Wireless Communication Systems
@CS.NCTU

Lecture 12: Video Streaming

Instructor: Kate Ching-Ju Lin (林靖茹)

Ch. 7-2 "Computer Networking: A Top-Down Approach"
Reference: http://www-users.cselabs.umn.edu/classes/Spring-2016/csci5221/

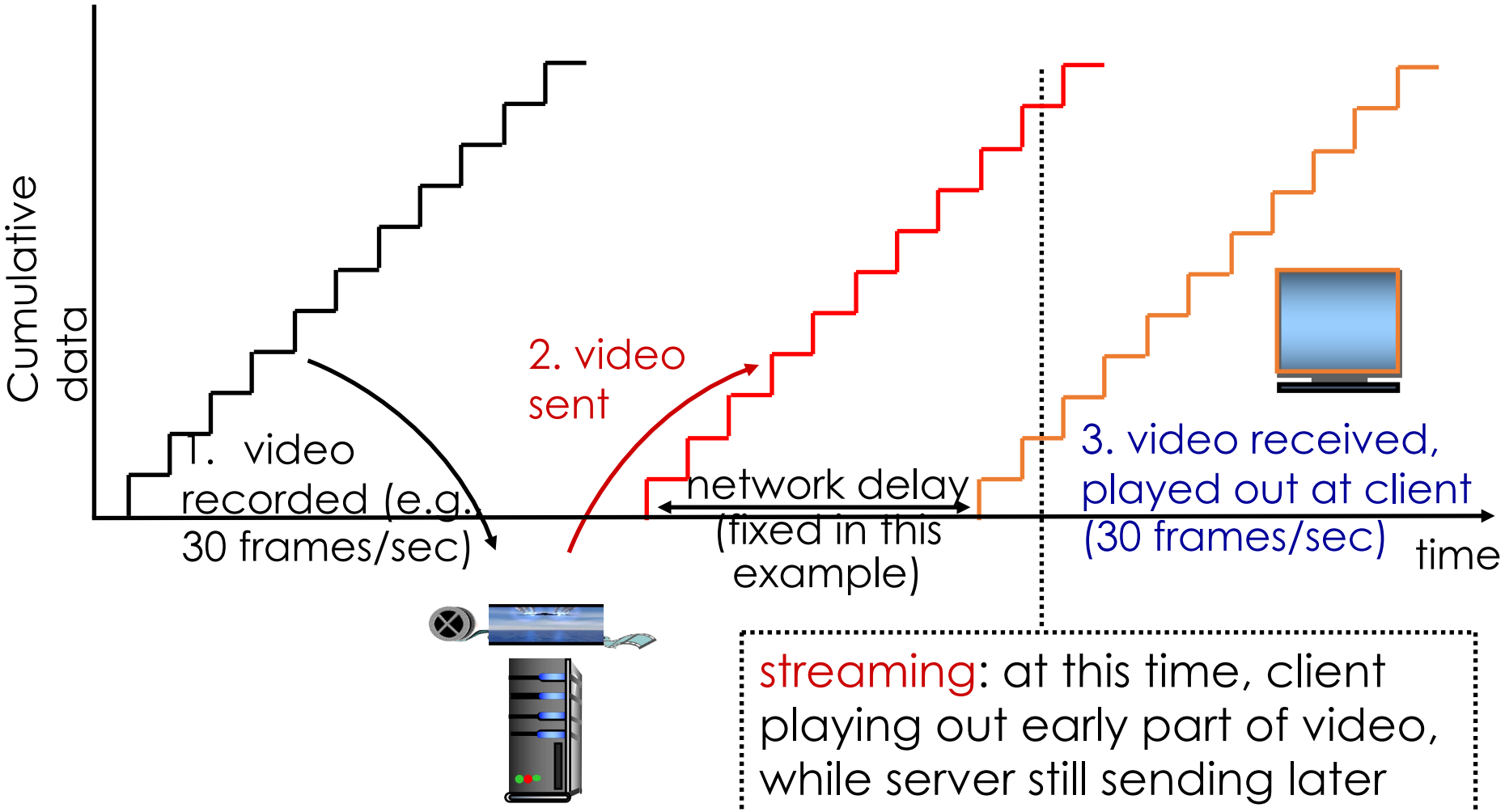
Outline

- **Streaming Basics**
- Performance Metrics
  - Reference:
  - https://nmsl.cs.nthu.edu.tw/images/courses/CS5263_2016/
- HTTP Streaming (DASH)
  - Reference:
  - https://bitmovin.com/dynamic-adaptive-streaming-http-mpeg-dash/
  - Dynamic Adaptive Streaming over HTTP - Standards and Design Principles
  - MPEG-DASH: The Standard for Multimedia Streaming Over Internet
- Video Rate Control

Multimedia Application Types

- **streaming, stored** audio, video
  - streaming: can begin playout before downloading entire file
  - stored (at server): can transmit faster than audio/video will be rendered (implies storing/buffering at client)
  - requested by client on demand
  - e.g., YouTube, Netflix, Hulu, occupying >50% of Internet traffic
- **conversational** voice/video over IP
  - interactive nature of human-to-human conversation limits delay tolerance, e.g., Skype, Google handout
  - highly delay-sensitive, but loss-tolerant
- **streaming live** audio, video
  - e.g., live sporting event (broadcasting)

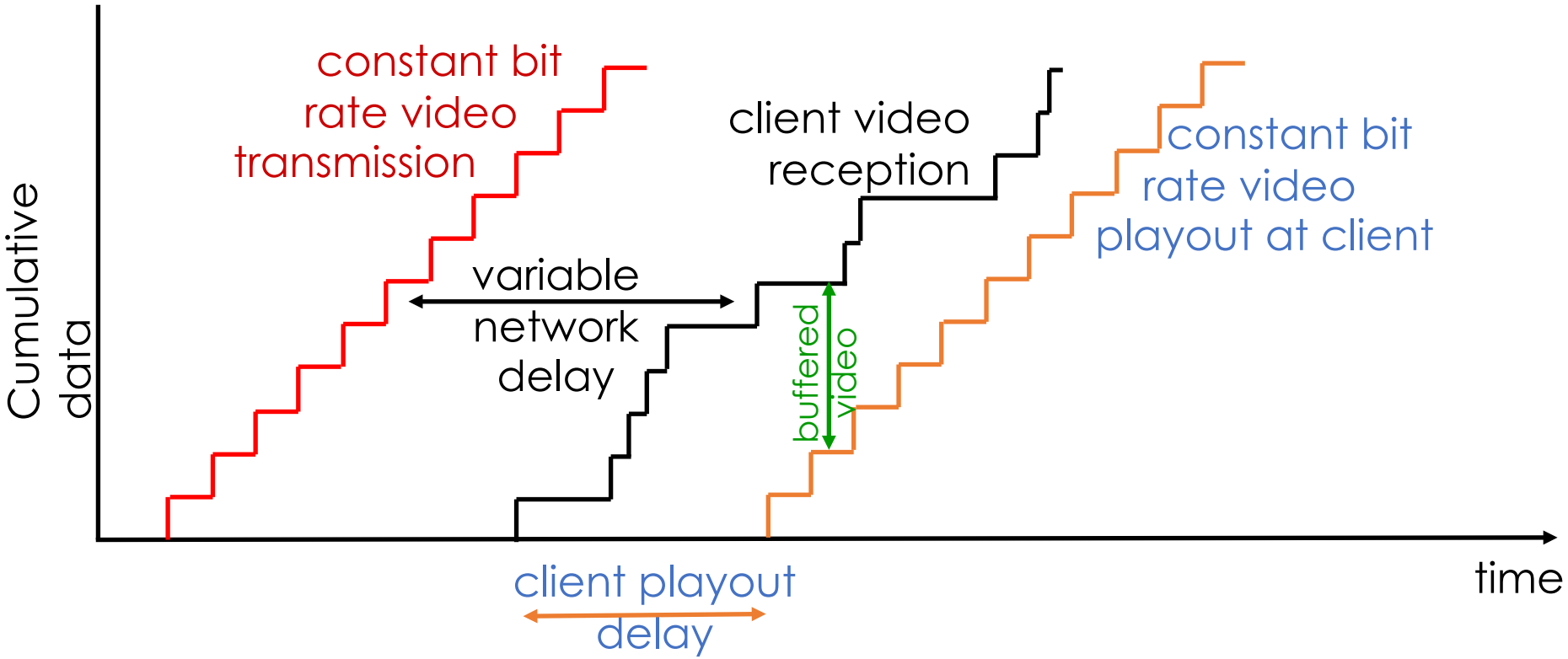
Streaming Stored Video



Streaming Stored Video: Challenges

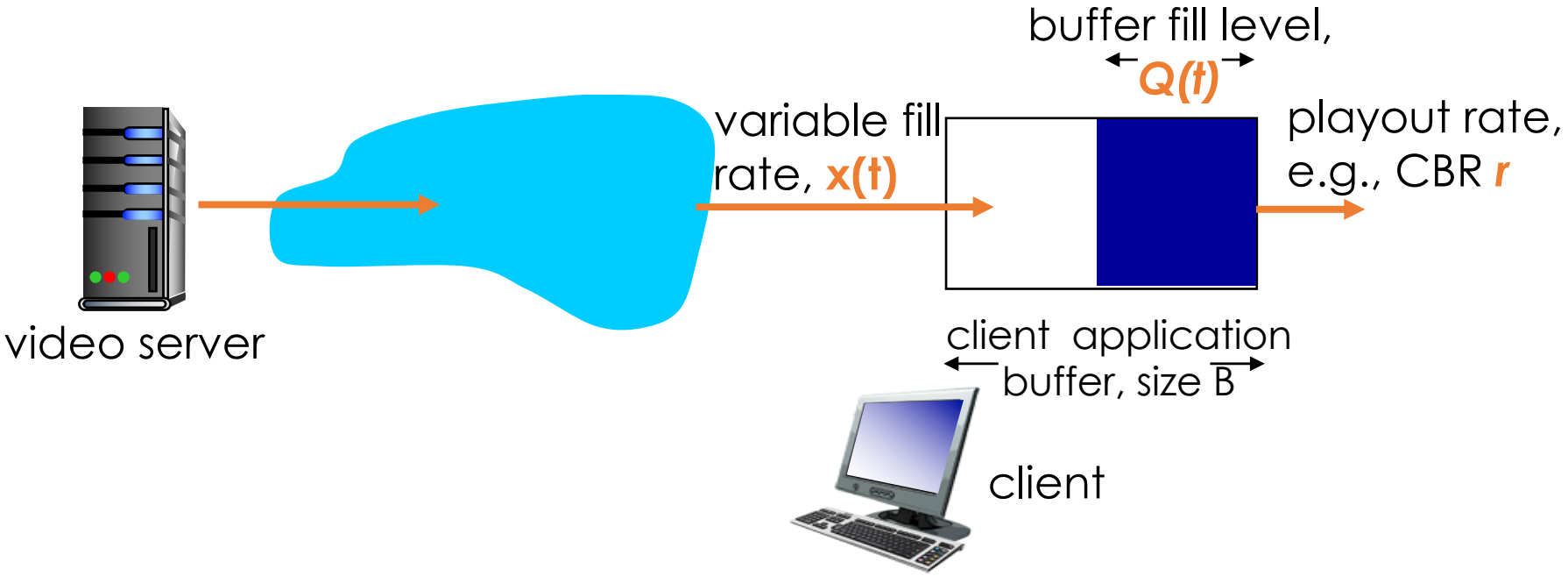
- Continuous playout constraint: once client playout begins, playback must match original timing
  - … but network delays are variable (jitter), so will need client-side buffer to match playout requirements
- Other challenges:
  - client interactivity: pause, fast-forward, rewind, jump through video
  - video packets may be lost, retransmitted

