

# Computer Networks

Instructor: Niklas Carlsson

Email: [niklas.carlsson@liu.se](mailto:niklas.carlsson@liu.se)

Notes derived from "*Computer Networking: A Top Down Approach*", by Jim Kurose and Keith Ross, Addison-Wesley.

The slides are adapted and modified based on slides from the book's companion Web site, as well as modified slides by Anirban Mahanti and Carey Williamson.

# Scalable Content Delivery

## Motivation

- **Use of Internet for content delivery is massive** ... and becoming more so (e.g., majority of all IP traffic is video content)
- **Variety of approaches:** HTTP-based Adaptive Streaming (HAS), broadcast/multicast, batching, replication/caching (e.g. CDNs), P2P, peer-assisted, ...
- In these slides, we only provide a few high-level examples

# Why Study Multimedia Networking?

- ❑ Exciting and fun!
- ❑ Multimedia is everywhere
- ❑ Industry-relevant research topic
- ❑ Lots of open research problems

# Requirements and quality of service

## □ Quality of Service (QoS)

- Real-time requirements (e.g., latency, jitter)
- Loss/stall requirements (e.g., drop rates, late packets)
- Bandwidth requirements (e.g., throughput)
- Service availability

## □ Quality of Experience (QoE)

- Measure of the users quality of experience
- Multimedia: Most negatively effected by stalls

# Requirements and quality of service

- ❑ Quality of Service (QoS)
  - Real-time requirements (e.g., latency, jitter)
  - Loss/stall requirements (e.g., drop rates, late packets)
  - Bandwidth requirements (e.g., throughput)
  - Service availability
- ❑ Quality of Experience (QoE)
  - Measure of the users quality of experience
  - Multimedia: Most negatively effected by stalls



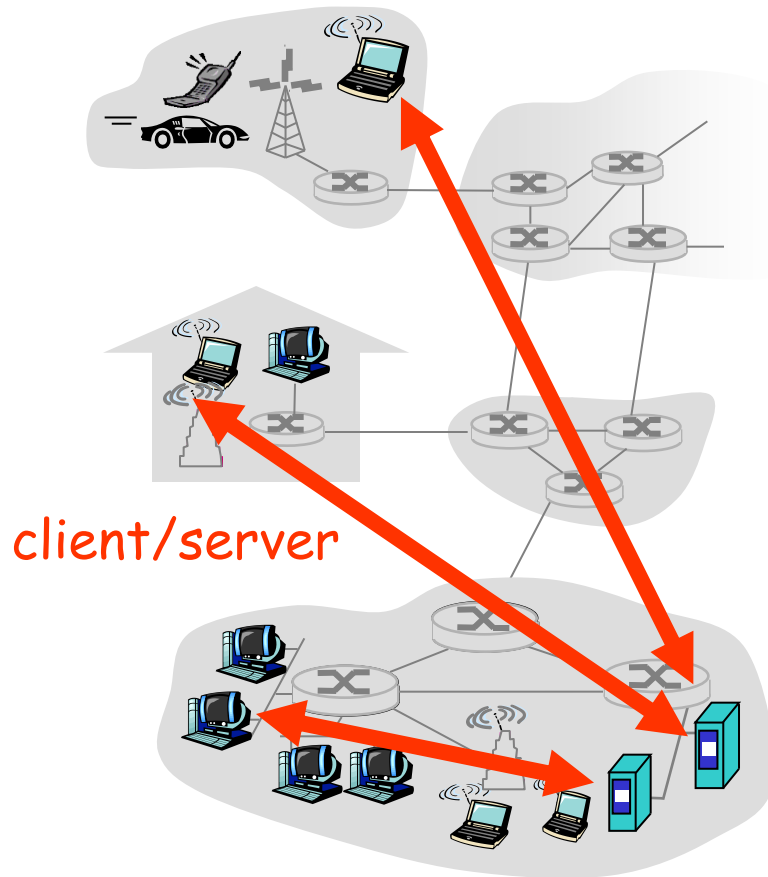


# Service models

- ❑ Client-server (one-to-one)
- ❑ Peer-to-peer (machines can act as both client and server)
- ❑ Multicast/broadcast (one-to-many and many-to-many)
  - Application layer, IP-based, and down at the MAC-layer
- ❑ ISP-based caching, CDNs, cloud, and other third-party solutions

# Client-server architecture

Client/server model has well-defined roles.



## server:

- always-on host
- permanent IP address
- server farms for scaling

## clients:

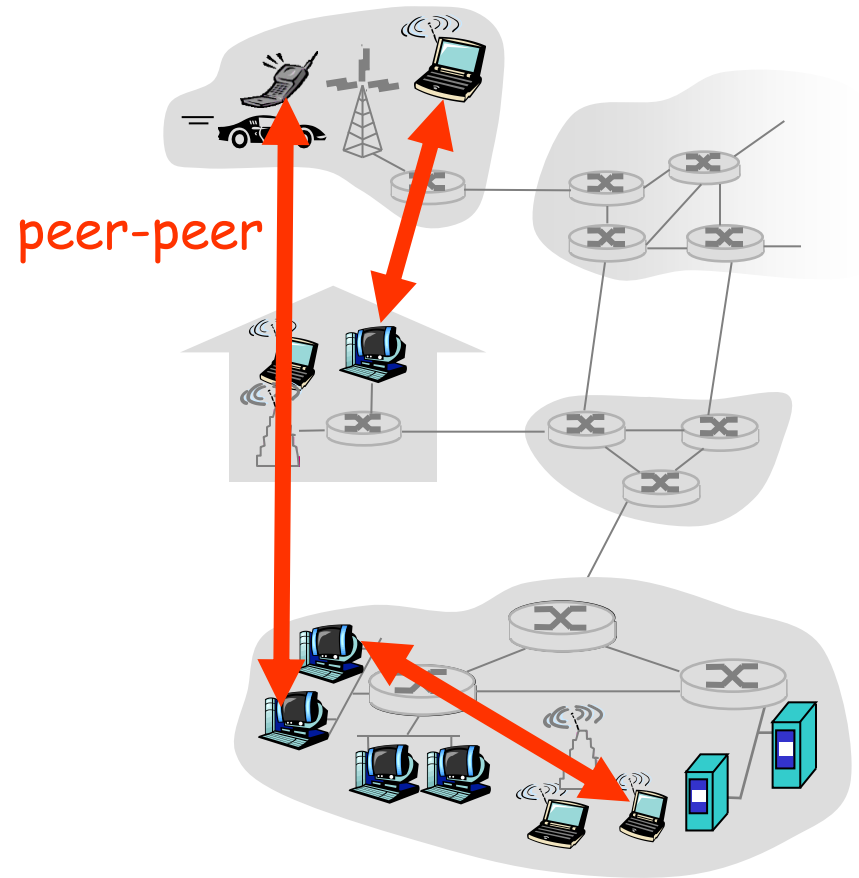
- communicate with server
- may be intermittently connected
- may have dynamic IP addresses
- do not communicate directly with each other



# Pure P2P architecture

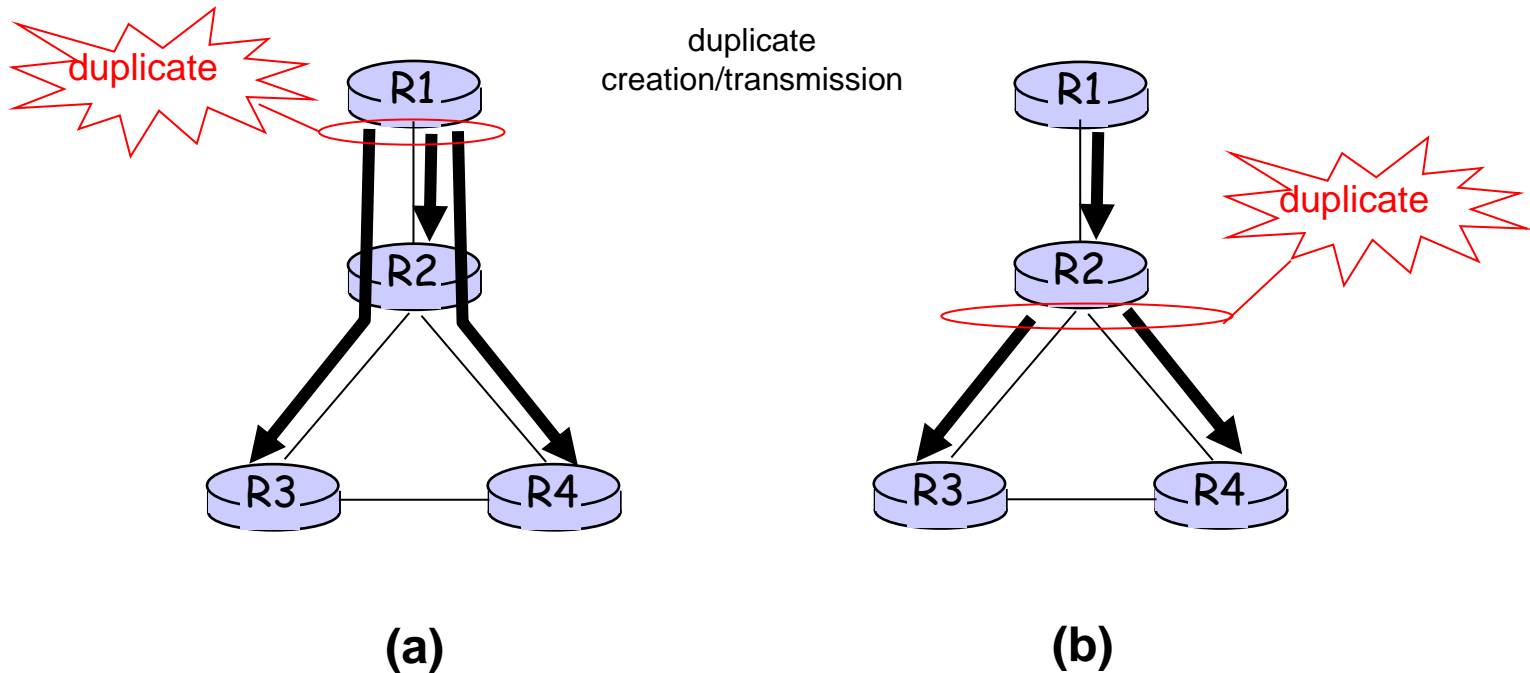
No fixed clients or servers: Each host can act as both client and server at any time

- ❑ *no* always-on server
- ❑ arbitrary end systems directly communicate
- ❑ peers are intermittently connected and change IP addresses



# Additional Multimedia Support

## Multicast/Broadcast



Source-duplication versus in-network duplication.  
(a) source duplication, (b) in-network duplication

Also, application-layer multicast ...

# Evolved Multimedia Broadcast/Multicast Service (eMBMS) in LTE-advanced

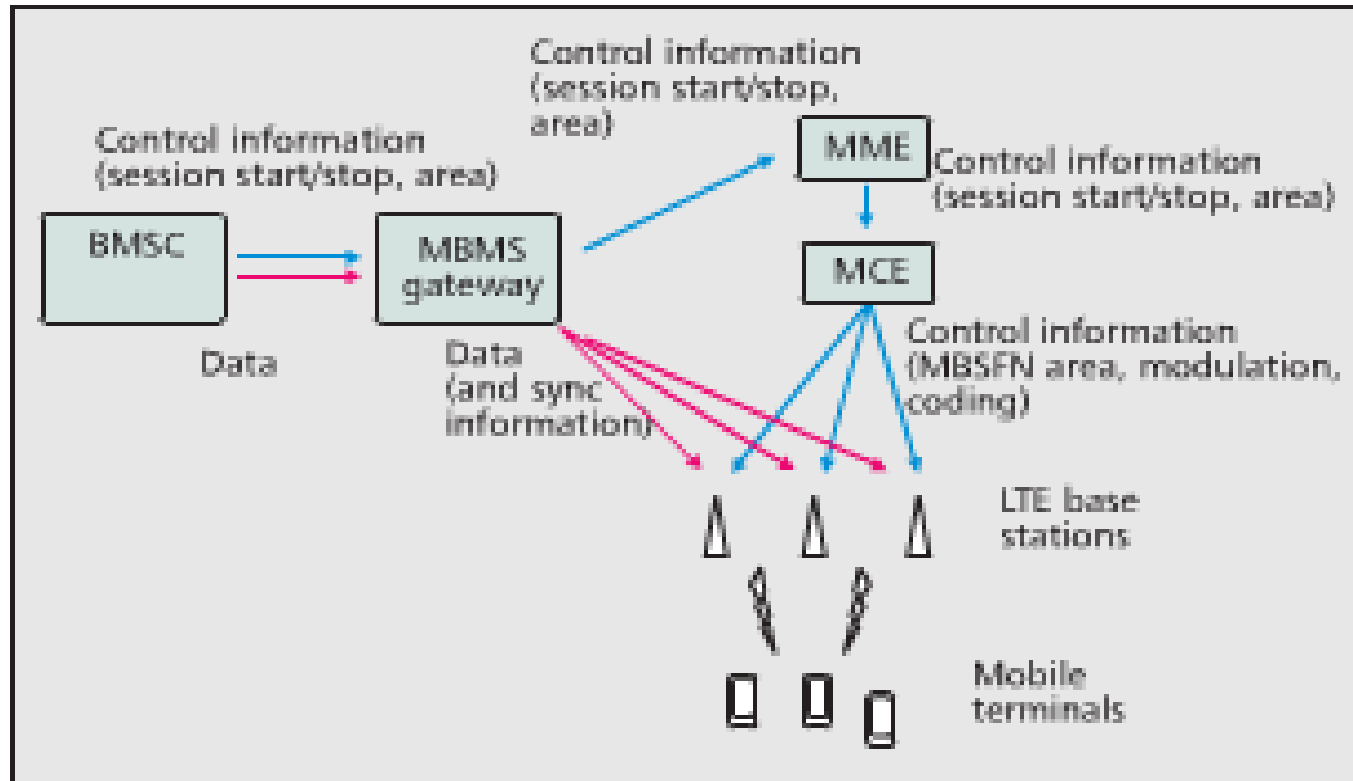


Figure 4. RAN architecture for SFN across LTE base stations.

## □ Separation of control plane and data plane

Image from: Lecompte and Gabin, Evolved Multimedia Broadcast/Multicast Service (eMBMS) in LTE-Advanced: Overview and Rel-11 Enhancements, IEEE Communications Magazine, Nov. 2012.

# Evolved Multimedia Broadcast/Multicast Service (eMBMS) in LTE-advanced

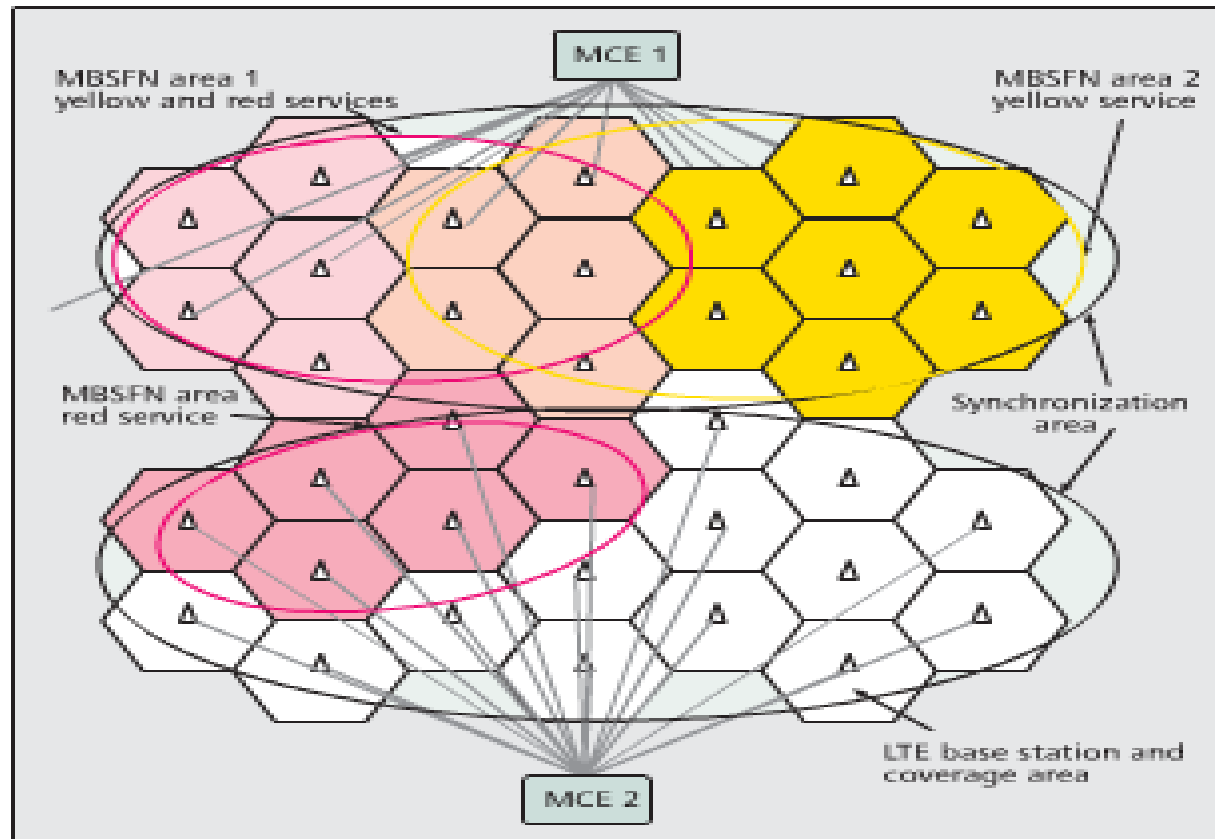


Figure 5. Example with two MBMS services with different services areas.

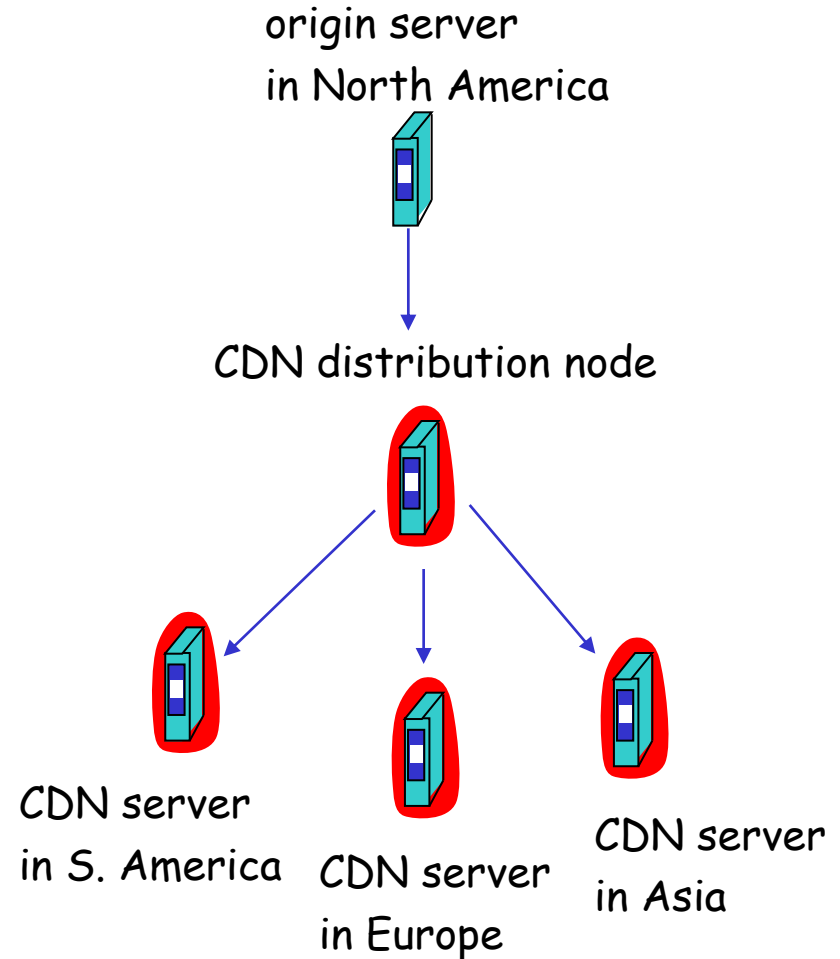
## □ MBMSFN and use of services areas

Image from: Lecompte and Gabin, Evolved Multimedia Broadcast/Multicast Service (eMBMS) in LTE-Advanced: Overview and Rel-11 Enhancements, IEEE Communications Magazine, Nov. 2012.

# Content distribution networks (CDNs)

## Content replication

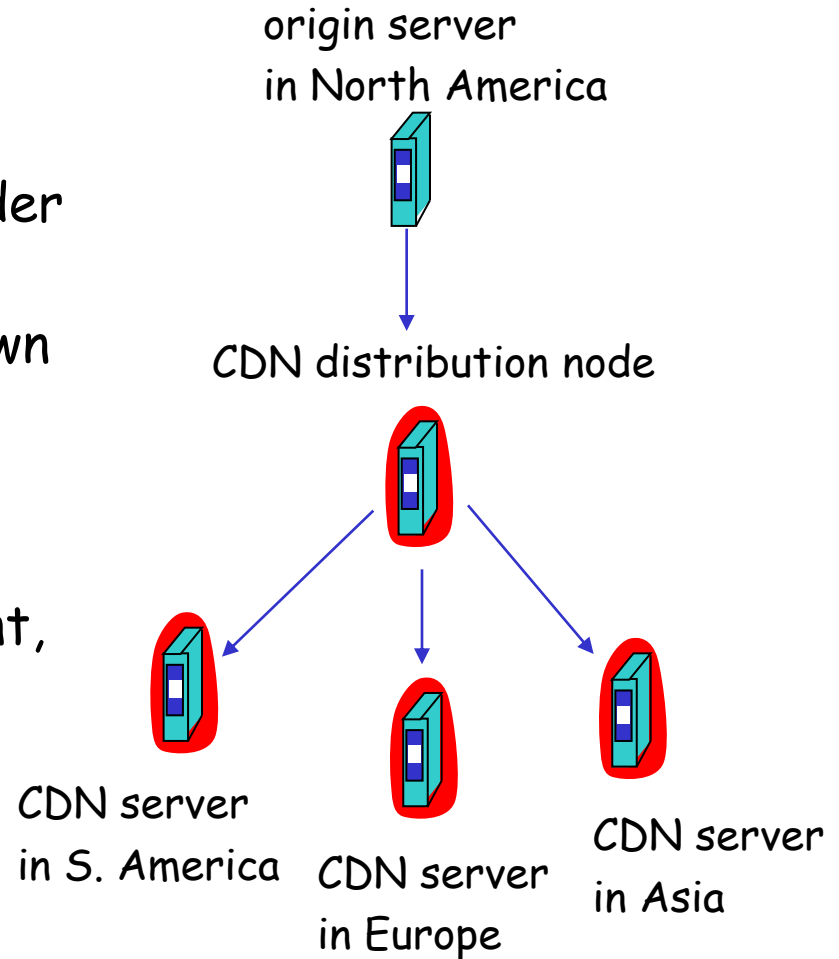
- ❑ replicate content at hundreds of servers throughout Internet (often in edge/access network)
- ❑ content "close" to user reduce impairments (loss, delay) of sending content over long paths



# Content distribution networks (CDNs)

## Content replication

- ❑ CDN (e.g., Akamai, Limewire) customer is the content provider (e.g., CNN)
- ❑ Other companies build their own CDN (e.g., Google)
- ❑ CDN replicates customers' content in CDN servers.
- ❑ When provider updates content, CDN updates servers



# DNS redirect example





# Multimedia Networking

## Principles

- ❑ Classify multimedia applications
- ❑ Identify the network services the apps need
- ❑ Making the best of "best effort" service
- ❑ Mechanisms for providing QoS

## Protocols and Architectures

- ❑ Specific protocols for best effort delivery
- ❑ Architectures for QoS

# Multimedia Networking

## Principles

- ❑ Classify multimedia applications
- ❑ Identify the network services the apps need
- ❑ Making the best of "best effort" service
- ❑ Mechanisms for providing QoS

