

CS 43: Computer Networks

10: Video Streaming

October 5, 2026



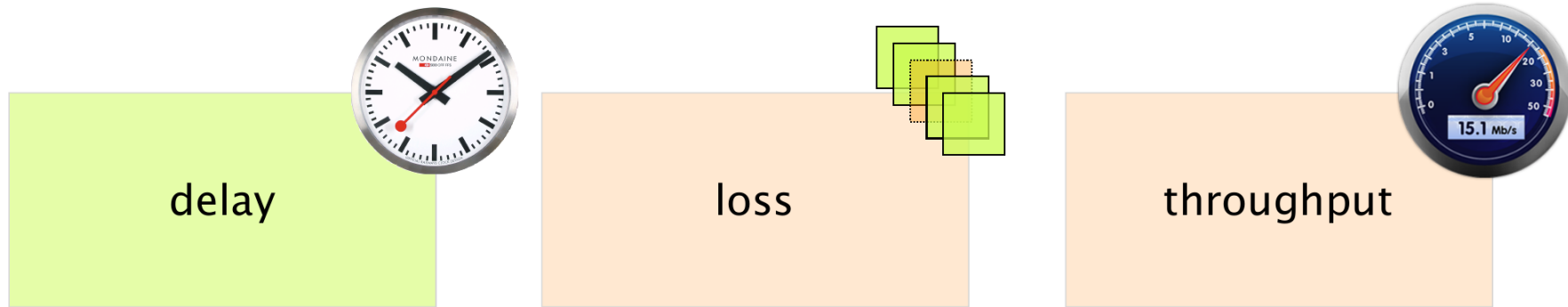
Slides adapted from: Kurose & Ross, K. Webb, V. Chaganti, D. Choffnes, L. Vanbever, S. Narayana

Announcements

- Please fill out Lab 3 partner form
- Thank you for your patience with grading...

Let's finish talking about network performance from last time...

A network *connection* is characterized by its delay, loss rate and throughput



The queuing delay is the amount of time a packet waits in a buffer to be transmitted on a link

Queuing delay is the hardest to evaluate
as it varies from packet to packet

It is characterized with statistical measures
e.g., average delay & variance, probability of exceeding x

Checkout this animation illustration:

[Queuing loss animation](#)

The time a packet has to sit in a buffer before being processed depends on the traffic pattern

Queueing delay depends on:

- arrival rate at the queue
- transmission rate of the outgoing link
- traffic burstiness

average packet arrival rate	a	[packet/sec]
-----------------------------	-----	--------------

transmission rate of outgoing link	R	[bit/sec]
------------------------------------	-----	-----------

fixed packets length	L	[bit]
----------------------	-----	-------

average bits arrival rate	La	[bit/sec]
---------------------------	------	-----------

traffic intensity	La/R	
-------------------	--------	--

Two links both have rate R and carry L -bit packets. On average, both receive one packet every $2L/R$ seconds.

Link X: one packet arrives every $2L/R$ seconds, evenly spaced. Link Y: 10 packets arrive at once every $20L/R$ seconds.

Which statement about average queueing delay is correct?

- A. Both are the same because average arrival rate is the same
- B. Both are zero, because each link keeps up on average
- C. X is larger because packets arrive more often
- D. Y is larger; X is zero
- E. Both are larger than zero, and Y is larger

Two links both have rate R and carry L -bit packets. On average, both receive one packet every $2L/R$ seconds.

Link X: one packet arrives every $2L/R$ seconds, evenly spaced. Link Y: 10 packets arrive at once every $20L/R$ seconds.

Which statement about average queueing delay is correct?

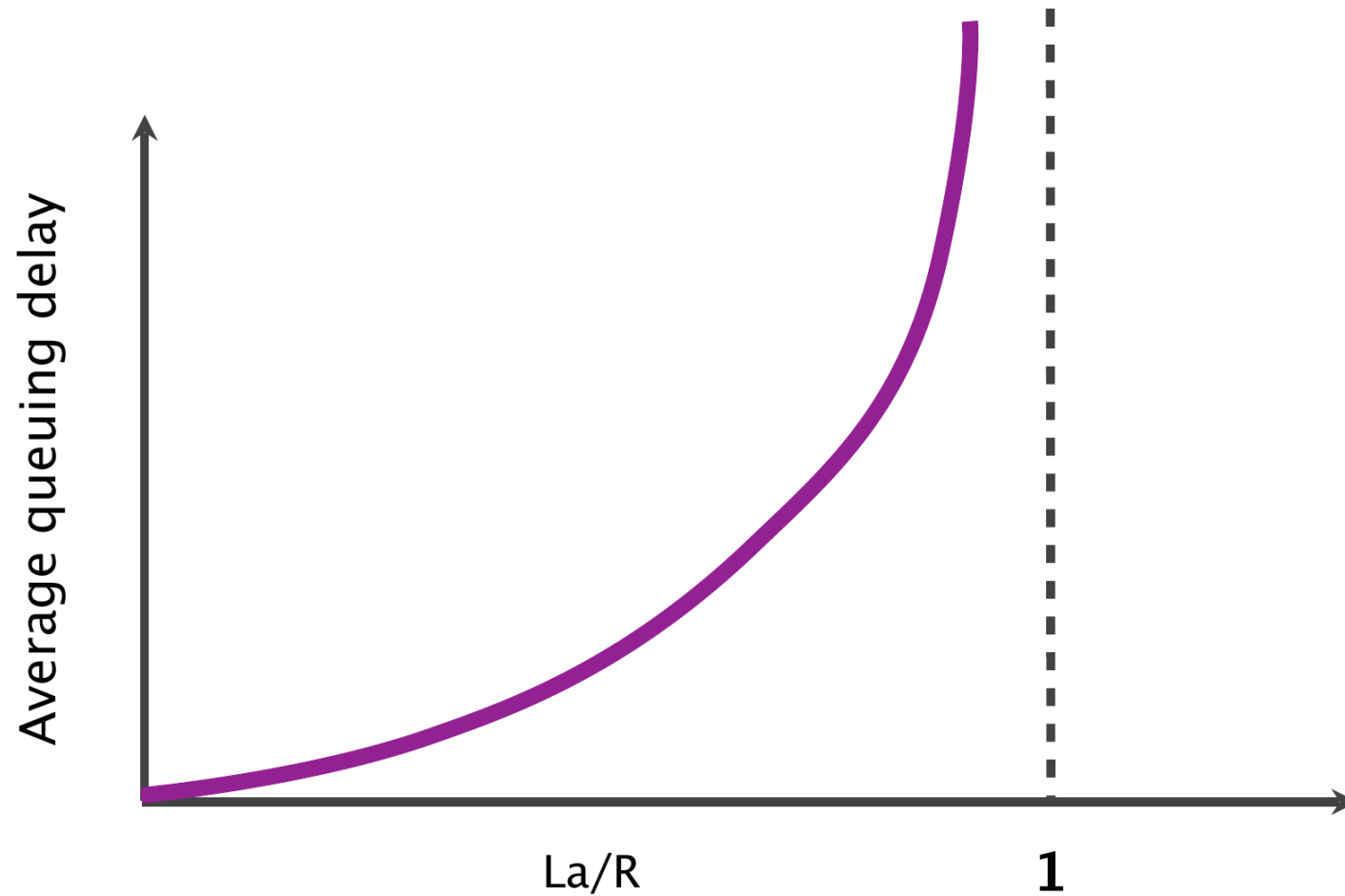
- A. Both are the same because average arrival rate is the same
- B. Both are zero, because each link keeps up on average
- C. X is larger because packets arrive more often
- D. Y is larger; X is zero**
- E. Both are larger than zero, and Y is larger

When the **traffic intensity is >1** , the queue will increase without bound, and so does the queuing delay

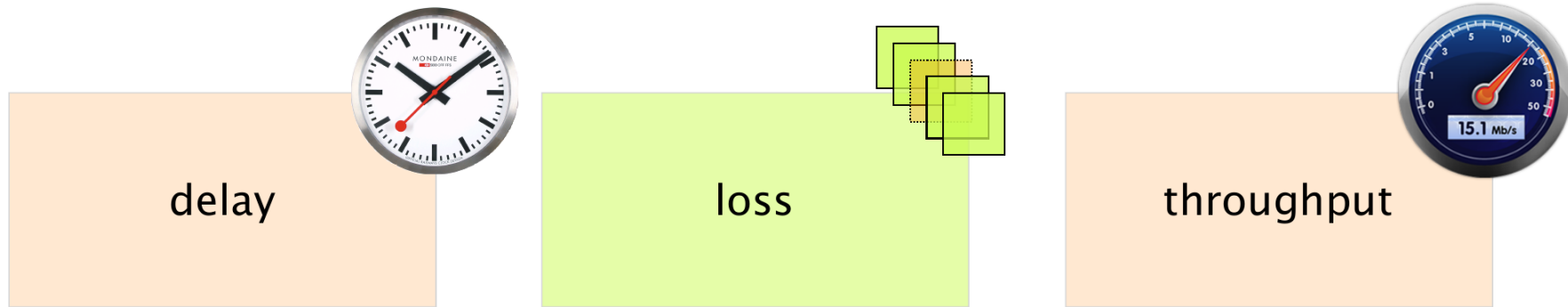
Golden rule

Design your queuing system,
so that it operates far from that point

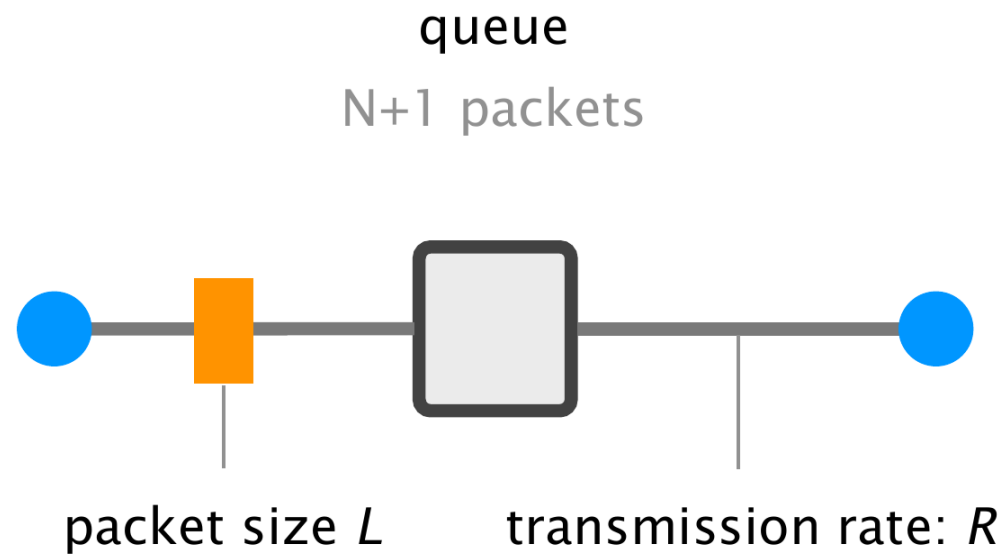
When the **traffic intensity is ≤ 1** ,
queueing delay depends on the burst size



