests -> must be served by more than one server.
- -> A straightforward extension to consistent hashing would be to map a popular object to the next *k* servers that appear in clockwise order on the unit circle, where *k* is a function of the popularity of the object.
  - One disadvantage of this approach, however, is that if two very popular objects happen to hash to nearby positions , a lot of buckets they map to will overlap
- -> the CDN uses a separate mapping of the buckets to the unit circle for each object
- In practice: Although an object is a single file to be served by a web server, the CDN does not hash each object independently.
- When a content provider signs up as a customer of the CDN, they are granted one or more serial numbers. The content provider's objects are grouped together by serial number, and all objects with the same serial number are hashed to the same bucket (or same set of buckets, if popular)
  - Beneficial when multiple objects must be fetched by to render a single web page.
  - The browser can make a persistent HTTP connection to just one randomly-chosen server in the set, and then fetch all of the objects from that server ->only 1 TCP cx

Load balancing within a single cluster: Consistent hashing

- Example:
- Serial numbers assigned to content providers range from 1 to 2048
- CDN has assigned the serial number 212 to a customer owning www.example.com.
- The DNS CNAME mechanism indicates that www.example.com is an alias for the canonical domain name a212.g.akamai.net.
- When an authoritative name server operated by the CDN receives a request to resolve the name a212.g.akamai.net from a resolving name server, it first determines which cluster should serve the ensuing HTTP requests from browsers sharing this resolving name server.
  - The cluster is chosen using stable allocations.
- Then consistent hashing is used to determine which server(s) within the cluster should serve objects under serial number 212.
- Depending on the current popularity of the content being served under serial number 212, the authoritative name server for the CDN returns the first *k* addresses, skipping any servers that are not operational.
- The k servers in the DNS response are listed in random order to help spread the load among them.

Getting an idea of whether the content is cached: Bloom Filters

- To approximately represent dynamically evolving sets in a space efficient manner.
- Useful in content delivery in: content summarization and content filtering.
- Basics:
- Suppose that we want to store a set $S = \{e_1, e_2, \cdots, e_n\}$ in a manner that allows us to efficiently insert new elements into the set and answer membership queries of the form "Is element *e* in set *S*?"
- A simple data structure is a hash function *h* that maps elements to a hash table $T[1 \cdots m]$, where each table entry stores 0 or 1, 0 first. When *e* is inserted into set S: $T[h(e)]$ is set to 1
- To check if $e \in S$, we simply need to check if $T[h(e)]$ is 1.
  - "false positives" are possible with this solution when e *and* $e' \in S$ are such that $h(e) = h(e')$
  - a "false negative" cannot occur
- Bloom filters are a generalization:
  - uses multiple hash functions $h_1, h_2, \cdots, h_k$ to reduce the probability of a false positive.
  - Still a table $T[1 \cdots m]$ with binary values, initialized to 0.
  - To insert *e*: the bits $T[hi(e)], 1 \le i \le k$, are set to 1.
  - To check if $e \in S$: are T [hi (e)] = 1, for all $1 \le i \le k$
  - False positives are still possible but less likely

Getting an idea of whether the content is cached: Bloom Filters

- Quantify the tradeoff between the false positive probability *p*, the number of elements *n*, the number of bits *m*, and the number of hash functions *k*

  -> exercise session

Bloom filters: for Content Summarization

- Cache summarization: a Bloom filter is used to succinctly summarize the list of objects stored in a CDN server.
- More space-efficient than storing a list of URLs associated with the objects in cache.
- Cache summarization can be used to locate which server cache has which objects.
  - For instance, if the CDN's servers periodically exchange cache summaries, a server that does not have a requested object in its cache can find it on other servers
- Example: implemented in popular web caching proxies like Squid, which create cache summaries called "digests"
  - Squid uses a Bloom filter with k = 4 where a single 128-bit MD5 hash of the object's identifier is partitioned into four 32-bit chunks and treated as four separate hashes.