## Protocols and Architectures

- ❑ Specific protocols for best effort delivery
- ❑ Architectures for QoS



# Multimedia Networking Applications

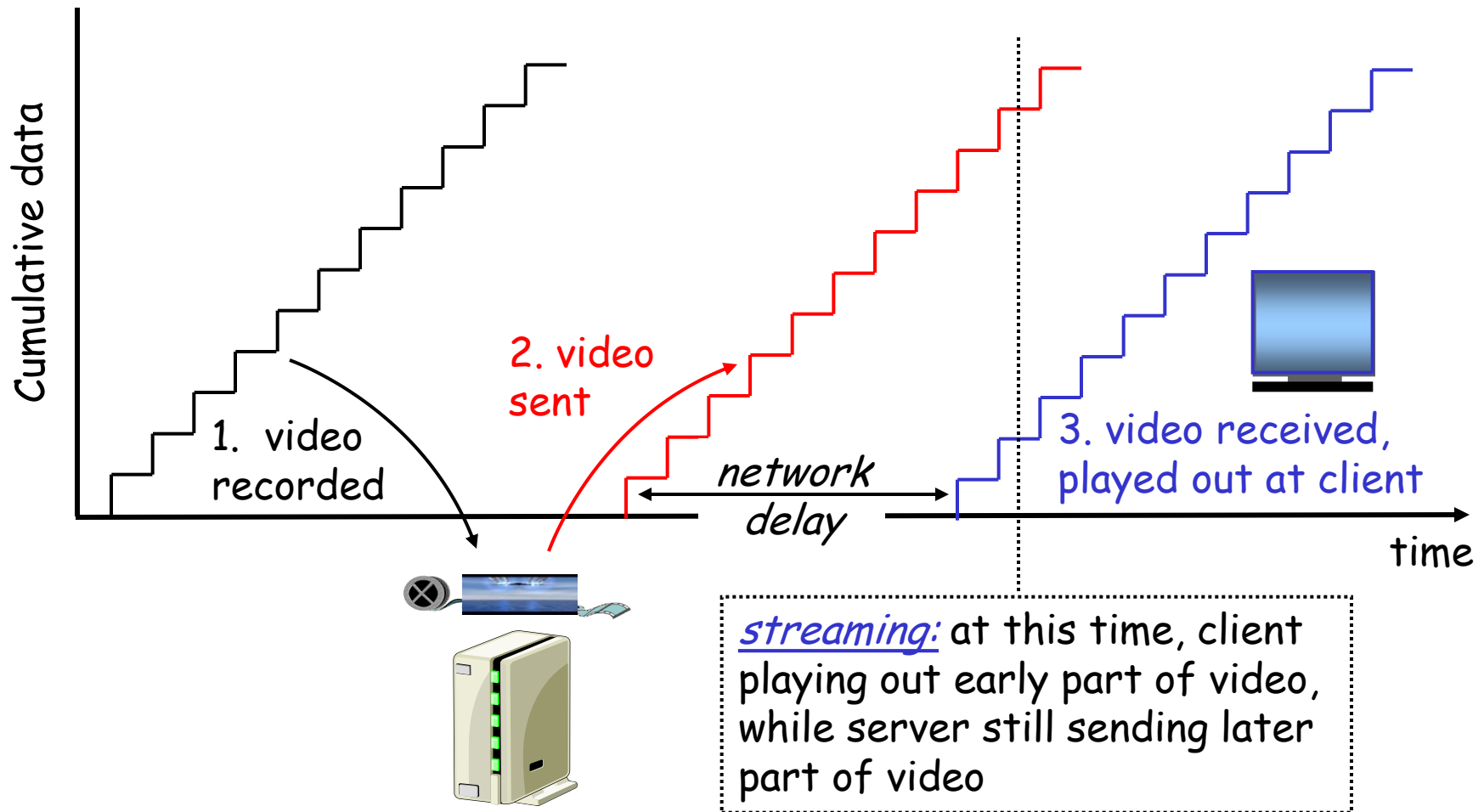
Classes of MM applications:

# Multimedia Networking Applications

## Classes of MM applications:

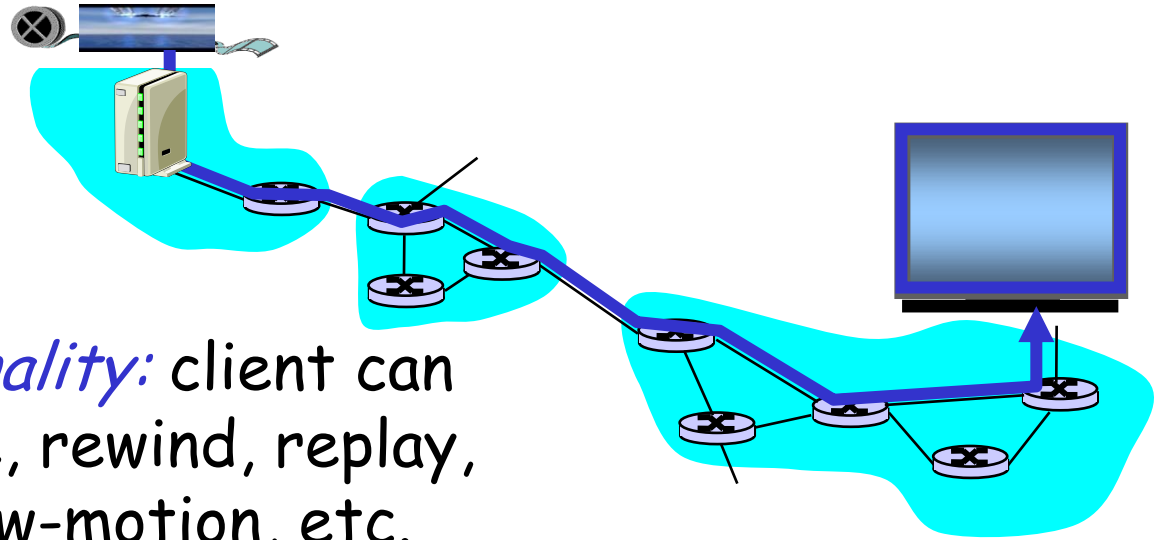
- 1) Streaming stored audio and video
- 2) Streaming live audio and video
- 3) Real-time interactive audio and video

# Streaming Stored Multimedia (1/2)

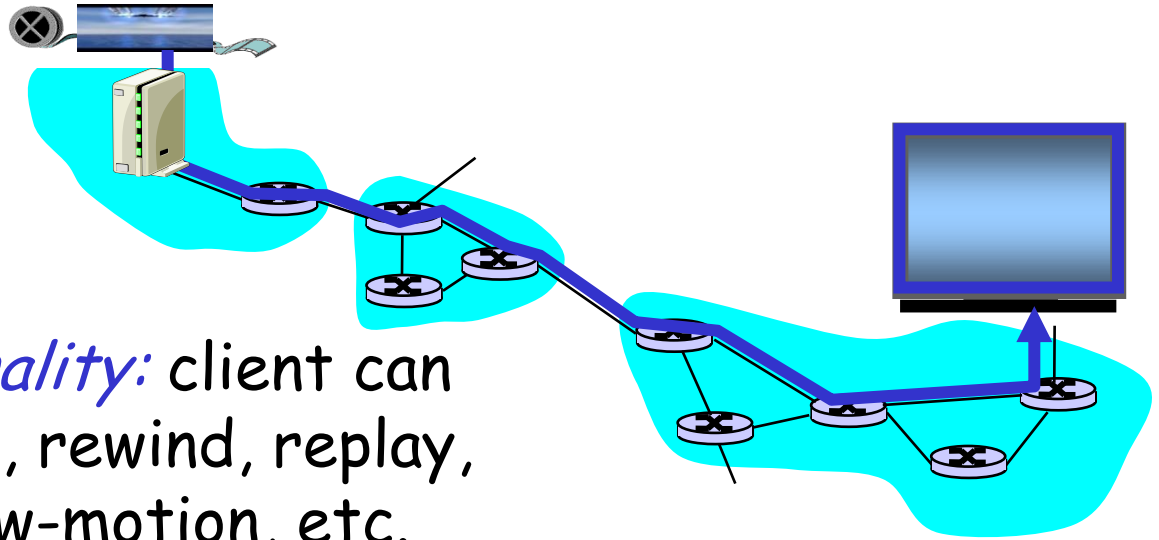


# Streaming Stored Multimedia (2/2)

- *VCR-like functionality*: client can start, stop, pause, rewind, replay, fast-forward, slow-motion, etc.



# Streaming Stored Multimedia (2/2)



- ❑ *VCR-like functionality*: client can start, stop, pause, rewind, replay, fast-forward, slow-motion, etc.
  - 10 sec initial delay OK
  - 1-2 sec until command effect OK
  - need a separate control protocol?
- ❑ timing constraint for data that is yet to be transmitted: must arrive in time for playback



# Streaming Live Multimedia

## Examples:

- ❑ Internet radio talk show
- ❑ Live sporting event

# 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

## Interactivity

- ❑ fast-forward is not possible
- ❑ rewind and pause possible!

# Interactive, Real-time Multimedia

- **applications:** IP telephony, video conference, distributed interactive worlds

# Interactive, Real-time Multimedia

- ❑ **applications:** IP telephony, video conference, distributed interactive worlds
- ❑ **end-end delay requirements:**
  - audio: < 150 msec good, < 400 msec OK
    - includes application-layer (packetization) and network delays
    - higher delays noticeable, impair interactivity
- ❑ **session initialization**
  - callee must advertise its IP address, port number, frame rate, encoding algorithms

And the future has more ...





# Multimedia Networking Applications

Fundamental characteristics:

# Multimedia Networking Applications

## Fundamental characteristics:

- Inherent frame rate



# Multimedia Networking Applications

## Fundamental characteristics:

- ❑ Inherent frame rate
- ❑ Typically **delay-sensitive**
  - end-to-end delay
  - delay jitter

# Multimedia Networking Applications

## Fundamental characteristics:

- ❑ Inherent frame rate
- ❑ Typically **delay-sensitive**
  - end-to-end delay
  - delay jitter
- ❑ But **loss-tolerant**: infrequent losses cause minor transient glitches
- ❑ Unlike data apps, which are often delay-tolerant but loss-sensitive.

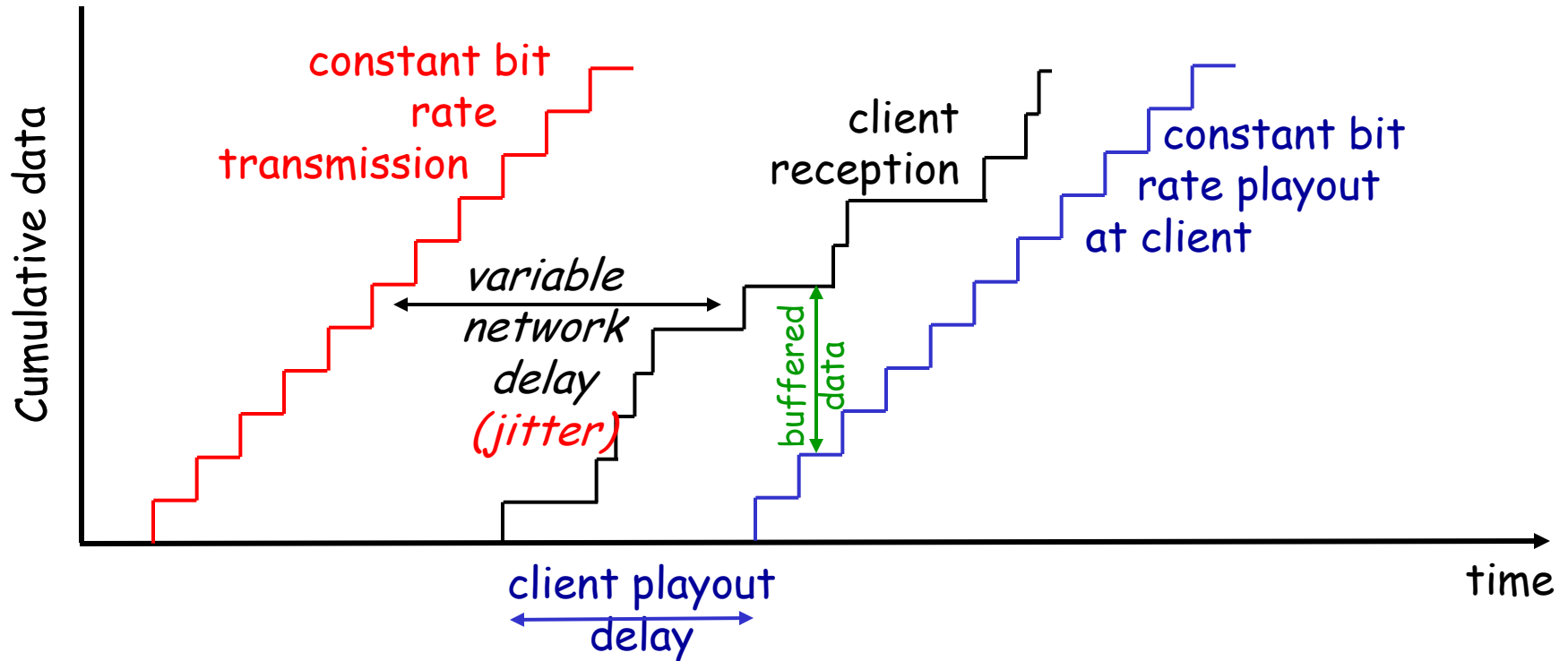
# Multimedia Networking Applications

## Fundamental characteristics:

- ❑ Inherent frame rate
- ❑ Typically **delay-sensitive**
  - end-to-end delay
  - delay jitter
- ❑ But **loss-tolerant**: infrequent losses cause minor transient glitches
- ❑ Unlike data apps, which are often delay-tolerant but loss-sensitive.

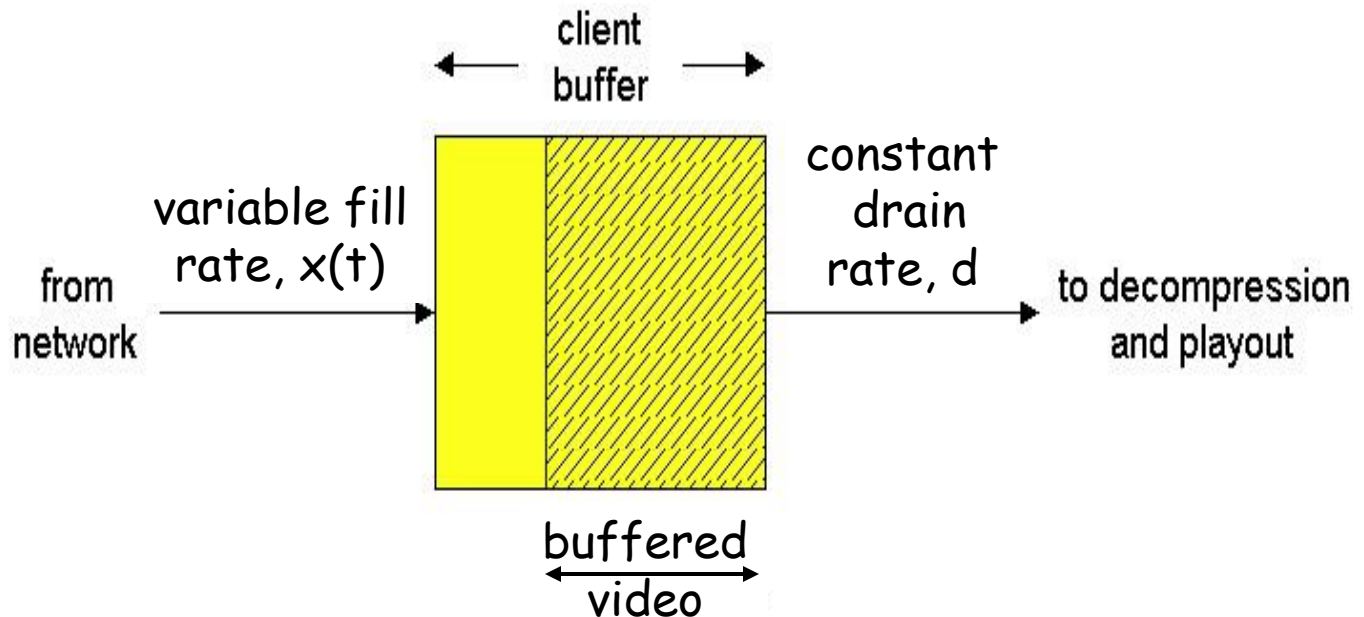
**Jitter** is the variability of packet delays within the same packet stream

# Delay Jitter



- consider end-to-end delays of two consecutive packets: difference can be more or less than 20 msec (transmission time difference)

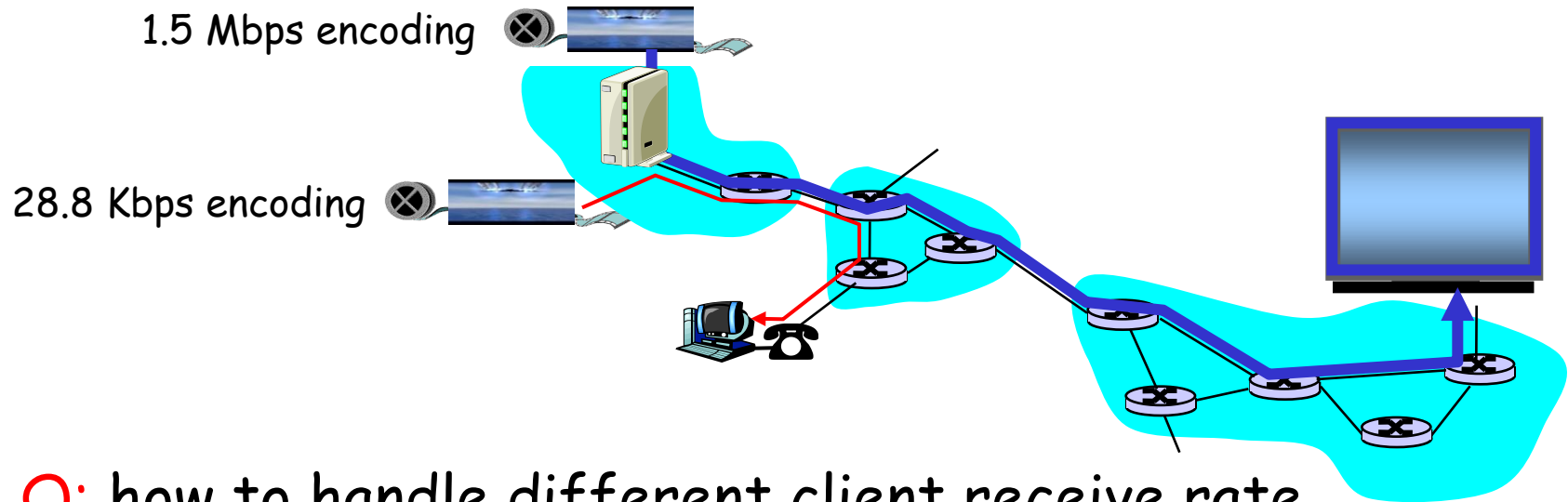
# Streaming Multimedia: Client Buffering



- Client-side buffering, playout delay compensate for network-added delay, delay jitter



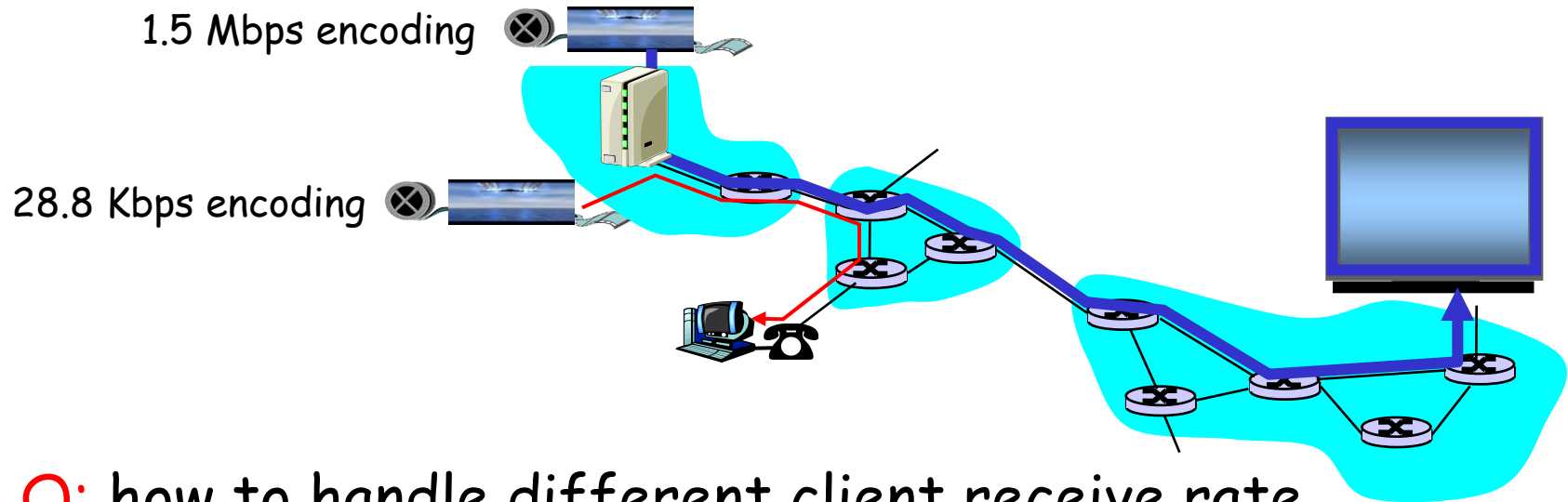
# Streaming Multimedia: client rate(s)



Q: how to handle different client receive rate capabilities?

- 28.8 Kbps dialup
- 100 Mbps Ethernet

# Streaming Multimedia: client rate(s)



Q: how to handle different client receive rate capabilities?

- 28.8 Kbps dialup
- 100 Mbps Ethernet

A1: server stores, transmits multiple copies of video, encoded at different rates

A2: layered and/or dynamically rate-based encoding





# Consider first ...

## Streaming Stored Multimedia

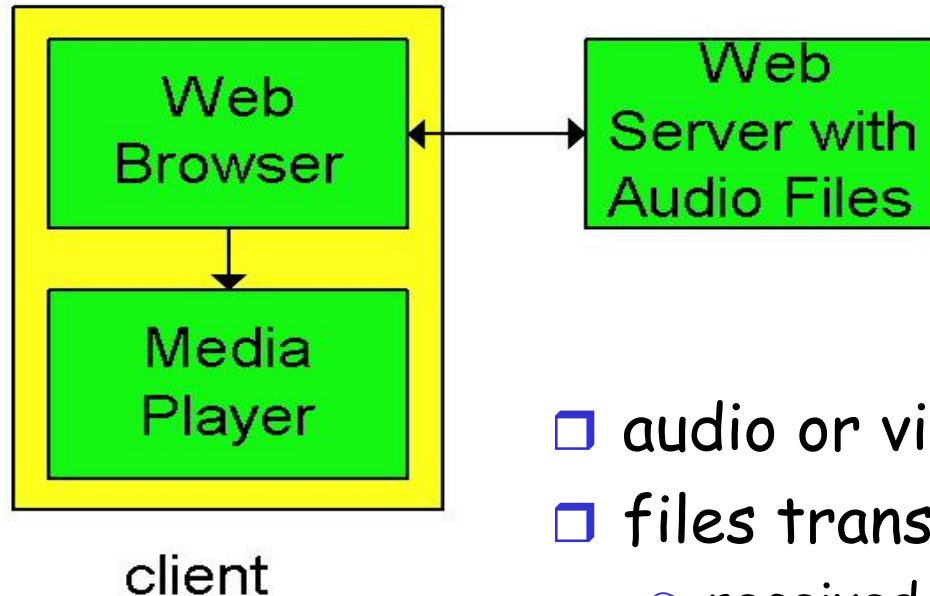
application-level  
streaming techniques for  
making the best out of  
best effort service:

- client-side buffering
- use of UDP versus TCP
- multiple encodings of multimedia

### Media Player

- ❑ jitter removal
- ❑ decompression
- ❑ error concealment
- ❑ graphical user interface  
w/ controls for  
interactivity

# Internet multimedia: simplest approach

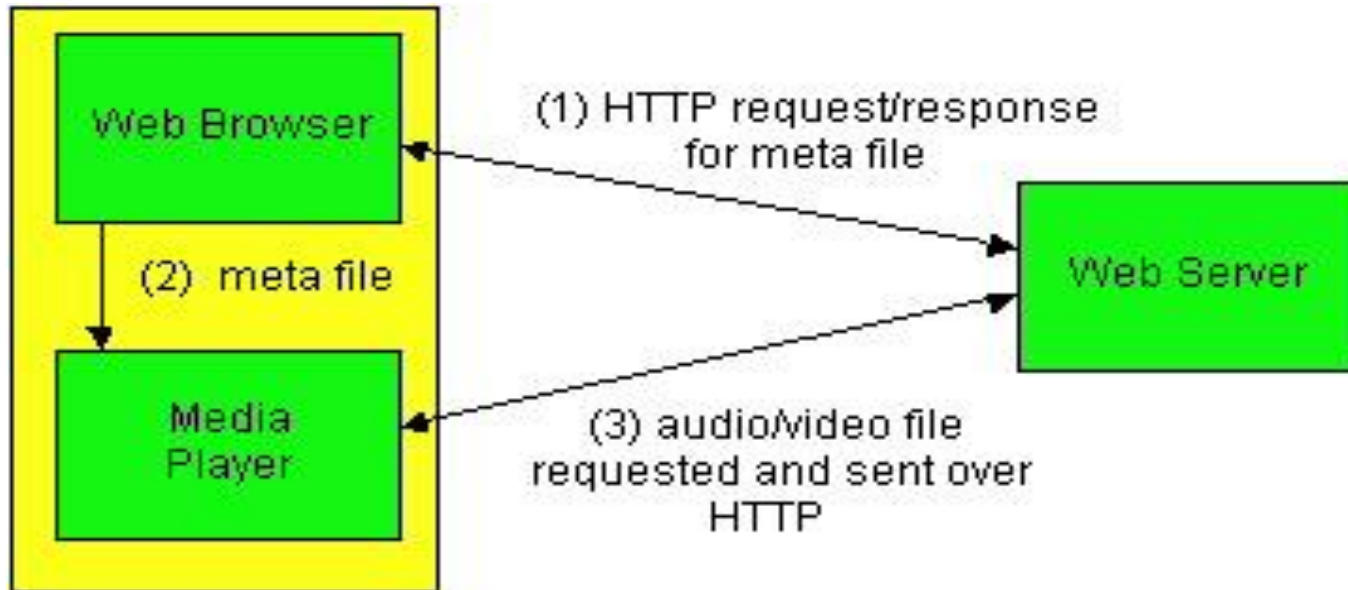


- ❑ audio or video stored in file
- ❑ files transferred as HTTP object
  - received in entirety at client
  - then passed to player

audio, video is downloaded, not streamed:

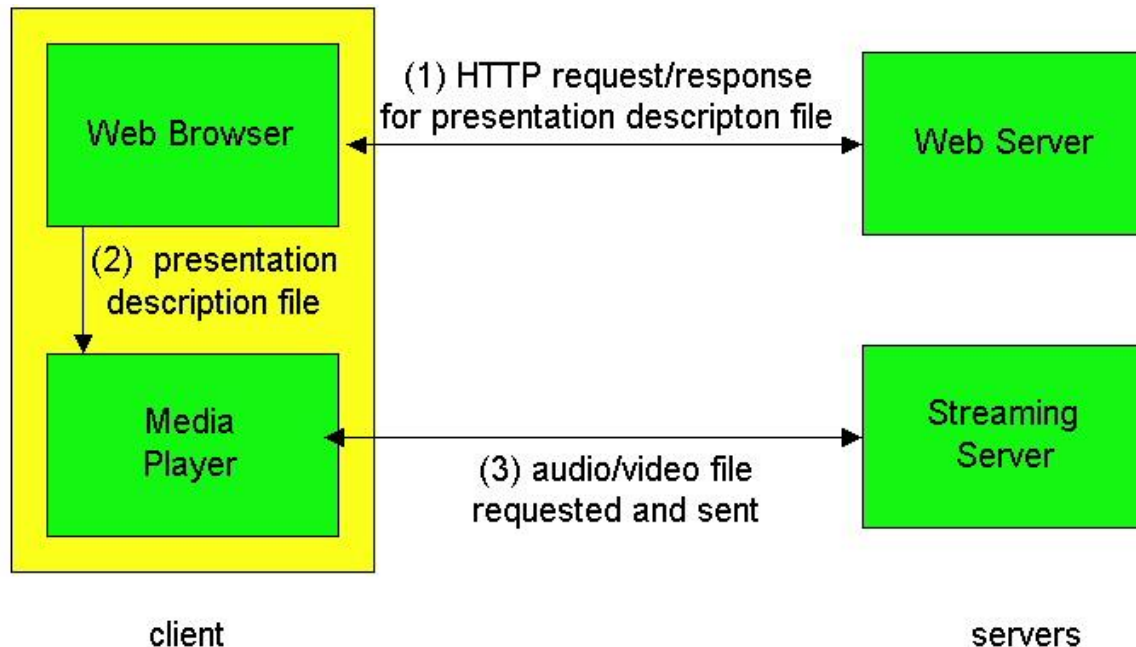
- ❑ long delays until playout, since no pipelining!