A network *connection* is characterized by its delay, loss rate and throughput

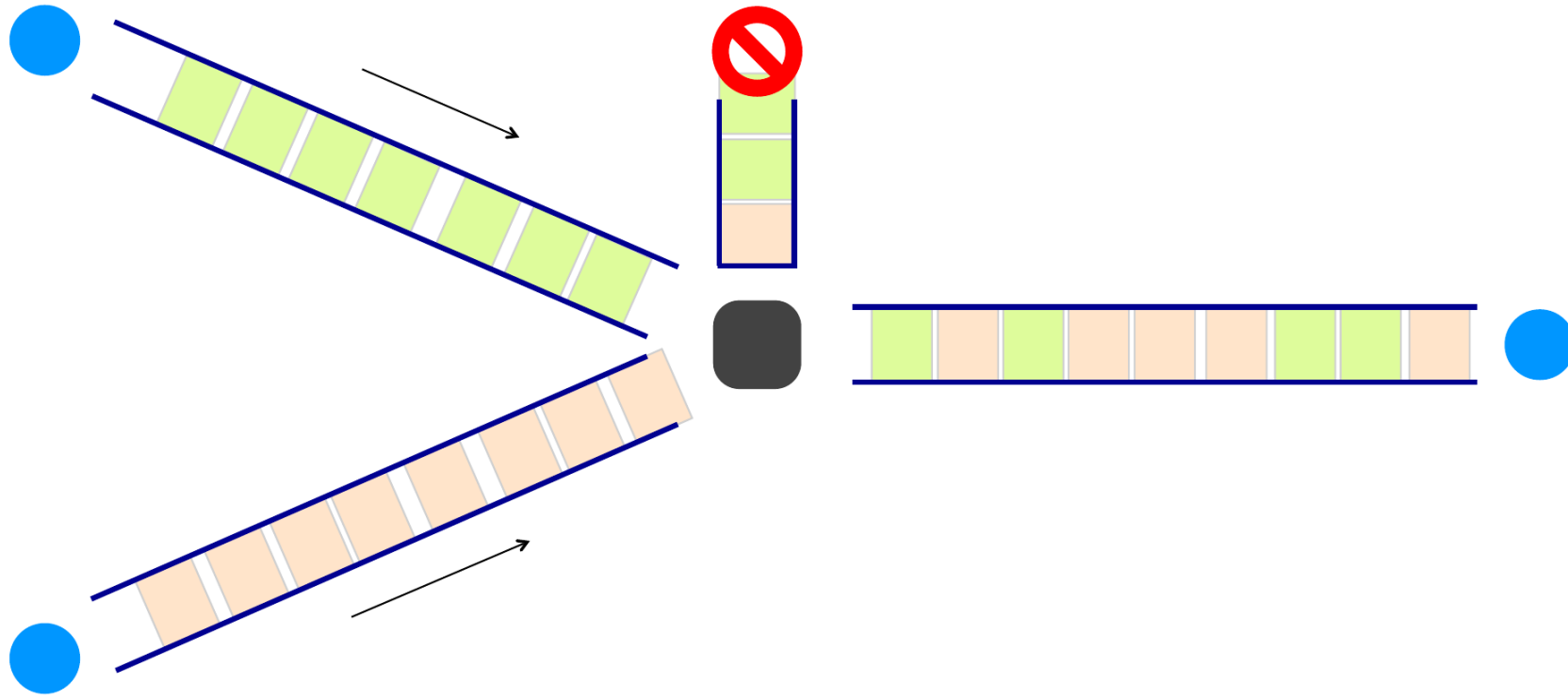


In practice, queues are not infinite.
There is an upper bound on queuing delay.

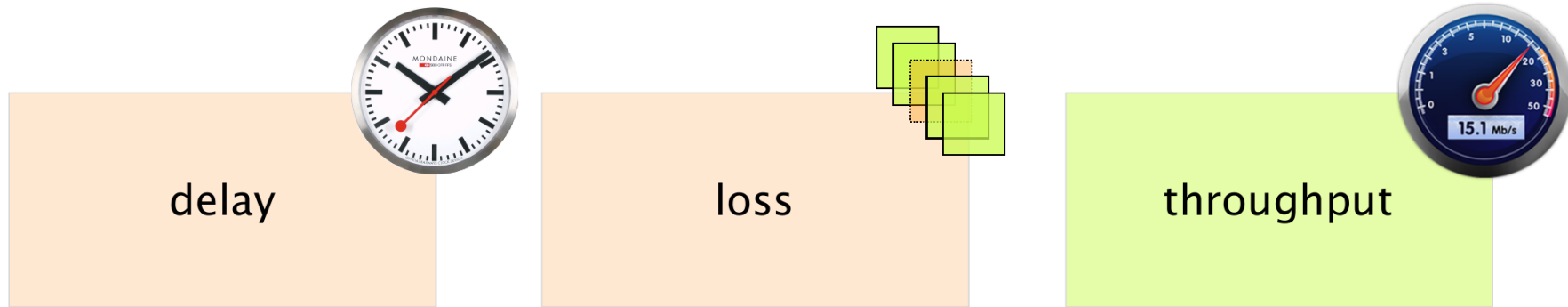


queuing delay upper bound: $N \cdot L / R$

If the queue is persistently overloaded,
it will eventually drop packets (loss)



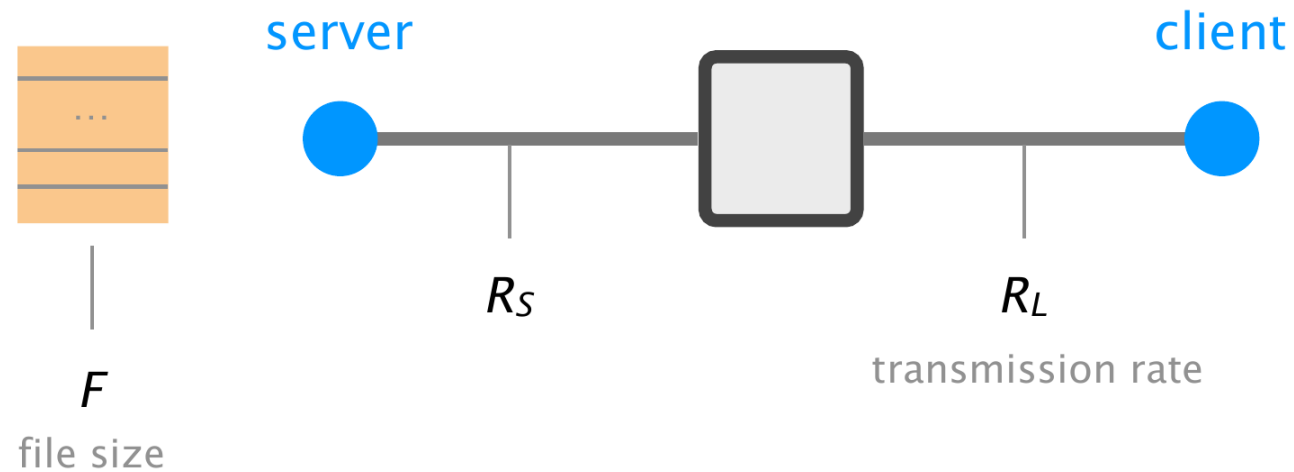
A network *connection* is characterized by its delay, loss rate and throughput



The throughput is the instantaneous rate at which a host receives data

$$\begin{array}{lcl} \text{Average throughput} & = & \frac{\text{data size}}{\text{transfer time}} \\ \text{[#bits/sec]} & & \begin{array}{l} \text{[#bits]} \\ \text{[sec]} \end{array} \end{array}$$

To compute throughput, one has to consider the bottleneck link

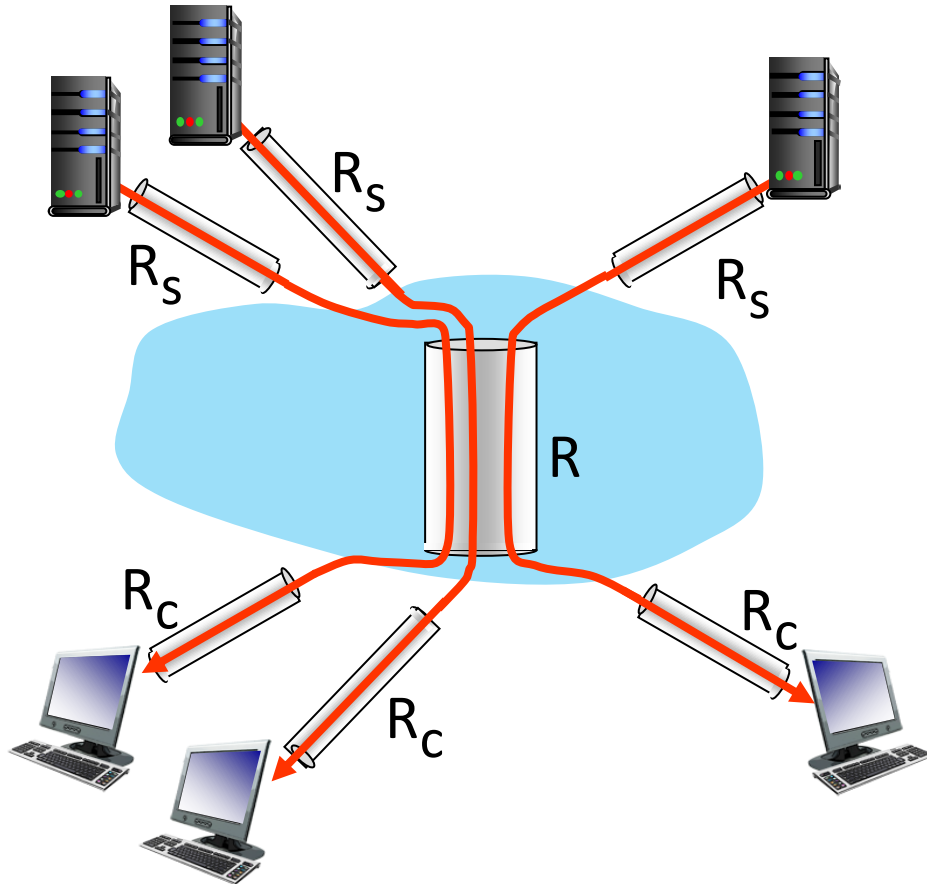


Average throughput

$$\min(R_S, R_L)$$

= transmission rate
of the bottleneck link

Throughput: network scenario



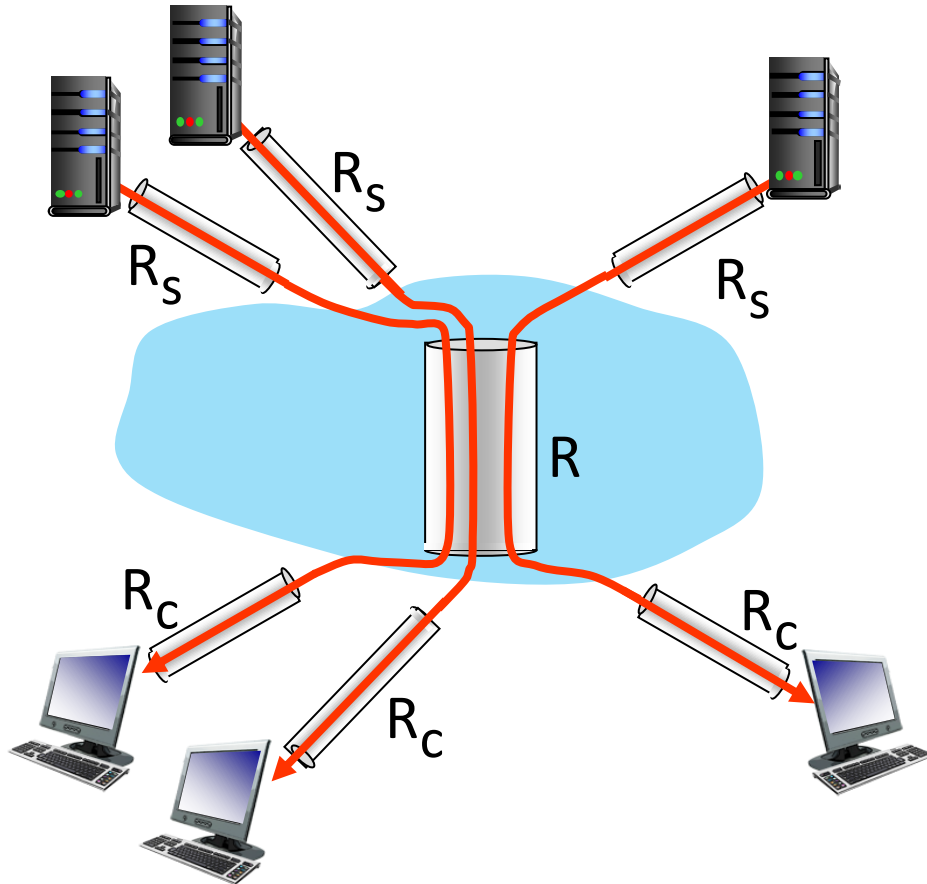
10 connections (fairly) share
backbone bottleneck link R bits/sec

Each server access link is $R_s = 2$ Mbps, each client access link is $R_c = 1$ Mbps, and all 10 share one core link that divides its rate equally. With a 5 Mbps shared link, each download gets 500 kbps.