Streaming Stored Video: Revisited

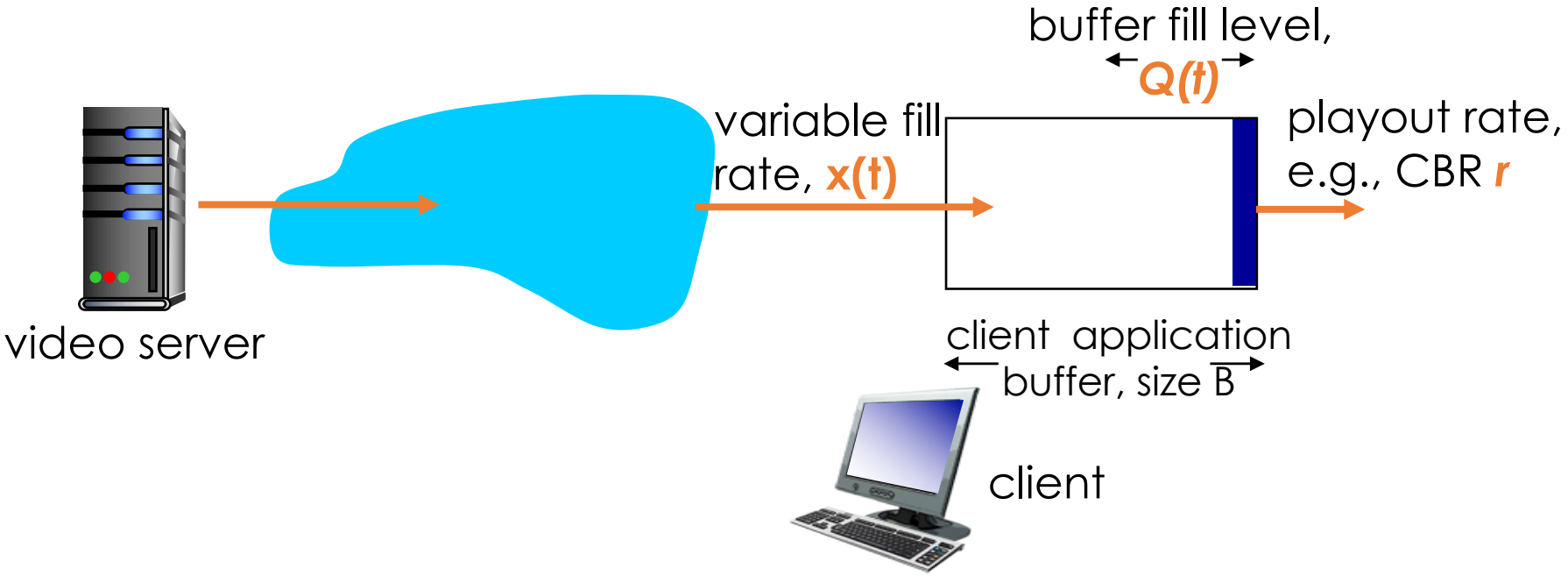


- **client-side buffering and playout delay:**
  compensate for network-added delay, delay jitter

Client-Side Buffering, Playout

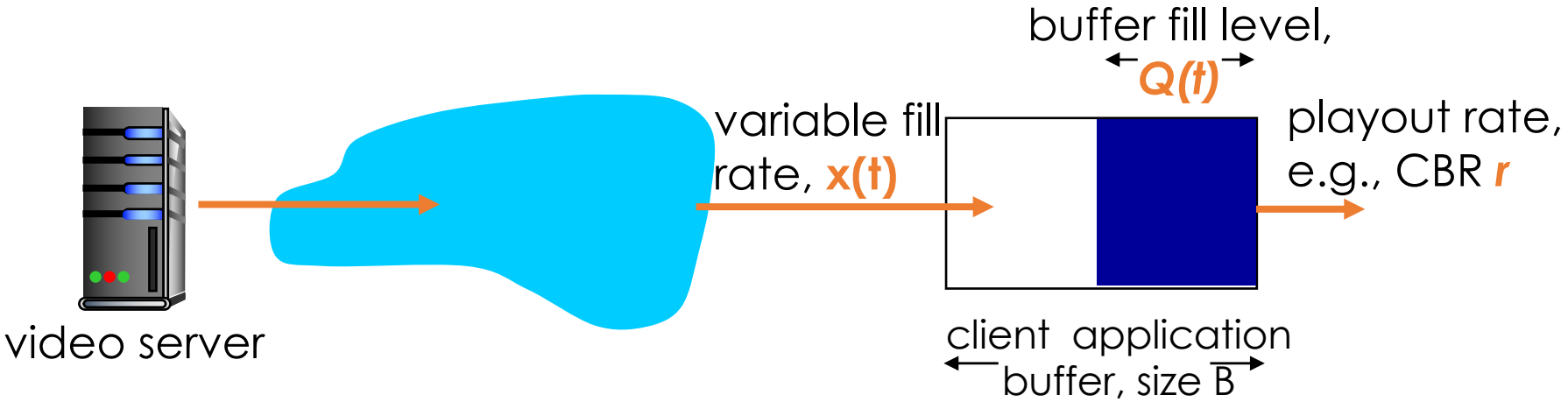


Client-Side Buffering, Playout



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 and playout rate r is constant

Client-Side Buffering, Playout



playout buffering: average fill rate (x), playout rate (r):

- $\overline{x} < r$: buffer eventually empties (causing freezing of video playout until buffer again fills)
- $\overline{x} > r$: buffer will not empty, provided initial playout delay is large enough to absorb variability in x(t)
  - initial playout delay tradeoff: buffer starvation less likely with larger delay, but larger delay until user begins watching

Prefetching

- When a user requests a media content, they may only be asking for a portion of a stream
- Prefetching attempts to load that data into the edge server's cache before it is requested by the user
- When it is successful, prefetching reduces latency
- How to determine whether to prefetch or not?
  - Estimate popularity of chunks
  - Predict according to popularity and user preference

Streaming Live Multimedia

Examples:

- Internet radio talk show
- Live sporting event

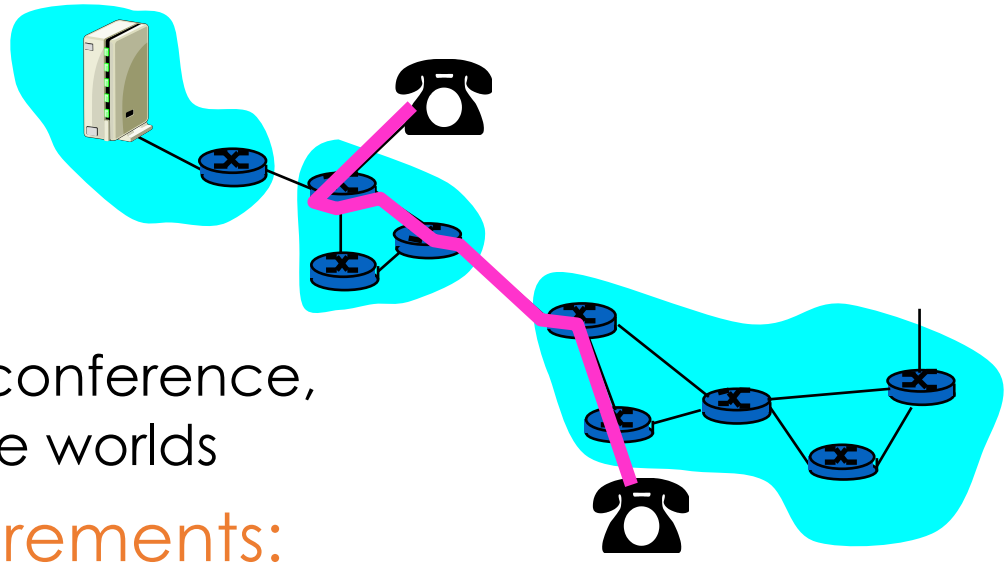
Streaming:

- Playback buffer
- Playback can lag tens of seconds after transmission
- Still have timing constraint → initial startup delay (playback buffer) should be small → smoothness is sacrificed

Interactivity:

- fast forward impossible
- rewind, pause possible!

Interactive, Real-Time Multimedia



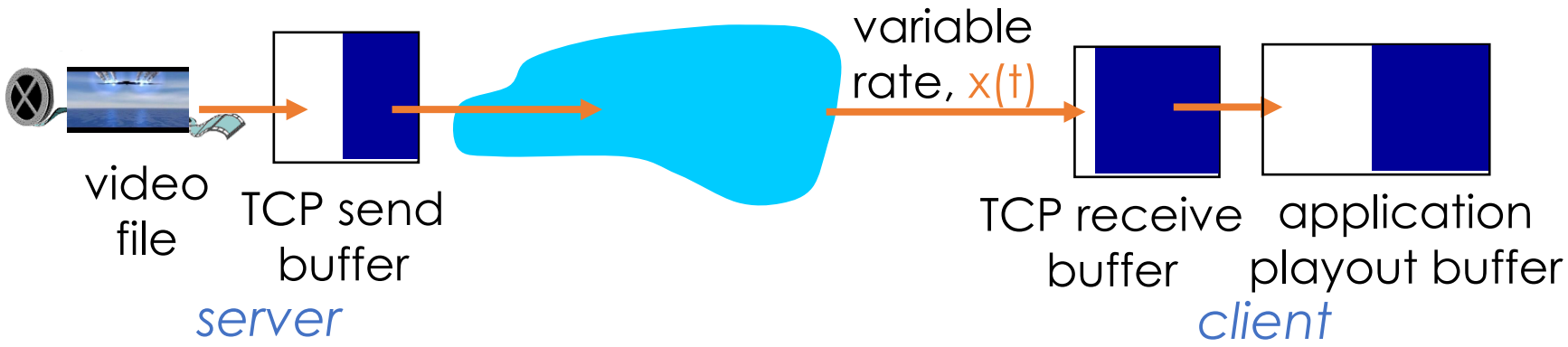
- Applications:
  - IP telephony, video conference, distributed interactive worlds
- End-end delay requirements:
  - audio: < 150 msec good, < 400 msec OK
  - Include application-level (packetization) and network delays
  - Higher delays noticeable, impair interactivity
- Session initialization
  - How does callee advertise its IP address, port number, encoding algorithms?

