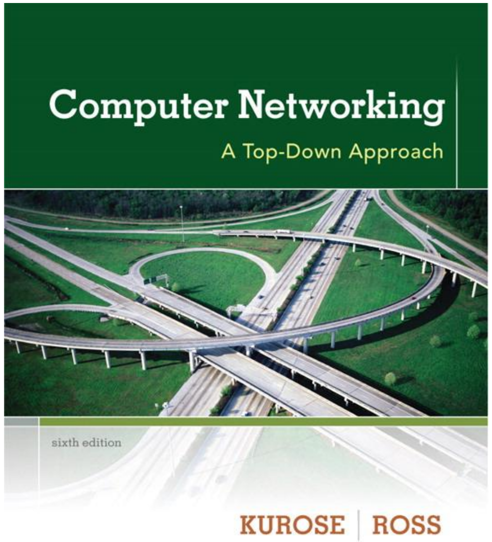
Computer Networking

谢 逸

中山大学・数据科学与计算机学院

2018. Spring

Chapter 7
Multimedia Networking



A note on the use of these ppt slides:

We're making these slides freely available to all (faculty, students, readers). They're in PowerPoint form so you see the animations; and can add, modify, and delete slides (including this one) and slide content to suit your needs. They obviously represent a lot of work on our part. In return for use, we only ask the following:

- If you use these slides (e.g., in a class) that you mention their source (after all, we'd like people to use our book!)
- If you post any slides on a www site, that you note that they are adapted from (or perhaps identical to) our slides, and note our copyright of this material.

Thanks and enjoy! JFK/KWR

Computer Networking: A Top Down Approach
6th edition
Jim Kurose, Keith Ross
Addison-Wesley
March 2012