The shared link is upgraded to 50 Mbps.
What is each download's throughput now?

- A. 500 Kbps
- B. 1 Mbps
- C. 2 Mbps
- D. 5 Mbps
- E. 50 Mbps

Throughput: network scenario

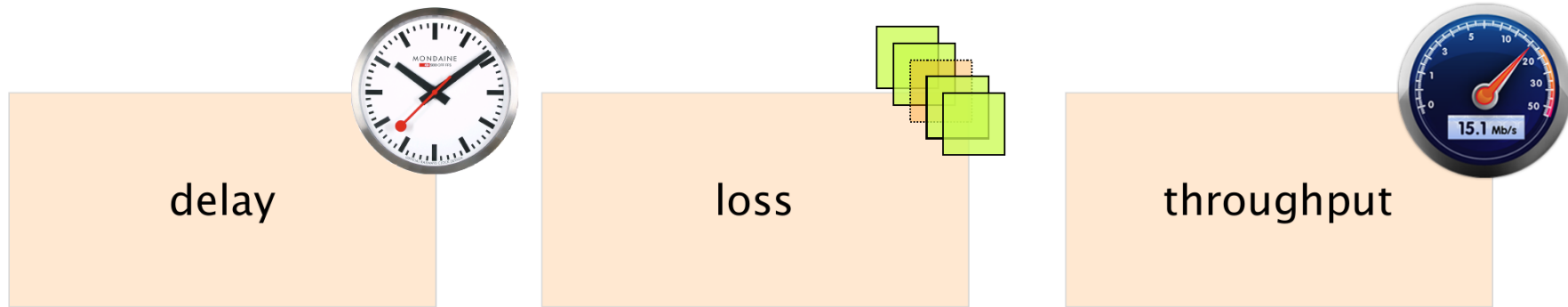


10 connections (fairly) share
backbone bottleneck link R bits/sec

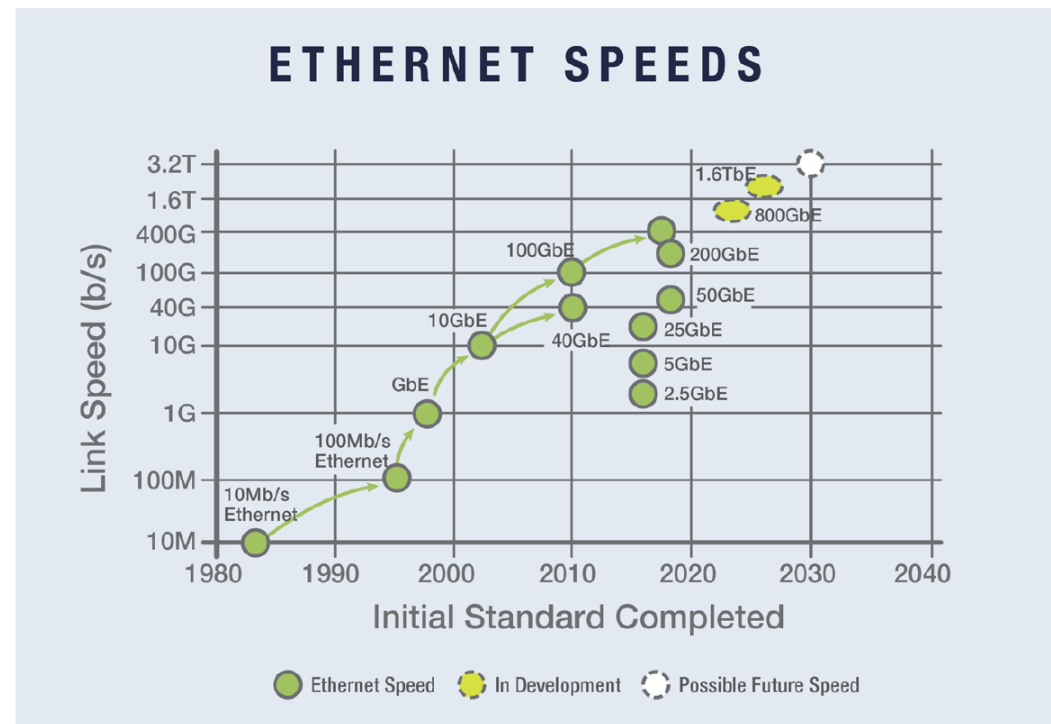
- per-connection end-end throughput:
 $\min(R_c, R_s, R/10)$
- in practice: R_c or R_s is often bottleneck

* Check out the online interactive exercises for more examples: http://gaia.cs.umass.edu/kurose_ross/

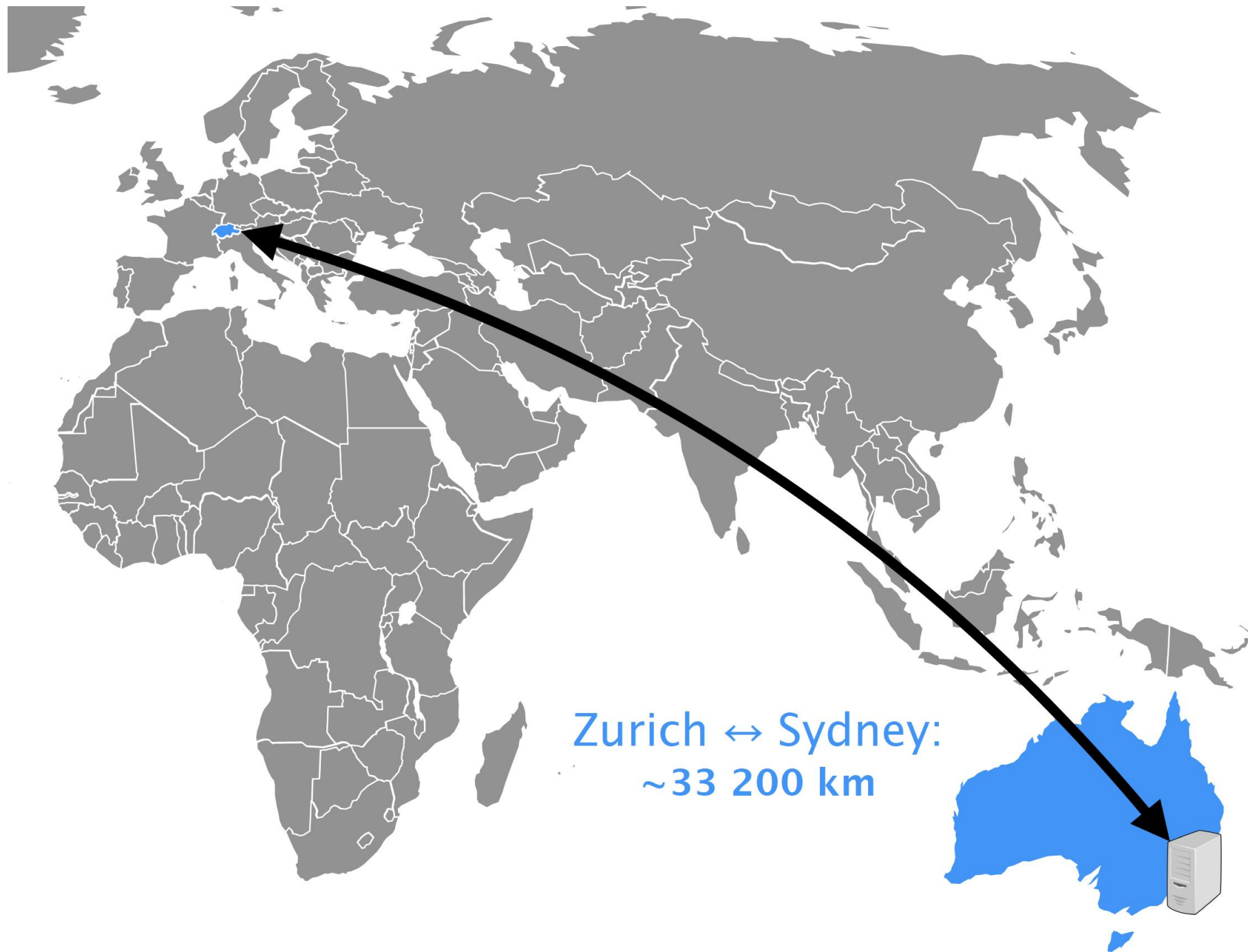
A network *connection* is characterized by its delay, loss rate and throughput



As technology improves, throughput increase & delays are getting lower except for propagation
(speed of light)



source: Ethernet Alliance



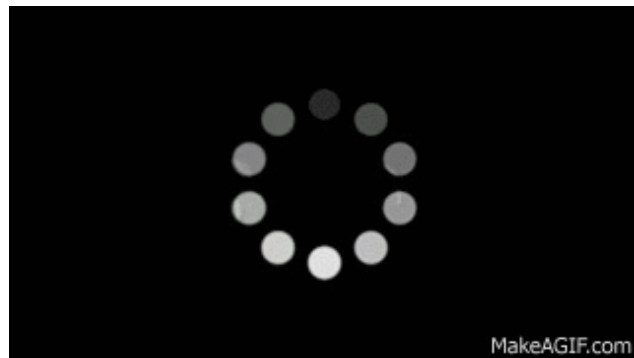
$$\frac{299,792,458 \text{ m/s}}{\text{refractive index optical fiber } 1.468} = \text{speed of light in optical fiber } 204,218,296 \text{ m/s}$$

$$\frac{33,200,000 \text{ m}}{204,218,296 \text{ m/s}} = 163 \text{ ms}$$

theoretical
round-trip time

Today: Explain what happens when you see “rebuffering” and how video streaming services work to avoid this

- Digital representation of video
- On-demand video streaming
- Dynamic Adaptive Streaming over HTTP (DASH)



Where we are

Application: the application (e.g., Video Streaming)

Transport: end-to-end connections, reliability

Network: routing

Link (data-link): framing, error detection

Physical: 1's and 0's/bits across a medium
(copper, the air, fiber)

2024 estimates found more than 65% of global Internet traffic and nearly 75% of mobile network traffic is video streaming.

Let's look at a demo of a DASH video player...