Streaming Multimedia: UDP

- Server sends at rate appropriate for client
  - often: send rate = encoding rate = constant rate
  - transmission rate can be oblivious to congestion levels
- Short playout delay (2-5 seconds) to remove network jitter
- Error recovery: application-level, time permitting
- RTP [RFC 2326]: multimedia payload types
- UDP may *not* go through firewalls

Streaming Multimedia: HTTP

- Multimedia file retrieved via HTTP GET
- Send at maximum possible rate under TCP



- Fill rate fluctuates due to TCP congestion control, retransmissions (in-order delivery)
- Larger playout delay: smooth TCP delivery rate
- HTTP/TCP passes more easily through firewalls

Multimedia Over Today's Internet

TCP/UDP/IP: "best-effort service"

- ***no*** guarantees on delay, loss

But you said multimedia apps requires QoS and level of performance to be effective!

Today's multimedia applications use application-level techniques to mitigate (as best possible) effects of delay, loss

Challenges

- TCP/UDP/IP suite provides best-effort
  - no guarantees on expectation or variance of packet delay
- Streaming applications delay of 5 to 10 seconds is typical and has been acceptable,
  - but performance deteriorate if links are congested (transoceanic)
- Real-Time Interactive requirements on delay and its jitter have been satisfied by over-provisioning (providing plenty of bandwidth)
  - what will happen when the load increases?...

Streaming Stored/Live Multimedia

Application-Level Techniques

- Application-level streaming techniques for making the best out of best-effort service:
  - Dividing a long video into chunks of smaller sizes
  - Client side buffering
  - Multiple encodings of each video chunks

Media Player

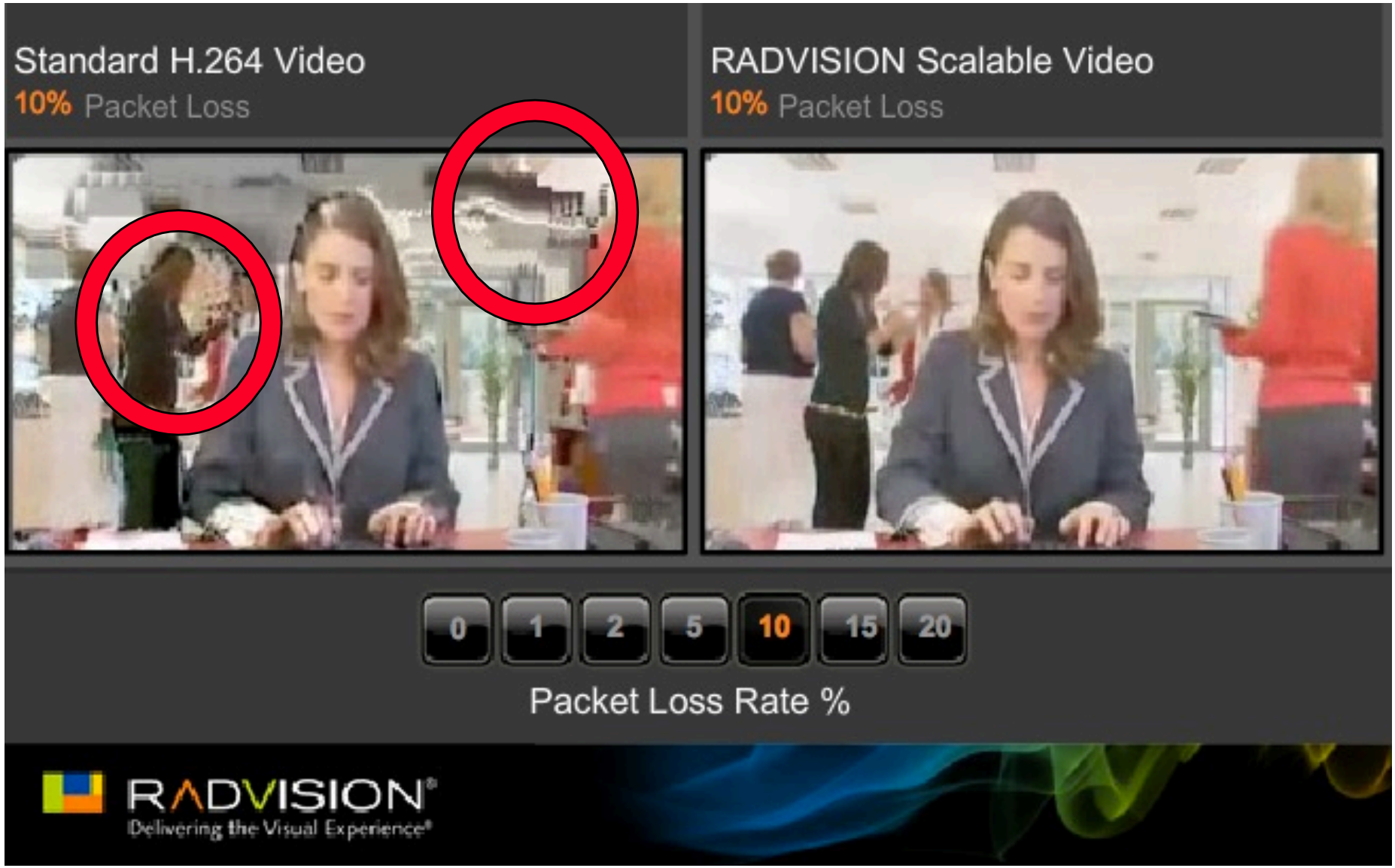
- Jitter removal
- Decompression
- Error concealment
- Graphical user interface w/ controls for interactivity

and with the help of CDNs! (See Lecture 13)

Outline

- Streaming Basics
- **Performance Metrics**
  - Reference:
  - https://nmsl.cs.nthu.edu.tw/images/courses/CS5263_2016/
- HTTP Streaming (DASH)
  - Reference:
  - https://bitmovin.com/dynamic-adaptive-streaming-http-mpeg-dash/
  - Dynamic Adaptive Streaming over HTTP - Standards and Design Principles
  - MPEG-DASH: The Standard for Multimedia Streaming Over Internet
- Video Rate Control

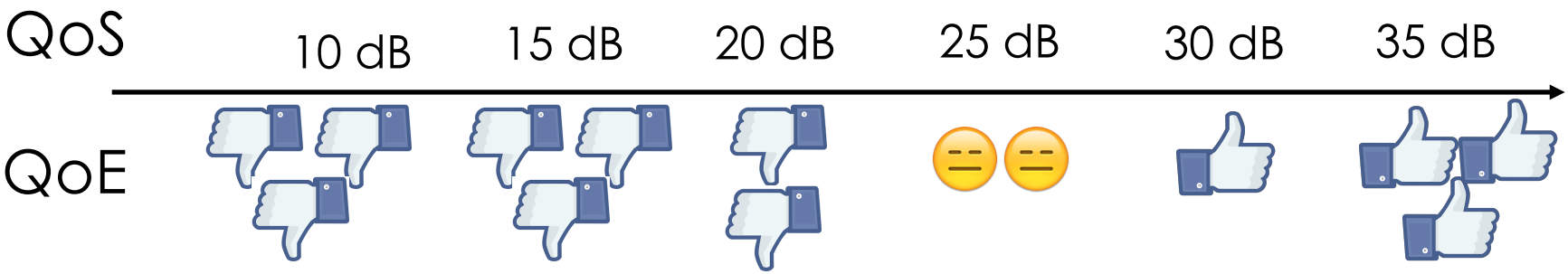
Visual Impairments due to Losses



Subjective or Objective?

- **Qualify of Server (QoS)**
  - Objective (quantitative) measurements
  - How good is the received content?
    → only consider content itself
- **Quality of Experience (QoE)**
  - Subjective (qualitative) measurements
  - What a user wants
    → Consider user perception (satisfaction)

High QoS may not imply high QoE! For example



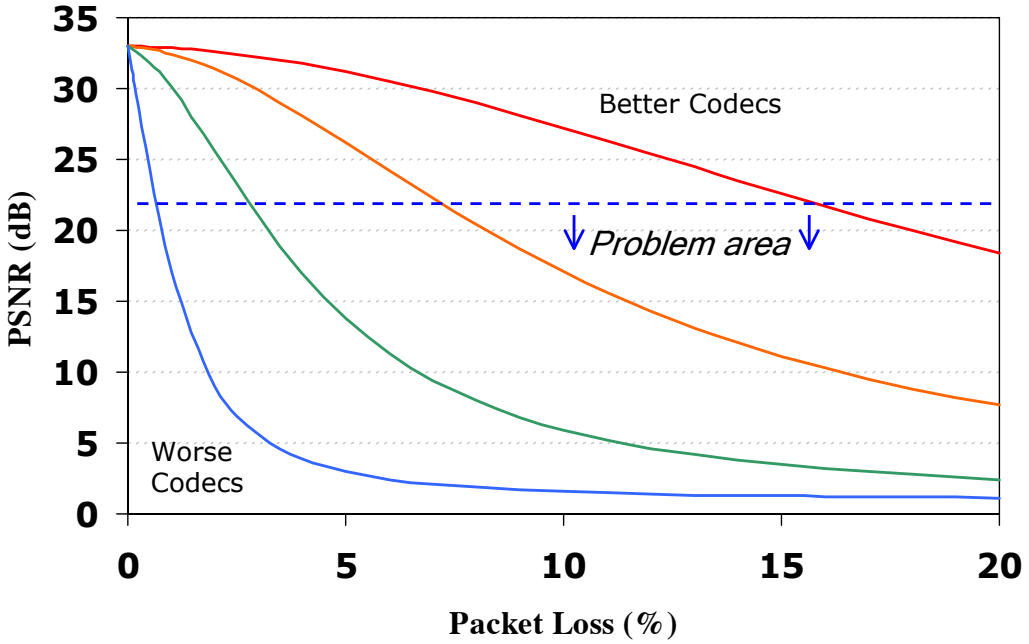
Subject Metrics

- **Voice: Mean Opinion Score (MOS)**
  - Users grade voice quality from 1 to 5
  - Above 4 is good quality
  - Various variations with difference score ranges

- **Video: ITU-R BT.500**
  - Several modes are defined
  - e.g., Double Stimulus Impairment Scale (DSIS)
    - first show the full-quality video, then show the impaired one
    - Viewers are informed the order
    - Viewers are asked to score the impaired video

Objective Metrics

- **Packet-based metrics**
  - Use network measurements and (optionally) codec properties to infer the degraded video quality
  - Low complexity and work without original videos
- **Example V-Factor**
  - V = f(QER, PLR, R)
  - QER: codec quality
  - PLR: packet loss ratio
  - R: video complexity
  - Adopted by Sprint



https://www.slideshare.net/RockyS11/iptv-deployment-performance-planning-and-management-architecture

Objective Metrics (cont.)