# Progressive Download



- ❑ browser retrieves **metafile** using HTTP GET
- ❑ browser launches player, passing metafile to it
- ❑ media player contacts server directly
- ❑ server **downloads** audio/video to player

# Streaming from a Streaming Server



- ❑ This architecture allows for non-HTTP protocol between server and media player
- ❑ Can also use UDP instead of TCP.



# Streaming Multimedia: UDP or TCP?

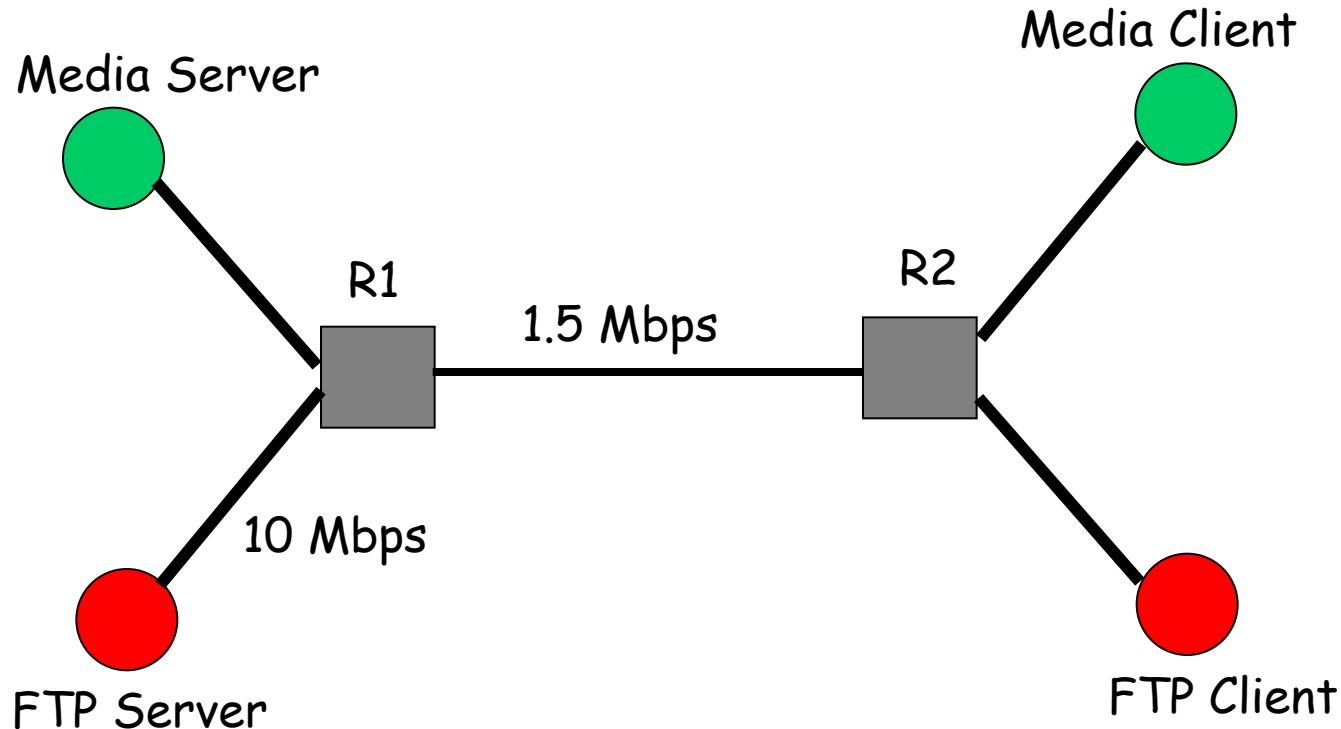
## UDP

- ❑ server sends at rate appropriate for client (oblivious to network congestion !)
  - often send rate = encoding rate = constant rate
  - then, fill rate = constant rate - packet loss
- ❑ short playout delay (2-5 seconds) to compensate for network delay jitter
- ❑ error recover: time permitting

## TCP

- ❑ send at maximum possible rate under TCP
- ❑ fill rate fluctuates due to TCP congestion control
- ❑ larger playout delay: smooth TCP delivery rate
- ❑ HTTP/TCP passes more easily through firewalls

# Fairness of UDP Streams (1/2)

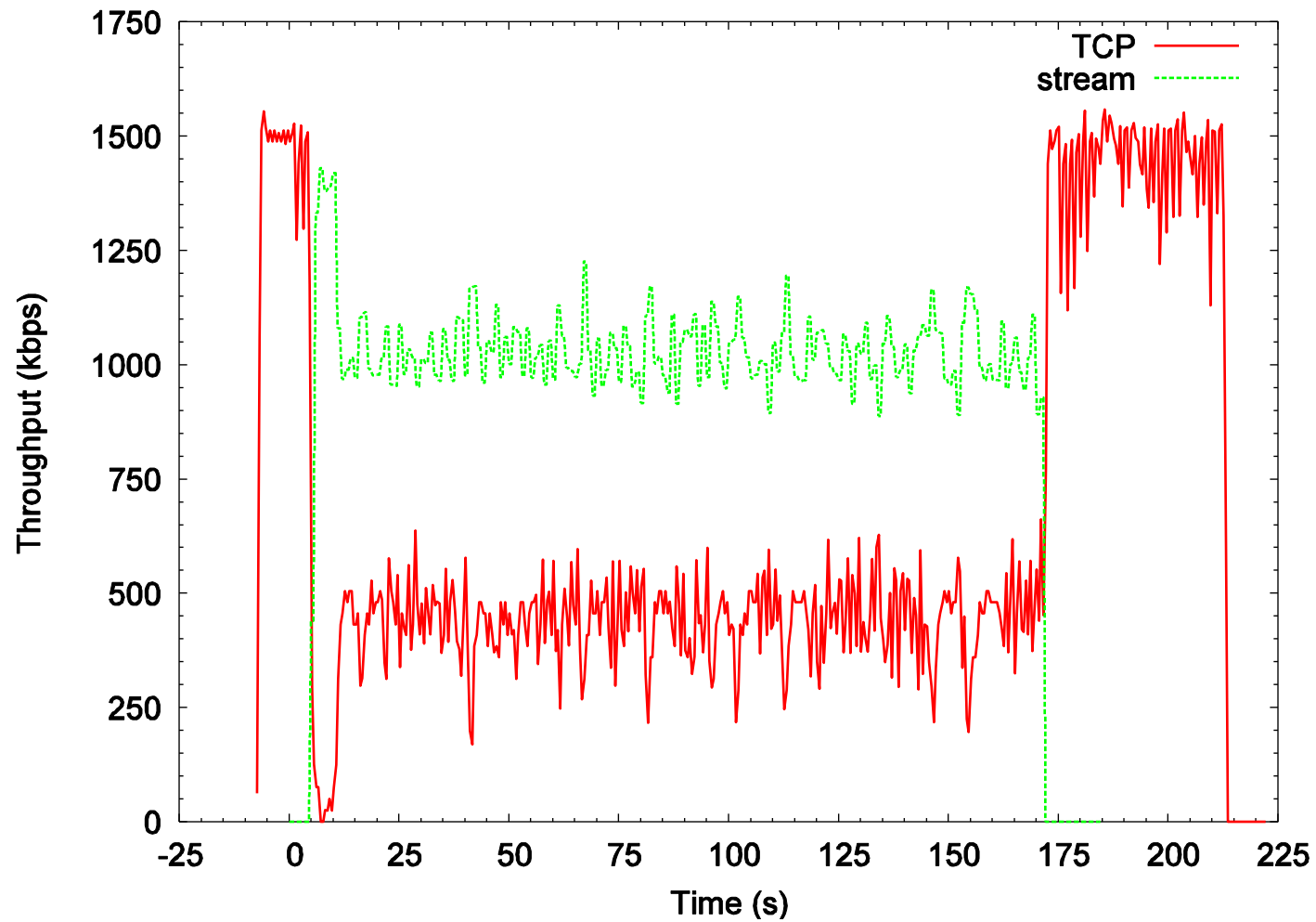


- R1-R2 is the bottleneck link
- Streaming uses UDP at the transport layer; requested media encoded at 1 Mbps
- What fraction of the bottleneck is available to FTP?

Credit: MSc thesis work by [Sean Boyden](#) (2006)

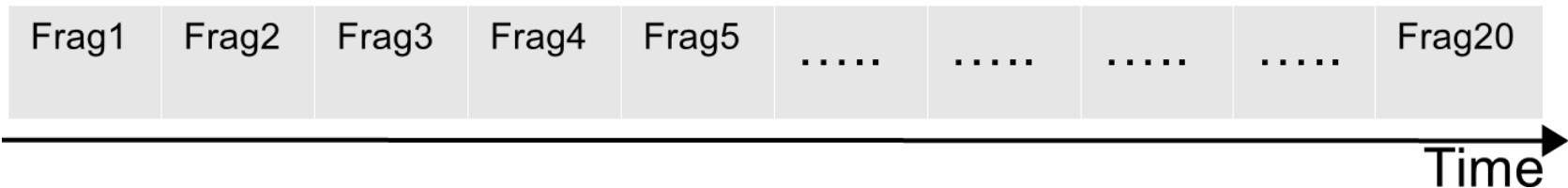


# Fairness of RealVideo Streams (2/2)





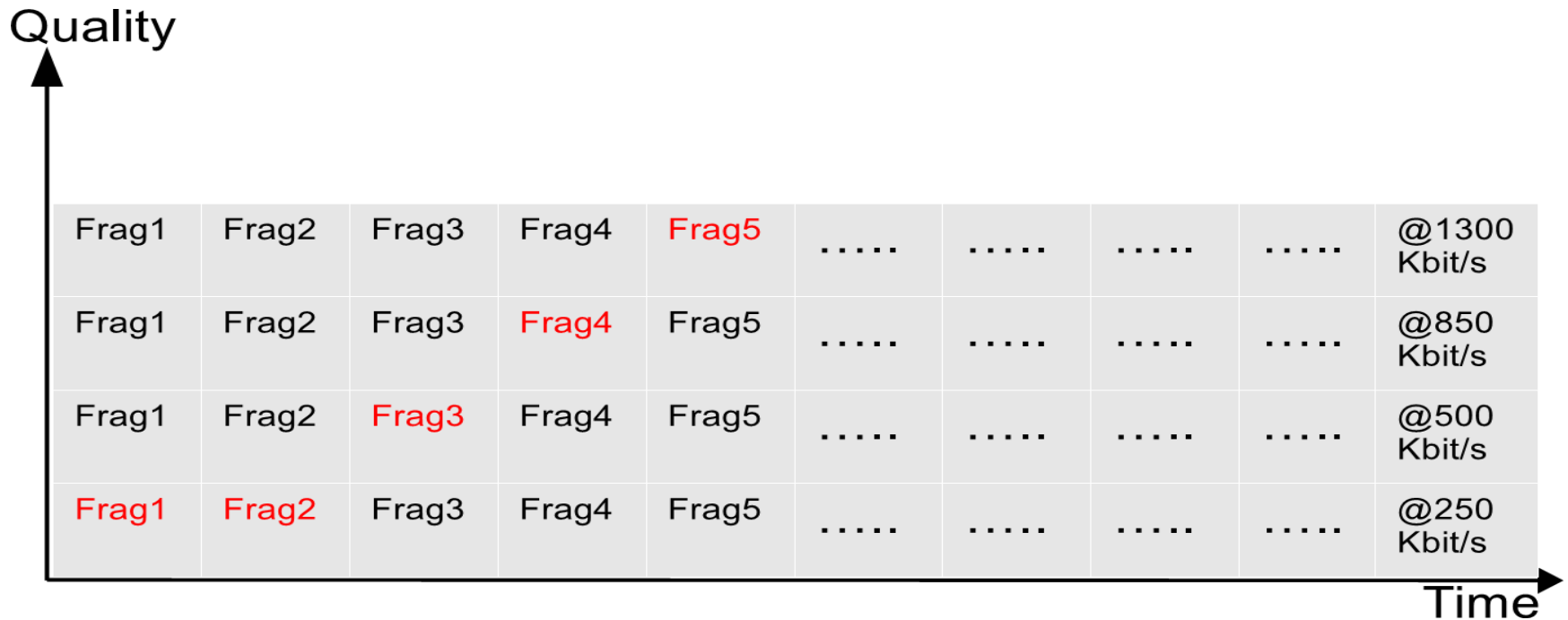
# HTTP-based streaming



## □ HTTP-based streaming

- Allows easy caching, NAT/firewall traversal, etc.
- Use of TCP provides natural bandwidth adaptation
- Split into fragments, download sequentially
- Some support for interactive VoD

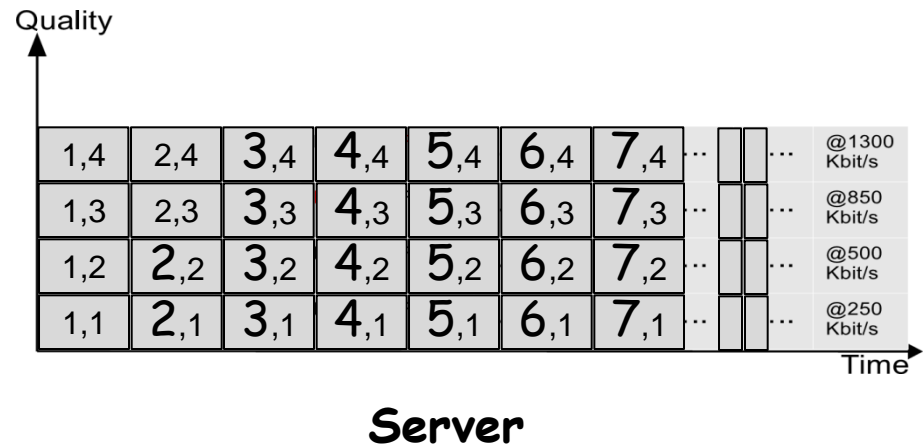
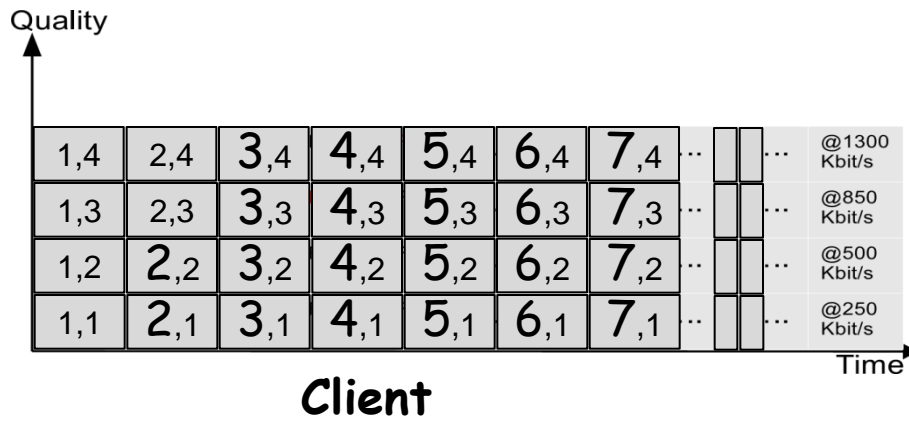
# HTTP-based adaptive streaming (HAS)



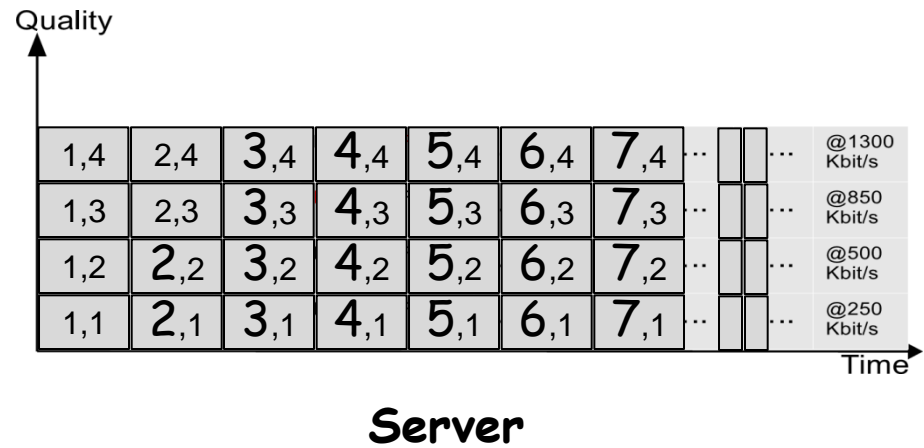
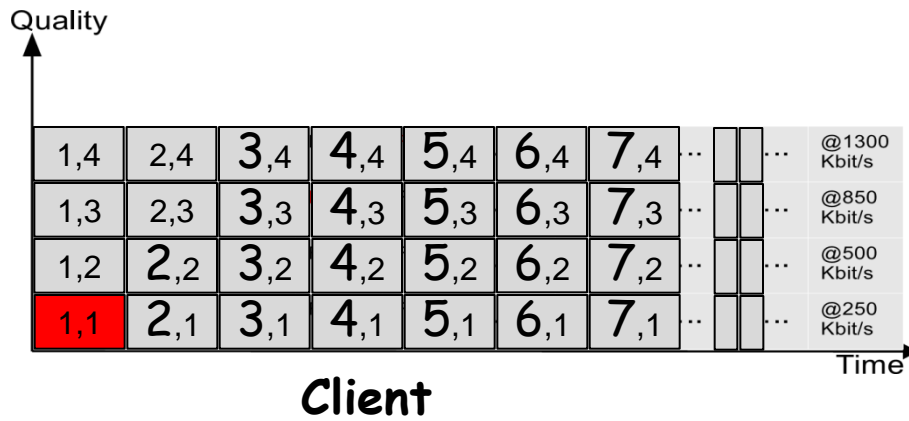
## □ HTTP-based adaptive streaming

- Multiple encodings of each fragment (defined in manifest file)
- Clients adapt quality encoding based on (buffer and network) conditions

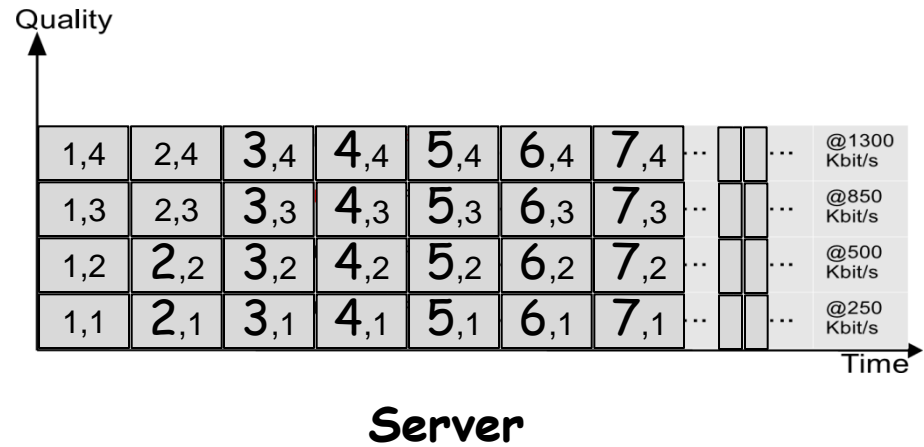
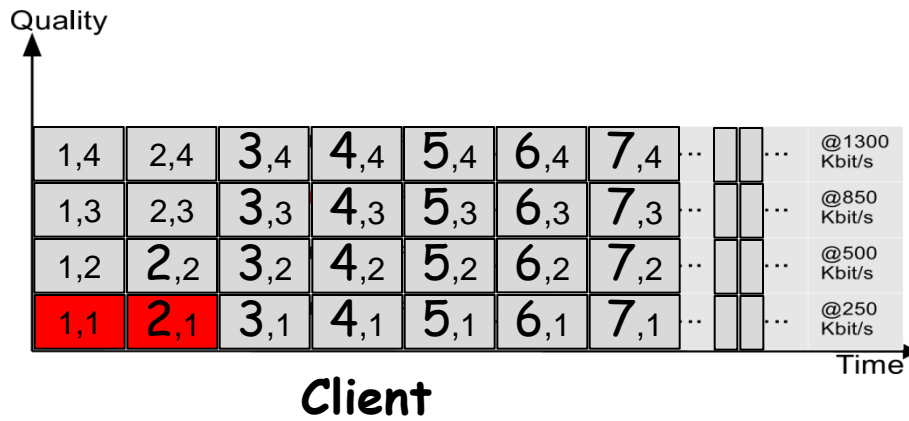
# Example