<https://reference.dashif.org/dash.js/latest/samples/dash-if-reference-player/index.html>

DIGITAL REPRESENTATION OF VIDEO

Video representation

- Video: sequence of images displayed at constant rate
 - e.g., 30 images/sec
 - Appear continuous due to the stroboscopic effect



frame i

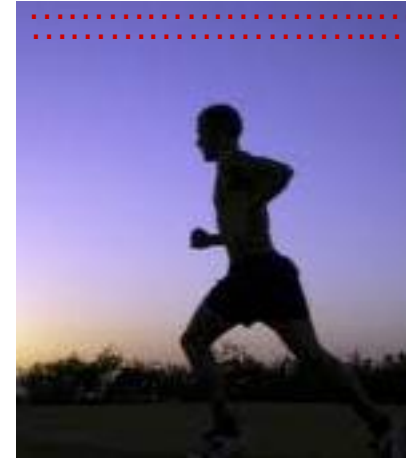


frame $i+1$

Video representation

- Digital image: array of pixels
 - each pixel represented by bits
 - Encode luminance and color
 - Number of pixels: **resolution**
- Coding: use redundancy **within** and **between** images to decrease # bits used to encode image
 - spatial (within image)
 - temporal (from one image to next)
- Encoding/decoding algorithm often called a **codec**

spatial coding example: instead of sending N values of same color (all purple), send only two values: color value (*purple*) and number of repeated values (N)



frame i

temporal coding example: instead of sending complete frame at $i+1$, send only differences from frame i (motion vectors)



frame $i+1$

Video codecs: terminology

- **Video bit rate:** effective number of bits per second of the video after encoding
- It depends on many factors
 - Resolution of each image: more pixels = more bits
 - Detail per pixel: more luminance & color detail = more bits
 - Amount of movement in the video. More movement = more bits
 - Quality of overall compression in the codec
- Video bit rate is typically correlated with quality of perception
 - Higher bit rate == better to perceive

Bitrates: terminology

- Bitrate of a video changes over the duration of the video
- **CBR: (constant bit rate):** fixed bitrate video
- **VBR: (variable bit rate):** different parts of the video have different bit rates, e.g., changes in color, motion, etc.
 - For VBR, we talk about **average bitrate** over video's duration

Examples of average video bitrates

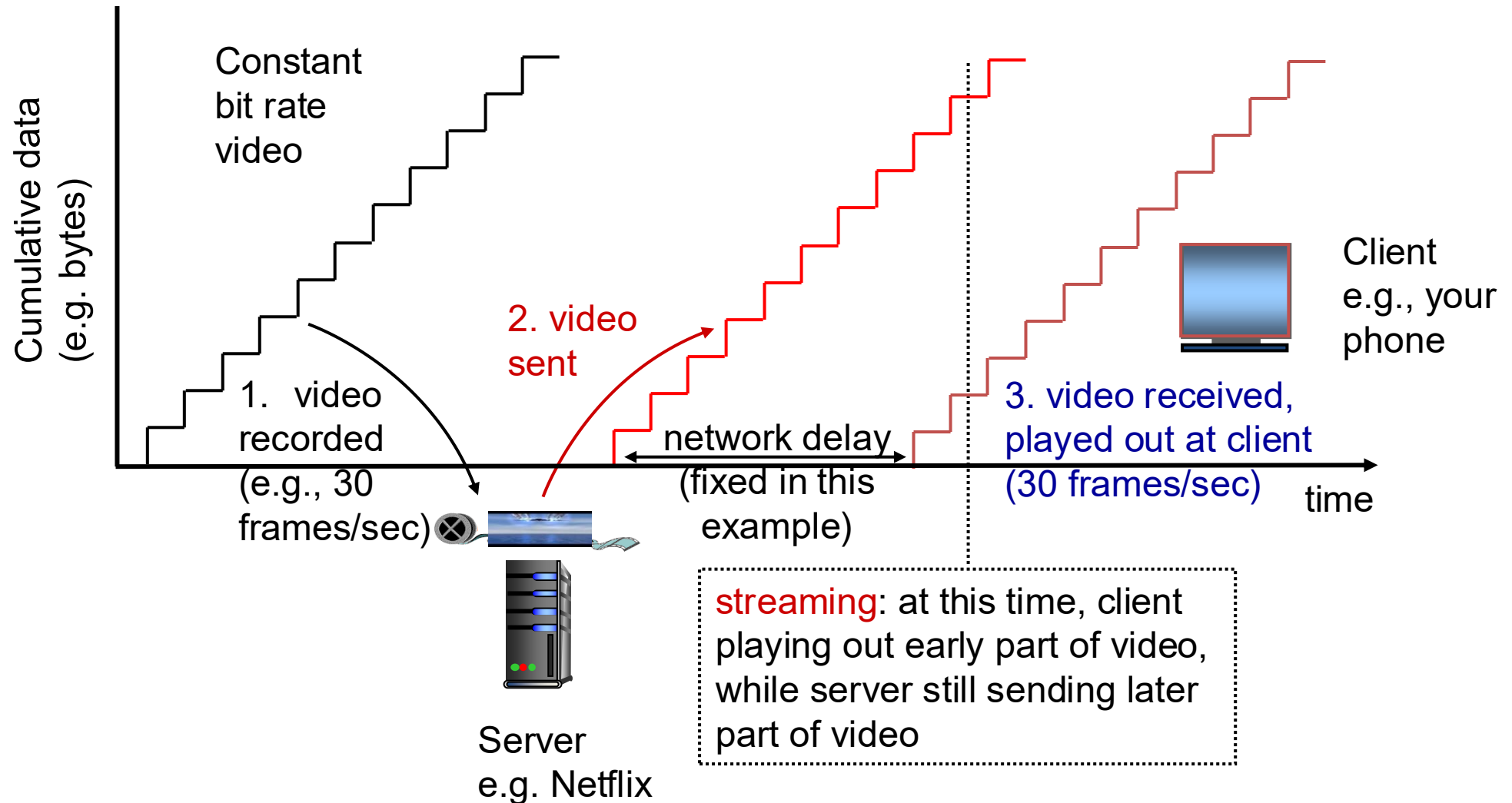
Resolution	Bitrate
1280x720 (720p)	5.0 Mbps
1920x1080 (1080p)	9.9 Mbps
3840x2160 (4K)	14.9 Mbps

Networking multimedia: 3 types

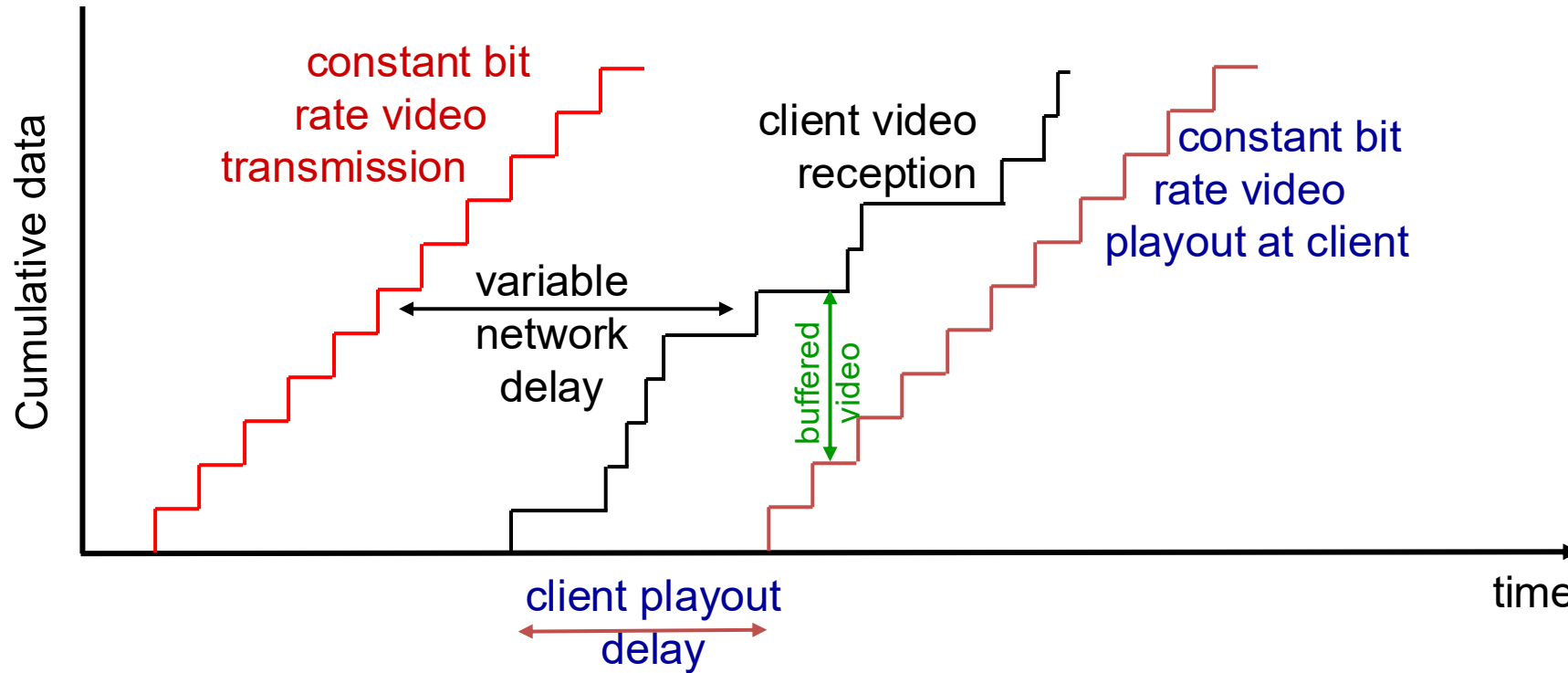
- **On-demand streamed video/audio**
 - Can begin playout before downloading the entire file
 - Full video/audio stored at the server: able to transmit faster than audio/video will be rendered (with storing/buffering at client)
 - e.g., Spotify, YouTube, Netflix
- **Conversational** voice or video over IP
 - interactive human-to-human communication limits delay tolerance
 - e.g., Zoom
- **Live streamed** audio, video
 - e.g., sporting event on sky sports
 - Can delay a little, but must be close to the “live edge” of content