Bloom filters: for Content Filtering

- Cache filtering: a Bloom filter to determine what objects to cache in the first place.
- Without cache filtering: a CDN's server caches each object that it serves.
  - When the disk cache fills up, the objects are evicted using a cache replacement policy.
  - But: no reason to cache objects likely to be accessed only once.
  - -> a significant amount of disk space would be saved (and number writes to disk)

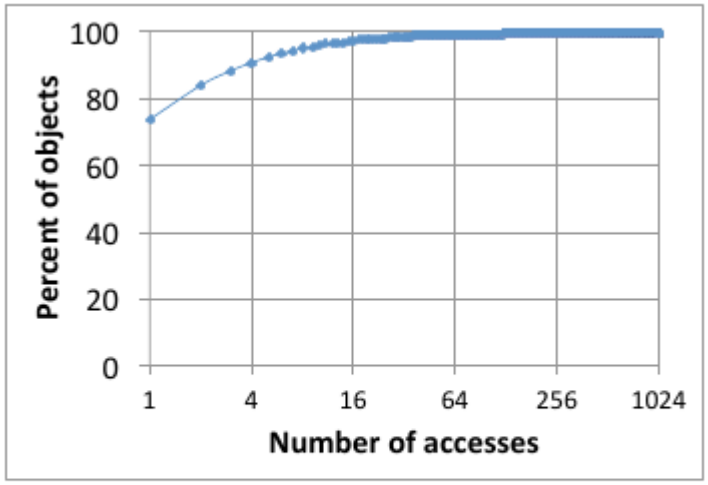


Figure 5: On a typical CDN server cluster serving web traffic over two days, 74% of the roughly 400 million objects in cache were accessed only once and 90% were accessed less than four times.

Bloom filters: for Content Filtering

- Cache-on-second-hit rule: a simple cache filtering rule
  - caches an object only when it is accessed for a second time within a specific time period.
  - can be implemented by storing the set of objects that have been accessed in a Bloom filter.
  - Upon request, the server first checks to see if the object has been accessed before by examining the Bloom filter.
    - If not, object fetched and served but not cached.
    - If so, the object has been accessed before, fetched, served, and stored

Bloom filters: for Content Filtering

- Most CDNs implement cache replacement algorithms such as LRU, which evict less popular objects such as one-hit-wonders when the cache is full.
- -> Filtering out the less popular objects and not placing them in cache at all provides additional disk space for more popular objects and increases the byte hit rate.

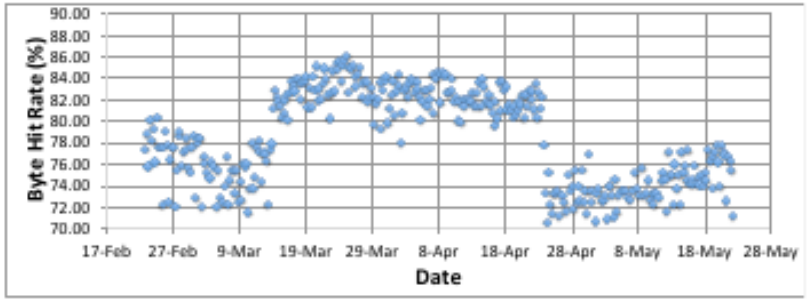


Figure 6: Byte hit rates increased when cache filtering was turned on between March 14th and April 24th because not caching objects that are accessed only once leaves more disk space to store more popular objects.

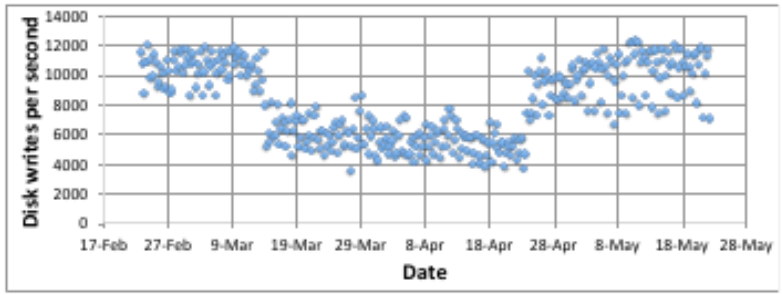
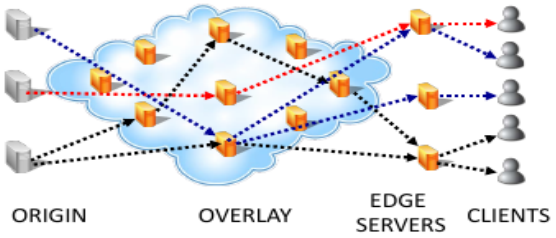


Figure 7: Turning on cache filtering decreases the rate of disk writes by nearly one half because objects accessed only once are not written to disk.