Homework

- **1, 7, 13, 18, 20**
- 多媒体应用的类型，流式多媒体系统的类型

Multimedia networking: outline

7.1 multimedia networking applications

7.2 streaming *stored* video

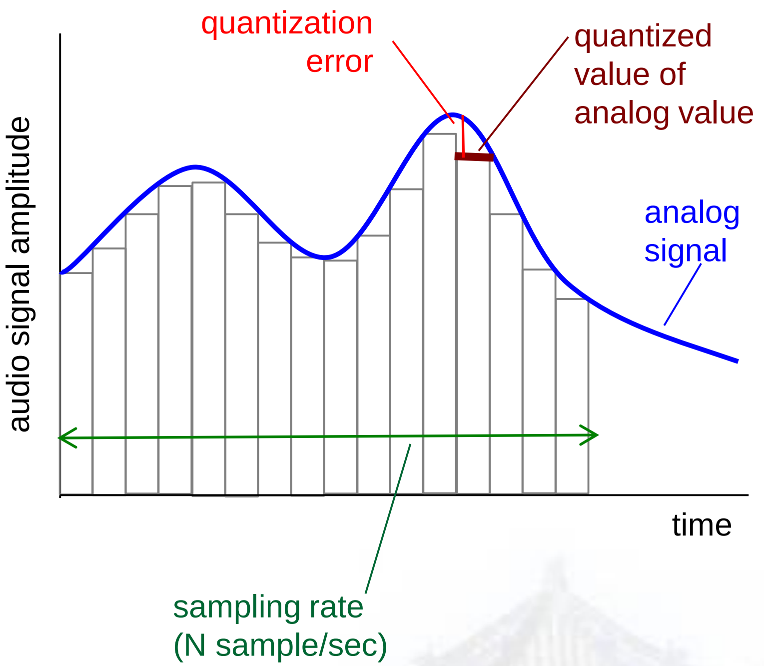
7.3 voice-over-IP

7.4 protocols for *real-time* conversational applications

7.5 network support for multimedia

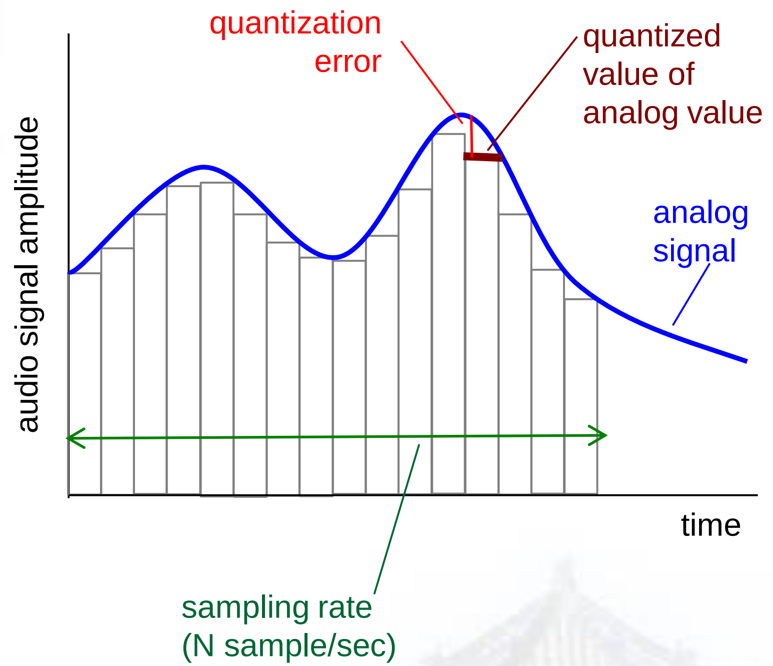
Multimedia: audio

- 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



Multimedia: audio

- example: 8,000 samples/sec, 256 quantized values: 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

Multimedia: video

- **video: sequence of images displayed at constant rate**
  - **e.g. 24 images/sec**
- **digital image: array of pixels**
  - **each pixel represented by bits**
- **coding: use redundancy *within* and *between* images to decrease # bits used to encode image**
  - **spatial (within image)**
  - **temporal (from one image to next)**

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

frame i+1

Multimedia: video

- CBR: (constant bit rate): video encoding rate fixed
- VBR: (variable bit rate): video encoding rate changes as amount of spatial, temporal coding changes
- examples:
  - MPEG 1 (CD-ROM) 1.5 Mbps
  - MPEG2 (DVD) 3-6 Mbps
  - MPEG4 (often used in Internet, < 1 Mbps)

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

frame i+1

Multimedia networking: 3 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)**
  - **e.g., YouTube, Netflix, Hulu**
- ***conversational*** **voice/video over IP**
  - **interactive nature of human-to-human conversation limits delay tolerance**
  - **e.g., Skype**
- ***streaming live*** **audio, video**
  - **e.g., live sporting event (futbol)**