- **Content-based metrics**
  - Compute the quality level using the video itself
  - Used in research labs for, e.g., comparing video codec performance
- Classified into three groups
  - Full reference: Assuming both the original and impaired videos are available
    → less practical, but widely used in research labs
  - Reduced reference: original videos are analyzed and a summary is compared against the impaired video
  - No reference: metrics that do not need original videos → ideal metrics

Full Reference Metrics

- Most quality metrics consider only Y-component
- Pixel-based metrics
  - **MSE (Mean Square Error)**

$$\sigma_n^2 = \frac{1}{N}\sum_{i=1}^{N}(x_i - y_i)^2$$

  - **PSNR (dB) (Peak Signal-to-Noise Ratio)**

$$\mathrm{PSNR} = 10\log_{10}\left(\frac{\sigma_{\mathrm{peak}}^2}{\sigma_n^2}\right)$$

Problems with PSNR

MSE=0, original picture

- Mean Structural Similarity (MSSIM)
- Cannot directly map to user-perceived quality

  → If so, why researchers still use it?

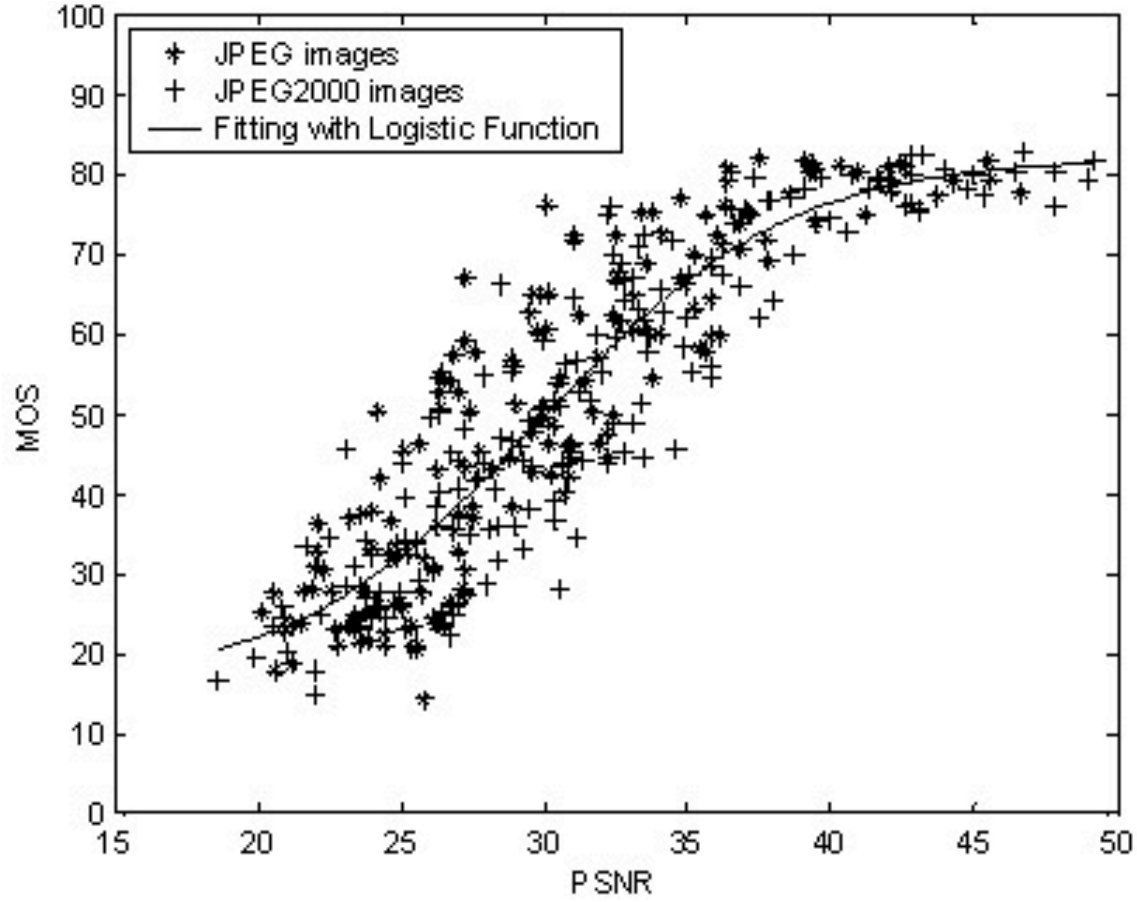
MSE=225, MSSIM=0.949

MSE=225, MSSIM=0.688

MSE=225, MSSIM=0.723

Performance Comparison

- Relationship between subjective and objective metrics



Source: https://ece.uwaterloo.ca/~z70wang/research/ssim/

Outline

- Streaming Basics
- Performance Metrics
  - Reference:
  - https://nmsl.cs.nthu.edu.tw/images/courses/CS5263_2016/
- **HTTP Streaming (DASH)**
  - Reference:
  - https://bitmovin.com/dynamic-adaptive-streaming-http-mpeg-dash/
  - Dynamic Adaptive Streaming over HTTP - Standards and Design Principles
  - MPEG-DASH: The Standard for Multimedia Streaming Over Internet
- Video Rate Control

HTTP Streaming

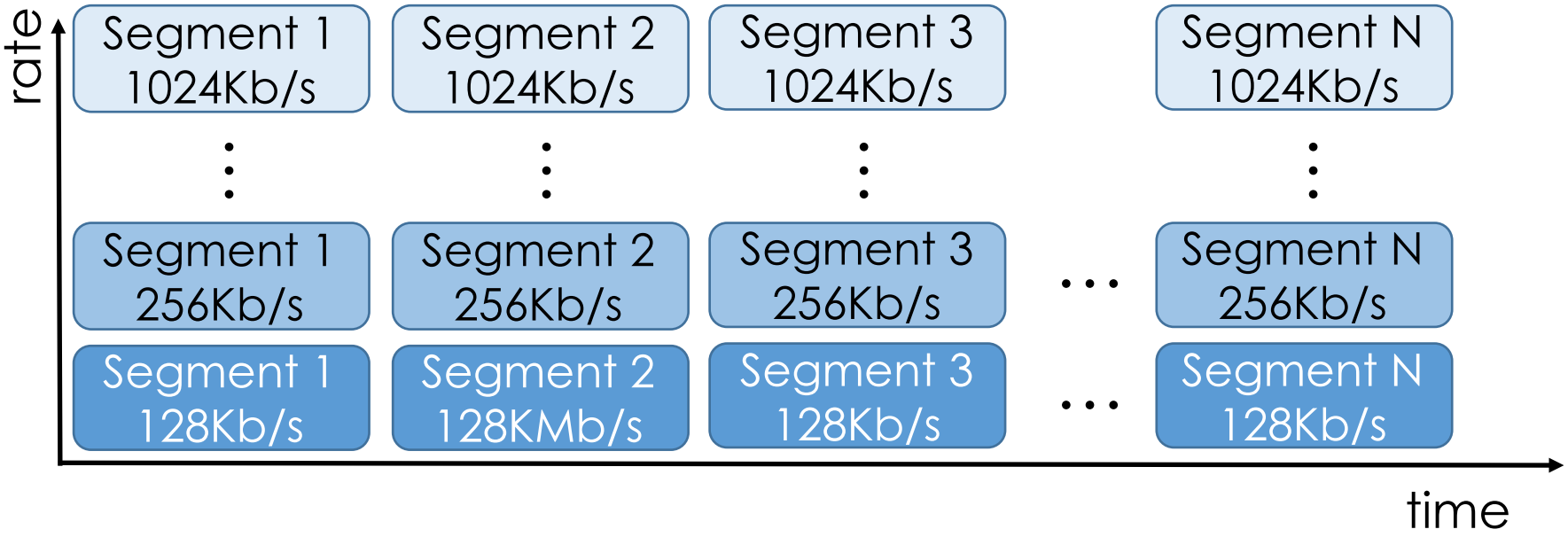
- ***DASH***: *D*ynamic, *A*daptive *S*treaming over *H*TTP
- server:
  - Divide video file into multiple chunks
  - Each chunk stored, encoded at different rates
  - Manifest file: provides URLs for different chunks
- client:
  - Periodically measures server-to-client bandwidth
  - Consulting manifest, requests one chunk at a time
    - Choose maximum coding rate sustainable given current bandwidth
    - Can choose different coding rates at different points in time (depending on available bandwidth at time)

Why HTTP Streaming

- Traditional streaming delivered over RTP/UDP
- However, in today's Internet, content objects are stored Content Delivery Networks (CDN)
  - Many CDNs do not support RTP streaming
  - RTP often does not allow traffic through firewall
  - Each RTP stream is a separate session → not scalable
- Benefits of HTTP streaming
  - Firewall friendly
  - No need to maintain the session state on the server
  - Has been standardized as "ISO/IEC 23009-1, also known as MPEG-DASH" in Apr. 2012

MPEG-DASH in a Nutshell

- Partition a media file into **segments**
- Each segment can be encoded at different bit-rates or spatial resolutions
- Each segment is downloaded through HTTP standard compliant GET requests