Bloom filters: Challenges

- Speeding up the Bloom filter: needs to be extremely fast because Bloom filter operations are on the critical path for client-perceived CDN performance
- Sizing the Bloom filter: space efficiency is the main reason why Bloom filters are used in place of simpler hash table based solutions
- Given the number *n* of objects and false positive proba *p*, the number of *k* of hash function and Bloom filter table size can be calculated.
- Complex rules may need to be implemented in practice that take into consideration other object popularity metrics and other characteristics such as size.

Overlay routing



- Most web sites have some content that is either not cacheable or cacheable for only short periods of time. Ex: a banking portal - dynamic web content
  - A significant part is personalized for individual users and cannot be cached,
  - though some page elements such as CSS, JS, and images may be common across users and cached.
- CDNs also host applications that users can interact with in real-time: cannot be cached
- Live streaming and teleconferencing are delivered by CDNs in real-time and are uncacheable.
- Even for cachable content, often time-to-live (TTL) to periodically refresh it (ex: stock chart)
- A common framework that can capture all of the above situations is shown in the fig:
  - origins that create the content,
  - edge servers that clients access to consume the content
  - an overlay network that is responsible for transporting the content from the origins to the edges.
- Clients request the content from a proximal edge server which downloads the requested content from the origin via the overlay network.
  - case of a web site: the origin is a collection of application servers, databases, and web servers deployed by the content provider in one or more data centers on the Internet.
  - case of a live stream: the origin denotes the servers that receive the stream in real-time from the encoders capturing the event.
- The edge servers are operated by the CDN and are deployed in more than a thousand data centers around the world, so as to be proximal to clients

Overlay routing

- A path from origin to the edge server that does not pass through any intermediate overlay servers is called the direct path. The direct path is always used when no overlay path is superior to it.
- Key problem: how to construct an overlay to provide efficient communication between origins and edge servers.
- An overlay construction algorithm takes as input
  - client demands, which dictate which origins need to send their content to which edge servers,
  - real-time network measurements of latency, loss, and available bandwidth for the paths through the Internet between origins, overlay servers, and edge servers.
- Algorithmic Solutions

Overlay routing

- Dynamic web content:
  - latency sensitive -> customized algorithms for the well-studied all-pairs shortest-path (APSP) problem.
  - Note that overlay construction involves both cost and capacity constraints.
  - A first step to constructing an APSP instance is to perform Lagrangian relaxation (i.e., penalizing violations of constraints rather than forbidding them) to encode the capacity constraints as a cost that can be added to existing link-performance-dependent cost terms.
- Live videos:
  - throughput sensitive -> account for the capacity constraints in the overlay and the bandwidth bottlenecks in the Internet.
  - One approach is to formulate a mixed integer program (MIP) that captures all the performance, capacity, and bandwidth constraints.
  - The MIP often cannot be solved efficiently as it is NP-hard.
  - -> To overcome this problem, we can “relax” the integral variables in the MIP so that they can take on real values, resulting in a linear program (LP).

Overlay routing: benefits

- On a consistent basis, overlay routing provides better QoE (faster download or fewer video rebuffers).
  - the overlay paths may have lower latency and/or higher throughput depending.
  - the CDN can save the overhead of establishing TCP cx between different nodes by holding them persistently.
  - the CDN may also use optimized transport protocols between their nodes.
- Catastrophic events such as a cable cut are not rare on the Internet -> overlays provide alternate paths when the direct path from origin to edge is impacted

Overlay routing: benefits

- E.g.: April 2010, a large-scale Internet outage occurred when a submarine communications cable system (linking Europe, Middle East and South Asia) was cut.
  - The cable underwent repairs from 25 April to 29 April during which time several cable systems were affected, severely impacting Internet connectivity in many regions across the Middle East, Africa, and Asia.
  - Figs shows the download time by clients in Asia
    - agents distributed across India, Malaysia, and Singapore to download a dynamic (i.e., uncacheable) web page approximately 70KB in size hosted at an origin in Boston.