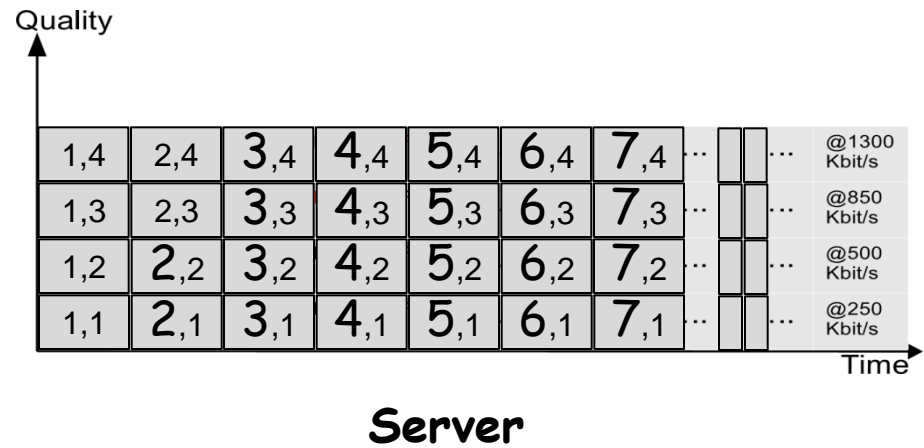
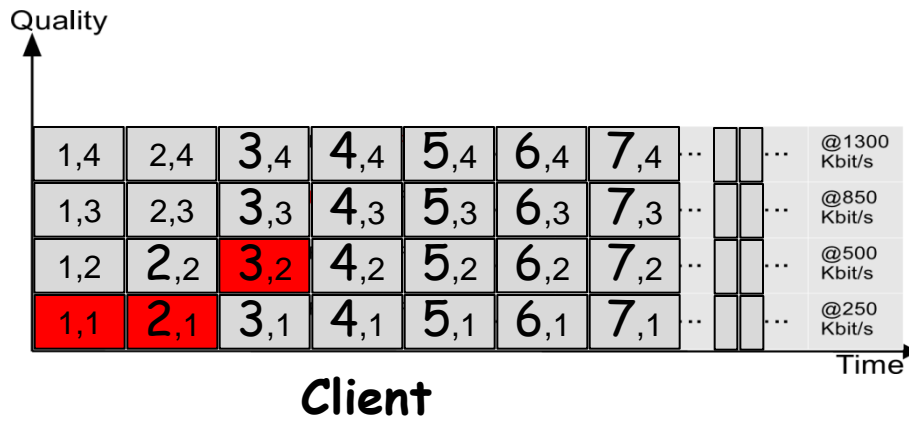
# Example



# Example

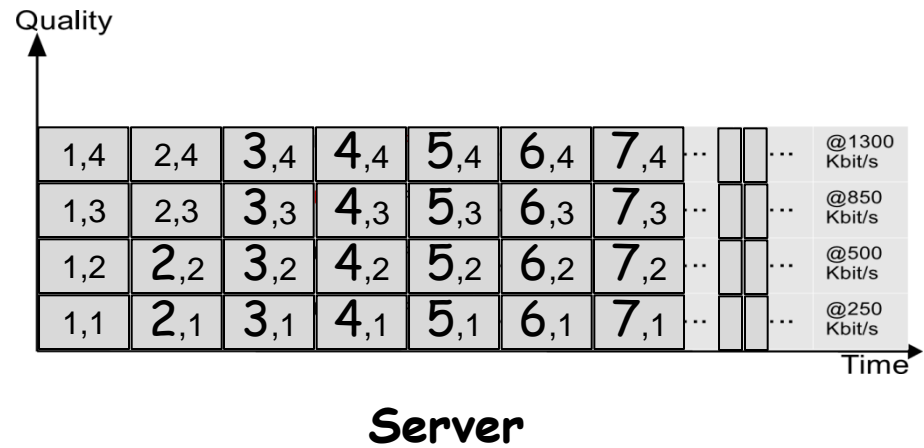
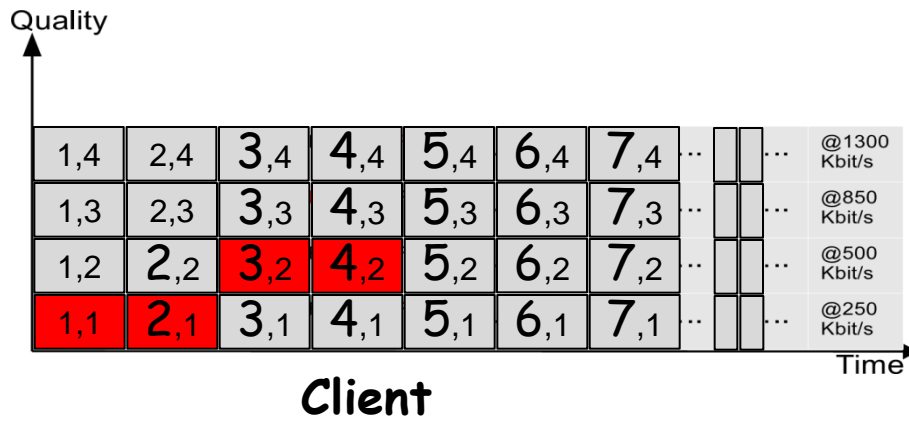


# Example

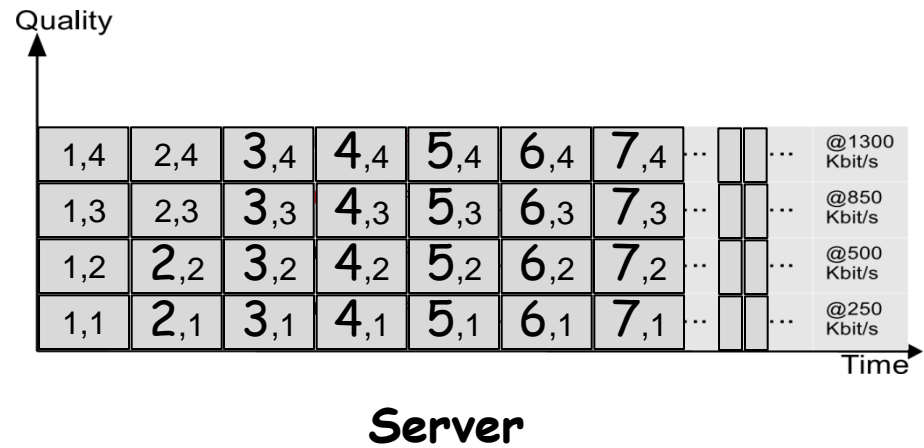
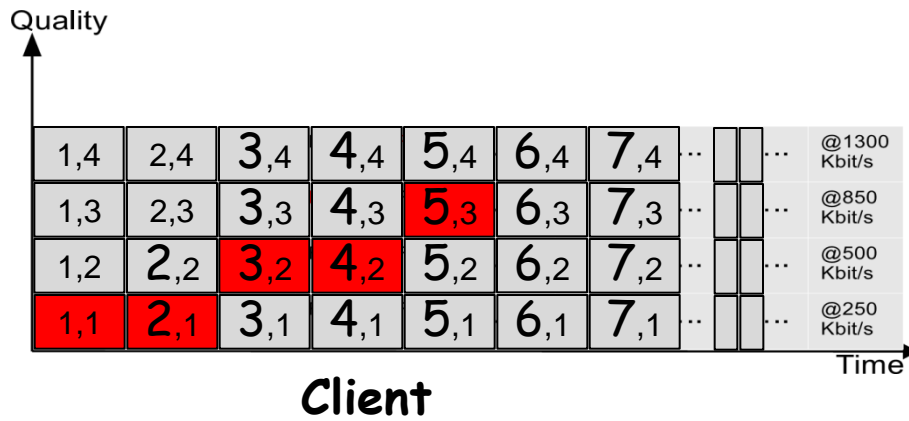




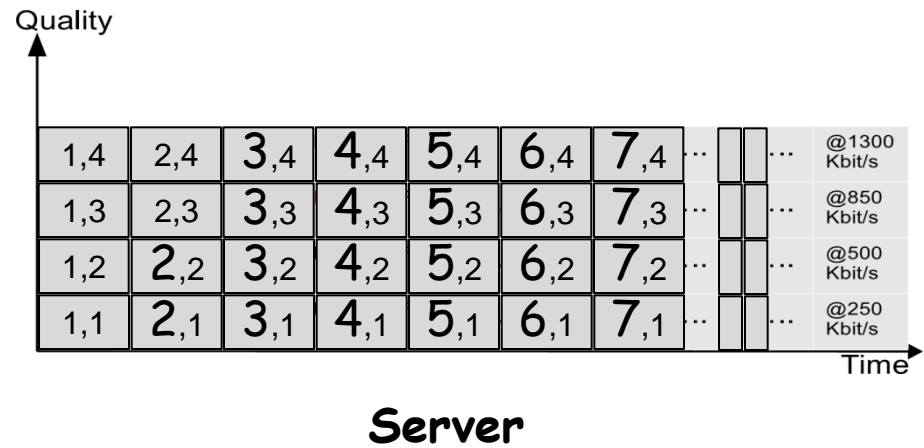
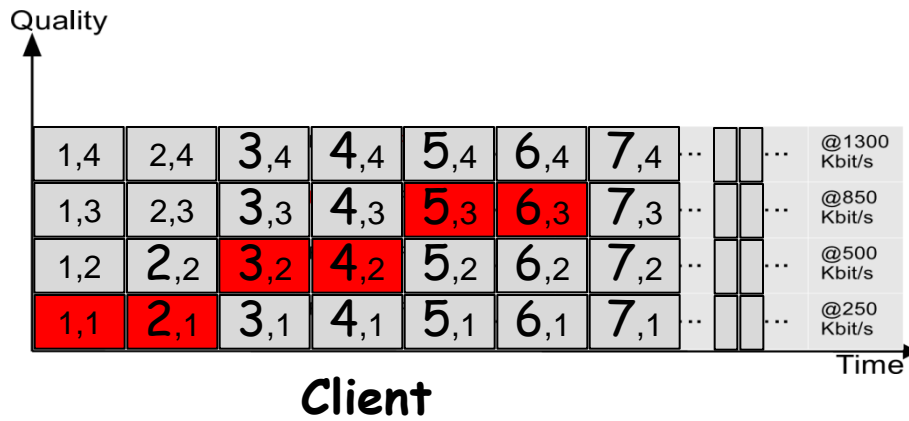
# Example



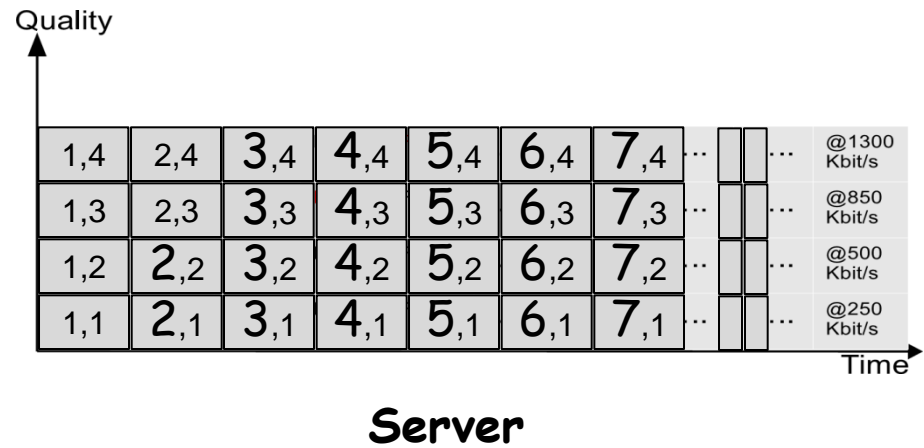
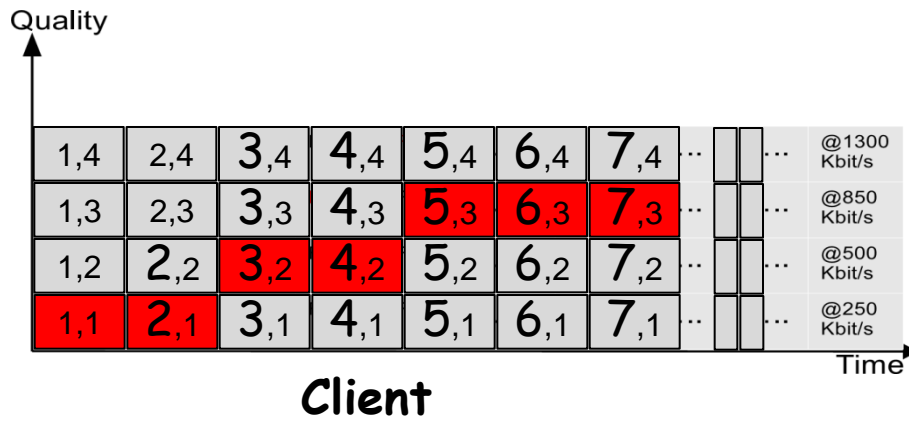
# Example



# Example



# Example



# Chunk-based streaming

- ❑ Chunks begin with keyframe so independent of other chunks
- ❑ Playing chunks in sequence gives seamless video
- ❑ Hybrid of streaming and progressive download:
  - Stream-like: sequence of small chunks requested as needed
  - Progressive download-like: HTTP transfer mechanism, stateless servers

# HTTP-based Adaptive Streaming (HAS)





- ❑ Other terms for similar concepts: Adaptive Streaming, Smooth Streaming, HTTP Chunking, DASH
- ❑ Actually a series of small progressive downloads of chunks (or range requests)
- ❑ No standard protocol ...

# HTTP-based Adaptive Streaming (HAS)

- ❑ Other terms for similar concepts: Adaptive Streaming, Smooth Streaming, HTTP Chunking, DASH
- ❑ Actually a series of small progressive downloads of chunks (or range requests)
- ❑ No standard protocol ...
  - Apple HLS: HTTP Live Streaming
  - Microsoft IIS Smooth Streaming: part of Silverlight
  - Adobe: Flash Dynamic Streaming
  - DASH: Dynamic Adaptive Streaming over HTTP

# Example players



	Player	Container	Type	Open Source
 Microsoft Silverlight™	Microsoft Smooth Streaming	Silverlight	Chunk	✗
	Netflix player	Silverlight	Range	✗
	Apple HLS	QuickTime	Chunk	✗
	Adobe HDS	Flash	Chunk	✓

.. and now YouTube (Google) is also using HAS!







# Example: HAS and proxy



## **Clients' want**

- ☐ High playback quality
- ☐ Small stall times
- ☐ Few buffer interruptions
- ☐ Few quality switches

# Example: HAS and proxy



## Clients' want

- ❑ High playback quality
- ❑ Small stall times
- ❑ Few buffer interruptions
- ❑ Few quality switches

HAS is increasingly responsible  
for larger traffic volumes  
... proxies to reduce traffic??

# Example: HAS and proxy



## **Clients' want**

- ☐ High playback quality
- ☐ Small stall times
- ☐ Few buffer interruptions
- ☐ Few quality switches

## **Network providers' want**

- High QoE of customers/clients

# Example: HAS and proxy



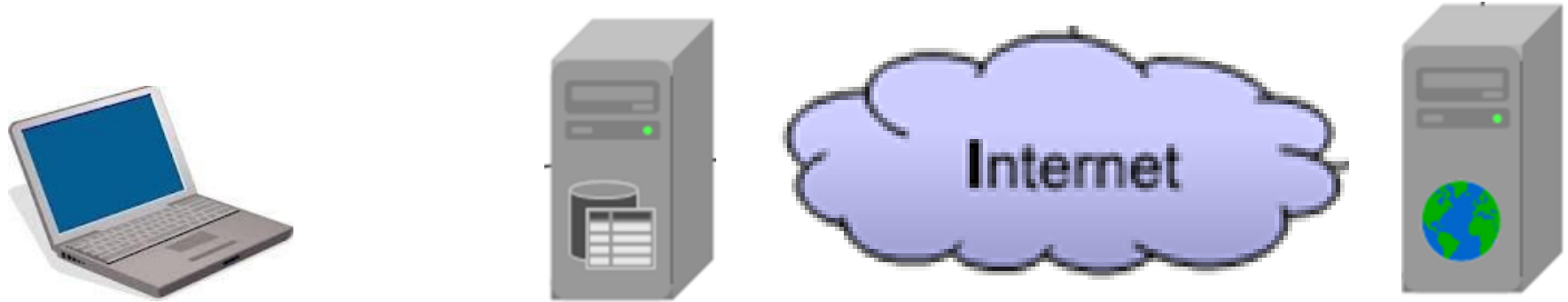
## **Clients' want**

- ☐ High playback quality
- ☐ Small stall times
- ☐ Few buffer interruptions
- ☐ Few quality switches

## **Network providers' want**

- High QoE of customers/clients
- Low bandwidth usage
- High hit rate

## Example: HAS and proxy



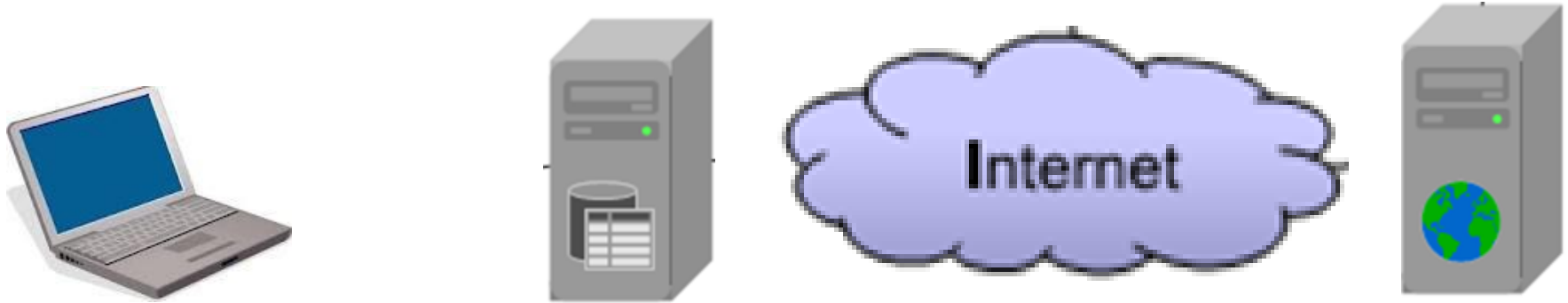
### **Clients' want**

- ☐ High playback quality
- ☐ Small stall times
- ☐ Few buffer interruptions
- ☐ Few quality switches

### **Network providers' want**

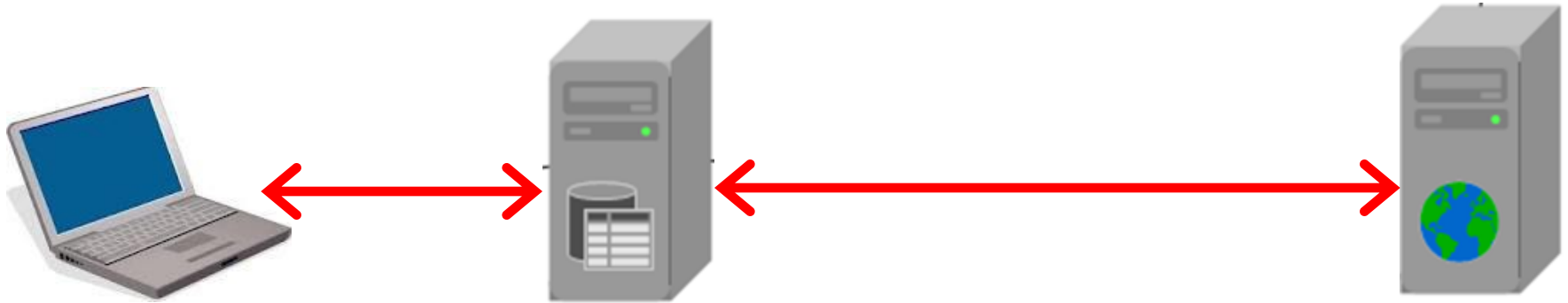
- High QoE of customers/clients
- Low bandwidth usage
- High hit rate

## Example: HAS and proxy



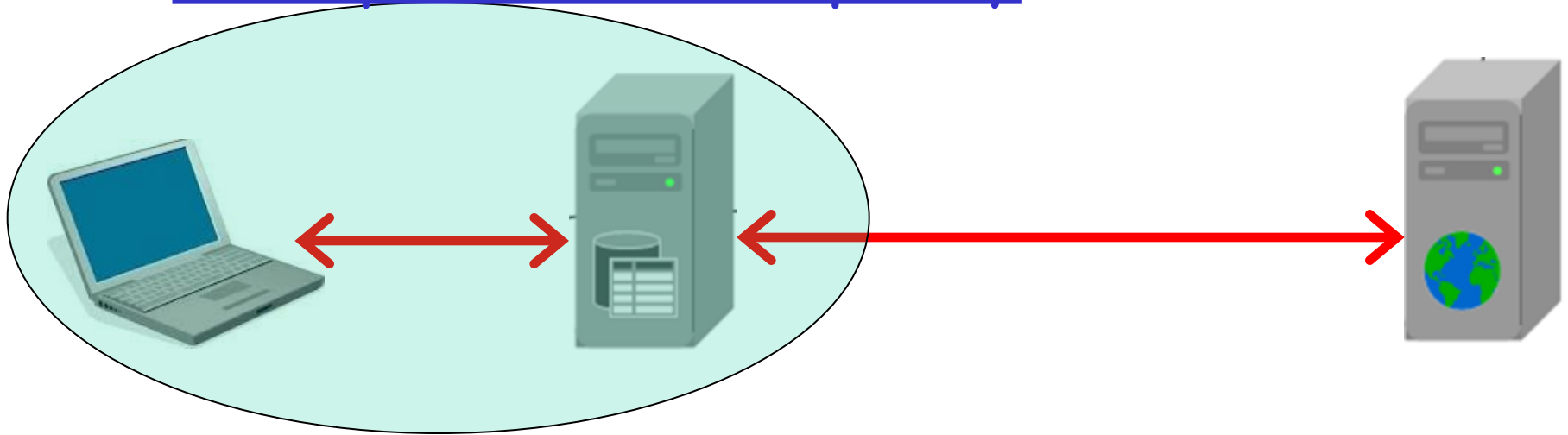
Proxy example ...