ON-DEMAND VIDEO STREAMING

Streaming stored video



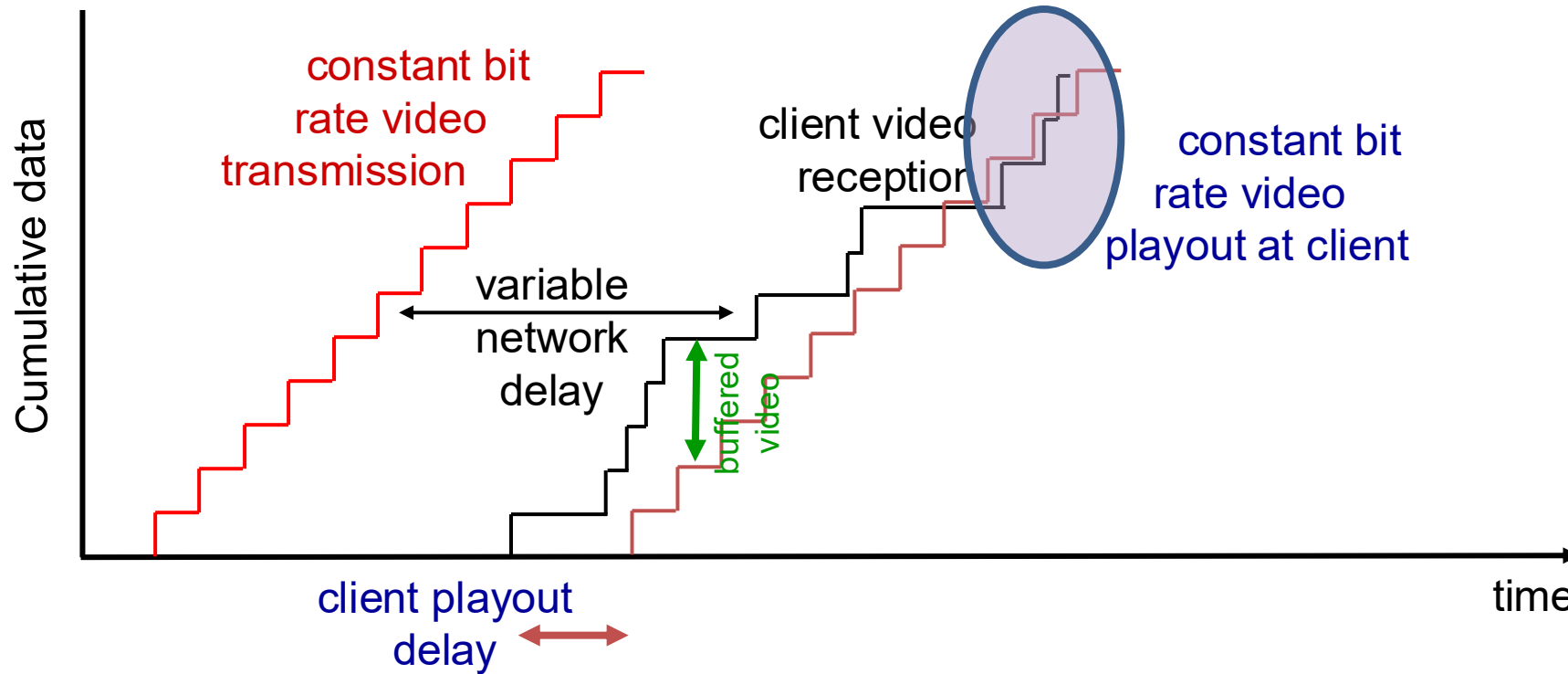
Introduce a delay for smooth playout



Client-side buffering with playout delay:

compensate for variations in the network delay

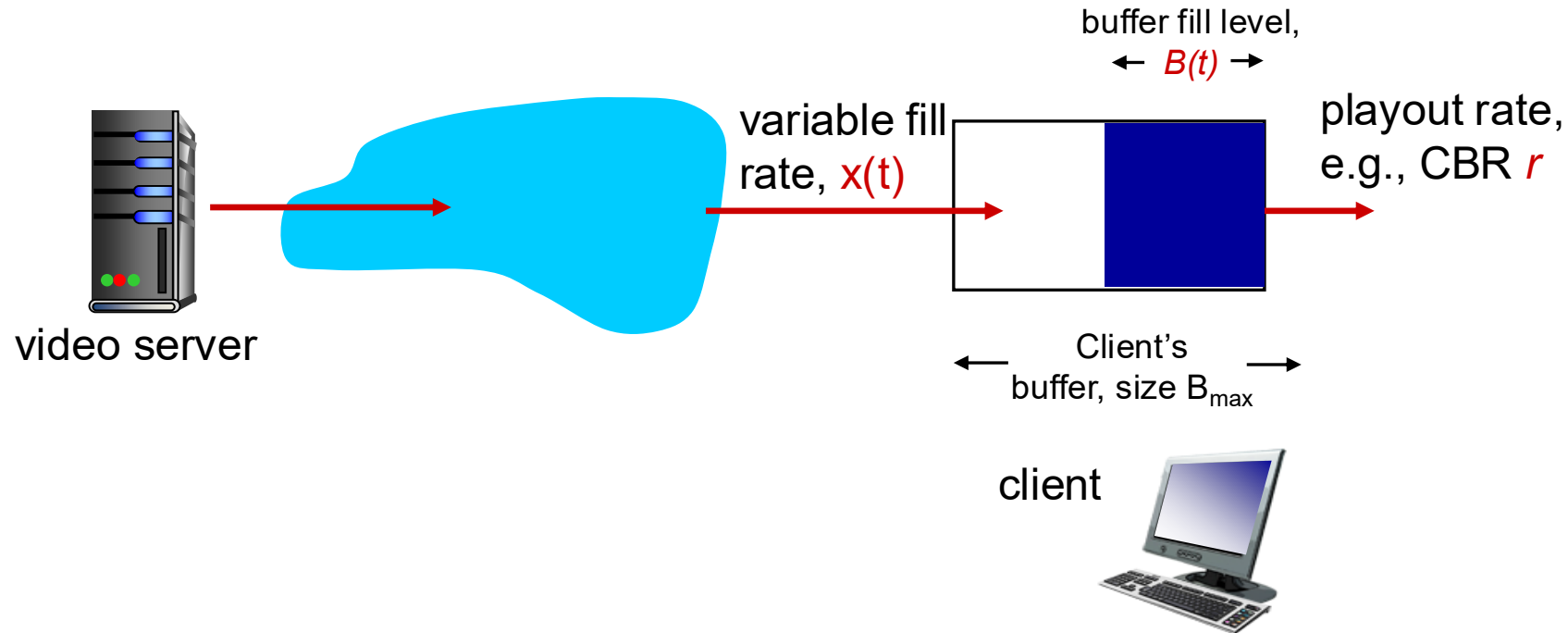
But not too small a delay



Playout delay that's too small can cause stalls

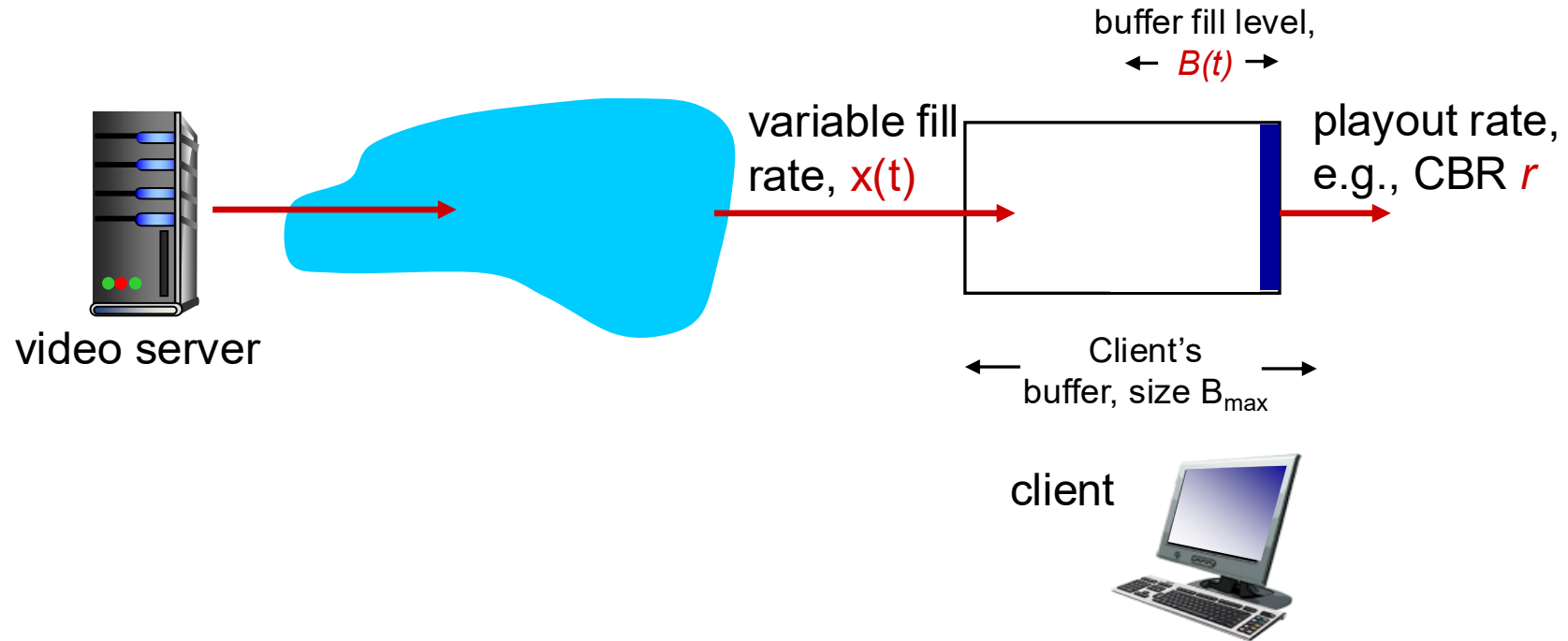
There's nothing in the buffer to show to the user

Client-side buffering, playout



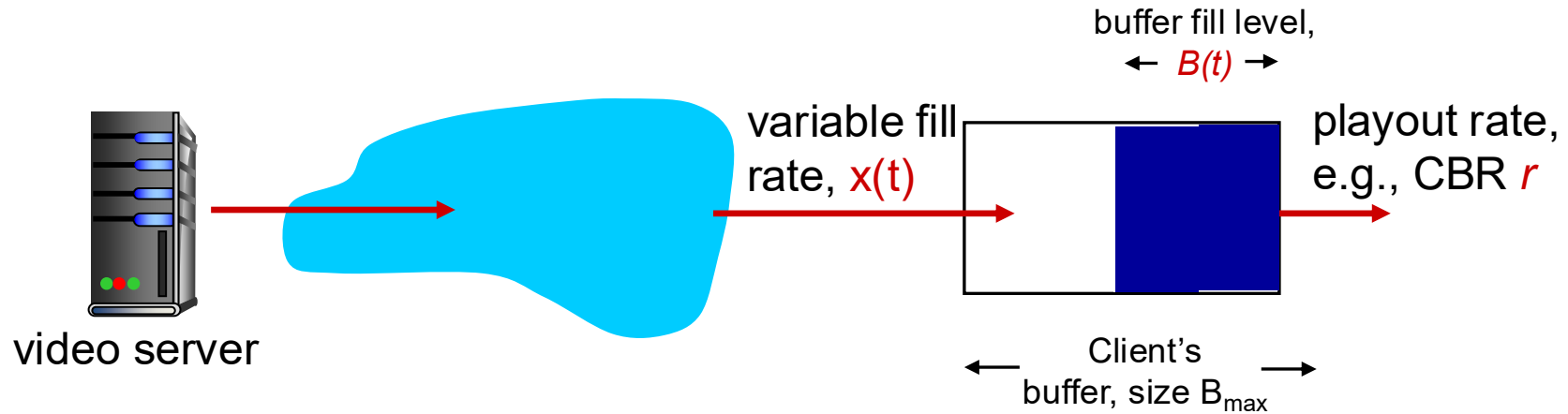
Most video is broken up in time into multiple **segments**
Client downloads video segment by segment
For example: a segment might be 4 seconds worth of video.