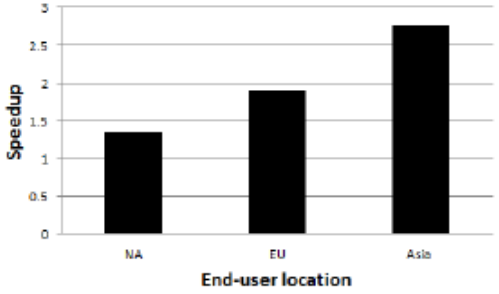


Figure 10: A routing overlay provides significant speedups by choosing better performing paths from the origin to the client. Key: North America (NA), Europe (EU), Asia.

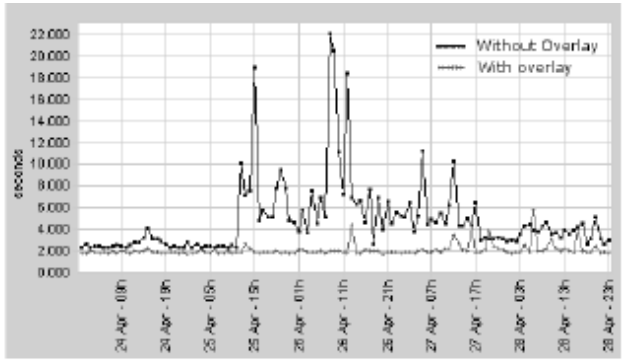


Figure 11: Performance of the routing overlay during a cable cut.

Wrapping up...

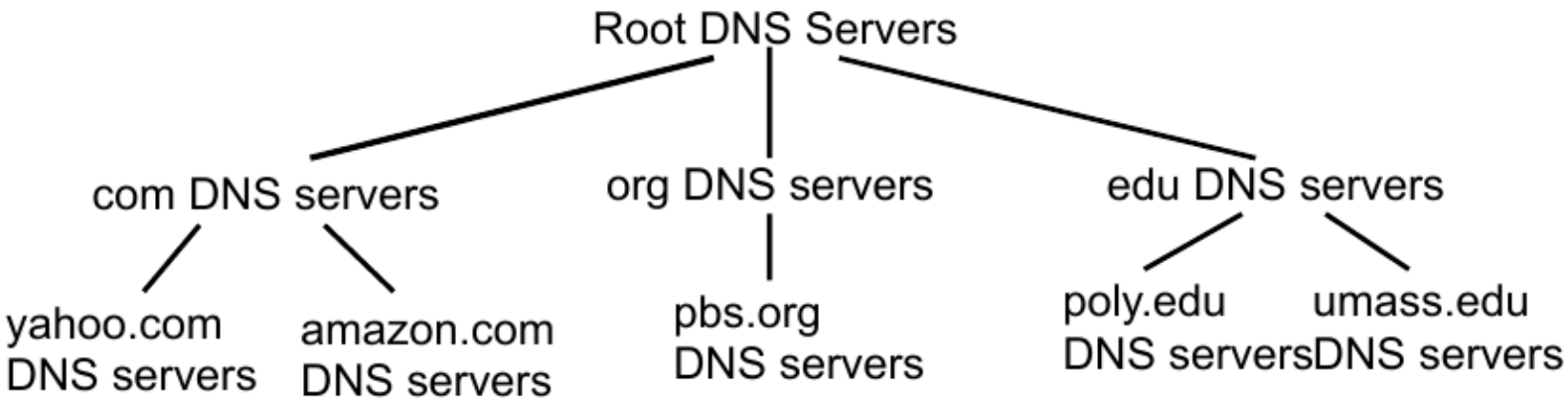
- Video streaming traffic dominates the Internet
  - Will grow to be even more important
  - High stakes at play for many companies (ISPs, video service providers, device manufacturers, etc.)
- Video delivery architectures have grown large and complex
  - Cache servers organized into multiple layers
  - Clever use of DNS and HTTP redirect to dynamically address load balancing, traffic locality, cache misses
- QoE matters a lot
  - Manifestation of many underlying factors
  - Large efforts to optimize it, as this is what clients see