## Example: HAS and proxy

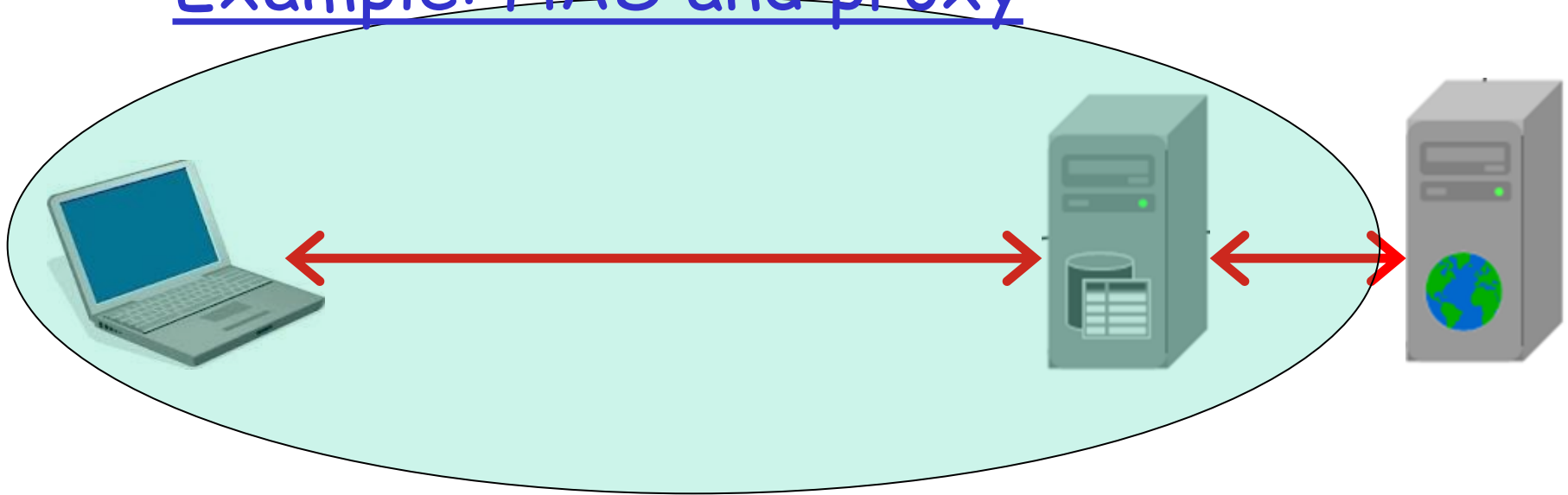




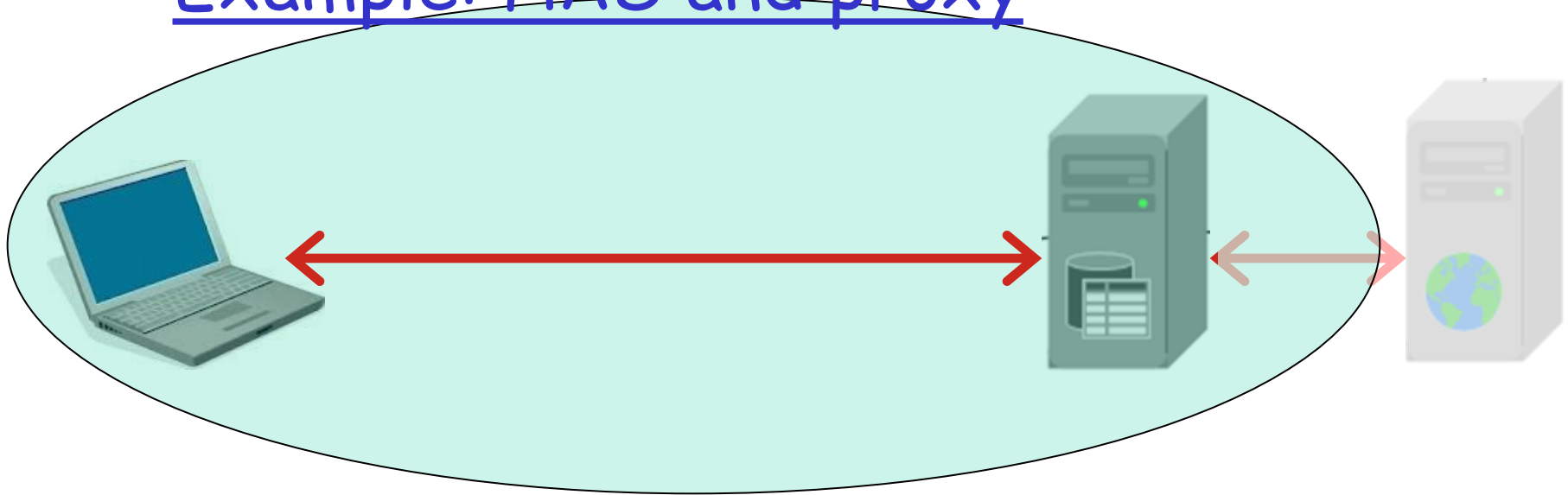
## Example: HAS and proxy



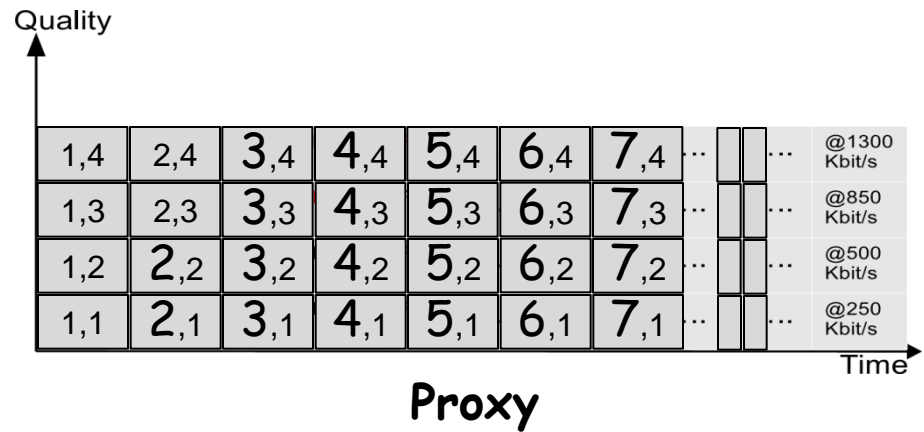
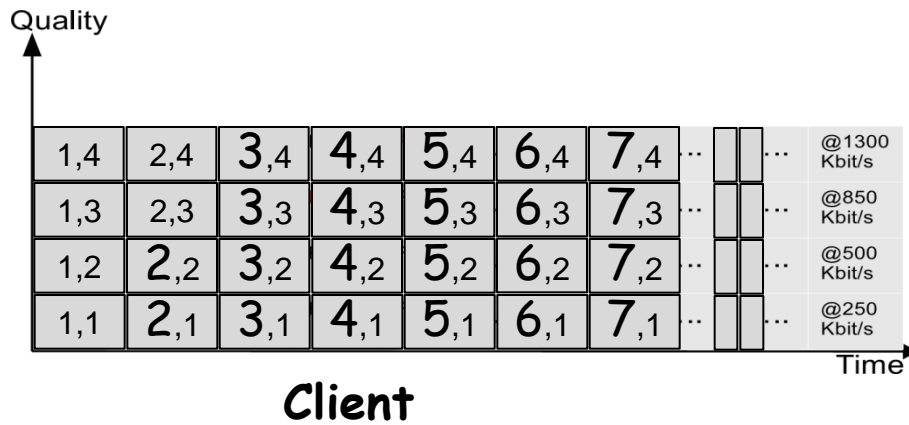
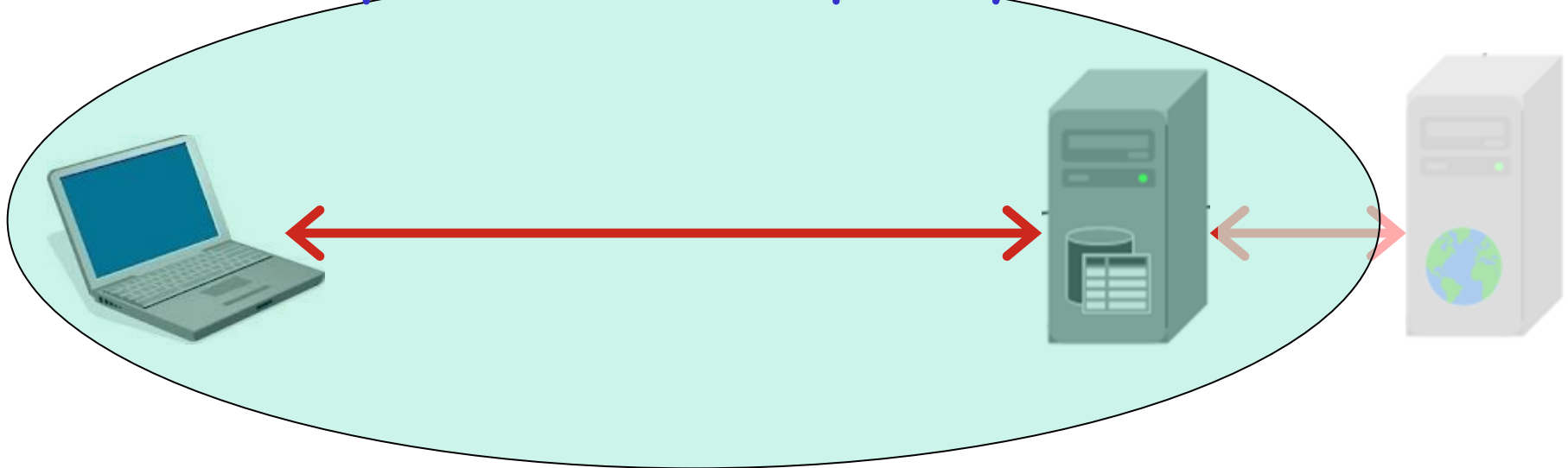
## Example: HAS and proxy



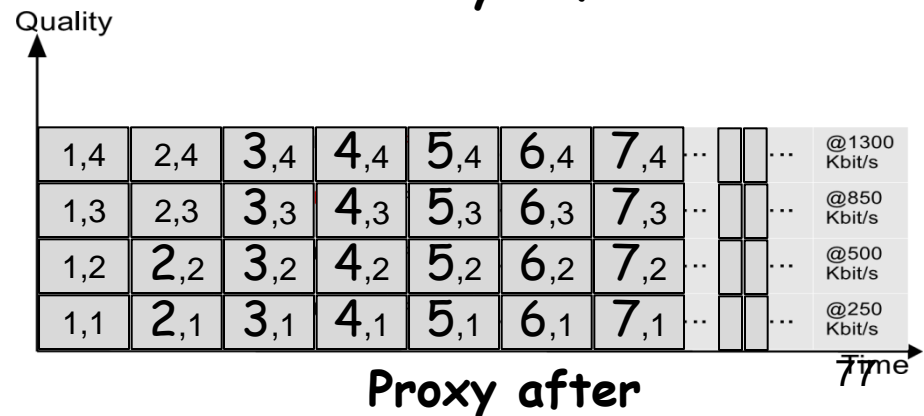
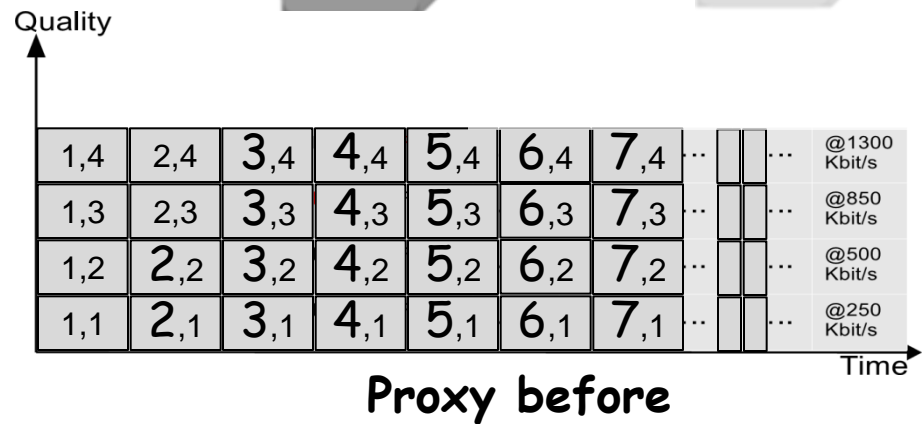
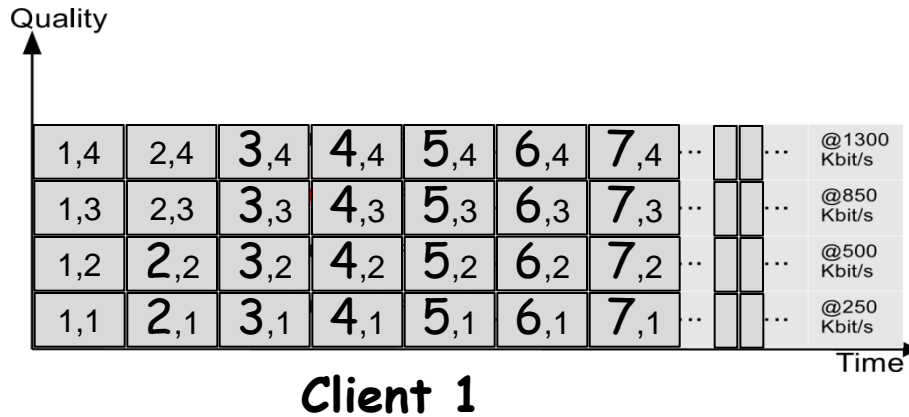
## Example: HAS and proxy



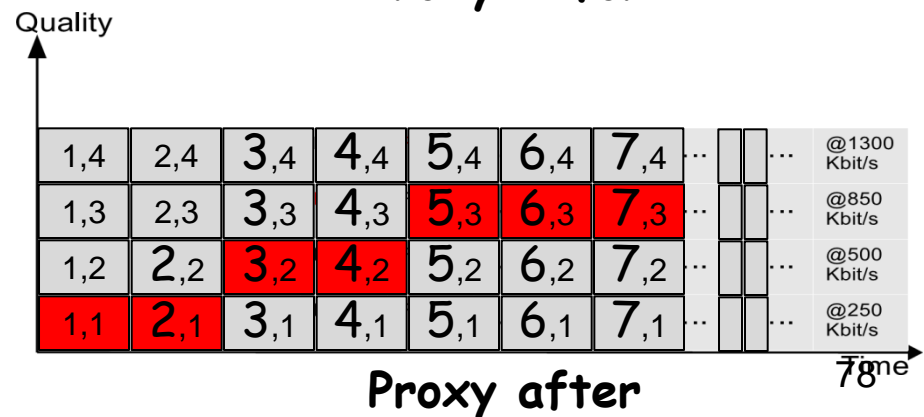
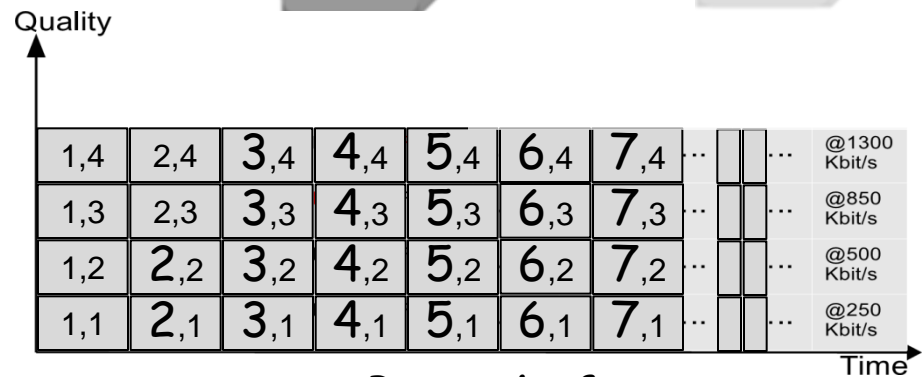
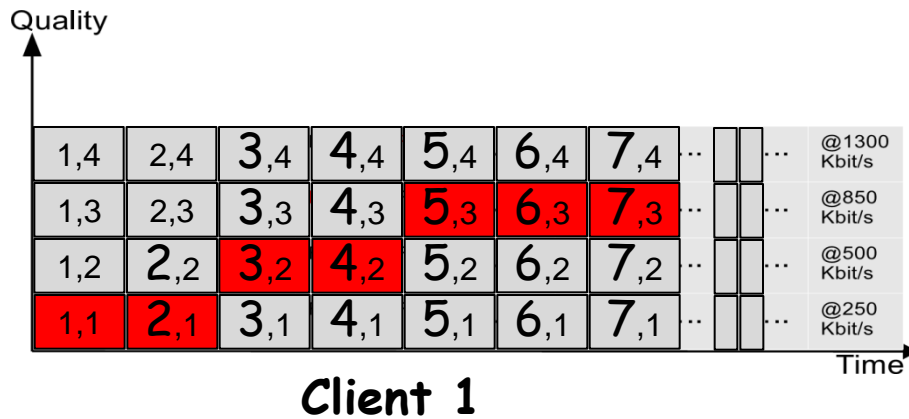
# Example: HAS and proxy



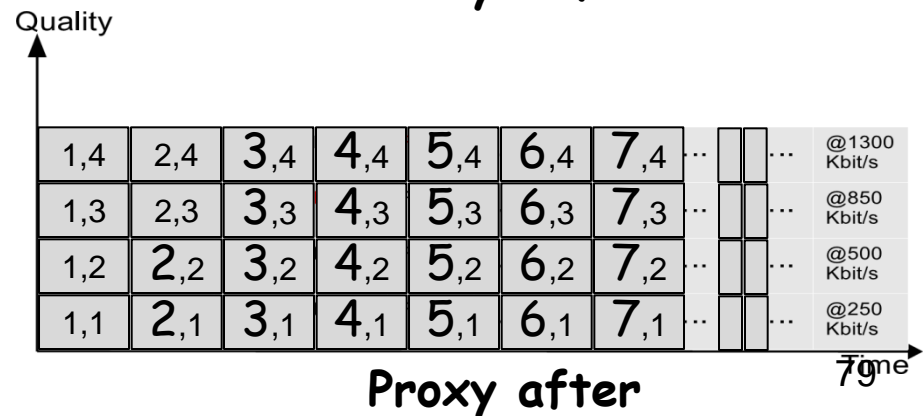
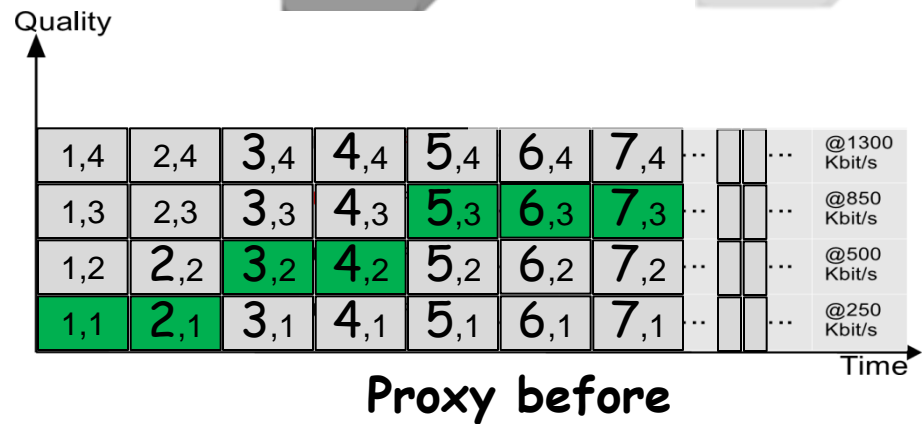
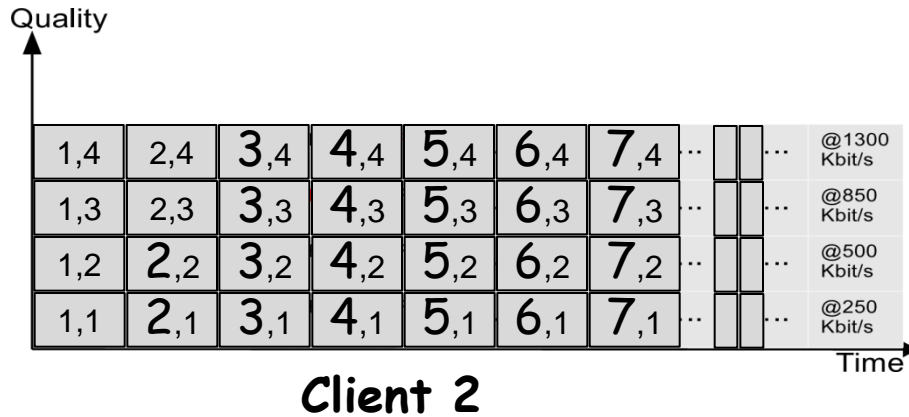
# Example: HAS and proxy



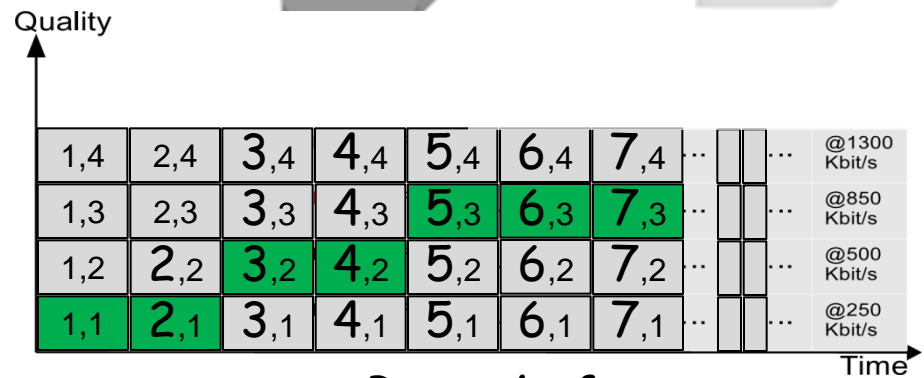
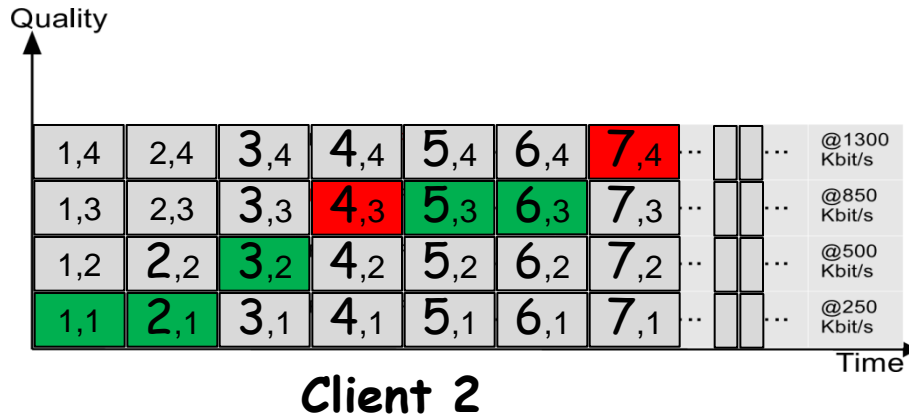
# Example: HAS and proxy



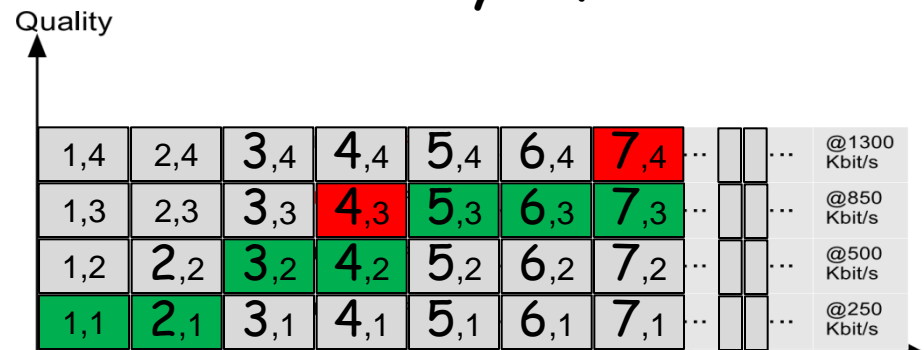
# Example: HAS and proxy



# Example: HAS and proxy



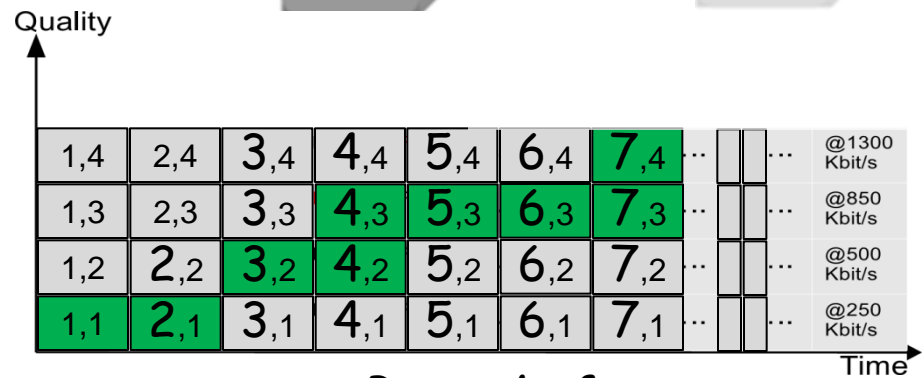
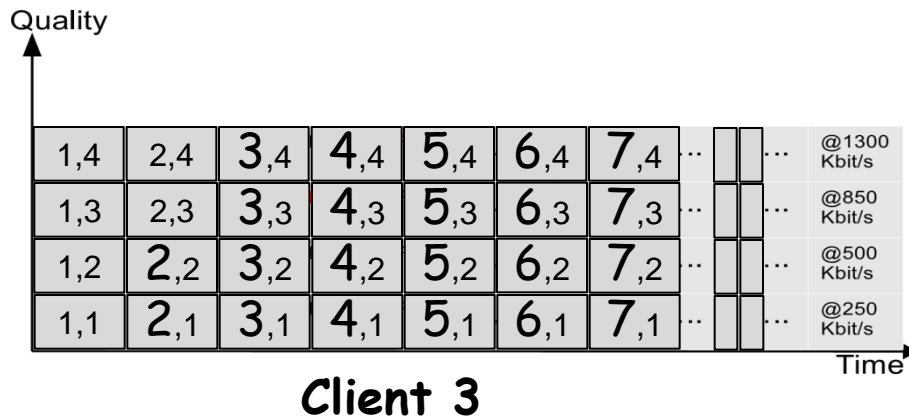
**Proxy before**