Client-side buffering, playout



1. Initial fill of buffer until playout begins at t_p
2. playout begins at t_p
3. buffer fill level varies over time as fill rate $x(t)$ varies (assume playout rate r is constant)

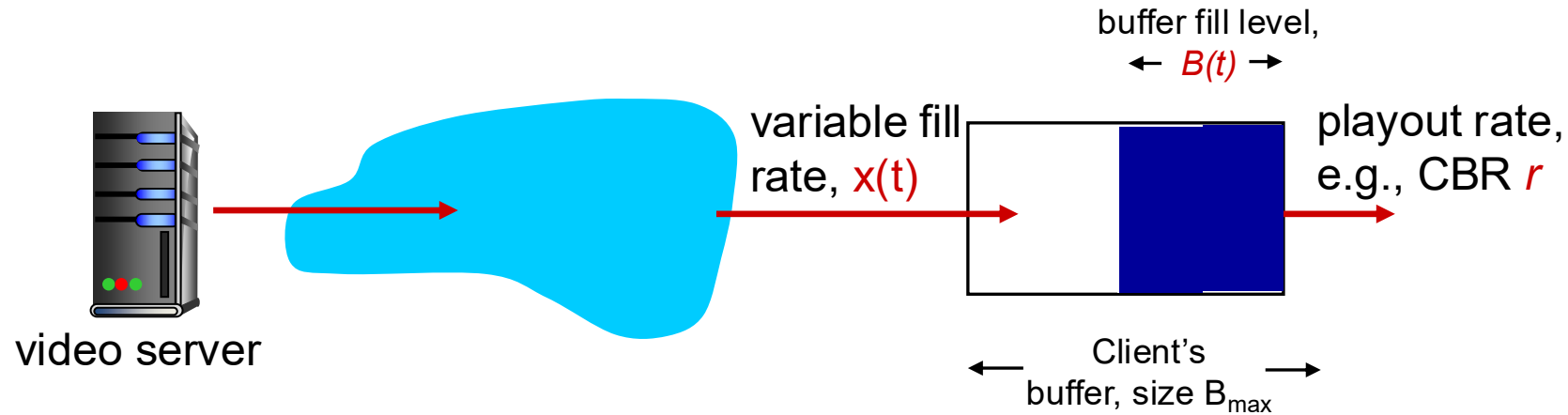
Client-side buffering, playout



playout buffering: average fill rate ($\bar{x}$), playout rate (r):

- $\bar{x} < r$: buffer eventually empties for a sufficiently long video.
 - Stall and rebuffering
- $\bar{x} > r$: buffer will not empty, provided the initial playout delay is large enough to absorb variability in $x(t)$
 - *initial playout delay tradeoff*: buffer starvation less likely with larger delay, but also incur a larger delay for the user to begin watching

Client-side buffering, playout



playout buffering: average fill rate ($\bar{x}$), playout rate (r):

- is $\bar{x} < r$ or $\bar{x} > r$ for a given network connection?
- It can be hard to control x or even predict it in general
 - Best-effort network inflicts long queues, low bandwidth, loss, etc.
- **How to set the playout rate r ?**
 - Too low a bitrate r : video has poorer quality than needed
 - Too high a bitrate r : buffer might empty out. Stall/rebuffering

A client is playing a 2 Mbps video stream and has 8 seconds of video in its buffer. The network throughput drops to 1 Mbps and stays there. The client keeps requesting the 2 Mbps segments. What happens?

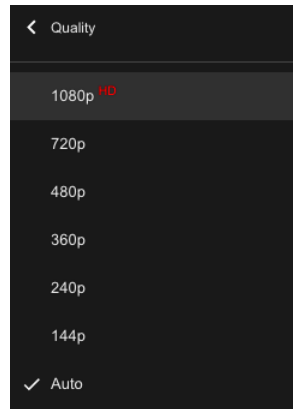
- A. Playback continues normally; data is still arriving
- B. Playback continues for a while, then stalls
- C. Playback stalls right away
- D. Playback continues, but at lower quality
- E. I don't know

A client is playing a 2 Mbps video stream and has 8 seconds of video in its buffer. The network throughput drops to 1 Mbps and stays there. The client keeps requesting the 2 Mbps segments. What happens?

- A. Playback continues normally; data is still arriving
- B. Playback continues for a while, then stalls**
- C. Playback stalls right away
- D. Playback continues, but at lower quality
- E. I don't know

Adaptive bit-rate video

- Motivation: Want to provide high quality video experience, without stalls
- Observations:
 - Videos come in different qualities (average bit rates)
 - Versions of the video for different quality levels readily available
 - Different segments of video can be downloaded separately
- **Adapt bitrate per segment** through collaboration between the video client (e.g., your browser) and the server (e.g., @ Netflix)
- **Adaptive bitrate (ABR) video**: change the bitrate (quality) of next video segment based on network and client conditions



A client is streaming at 4 Mbps and has 30 seconds of video buffered. The last segment downloaded slowly: the client measured the throughput was 1 Mbps. What should it request for the next segment?

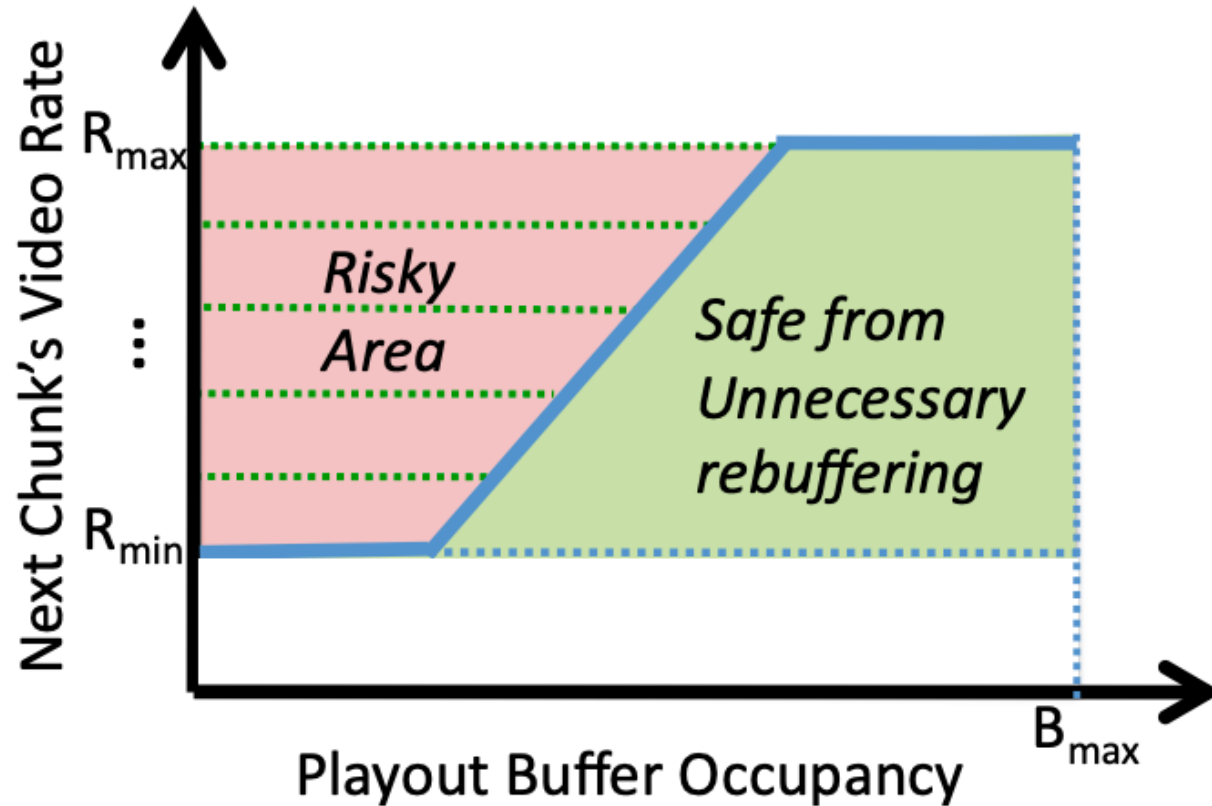
- A. The 1 Mbps version. The measurement says that's what the network can deliver
- B. Stay at 4 Mbps. With 30 seconds buffered it can afford to wait and see
- C. The 2 Mbps version, as a compromise
- D. The lowest version available, to be safe
- E. I don't know

Typical strategy:

Buffer-based bitrate adaptation

- Key idea: If there is a large stored buffer of video at the client, **optimize for video quality**, i.e., high bit rates
- Else (i.e., client video buffer has low occupancy), **avoid stalls** by being conservative and asking for a lower quality (bitrate)
 - The hope: the lower bandwidth requirement of a lower-quality stream is more easily met; stalls averted
- Buffer is measured in seconds of playout left before stalling

Buffer-based bitrate adaptation



Provide high video quality overall despite variable and intermittently poor network conditions.

<http://yuba.stanford.edu/~nickm/papers/sigcomm2014-video.pdf>

A Buffer-Based Approach to Rate Adaptation
(used by Netflix)

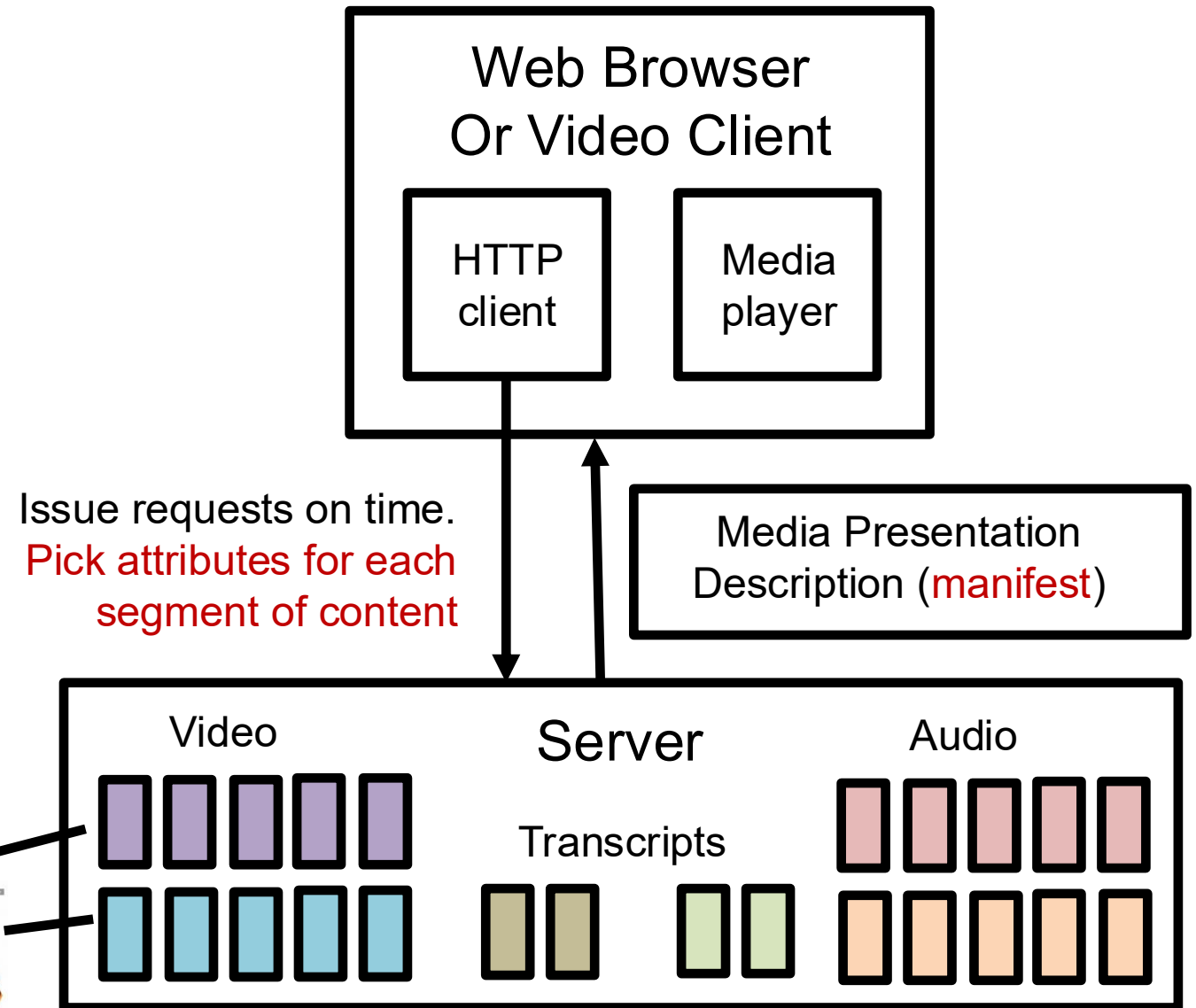
DYNAMIC ADAPTIVE STREAMING OVER HTTP (DASH)

Streaming multimedia with HTTP

- Early video: basic UDP
 - Problems: reliability, blocking
- Today: repurpose **web** infrastructure and protocols for video
- **DASH: Dynamic Adaptive Streaming over HTTP**
 - Used by Netflix, YouTube, and other video streaming services

DASH: Key ideas

- Content (video, audio, transcript, etc.) divided into **segments (time)**
- Algorithms to determine and request **varying** attributes (e.g., bitrate, language) for each segment
- Goal: ensure good quality of service, match user preferences, etc.