Multimedia networking: outline

7.1 multimedia networking applications

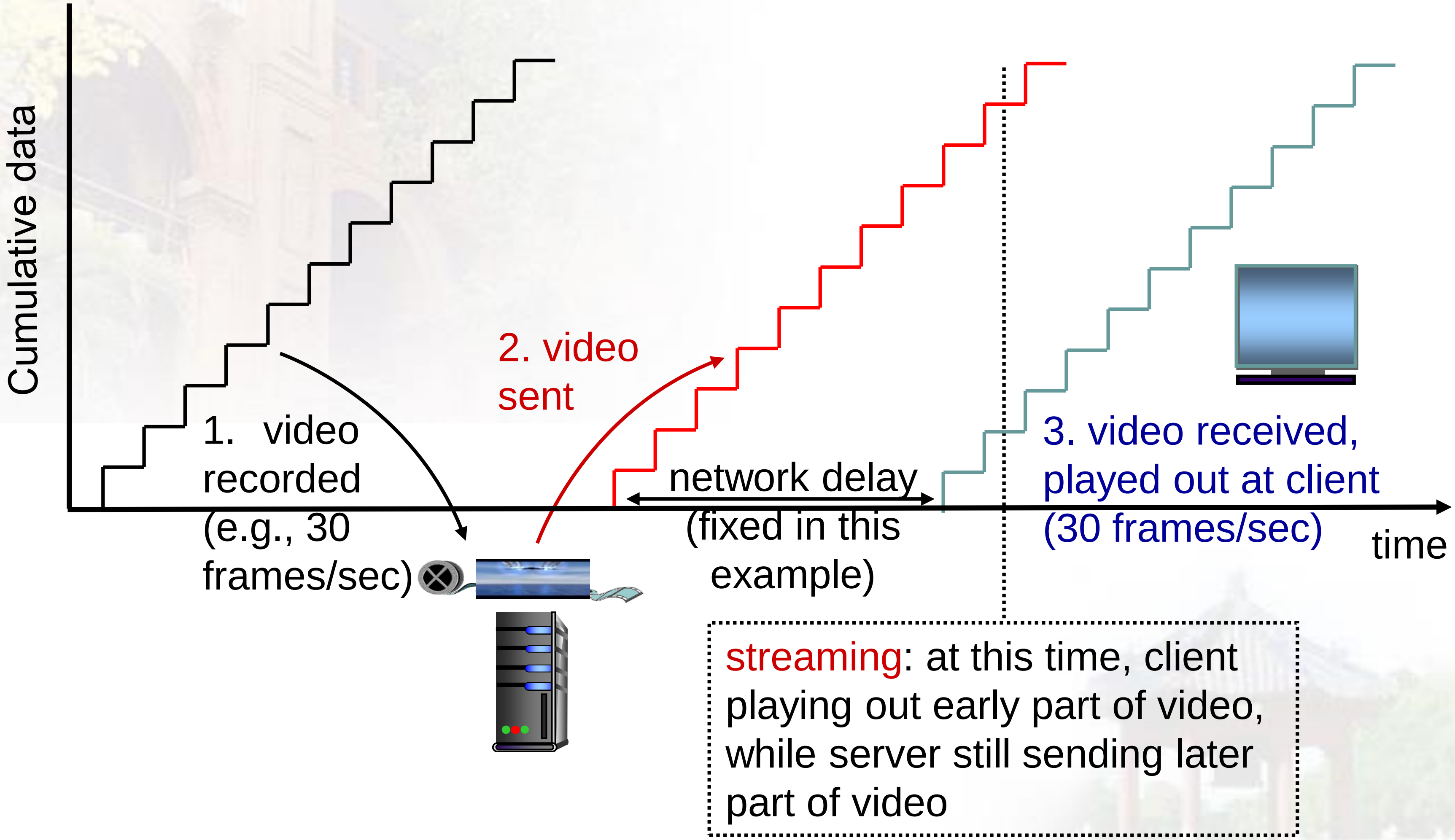
7.2 streaming *stored* video