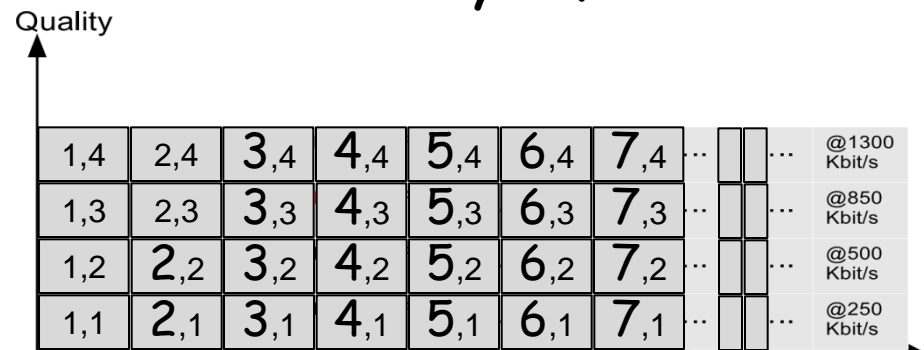
**Proxy after**



# Example: HAS and proxy

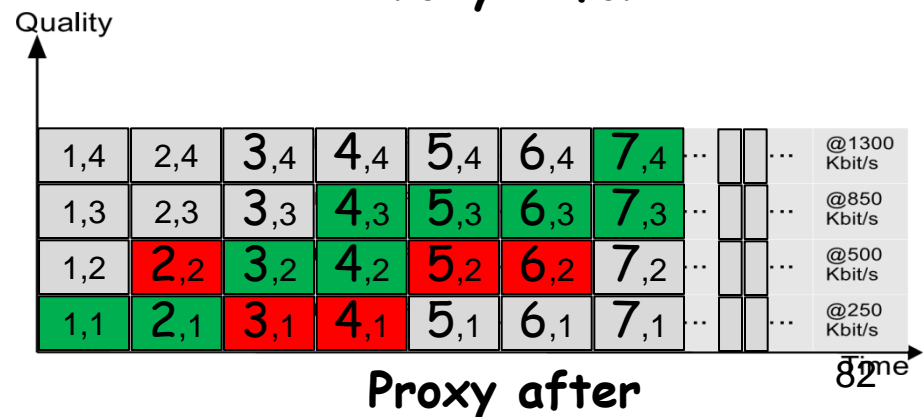
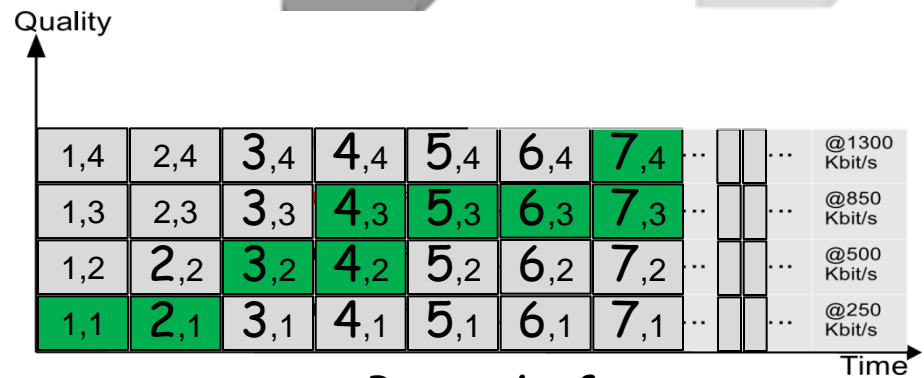
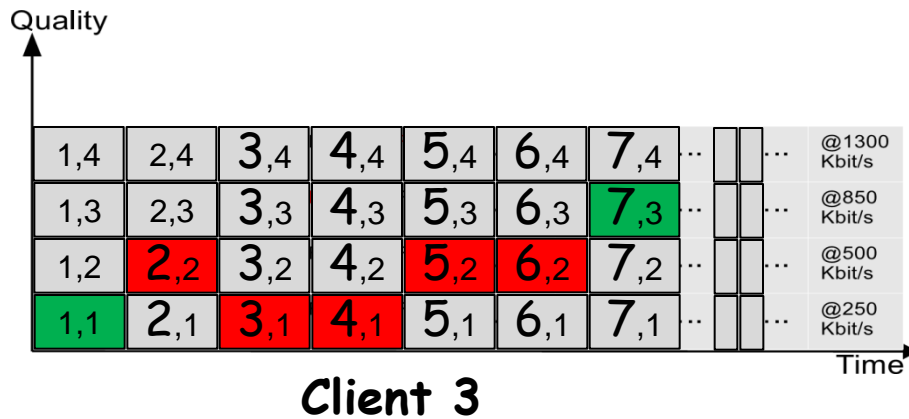


**Proxy before**



**Proxy after**

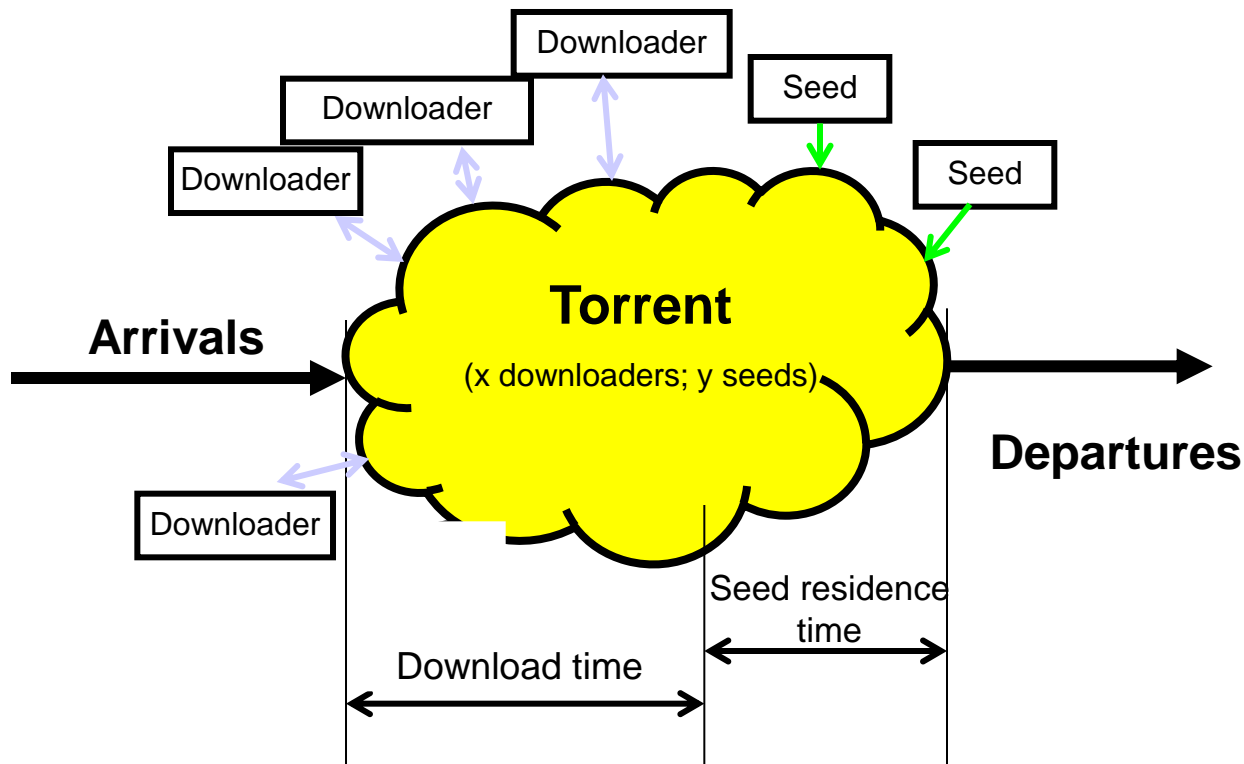
# Example: HAS and proxy





# BitTorrent-like systems

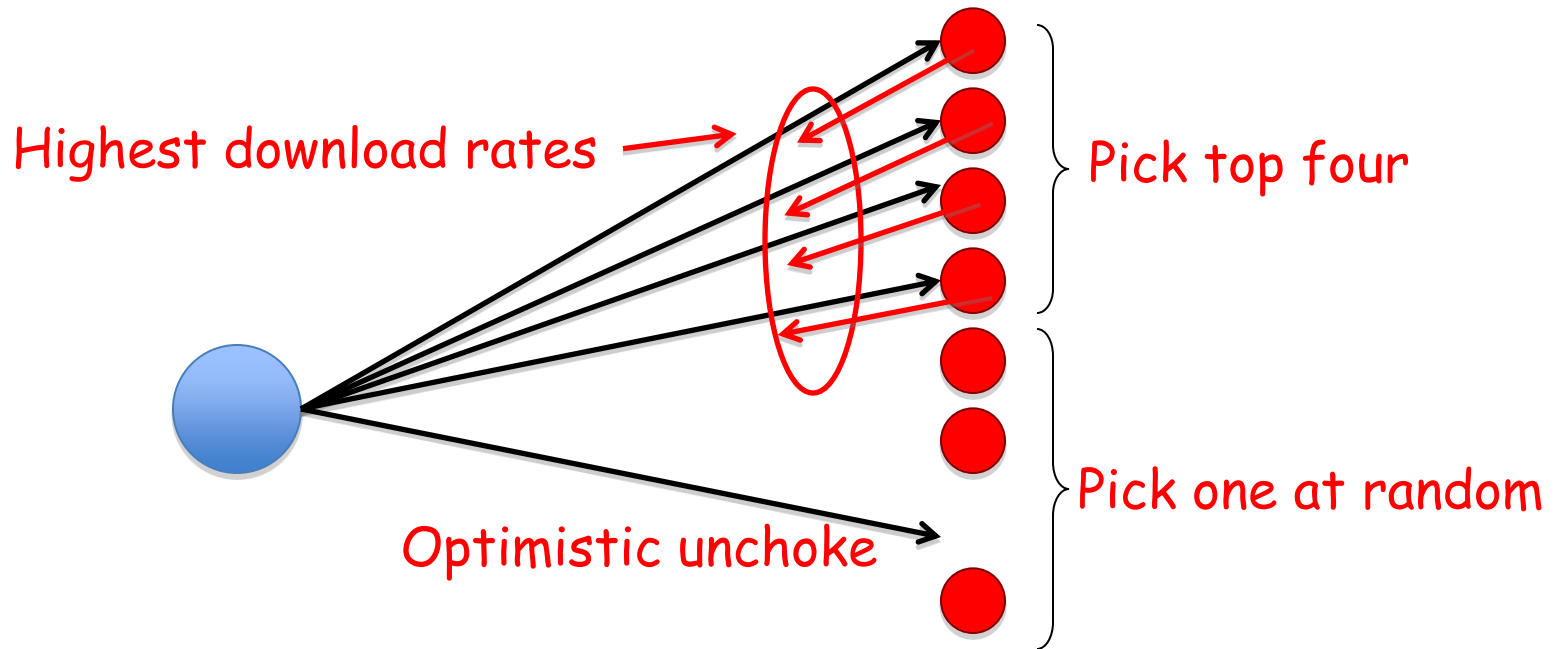
- ❑ File split into many smaller pieces
- ❑ Pieces are downloaded from both seeds and downloaders
- ❑ Distribution paths are dynamically determined
  - Based on data availability



# Download using BitTorrent

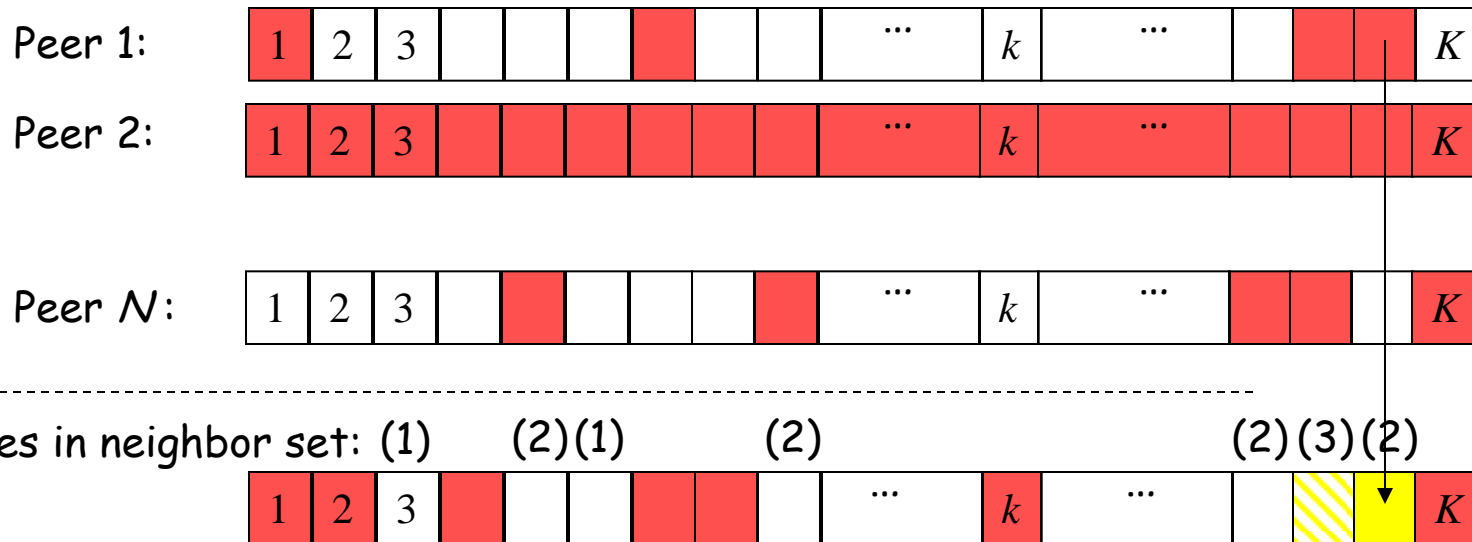
## Background: Incentive mechanism

- ❑ Establish connections to large set of peers
  - At each time, only upload to a small (changing) set of peers
- ❑ Rate-based tit-for-tat policy
  - Downloaders give upload preference to the downloaders that provide the highest download rates



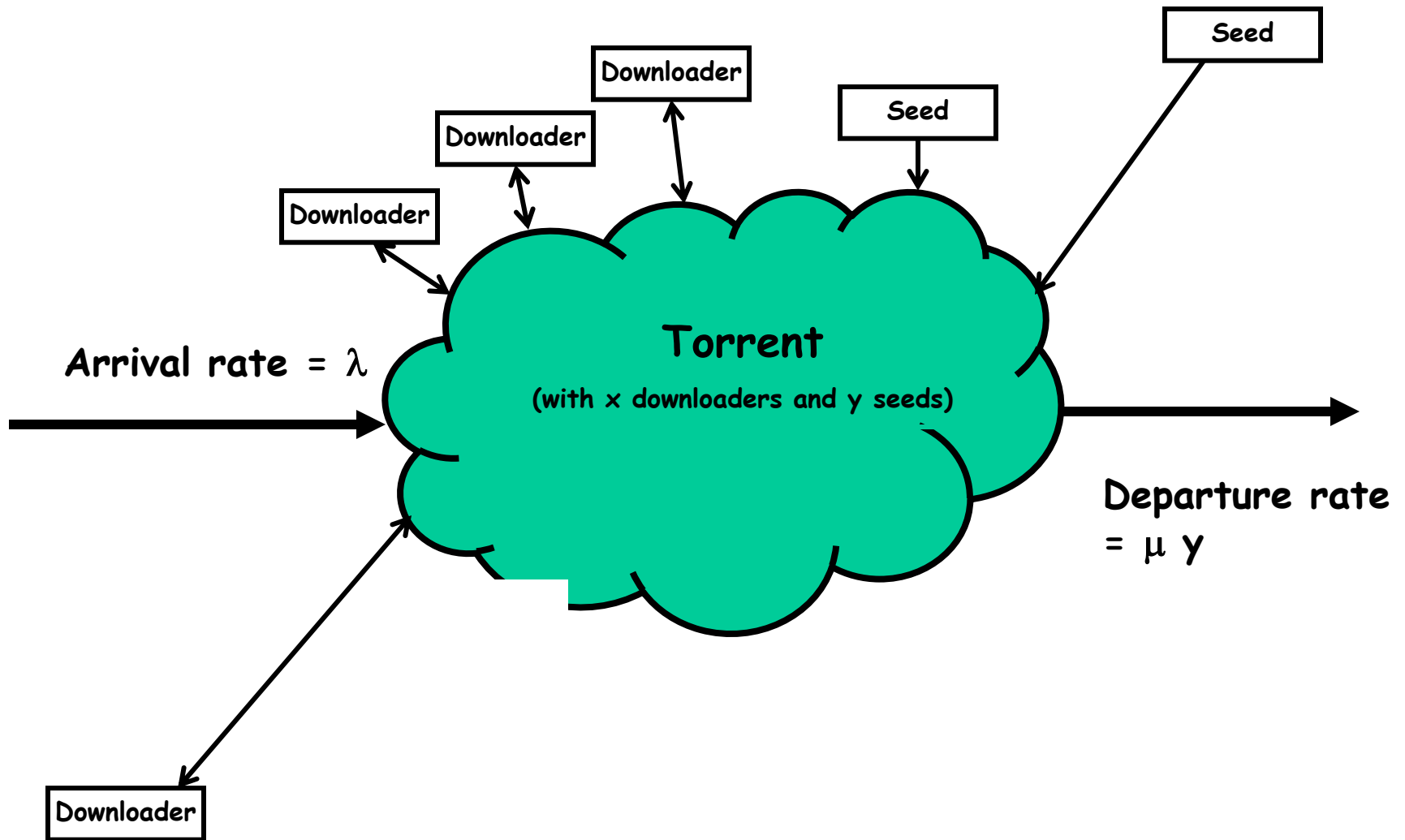
# Download using BitTorrent

## Background: Piece selection

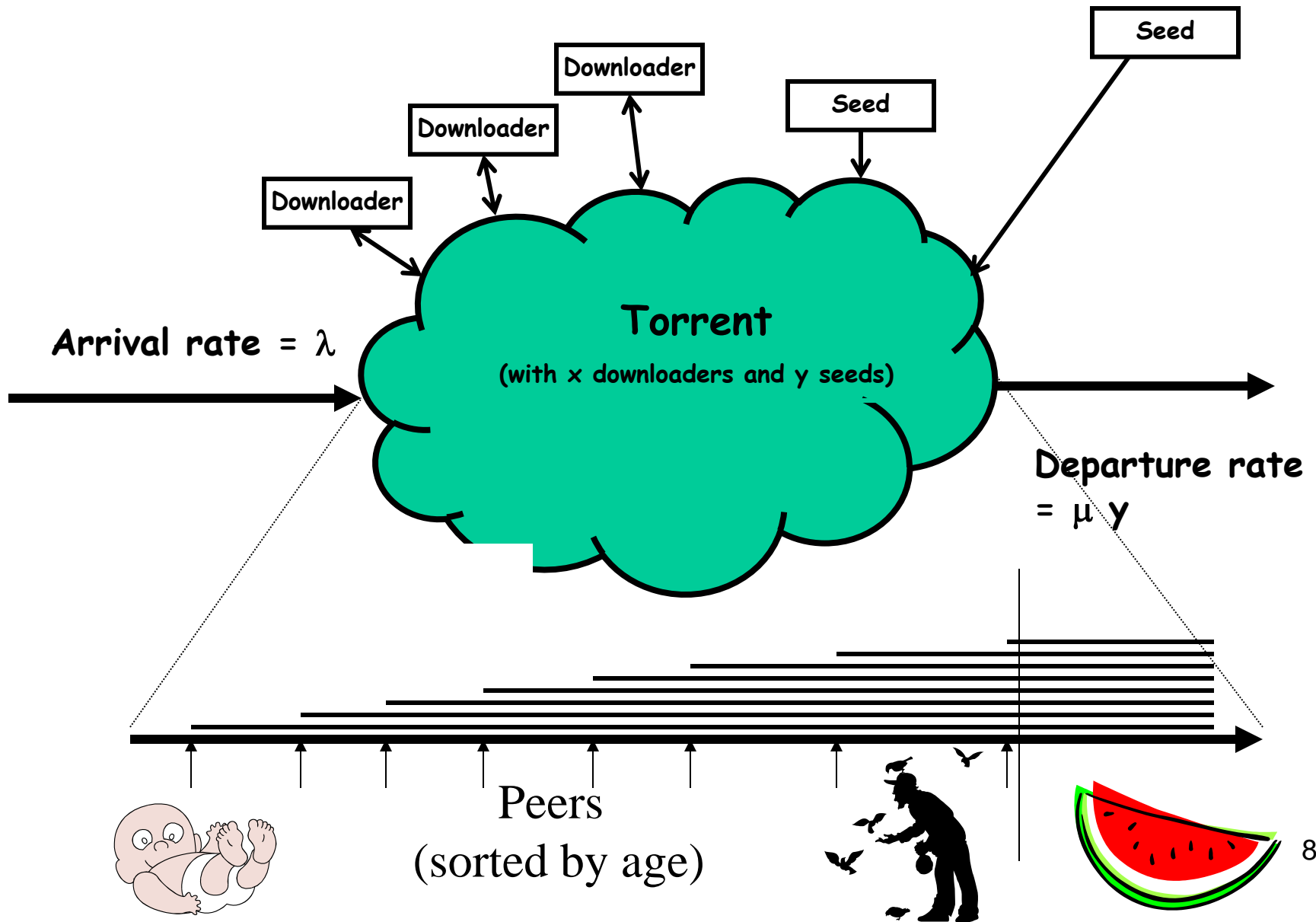


- ❑ Rarest first piece selection policy
  - Achieves high piece diversity
- ❑ Request pieces that
  - the uploader has;
  - the downloader is interested (wants); and
  - is the rarest among this set of pieces

# BitTorrent Model

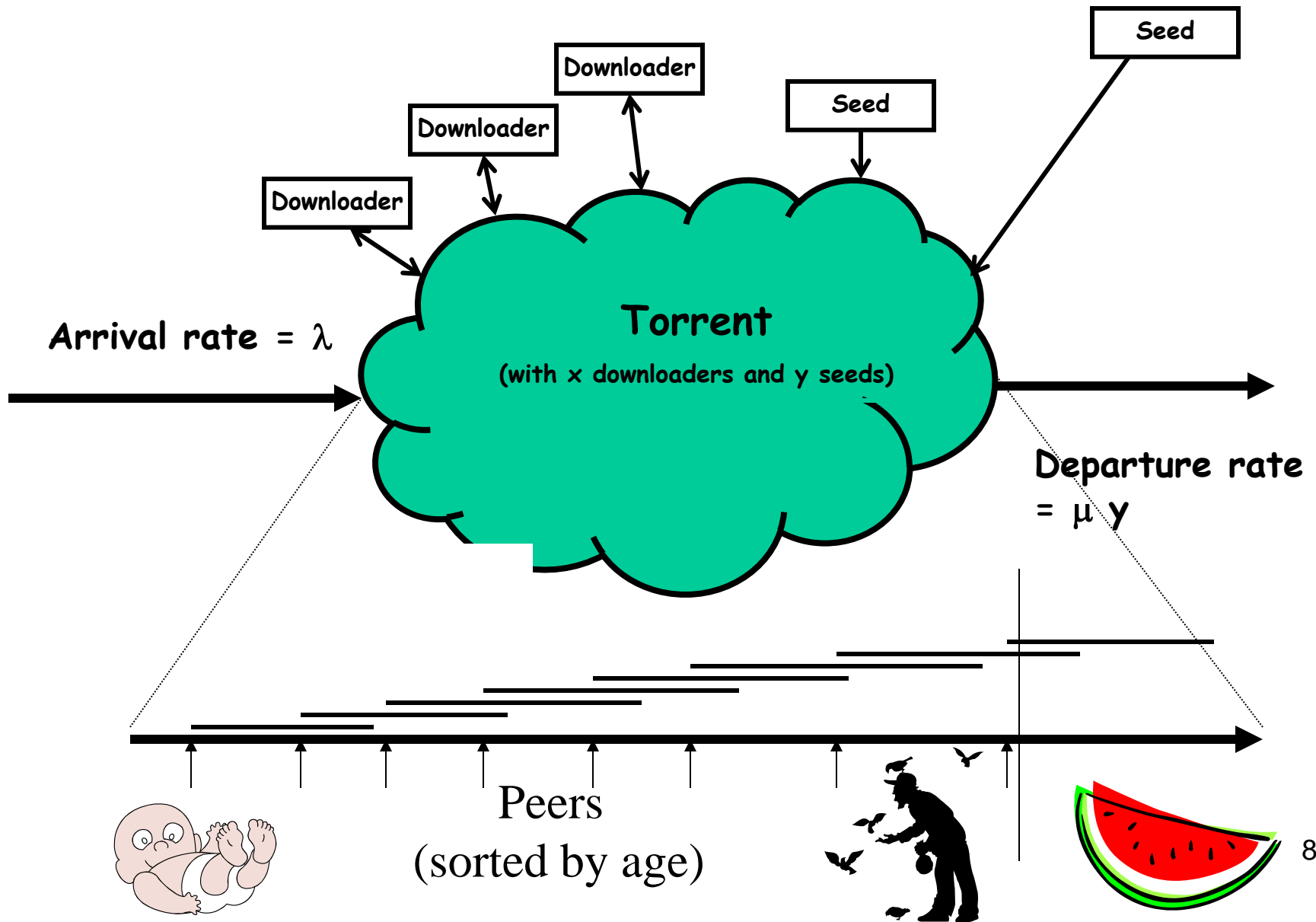


# BitTorrent Model (random)

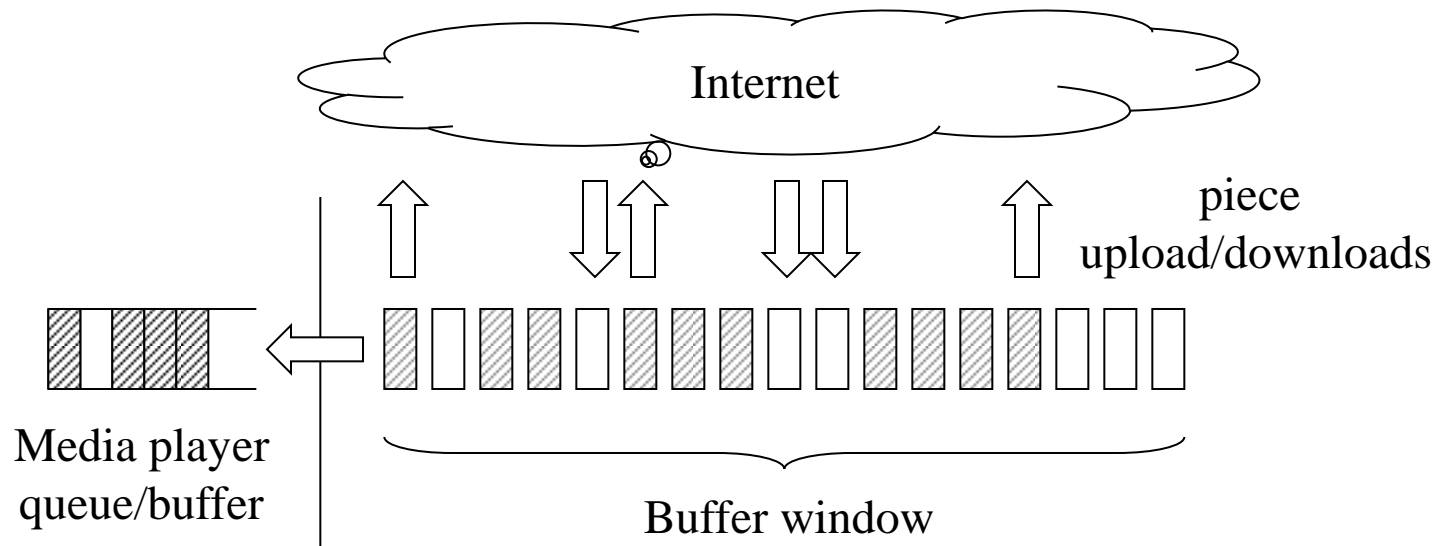




# BitTorrent Model (chaining)



# Live Streaming using Bittorrent-like systems



## ❑ Live streaming (e.g., CoolStreaming)

- All peers at roughly the same play/download position
  - High bandwidth peers can easily contribute more ...
- (relatively) Small buffer window
  - Within which pieces are exchanged