ANNEX SLIDES

Mechanisms for client to server allocation

- Service provider can dynamically control which server name is provided by video web page
  - YouTube: web page is www.youtube.com/...”videoid”...
  - YouTube: server name (e.g. v23.lscache5.c.youtube.com) embedded in web page
  - Gives way to choose the CDN (and server in CDN)
- CDN provider has a couple of different ways to control server allocation
  - Dynamic DNS mappings
  - HTTP redirect
  - Can choose the ISP through which a given client is served
  - Can choose the individual server within CDN

Refresher: DNS – Distributed, Hierarchical Database



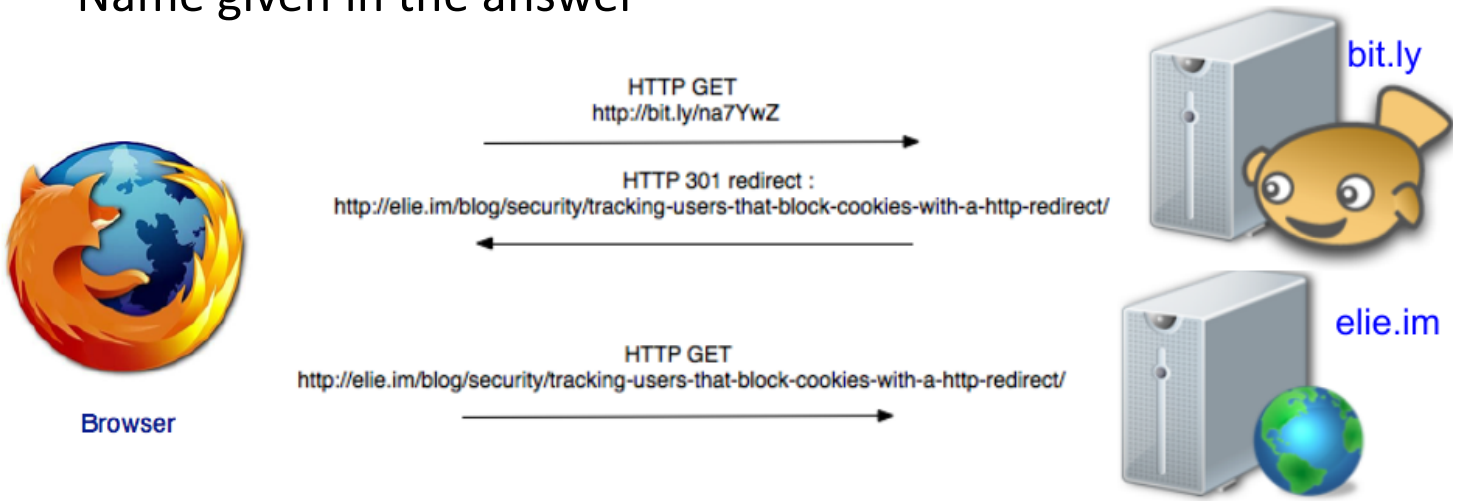
- Client wants IP for www.amazon.com:
  1. client queries a root server to find com DNS server
  2. client queries com DNS server to get amazon.com DNS server
  3. client queries amazon.com DNS server to get IP address for www.amazon.com
- Local DNS servers may do this on behalf of client (recursively)
  - They also cache results which get stale over time -> ask again from authoritative server

Role of DNS in content distribution

- DNS is typically used to perform initial assignments of clients to servers
  - Client resolves video server's DNS name to IP address
- DNS name to address mapping under control of the content provider
  - Provider's DNS servers provide authoritative answers to DNS requests
- DNS mapping often anycast -> one name maps to one of several addresses
  - Suitable server can be chosen for a given client at a given time instance

Role of HTTP redirection

- Another standard way to dynamically change client to server allocation
- Server replies with “HTTP redirect” to guide client to another server
  - Name given in the answer



Picture source : http://www.elie.net/