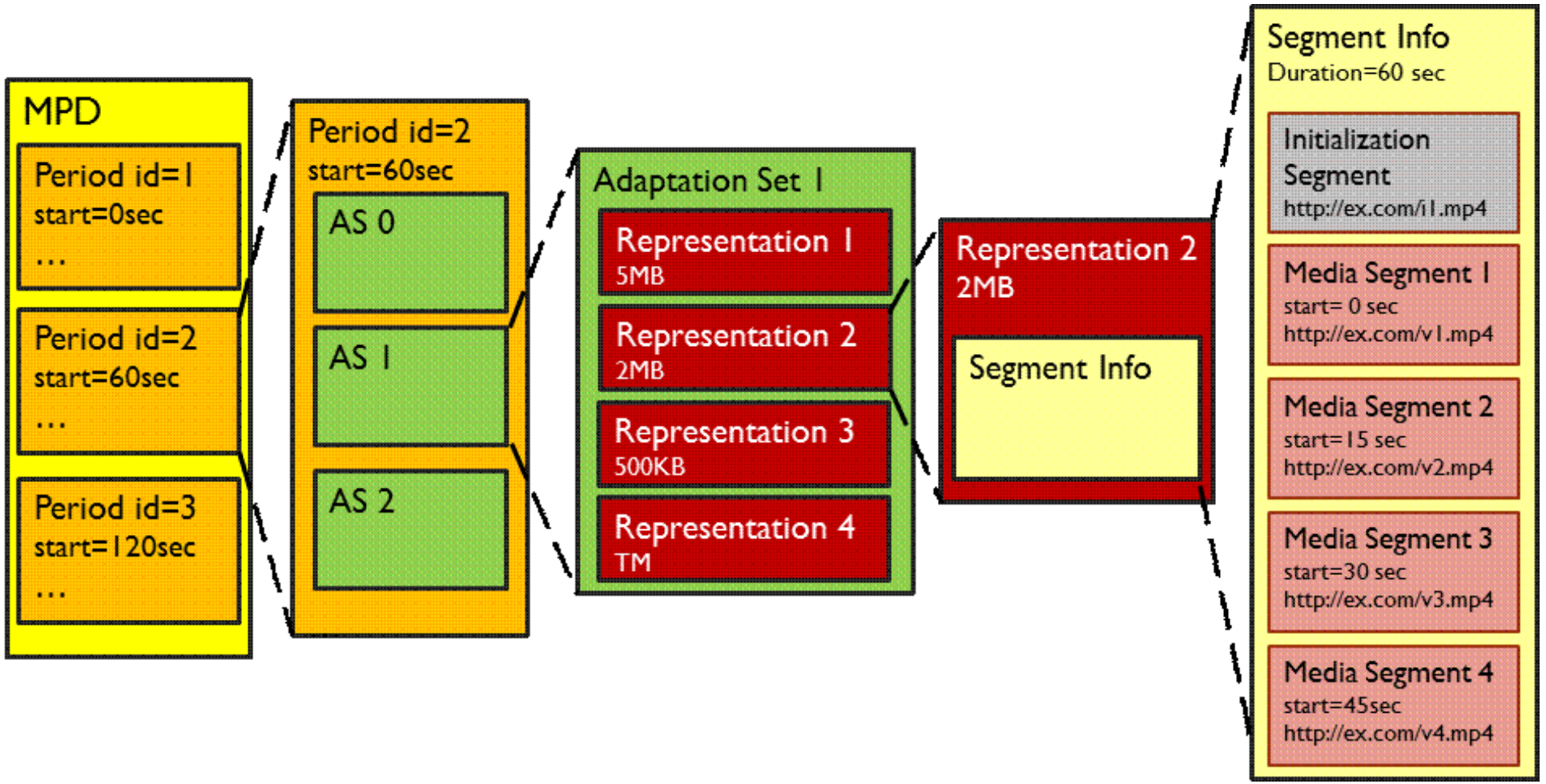


Media Presentation Description

- A.k.a **MPD**, a XML document describing the manifest of the available content
- Used to describe the temporal and structural relationships between segments
- Represented as a hierarchical data model
  - Period → Adaptation set → Representation → Segments
- Each MPD could contain one or more *periods*, each of which contains media components, e.g.,
  - Video components (e.g., different view angles or codecs)
  - Audio components (e.g., for different language)
  - Subtitle components
- Client can adapt during a period according to the available bitrates, resolutions, codecs, etc.

MPD Manifest

- **manifest file:** provides URLs for different chunks

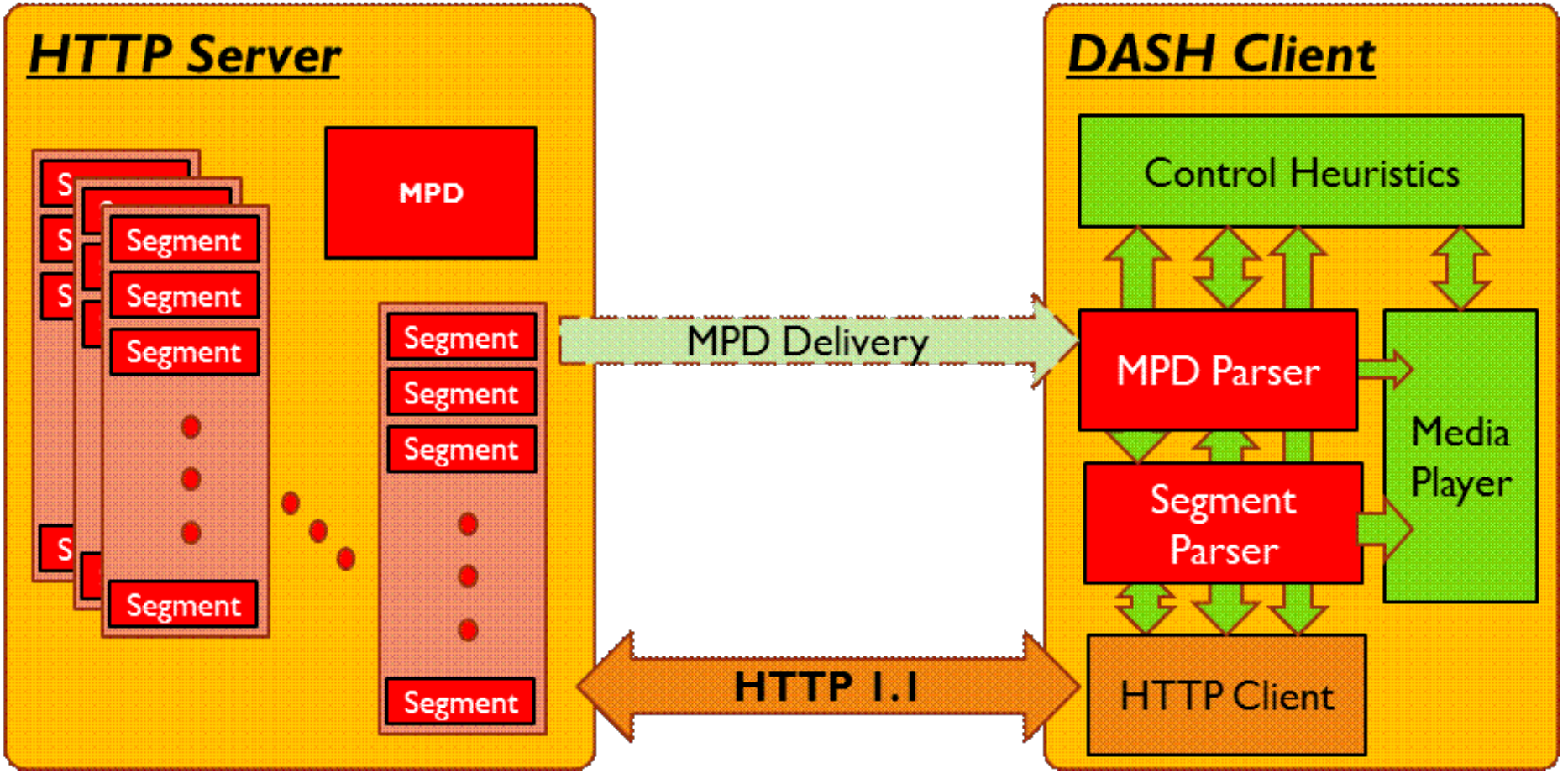


Iraj Sodagar, "MPEG-DASH: The Standard for Multimedia Streaming Over Internet"

Segments

- Each segment is described by a URL
- Segments in a Representation usually have the same length
- MPEG-DASH does not restrict the segment length or give advice on the optimal length
  - Can be chosen depending on the given scenario
  - Longer segments allow more efficient compression since a longer GOP needs less overhead
  - Shorter segments are preferable for live streaming or highly dynamic network conditions

HTTP Server-Client Framework



Server:

- MPD describes a manifest of content and their URL/characteristics
- Segments contain chunks of the actual multimedia bitstreams

Client:

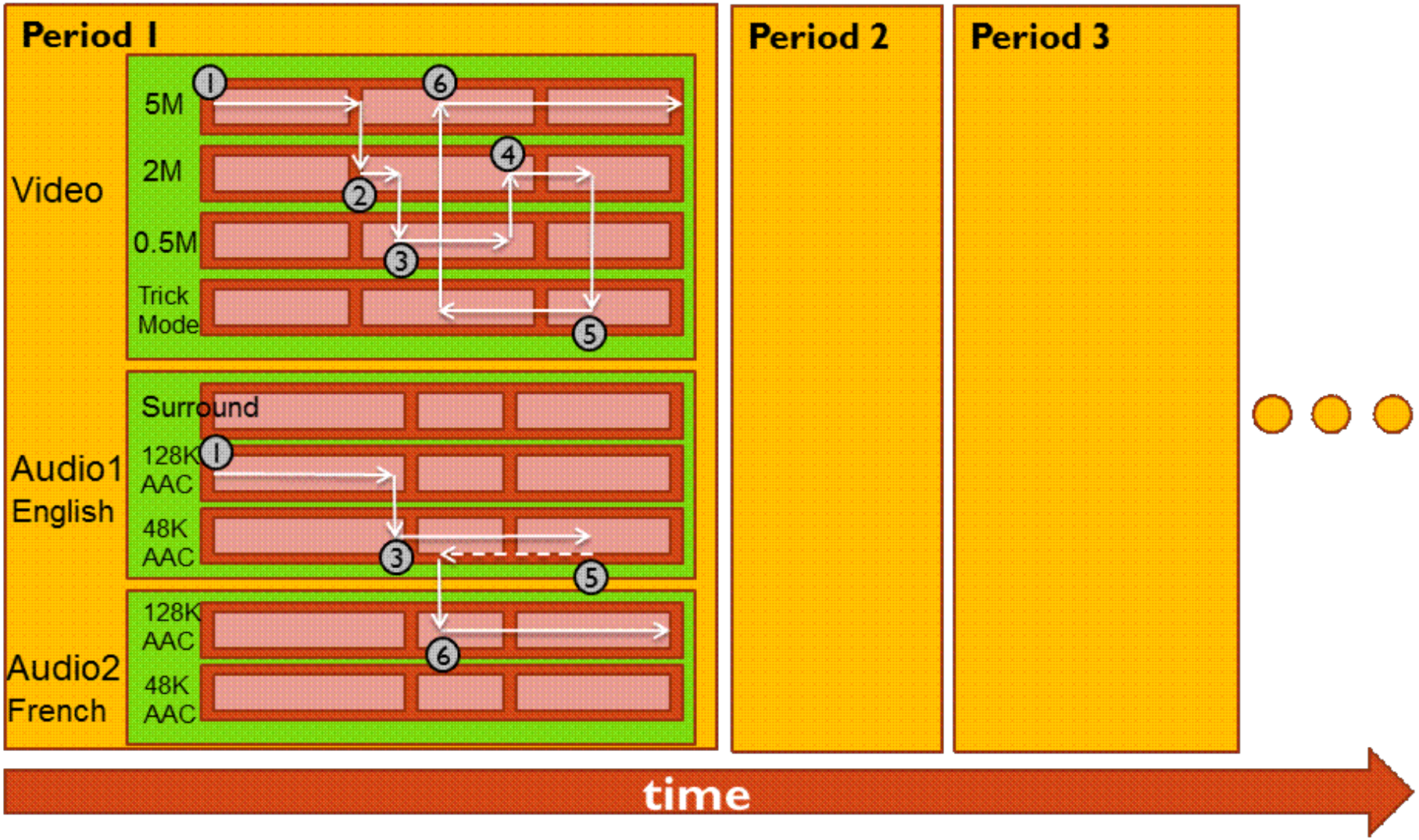
- Fetch segments using HTTP GET requests