7.3 voice-over-IP

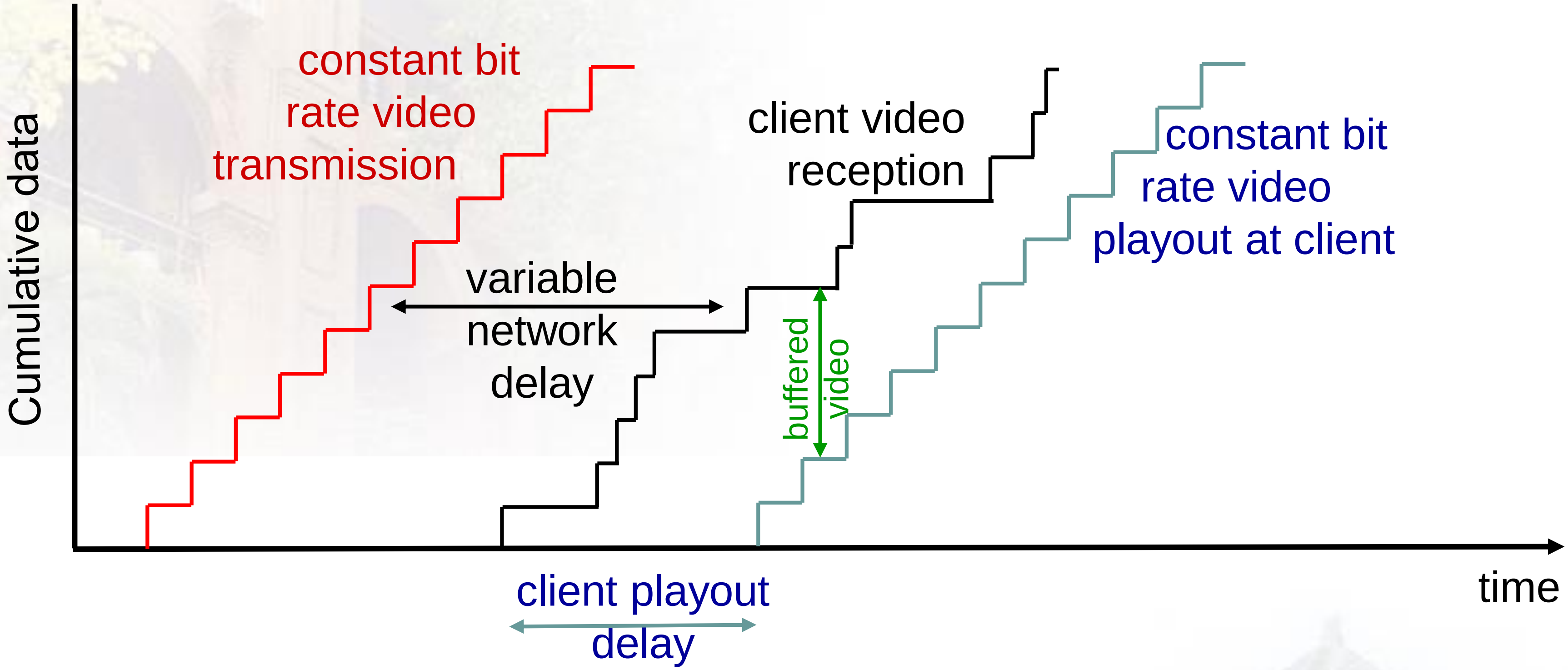
7.4 protocols for *real-time* conversational applications