Streaming multimedia with HTTP

- Dividing up a video into multiple segments enables a few things
- Possible to decide the quality **per segment**
 - Enables **bit rate adaptation**
 - Typically done on the client, but possible on the server (with feedback)
 - Change language, receive sub-titles, etc. mid-stream
- Retrieve segments of a single video from multiple sources
- DASH video server is just a standard HTTP server
 - Client issue HTTP GET requests (typically with the **Range** request header)
 - Leverage existing web infrastructure (CDN, DNS)
- Send different clients to different video sources (CDN)

- [DEMO] Let's take a look at the DASH player again...

<https://reference.dashif.org/dash.js/latest/samples/dash-if-reference-player/index.html>

In the demo, the player downshifted from 4K to lower quality when the link bandwidth was lower. Which of these did the server do to make that happen?

- A. Measured the client's throughput and chose a lower representation
- B. Re-encoded the next segment at a lower bitrate
- C. Read the client's buffer level from the HTTP request
- D. Nothing beyond answering HTTP GETs for files that already existed
- E. I don't know

DASH Summary

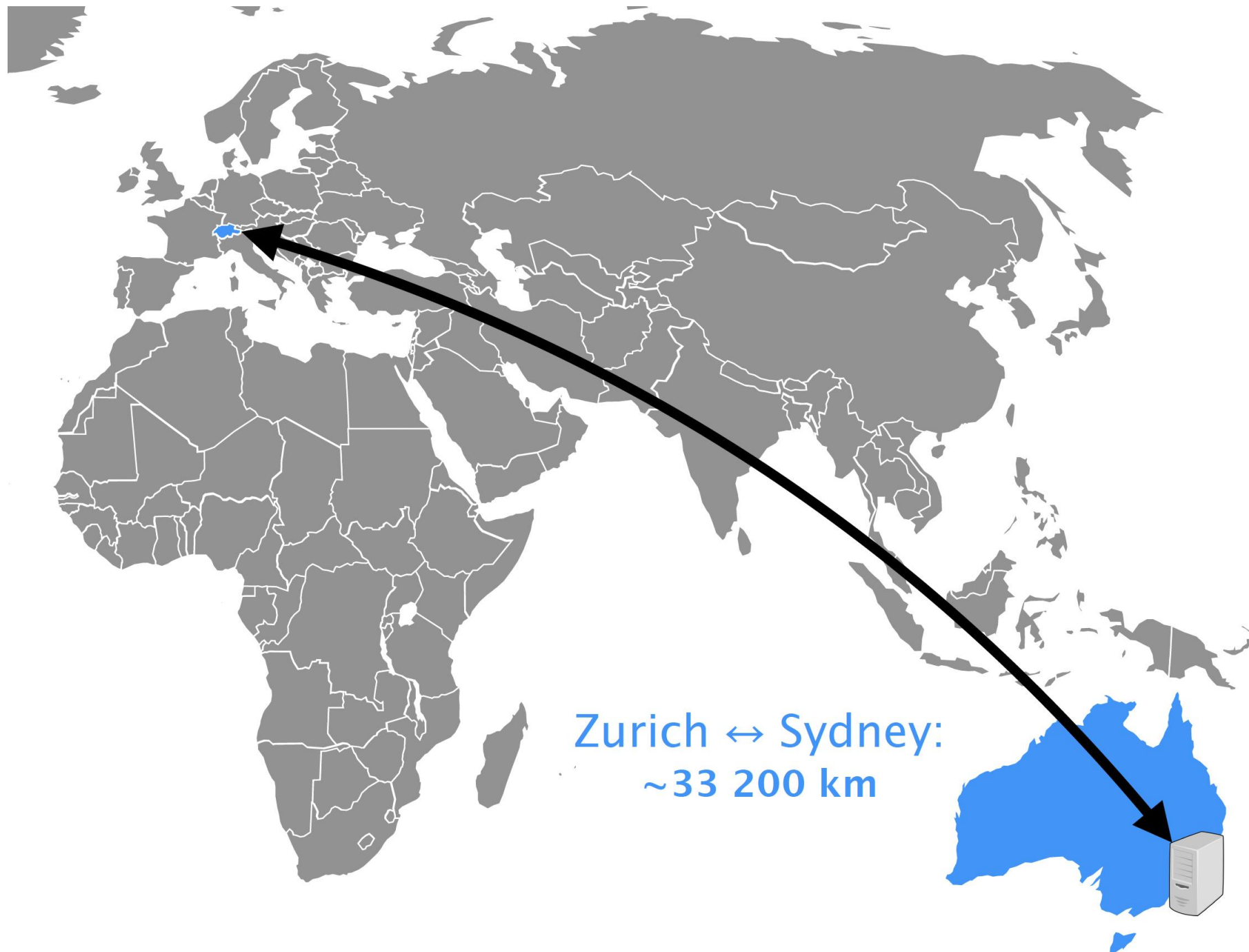
- Piggyback video on HTTP: **widely used**
- Enables independent HTTP requests per segment
 - Works well with CDNs
 - Adapt quality with time, location per client, possibly location over time
 - May use HTTP range requests to ask byte ranges in a segment URL
 - Fetch segments from locations other than the origin server
- More resources on DASH
 - https://www.w3.org/2010/11/web-and-tv/papers/webtv2_submission_64.pdf
 - <https://www.youtube.com/watch?v=xgowGnH5kUE>

Content Distribution Networks (CDNs) (1 of 3)

challenge: how to stream content (selected from millions of videos) to hundreds of thousands of **simultaneous** users?

- **option 1:** single, large “mega-server”
 - single point of failure
 - point of network congestion
 - long (and possibly congested) path to distant clients

....quite simply: this solution **doesn't scale**



$$\frac{299,792,458 \text{ m/s}}{\text{refractive index optical fiber } 1.468} = \text{speed of light in optical fiber } 204,218,296 \text{ m/s}$$

$$\frac{33,200,000 \text{ m}}{204,218,296 \text{ m/s}} = 163 \text{ ms}$$

theoretical
round-trip time

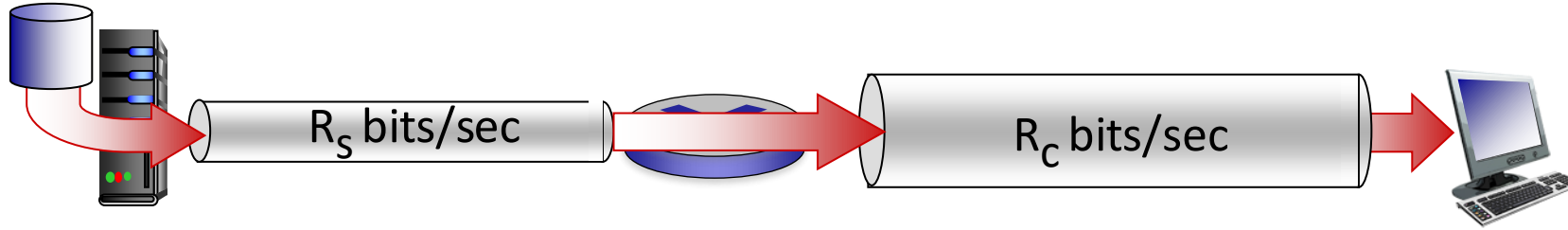
Because of propagation delays,
Content Delivery Networks move content closer to you



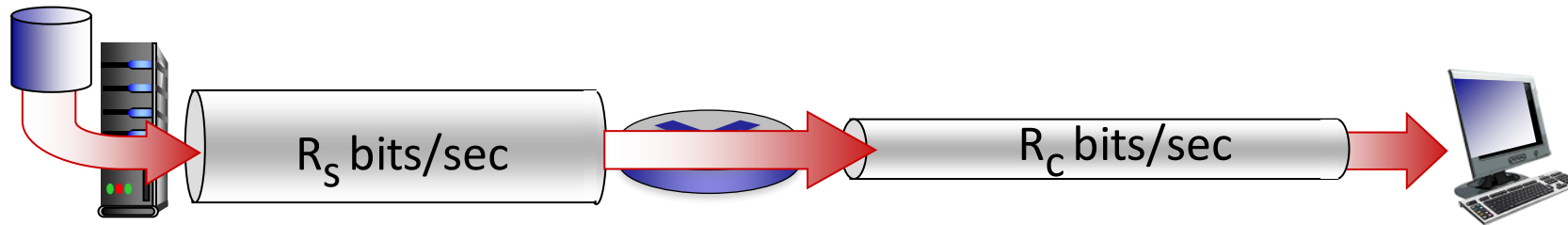
Reference material (not covered in class)

Throughput

$R_s < R_c$ What is average end-end throughput?



$R_s > R_c$ What is average end-end throughput?

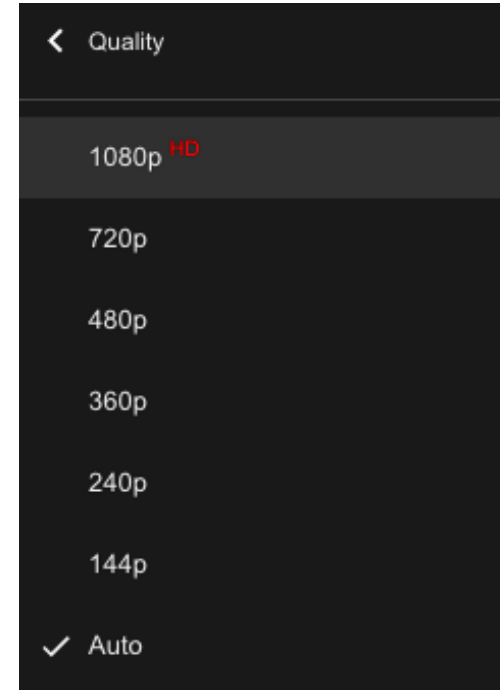


bottleneck link

link on end-end path that constrains end-end throughput

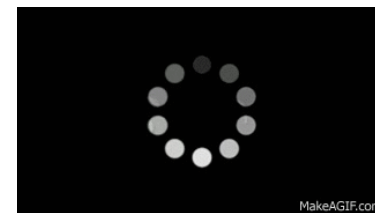
Streaming (stored) video

- Media is prerecorded at different qualities
 - Available in storage at the server
- Client downloads an initial portion and starts viewing
 - The rest is downloaded as time progresses
 - No need for user to wait for entire content to be downloaded (**streaming**)
- Can change the quality of the content and where it's fetched mid-stream