Adaptation at Clients

"intelligence" at client: client determines

- *When* to request chunk (so that buffer starvation, or overflow does not occur)
- *What encoding rate* to request (higher quality when more bandwidth available)
- *Where* to request chunk (can request from URL server that is "close" to client or has high available bandwidth)

Example of Adaptation



Iraj Sodagar, "MPEG-DASH: The Standard for Multimedia Streaming Over Internet"

Try to trace how it works …

Use wireshark to capture and view packets being sent to/from Netflix or Youtube

- Logging in (authorization), content selection
  - Main webpage hosted by amazon cloud?
  - Use whois to see who server is
- Select video and begin playing
  - Use whois to see who is serving video
  - How far away is server (use traceroute, no info? Ip2location.com)

Case Study: Netflix

- 30% downstream US traffic in 2013
- $1B quarterly revenue
- 2B hours viewing streamed video
- Owns very little infrastructure, uses 3rd party services:
  - Own registration, payment servers
  - Amazon (3rd party) cloud services:
    - Netflix uploads studio master to Amazon cloud
    - create multiple version of movie (different encodings) in cloud, move to CDNs
    - Cloud hosts Netflix web pages for user browsing

http://www.statisticbrain.com/netflix-statistics/

Case Study: Netflix

- Uses *three* 3rd party CDNs to store, host, stream Netflix content:
  - Akamai, Limelight, Level-3
  - Can play each of the CDNs against each other in terms of customer experience
- Developing its own CDN (2012)

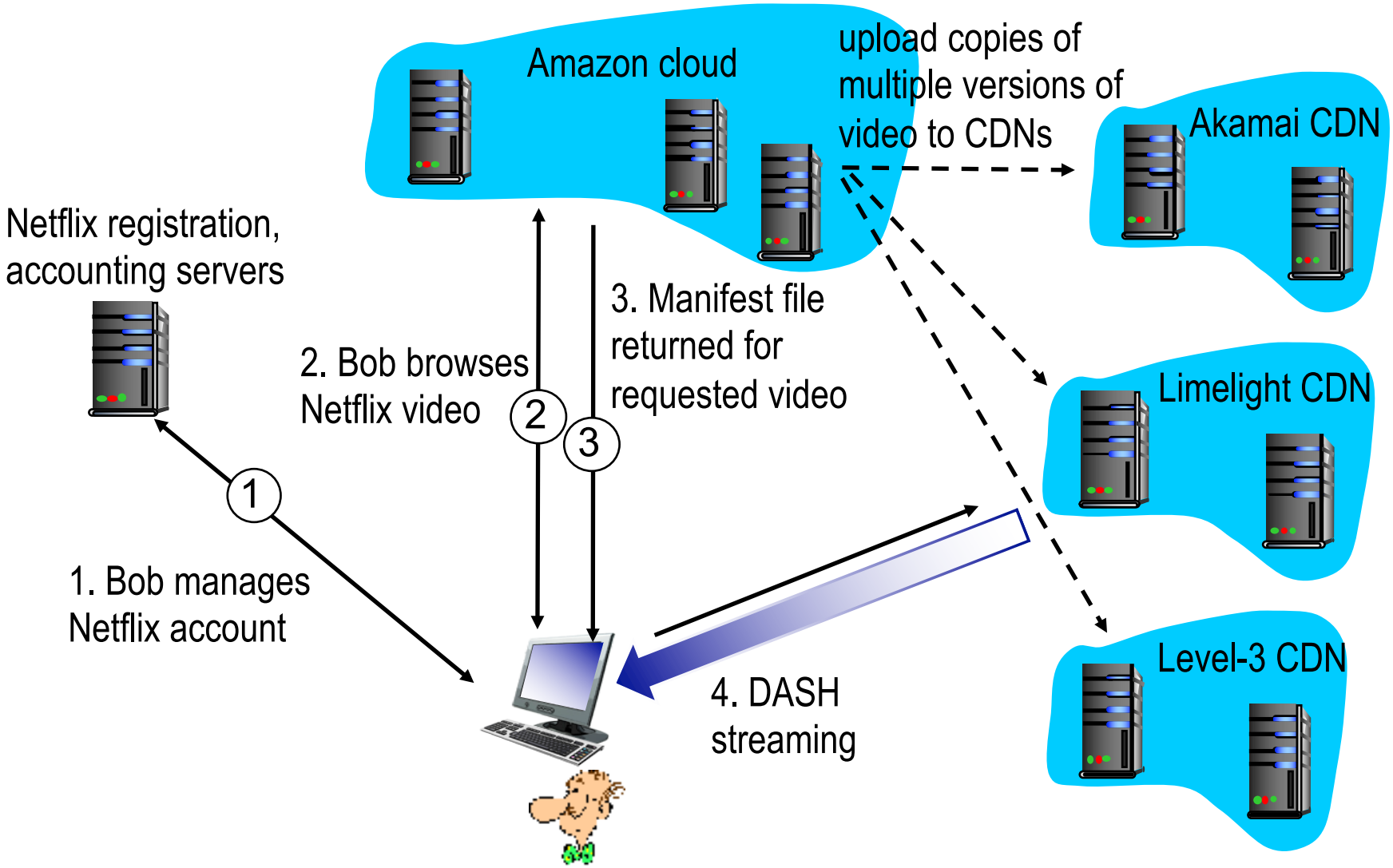
	Upstream Traffic		Downstream Traffic		Total Traffic	
Rank	Application	Share	Application	Share	Application	Share
1	BitTorrent	52.01%	Netflix	29.70%	Netflix	24.71%
2	HTTP	8.31%	HTTP	18.36%	BitTorrent	17.23%
3	Skype	3.81%	YouTube	11.04%	HTTP	17.18%
4	Netflix	3.59%	BitTorrent	10.37%	YouTube	9.85%
5	PPStream	2.92%	Flash Video	4.88%	Flash Video	3.62%

SOURCE: SANDVINE NETWORK DEMOGRAPHICS

sandvine

Table 1 - North America - Top Applications by Bytes (Peak Period, Fixed Access)

Case Study: Netflix



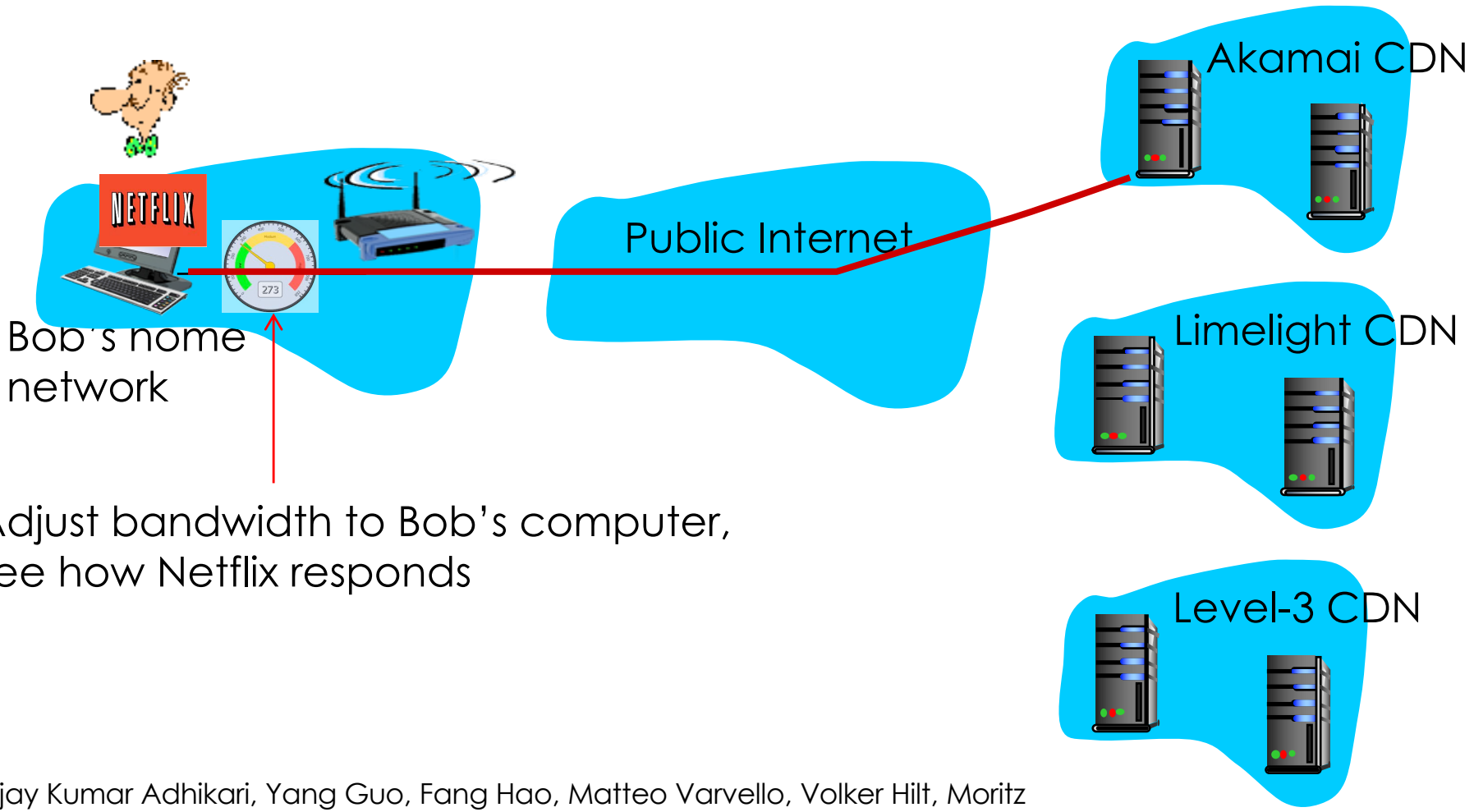
Client Adaptation

Users adapts to Bandwidth Changes!