# Peer-assisted VoD streaming

- ❑ Can BitTorrent-like protocols provide scalable on-demand streaming?
- ❑ How sensitive is the performance to the application configuration parameters?
  - Piece selection policy (rarest vs. in-order tradeoff)
  - Peer selection policy
  - Upload/download bandwidth
- ❑ What is the user-perceived performance?
  - Start-up delay
  - Probability of disrupted playback



# A protocol family for streaming

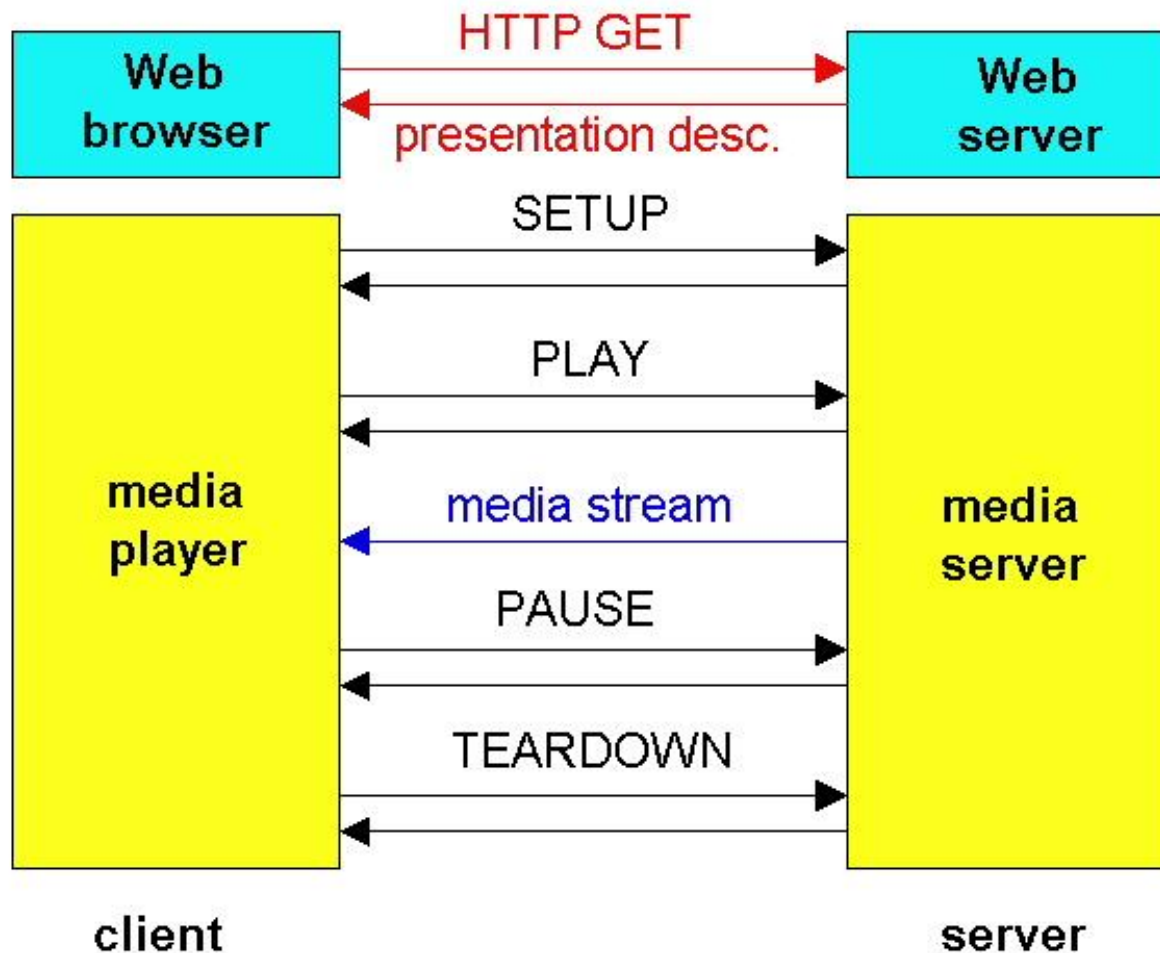
- ❑ RTSP
- ❑ RTP
- ❑ RTCP

# RTSP Example

## Scenario:

- ❑ metafile communicated to web browser
- ❑ browser launches player
- ❑ player sets up an RTSP control connection, data connection to streaming server

# RTSP Operation



Control messages  
"out-of-band"

○ port 554

Media stream "in-band".

# Real-Time Protocol (RTP)

- ❑ RTP specifies packet structure for packets carrying audio, video data
- ❑ RFC 3550
- ❑ RTP runs in end systems
- ❑ RTP packets encapsulated in UDP segments



# RTP Header



RTP Header

*Payload Type (7 bits)*: Indicates type of encoding currently being used. If sender changes encoding in middle of conference, sender informs receiver via payload type field.

- Payload type 0: PCM mu-law, 64 kbps
- Payload type 3, GSM, 13 kbps
- Payload type 7, LPC, 2.4 kbps
- Payload type 26, Motion JPEG
- Payload type 31. H.261
- Payload type 33, MPEG2 video

*Sequence Number (16 bits)*: Increments by one for each RTP packet sent, and may be used to detect packet loss and to restore packet sequence.

# Real-time Control Protocol (RTCP)

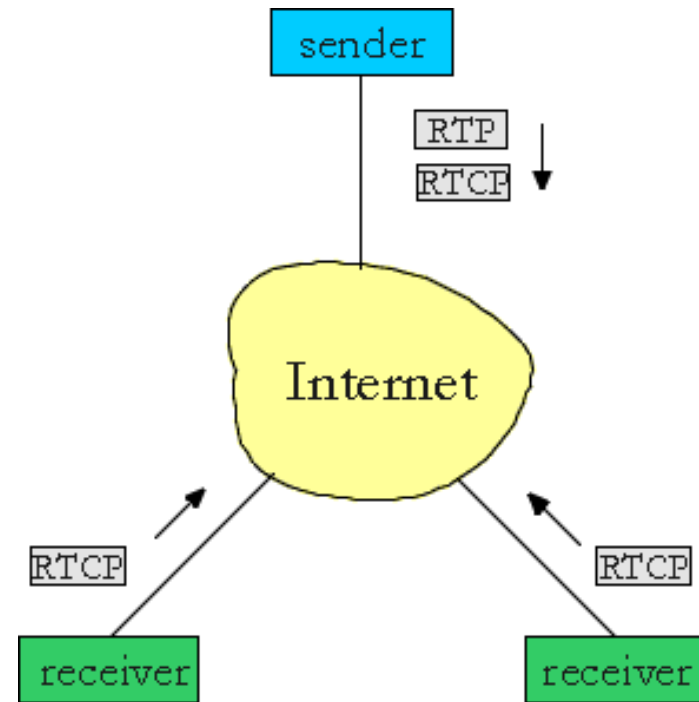
## Receiver report packets:

- ❑ fraction of packets lost, last sequence number, average interarrival jitter

## Sender report packets:

- ❑ SSRC of RTP stream, current time, number of packets sent, number of bytes sent

- ❑ feedback can be used to control performance



- ❑ RTCP attempts to limit its traffic to 5% of session bandwidth



# Multimedia Over "Best Effort" Internet

- **TCP/UDP/IP:** *no guarantees on delay, loss*



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



Today's multimedia applications implement functionality at the app. layer to mitigate (as best possible) effects of delay, loss

# Packet Loss

- ❑ **network loss:** IP datagram lost due to network congestion (router buffer overflow) or losses at wireless link(s)
- ❑ **delay loss:** IP datagram arrives too late for playout at receiver (effectively the same as if it was lost)
  - delays: processing, queueing in network; end-system (sender, receiver) delays
  - Tolerable delay depends on the application
- ❑ How can packet loss be handled?
  - We will discuss this next ...

# Receiver-based Packet Loss Recovery

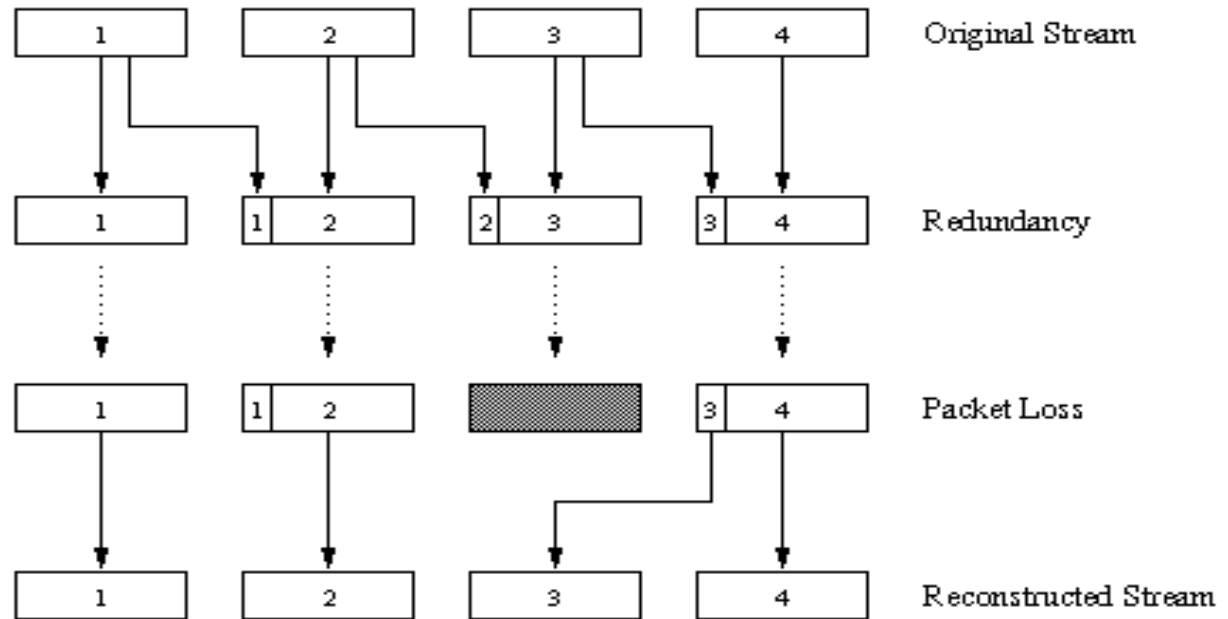
- ❑ Generate replacement packet
  - Packet repetition
  - Interpolation
  - Other sophisticated schemes
- ❑ Works when audio/video streams exhibit short-term correlations (e.g., self-similarity)
- ❑ Works for relatively low loss rates (e.g.,  $< 5\%$ )
- ❑ Typically, breaks down on “bursty” losses

# Forward Error Correction (FEC)

- ❑ For every group of  $n$  actual media packets, generate  $k$  additional redundant packets
- ❑ Send out  $n+k$  packets, which increases the bandwidth consumption by factor  $k/n$ .
- ❑ Receiver can reconstruct the original  $n$  media packets provided at most  $k$  packets are lost from the group
- ❑ Works well at high loss rates (for a proper choice of  $k$ )
- ❑ Handles "bursty" packet losses
- ❑ Cost: increase in transmission cost (bandwidth)

# Another FEC Example

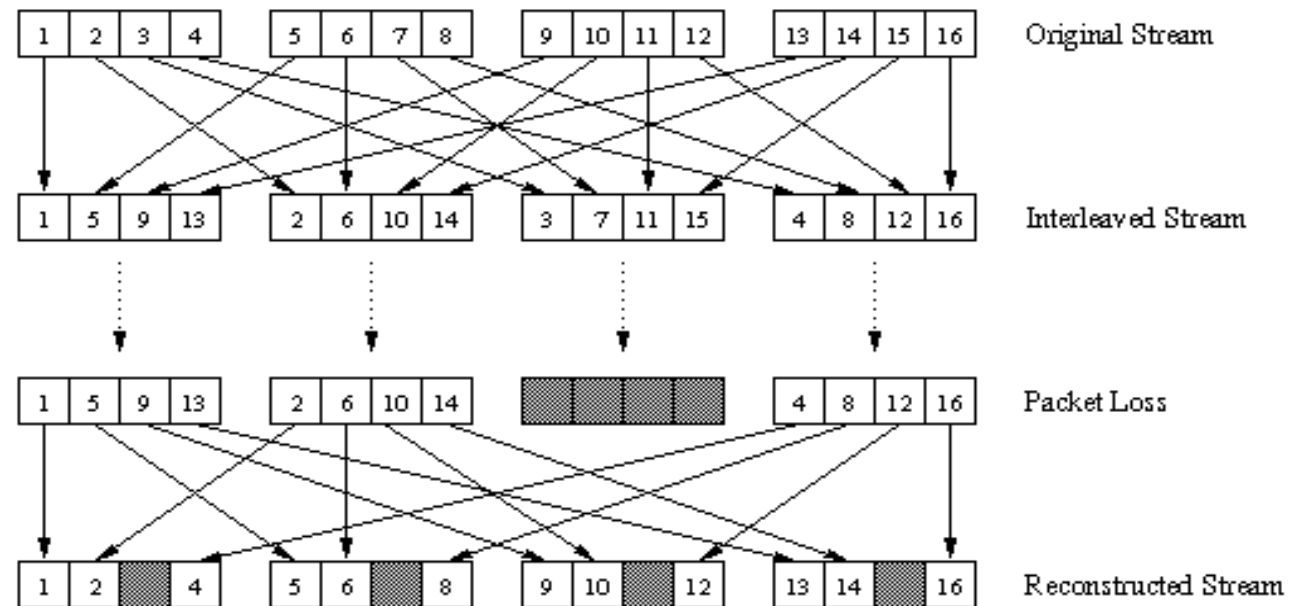
- “piggyback lower quality stream”



- Whenever there is non-consecutive loss, the receiver can conceal the loss.
- Can also append (n-1)st and (n-2)nd low-bit rate chunk



# Interleaving: Recovery from packet loss



## Interleaving

- ❑ Intentionally alter the sequence of packets before transmission
- ❑ Better robustness against "burst" losses of packets
- ❑ Results in increased playout delay from inter-leaving



More slides ....



# Outline

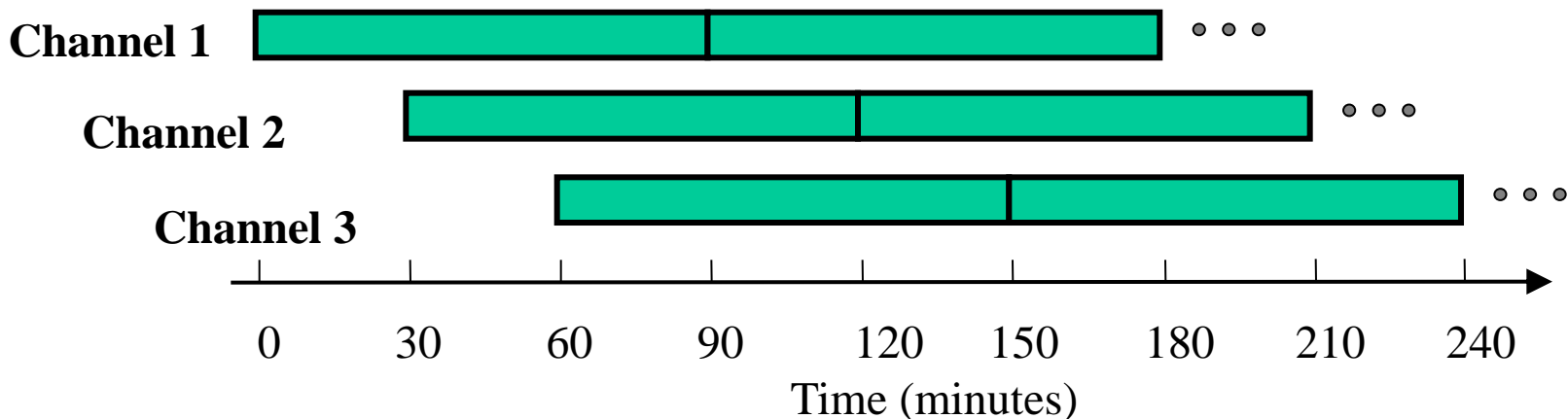
- ❑ Multimedia Networking Applications
- ❑ Streaming stored audio and video
- ❑ Scalable Streaming Techniques
- ❑ Content Distribution Networks
- ❑ Beyond Best Effort

# Streaming Popular Content

- ❑ Consider a popular media file
  - Playback rate: 1 Mbps
  - Duration: 90 minutes
  - Request rate: once every minute
- ❑ How can a video server handle such high loads?
  - **Approach 1:** Start a new "stream" for each request
  - Allocate server and disk I/O bandwidth for each request
  - Bandwidth required at server =  $1 \text{ Mbps} \times 90$

# Streaming Popular Content using Batching

- ❑ **Approach 2:** Leverage the multipoint delivery capability of modern networks
- ❑ Playback rate = 1 Mbps, duration = 90 minutes
- ❑ Group requests in non-overlapping intervals of 30 minutes:
  - Max. start-up delay = 30 minutes
  - Bandwidth required = 3 channels = 3 Mbps



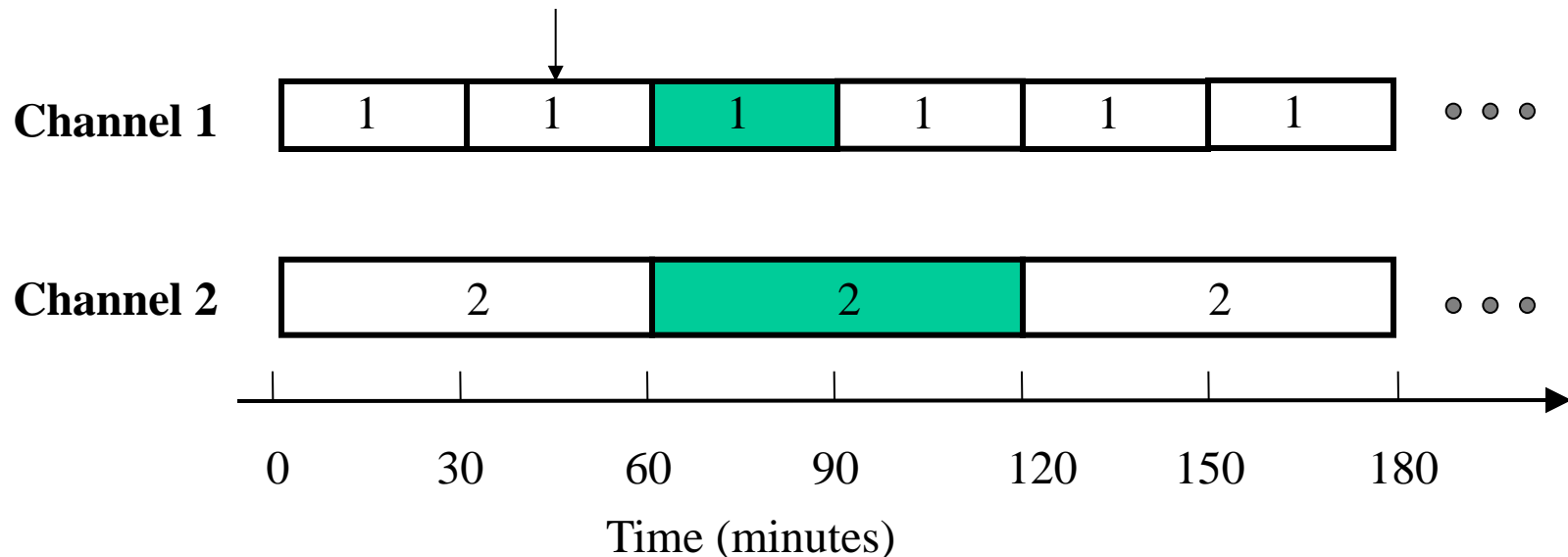
# Batching Issues

- ❑ Bandwidth increases linearly with decrease in start-up delays
- ❑ Can we reduce or eliminate “start-up” delays?
  - Periodic Broadcast Protocols
  - Stream Merging Protocols



# Periodic Broadcast Example

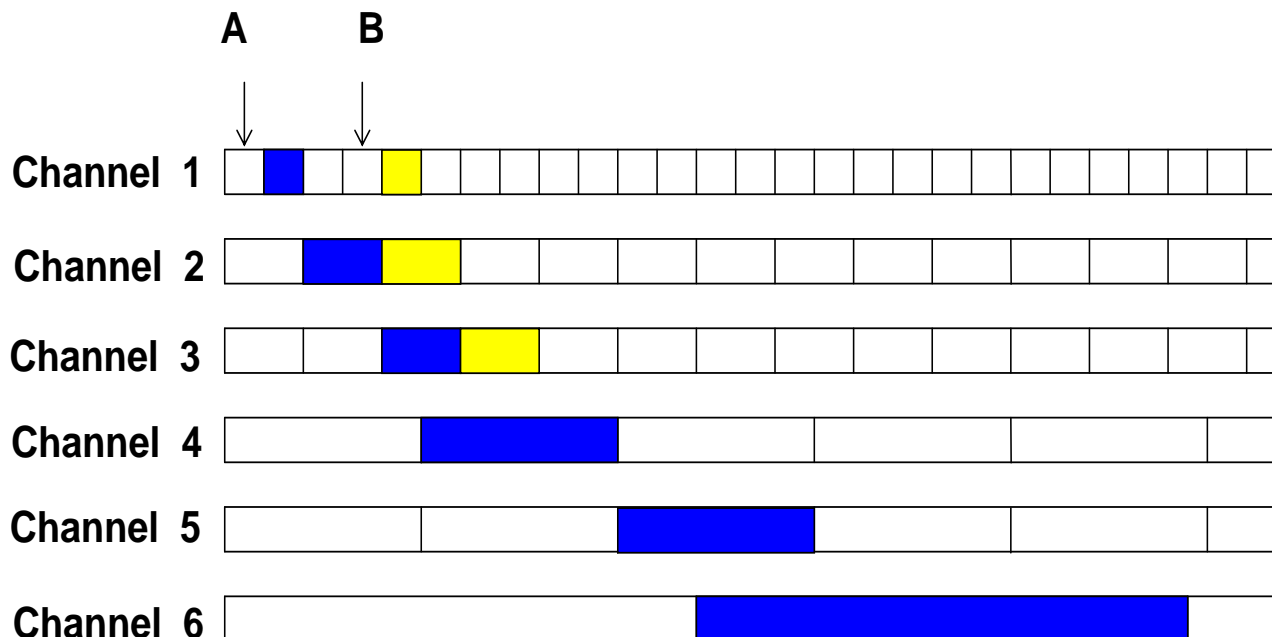
- ❑ Partition the media file into 2 segments with relative sizes {1, 2}. For a 90 min. movie:
  - Segment 1 = 30 minutes, Segment 2 = 60 minutes
- ❑ Advantage:
  - Max. start-up delay = 30 minutes
  - Bandwidth required = 2 channels = 2 Mbps
- ❑ Disadvantage: Requires increased client capabilities



# Skyscraper Broadcasts (SB)

[Hua & Sheu 1997]

- Divide the file into *K* segments of increasing size
  - Segment size progression: 1, 2, 2, 5, 5, 12, 12, 25, ...
- Multicast each segment on a separate channel at the playback rate
- Aggregate rate to clients: *2 x playback rate*