7.5 network support for multimedia

Streaming stored video:

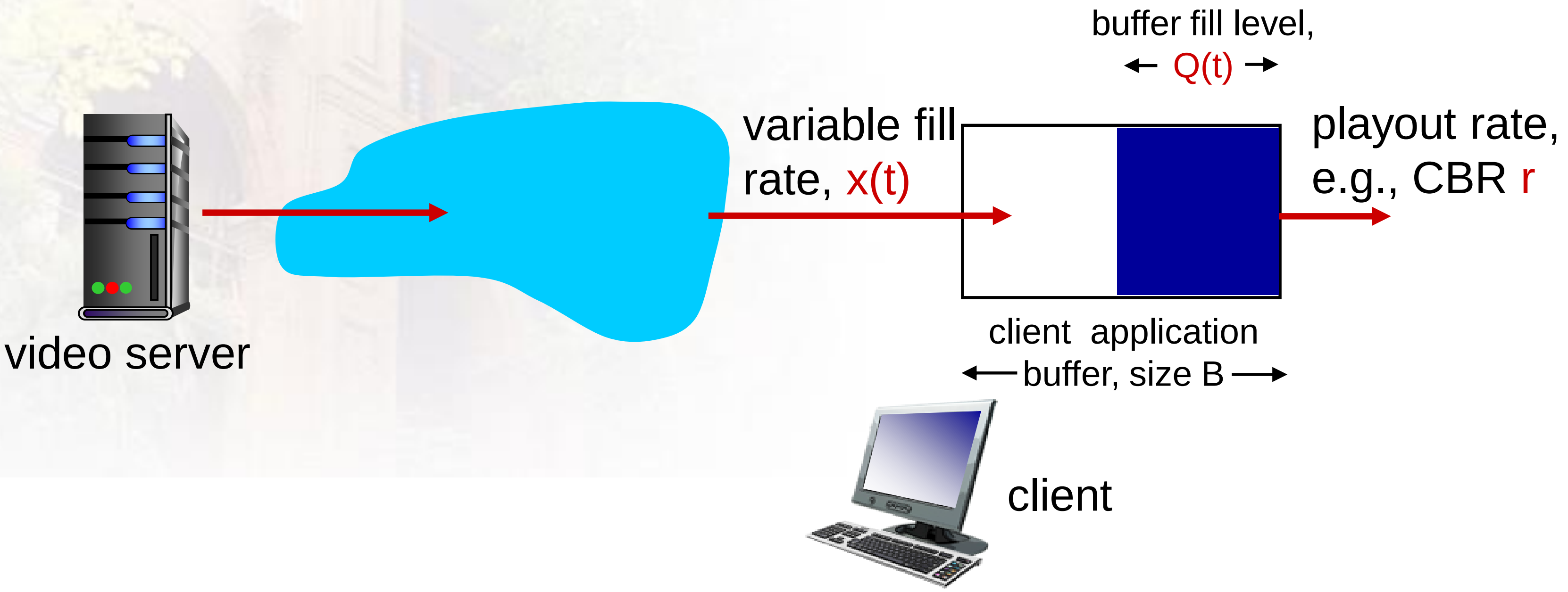


Streaming stored video: revisited

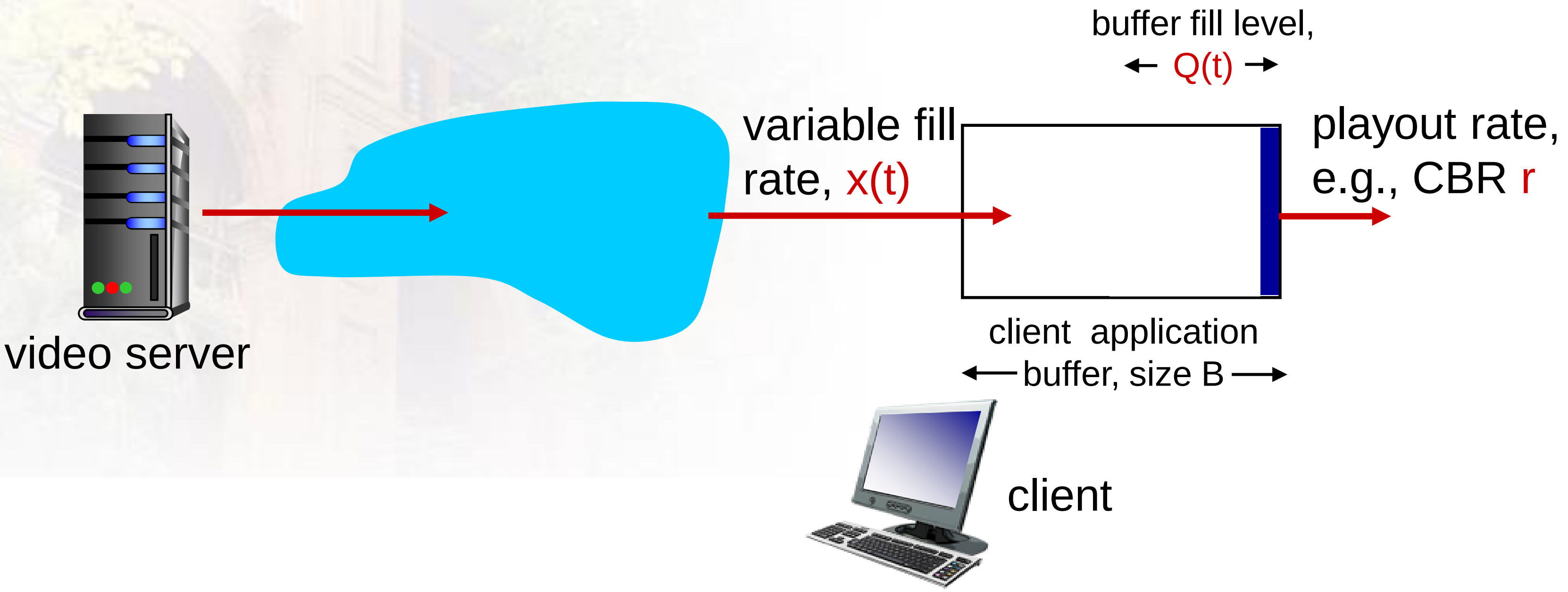


❖ ***client-side buffering and playout delay:*** **compensate for network-added delay, delay jitter**