- *Question:* how does Netflix client adapt to changes in bandwidth (due to changing congestion levels on network path)

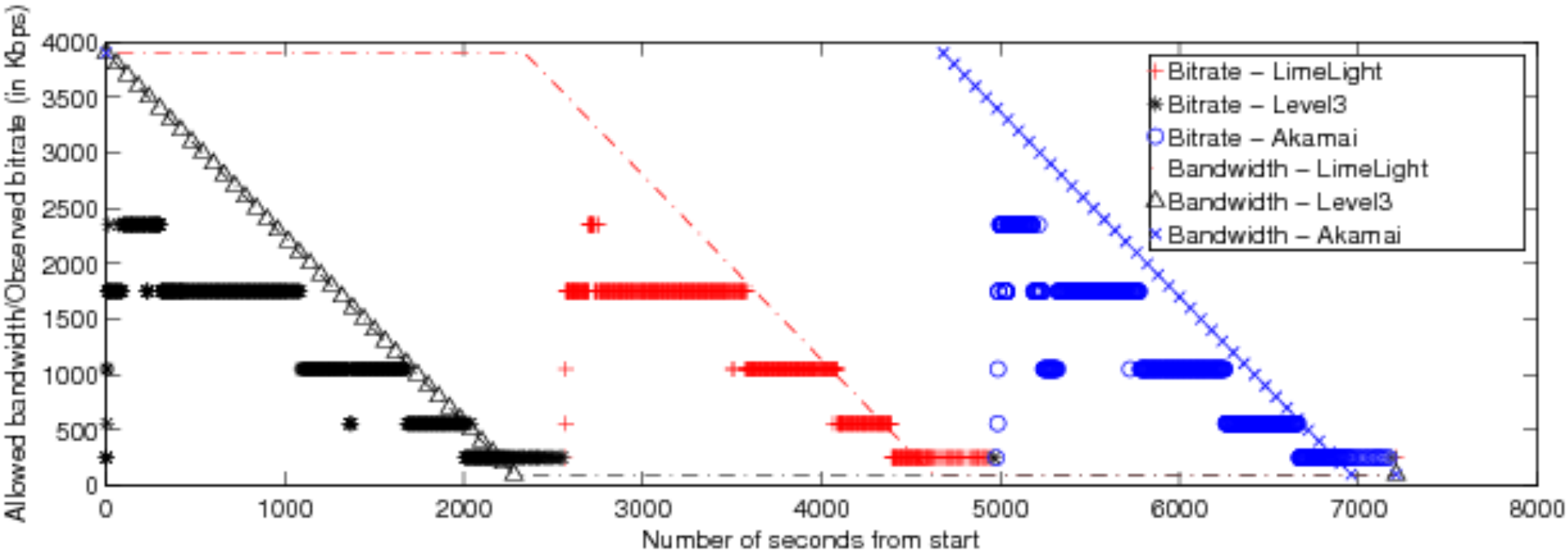
- Two client-based alternatives:
  - Get video from new CDN server (over new path)?
  - Change video streaming rate via DASH?

Netflix Experiment



Vijay Kumar Adhikari, Yang Guo, Fang Hao, Matteo Varvello, Volker Hilt, Moritz Steiner, Zhi-Li Zhang, "Unreeling Netflix: Understanding and Improving Multi-CDN Movie Delivery", *INFOCOM*, Orlando, FL, USA, March, 2012.

Netflix Experiment



Netflix seems to *dynamically* use alternate CDNs only for fail-over

Secure HTTP Streaming

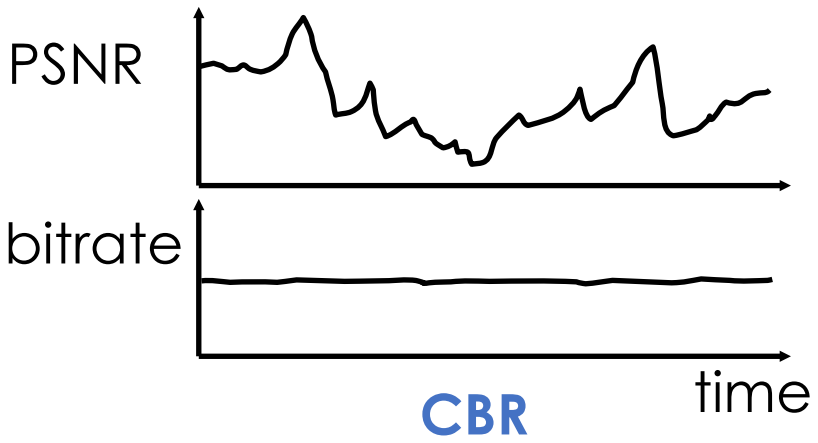
- Most of content providers, , e.g., Netflix, now deliver content over HTTPs
- Become harder to trace and monitor streaming packets
- If so, how can we evaluate our design?
  - Option 1: Setup a proxy server by yourself to cache the video segments
  - Option 2: Build your own HTTP and content server using Apach and Dash.js https://github.com/Dash-Industry-Forum/dash.js/wiki

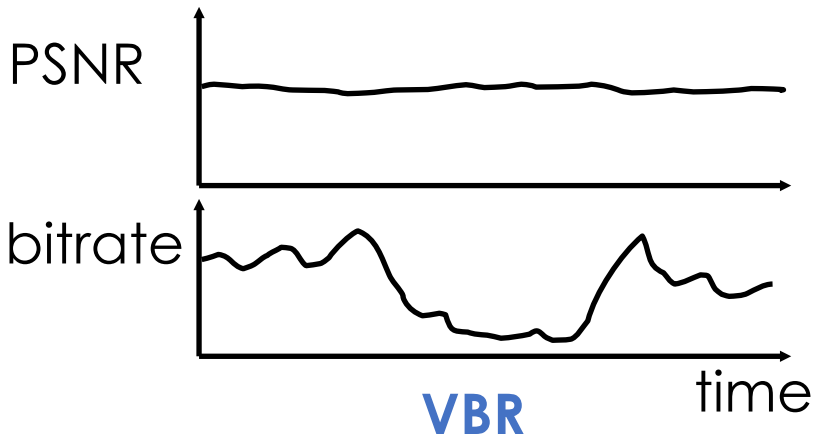
Outline

- Streaming Basics
- Performance Metrics
  - Reference:
  - https://nmsl.cs.nthu.edu.tw/images/courses/CS5263_2016/
- HTTP Streaming (DASH)
  - Reference:
  - https://bitmovin.com/dynamic-adaptive-streaming-http-mpeg-dash/
  - Dynamic Adaptive Streaming over HTTP - Standards and Design Principles
  - MPEG-DASH: The Standard for Multimedia Streaming Over Internet
- **Video Rate Control**

Rate Control

- Variable-bitrate nature of video
- **Channel:** constant bit rate (CBR) (e.g., PSTN, ISDN, ...) or variable bit rate (VBR) (e.g., IP, ...)
- Bit-rate and delay constraints
- Rate-control determines the quantization step-size for each macroblock or frame-skipping to produce good video quality without encoder buffer overflow