# Comparing Batching and SB

Server Bandwidth	Start-up Delay	
	Batching	SB
1 Mbps	90 minutes	90 minutes
2 Mbps	45 minutes	30 minutes
6 Mbps	15 minutes	3 minutes
10 Mbps	9 minutes	30 seconds

❑ Playback rate = 1 Mbps, duration = 90 minutes

❑ Limitations of Skyscraper:

- Ad hoc segment size progress
- Does not work for low client data rates

# Reliable Periodic Broadcasts (RPB)

[Mahanti *et al.* 2001, 2003, 2004]

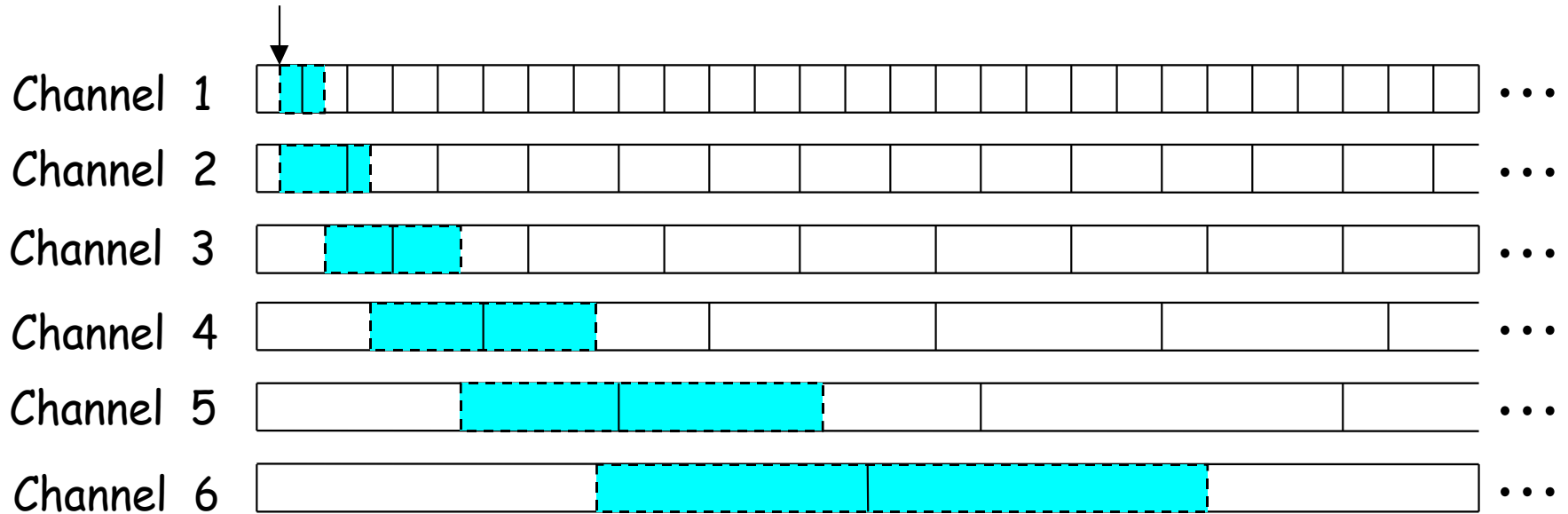
- ❑ **Optimized PB protocols** (no packet loss recovery)
  - client fully downloads each segment before playing
  - required server bandwidth near minimal
  - Segment size progression is *not* ad hoc
  - Works for client data rates  $< 2 \times \text{playback rate}$
- ❑ extend for packet loss recovery
- ❑ extend for “bursty” packet loss
- ❑ extend for client heterogeneity

# Reliable Periodic Broadcasts (RPB)

[Mahanti *et al.* 2001, 2003, 2004]

- ❑ **Optimized PB protocols** (no packet loss recovery)
  - client fully downloads each segment before playing
  - required server bandwidth near minimal
  - Segment size progression is *not* ad hoc
  - Works for client data rates  $< 2 \times \text{playback rate}$
- ❑ extend for packet loss recovery
- ❑ extend for “bursty” packet loss
- ❑ extend for client heterogeneity

# Optimized Periodic Broadcasts



- $r$  = segment streaming rate = 1
- $s$  = maximum # streams client listens to concurrently = 2
- $b$  = client data rate =  $s \times r = 2$

□ length of first  $s$  segments:  $\frac{1}{r}l_k = \frac{1}{r}l_1 + \sum_{j=1}^{k-1} l_j$

□ length of segment  $k > s$ :  $\frac{1}{r}l_k = \sum_{j=k-s}^{k-1} l_j$









# Outline

- ❑ Multimedia Networking Applications
- ❑ Streaming stored audio and video
- ❑ Scalable Streaming Techniques
- ❑ Content Distribution Networks
- ❑ Beyond Best Effort

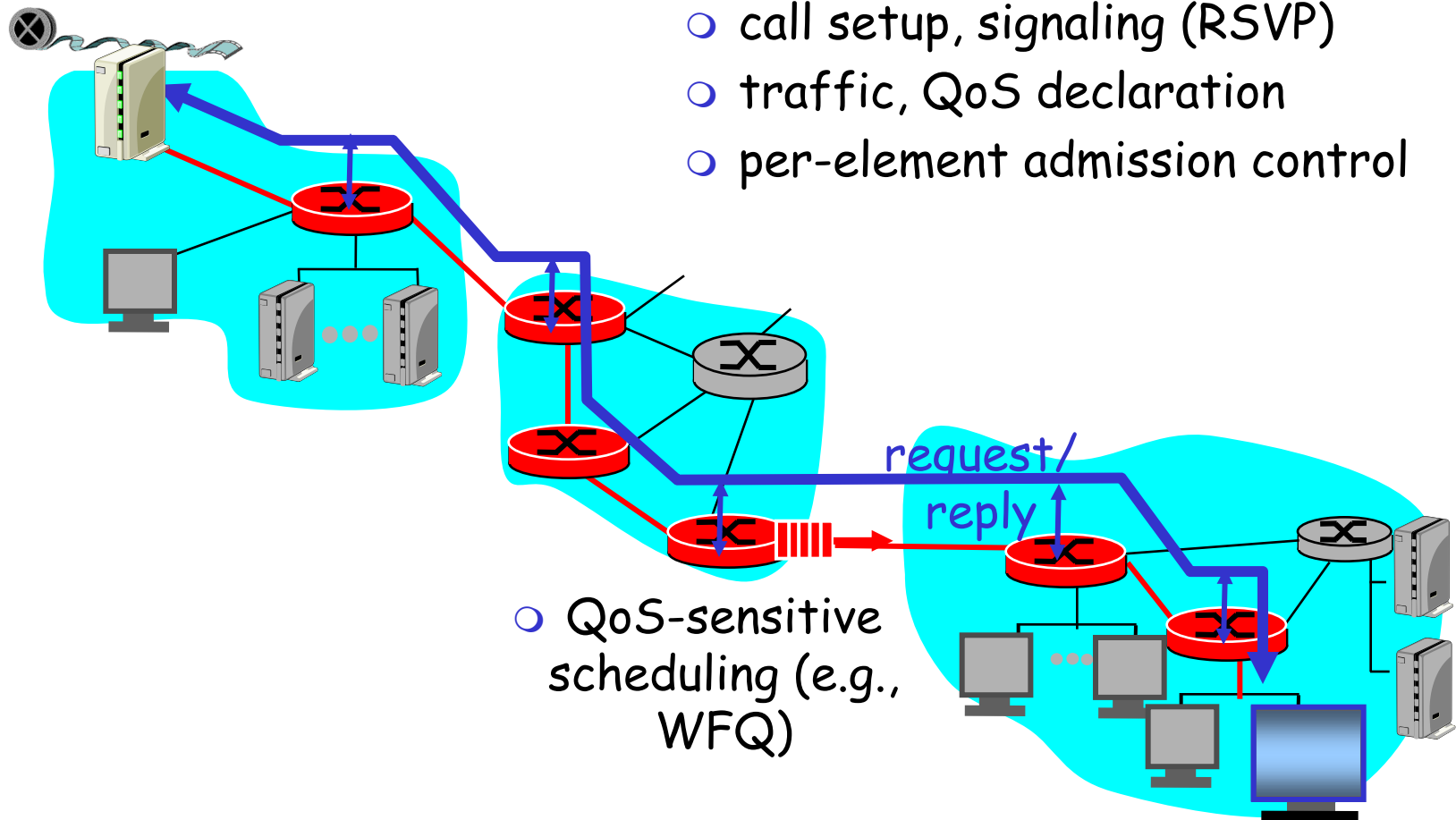
# Integrated Services (IntServ) Architecture

- ❑ architecture for providing QOS guarantees in IP networks for individual flows
- ❑ flow: a distinguishable stream of distinct IP datagrams
  - Unidirectional
  - Multiple recipient
- ❑ Components of this architecture:
  - Admission control
  - Reservation protocol
  - Routing protocol
  - Classifier and route selection
  - Packet scheduler

# Intserv: QoS guarantee scenario

## □ Resource reservation

- call setup, signaling (RSVP)
- traffic, QoS declaration
- per-element admission control



# Call Admission

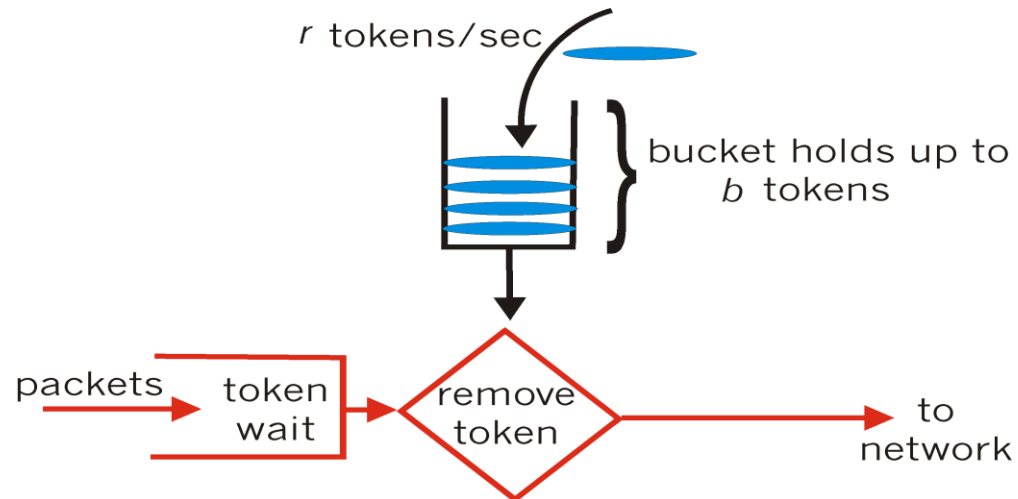
Arriving session must :

- ❑ declare its QoS requirement
  - **R-spec**: defines the QoS being requested
- ❑ characterize traffic it will send into network
  - **T-spec**: defines traffic characteristics
- ❑ signaling protocol: needed to carry R-spec and T-spec to routers (where reservation is required)
  - **RSVP**

Need Scheduling and Policing Policies to provide QoS

# Policing: Token Bucket

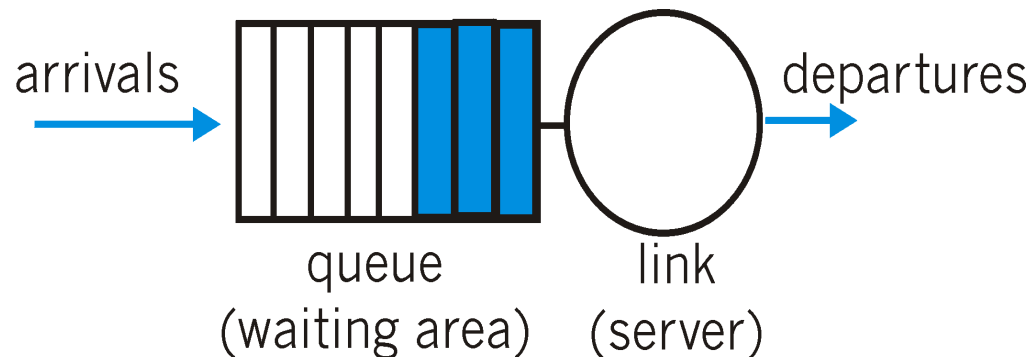
Token Bucket: limit input to specified Burst Size and Average Rate.



- ❑ bucket can hold  $b$  tokens
- ❑ tokens generated at rate  $r$  token/sec unless bucket full
- ❑ *over interval of length  $t$ : number of packets admitted less than or equal to  $(r t + b)$ .*

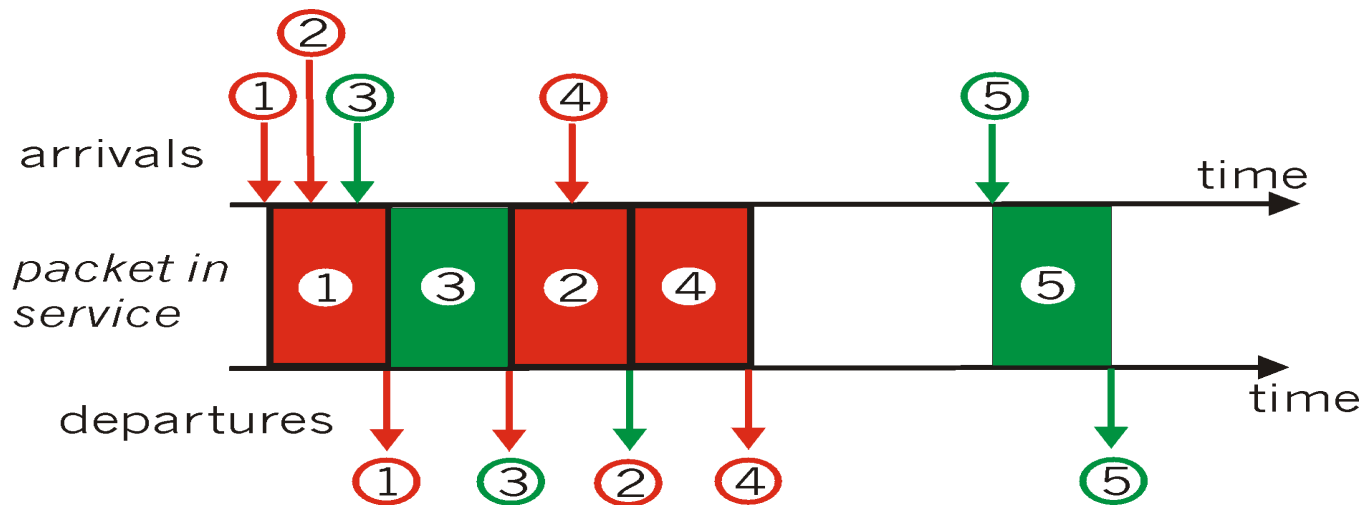
# Link Scheduling

- ❑ **scheduling**: choose next packet to send on link
- ❑ **FIFO (First In First Out) scheduling**: send in order of arrival to queue
  - **discard policy**: if packet arrives to full queue: who to discard?
    - DropTail: drop arriving packet
    - Priority: drop/remove on priority basis
    - Random: drop/remove randomly (e.g., RED)



# Round Robin

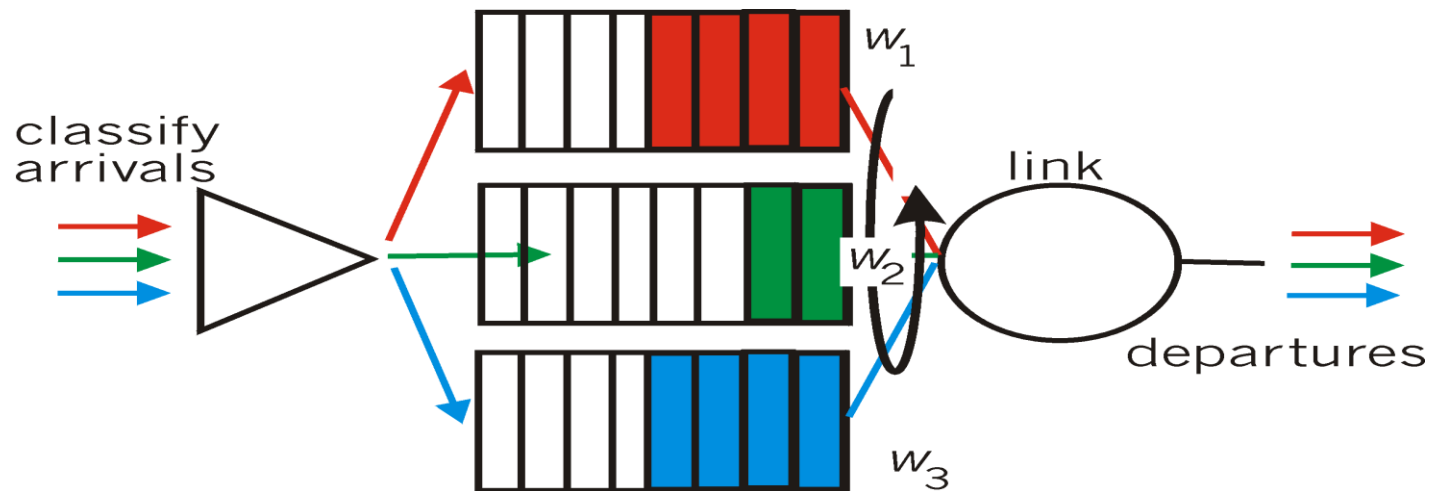
- multiple classes
- cyclically scan class queues, serving one from each class (if available)





# Weighted Fair Queuing

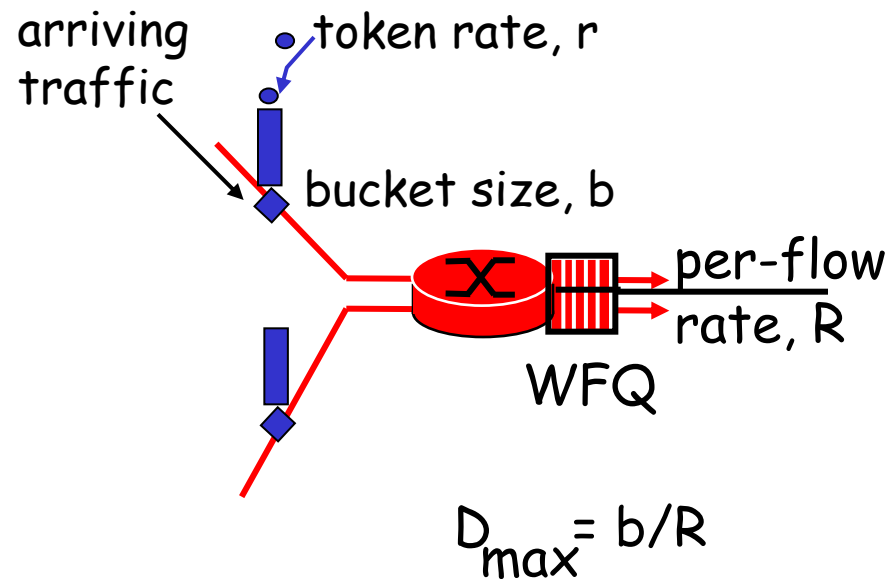
- generalized Round Robin
- each class gets weighted amount of service in each cycle



# IntServ QoS: Service models [rfc2211, rfc 2212]

## Guaranteed service:

- ❑ Assured data rate
- ❑ A specified upper bound on queuing delay



## Controlled load service:

- ❑ "a quality of service closely approximating the QoS that same flow would receive from an unloaded network element."
- ❑ Similar to behavior best effort service in an unloaded network

# Differentiated Services

## Concerns with IntServ:

- ❑ **Scalability:** signaling, maintaining per-flow router state difficult with large number of flows
- ❑ **Flexible Service Models:** Intserv has only two classes. Desire “qualitative” service classes
  - E.g., Courier, xPress, and normal mail
  - E.g., First, business, and cattle class ☺

## DiffServ approach:

- ❑ simple functions in network core, relatively complex functions at edge routers (or hosts)
- ❑ Don't define service classes, just provide functional components to build service classes

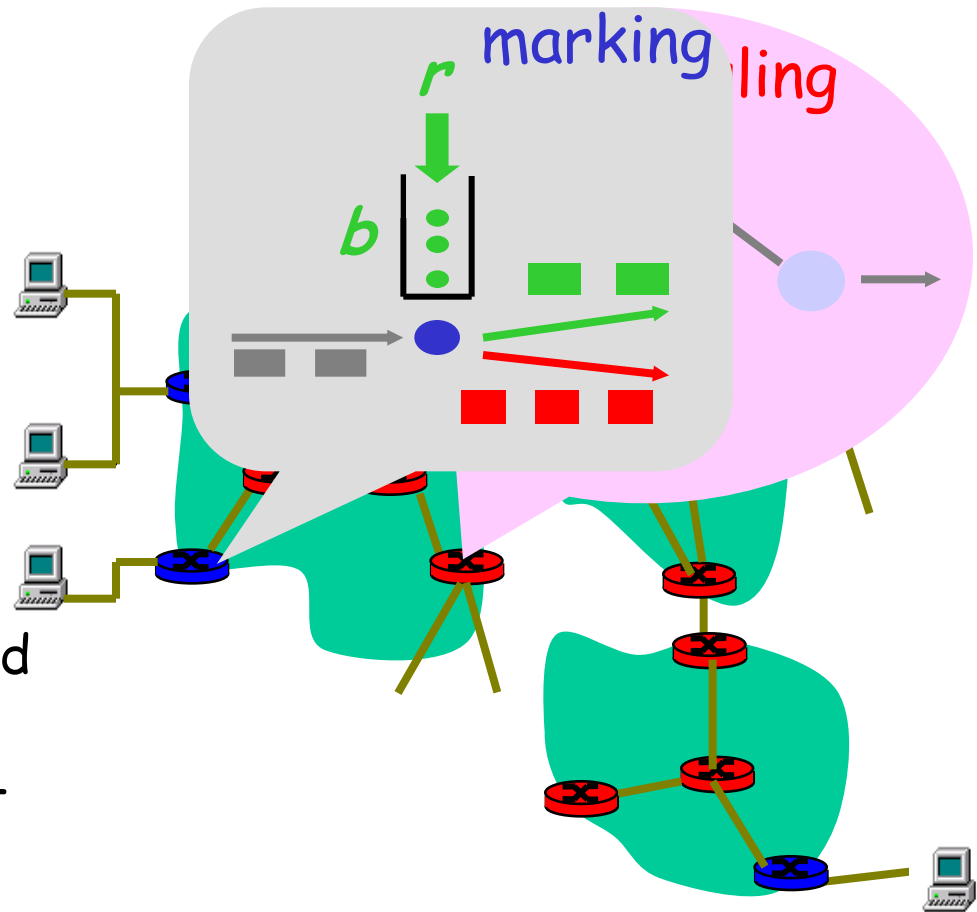
# DiffServ Architecture

## Edge router:

- per-flow traffic management
- Set the DS field; value determines type of service (PHB: Per-Hop Behavior)

## Core router:

- buffering and scheduling based on **marking** at edge
- per-class traffic management



# Traffic Classification/Conditioning

- ❑ How can packet markings be carried in IPv4 datagrams?
- ❑ Sender may agree to conform to a “traffic profile”, thus a leaky bucket policer may be used at the network edge to enforce
  - Peak rate
  - Average rate
  - Burst size
- ❑ What happens when traffic profile is violated?
  - Employ traffic shaping?

# Deployment Issues

- ❑ Single administrative domain
- ❑ Incremental deployment
- ❑ Traffic policing/shaping complexity
- ❑ Charging models

# Signaling in the Internet

connectionless  
(stateless)  
forwarding by IP  
routers      +      best effort  
service      =      no network  
signaling protocols  
in initial IP  
design

- ❑ **New requirement:** reserve resources along end-to-end path (end system, routers) for QoS for multimedia applications
- ❑ **RSVP:** Resource reSerVation Protocol [RFC 2205]
  - “ ... allow users to communicate requirements to network in robust and efficient way.” i.e., signaling !
- ❑ earlier Internet Signaling protocol: ST-II [RFC 1819]

# RSVP Design Goals

1. accommodate **heterogeneous receivers** (different bandwidth along paths)
2. accommodate different applications **with different resource requirements**
3. make **multicast a first class service**, with adaptation to multicast group membership
4. **leverage existing multicast/unicast routing**, with adaptation to changes in underlying unicast, multicast routes
5. **control protocol overhead** to grow (at worst) linear in # receivers
6. **modular design** for heterogeneous underlying technologies



## RSVP: does not...

- ❑ specify how resources are to be reserved
  - ❑ rather: a mechanism for communicating needs
- ❑ determine routes packets will take
  - ❑ that's the job of routing protocols
  - ❑ signaling decoupled from routing
- ❑ interact with forwarding of packets
  - ❑ separation of control (signaling) and data (forwarding) planes

# Multimedia Networking: Summary

- ❑ multimedia applications and requirements
- ❑ making the best of today's "best effort" service
- ❑ scheduling and policing mechanisms
- ❑ next generation Internet: IntServ, RSVP, DiffServ, IPv6, IP-QoS