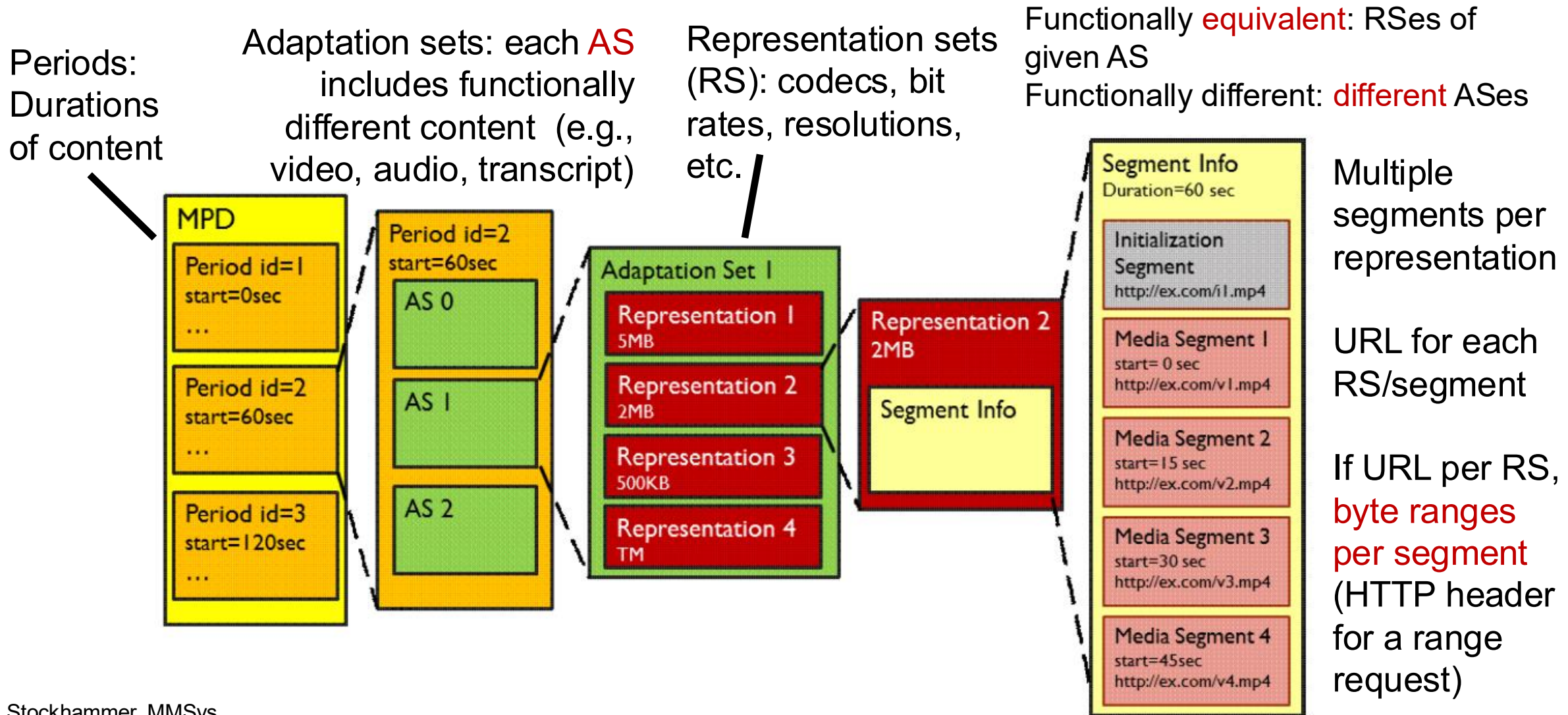


Streaming stored video: challenges

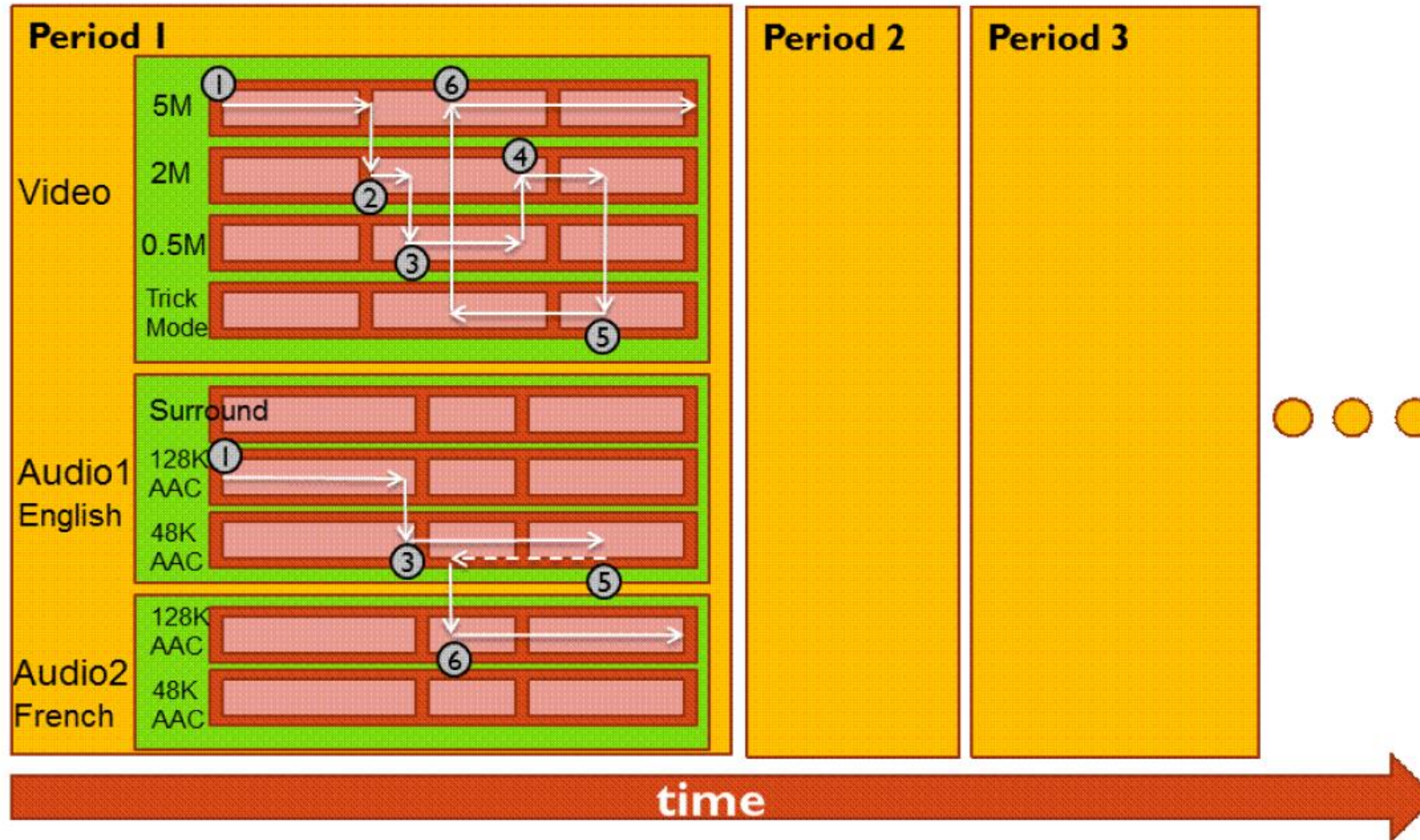
- **Continuous playout constraint**: once video playout begins at client, time gap between frames must match the original time gap in the video (why?)
- But **network delays are variable!**
- Clients have a **client-side buffer** of downloaded video to absorb variation in network delay, available bandwidth
- The video buffer also helps with user interactions: pause, fast-forward, rewind, jump through video



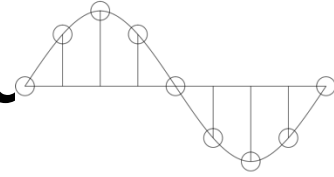
What does the manifest contain?



Adaptive changes in quality



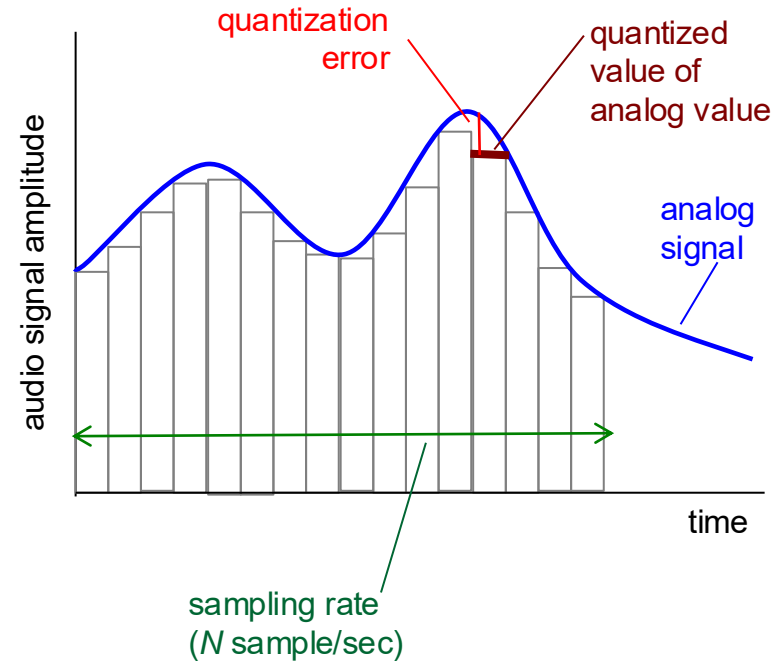
Digital representation of an analog signal



- Must convert analog signal to digital representation
- Sample
 - How many times (twice the max frequency in the signal)
- Quantize
 - How many levels or bits to represent each sample
 - More levels → more accurate representation of signal
 - More levels → more bits to store & need more bandwidth to transmit
- Compress
 - Compact representation of quantized values

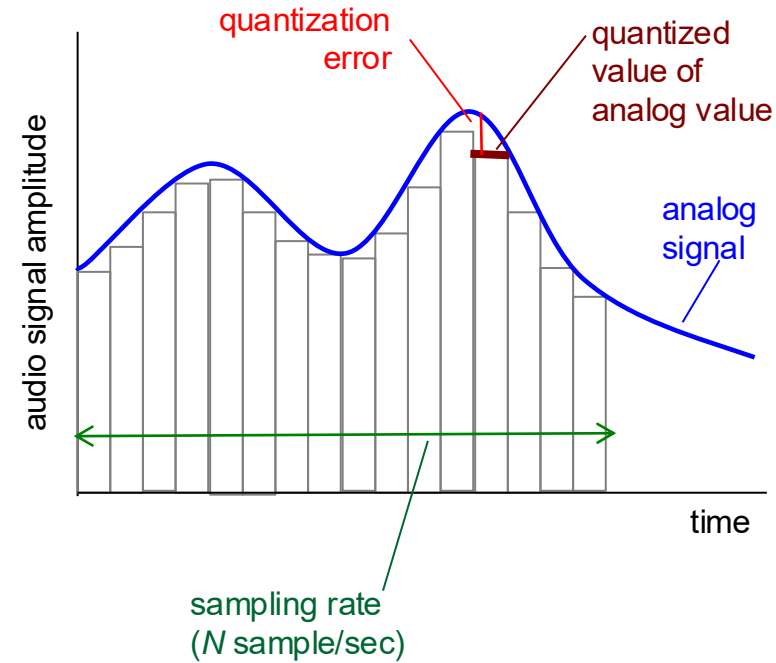
Audio representation

- analog audio signal sampled at constant rate
 - telephone: 8,000 samples/sec
 - CD music: 44,100 samples/sec
- each sample quantized, i.e., rounded
 - e.g., $2^8=256$ possible quantized values
 - each quantized value represented by bits, e.g., 8 bits for 256 values



Audio representation

- example: 8,000 samples/sec, 256 quantized values
- Bandwidth needed: 64,000 bps
- receiver converts bits back to analog signal:
 - some quality reduction



Example rates

- CD: 1.411 Mbps
- MP3: 96, 128, 160 Kbps
- Internet telephony: 5.3 Kbps and up