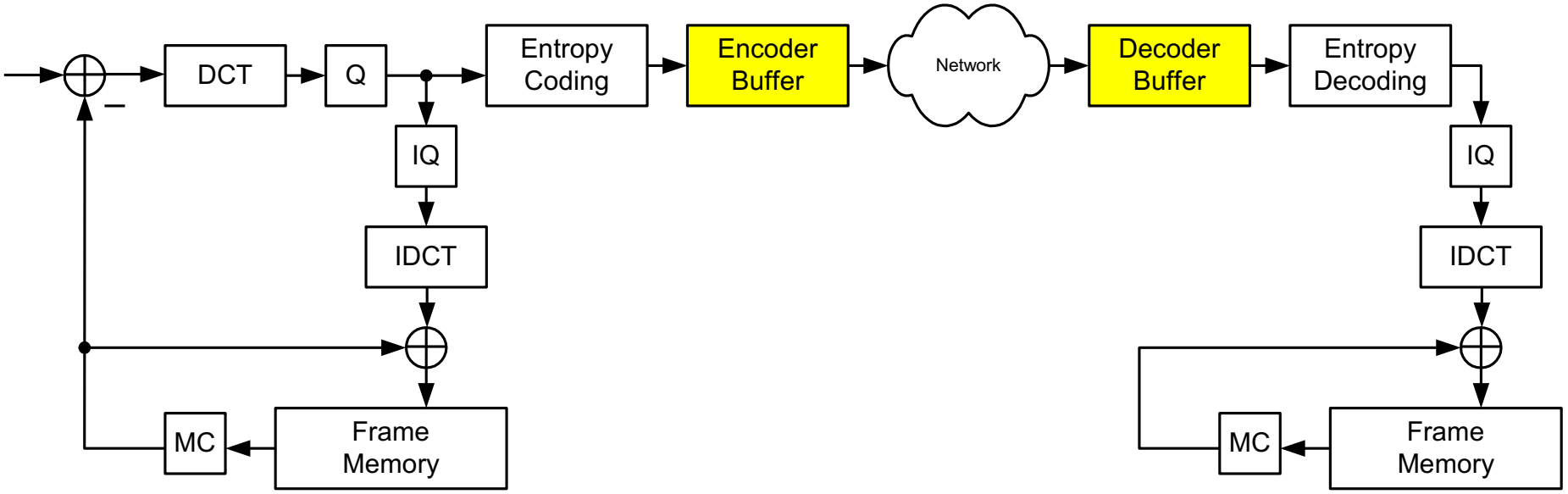


Rate Control

- Use a buffer to
  - smooth out the bit-rate
  - match the channel bitrate

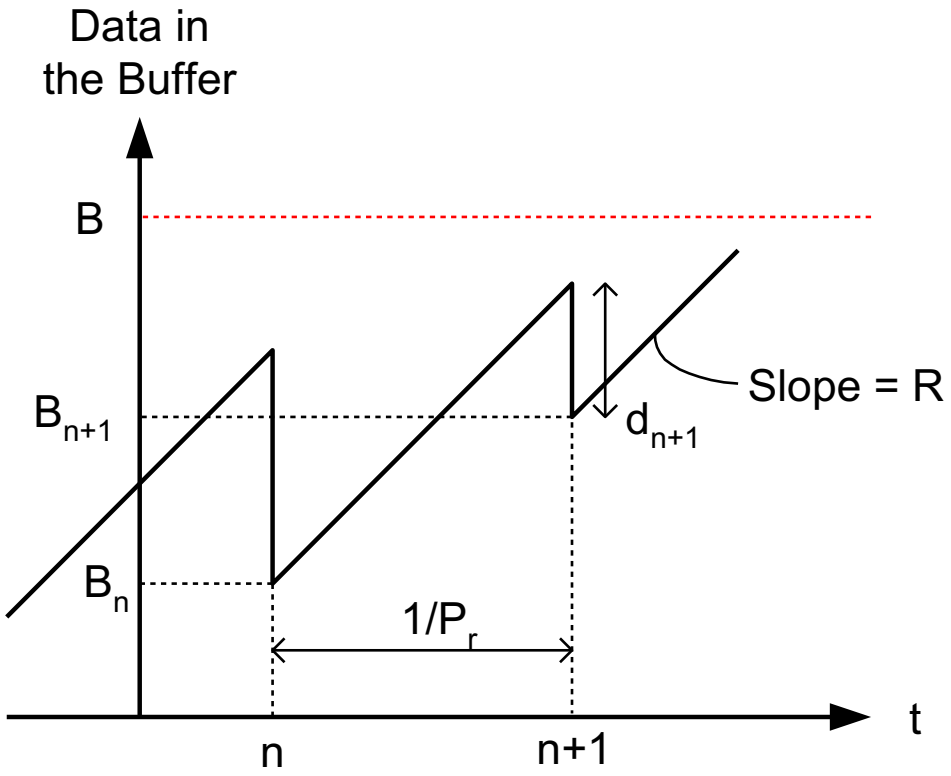
- Control the quantization step-size to
  - prevent the buffer from overflow/underflow
  - optimize the video quality

Video Buffer Control



- Decoder buffer is the major concern
- collapse when overflow/underflow since it decodes one frame every 1/30 second

Video Buffer Verifier



B: maximum buffer size
R: bit-rate
P_r: number of picture per second
d_n: decoded bits in time n

$$B - \frac{R}{P_r} > B_n + \frac{R}{P_r} - d_{n+1} \geq 0$$

Overflow Constraint

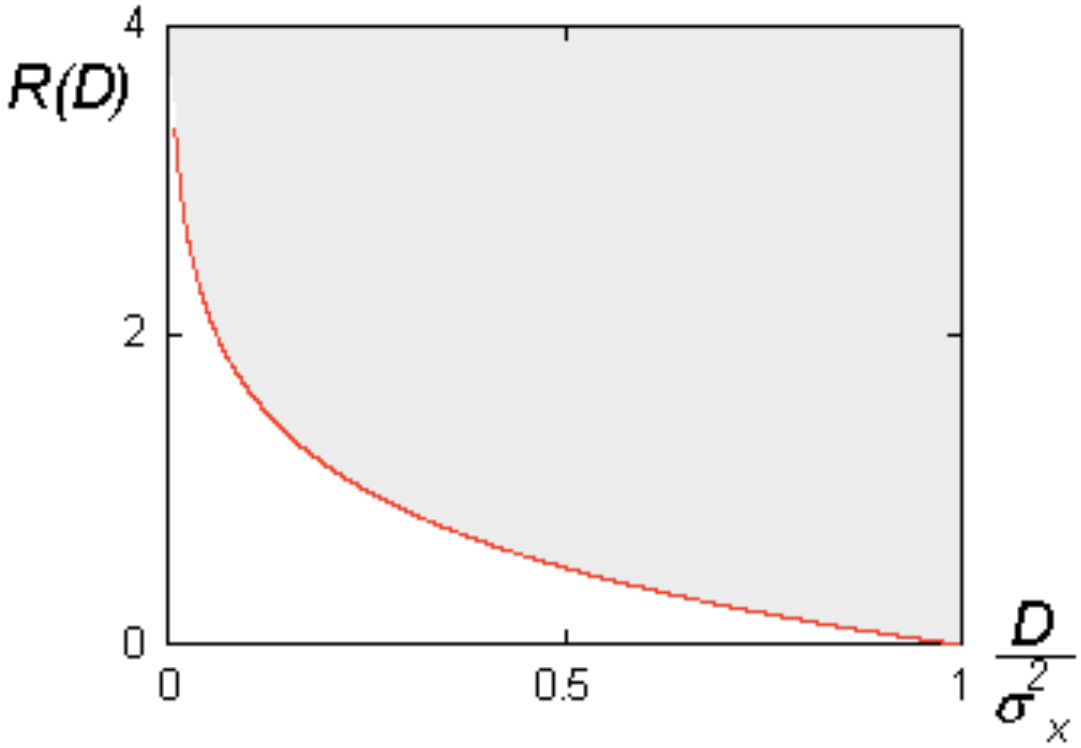
Underflow Constraint

Rate Control Schemes

- Rate control adapts quantizer to meet bitrate and delay constraints based on
  - Current frame complexity
  - Channel bit-rate
  - Buffer fullness
  - Mathematical modeling
- Examples of rate control schemes in video standards:
  - H.261: RM8
  - MPEG2: TM5
  - H.263+: TMN8
  - MPEG-4: VM5
  - H.264: JM8.1a

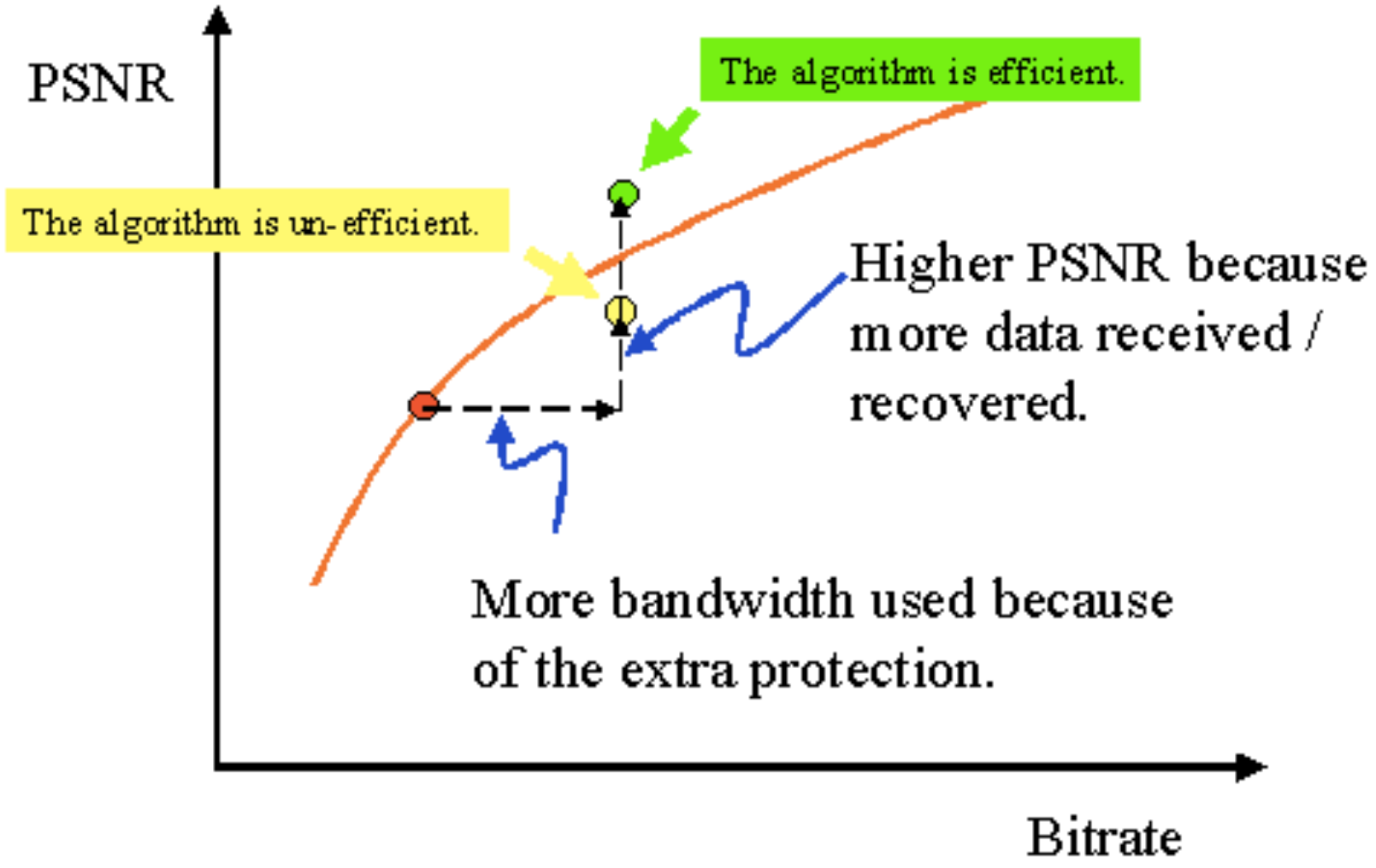
Rate Distortion Curve

- no compression system exists that performs outside the gray area
- Closer to the red curve (bound), better performance



https://en.wikipedia.org/wiki/Rate%E2%80%93distortion_theory

Rate PSNR Curve



Rate-Distortion Optimization

- Constrained Optimization Problem:

  choose $q=\{q_i\}$, to minimize $D(q)$, given constraint $R(q)=B$ $(i=1,2,\ldots,N)$

  - $D(q)$: the distortion
  - q: the quantization parameters
  - $R(q)$ is the rate (number of bits)
  - B is the number of bits for the frame
  - N is the number of macroblocks in the frame

Introduce Lagrange Multiplier λ and convert it to an unconstrained problem:

$$\text{Min}_q\ [J=D(q)+\lambda(R(q)-B)]$$