Client-side buffering, playout

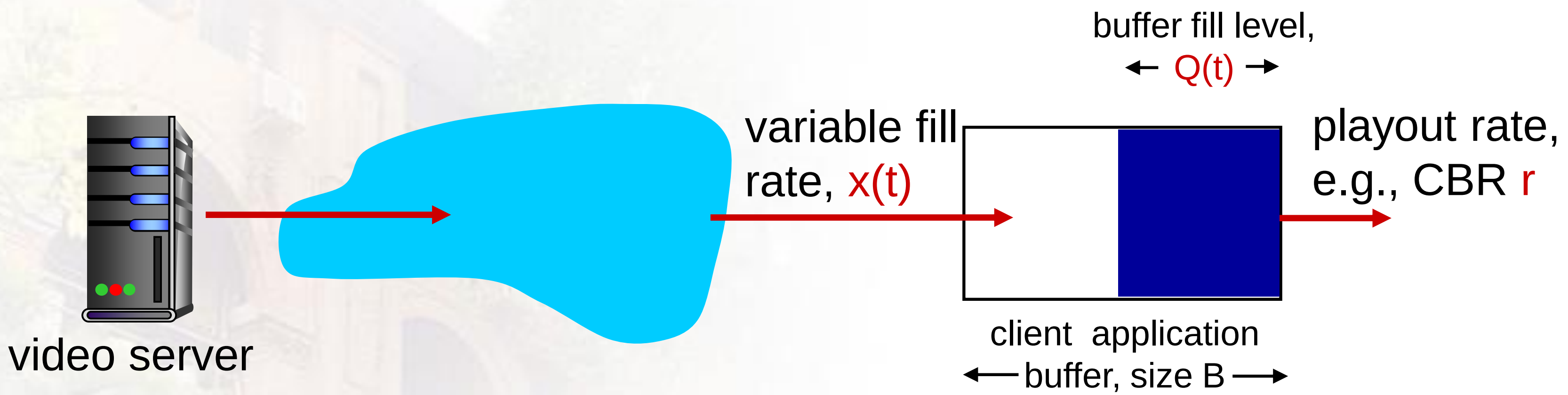


Client-side buffering, playout



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

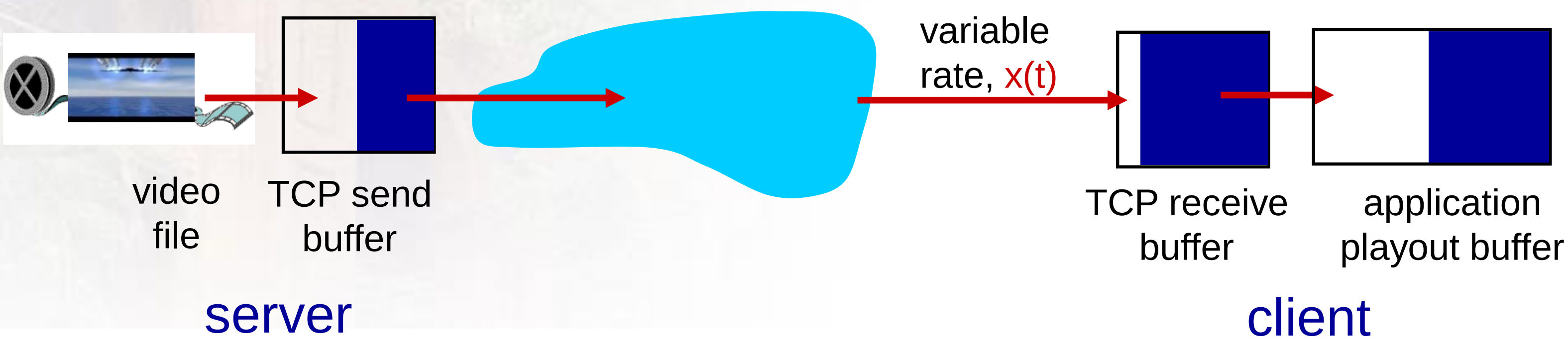


playout buffering: average fill rate (x), playout rate (r):

- **x < r: buffer eventually empties (causing freezing of video playout until buffer again fills)**
- **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**

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

Streaming multimedia: DASH

- *DASH: D*ynamic, *A*daptive *S*treaming over *H*TTP
- *server:*
  - divides 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
    - chooses maximum coding rate sustainable given current bandwidth
    - can choose different coding rates at different points in time (depending on available bandwidth at time)

Streaming multimedia: DASH

- ***DASH: Dynamic, Adaptive Streaming over HTTP***
- ***"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)**

Content distribution networks

- ***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 path to distant clients**
  - **multiple copies of video sent over outgoing link**

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

Content distribution networks

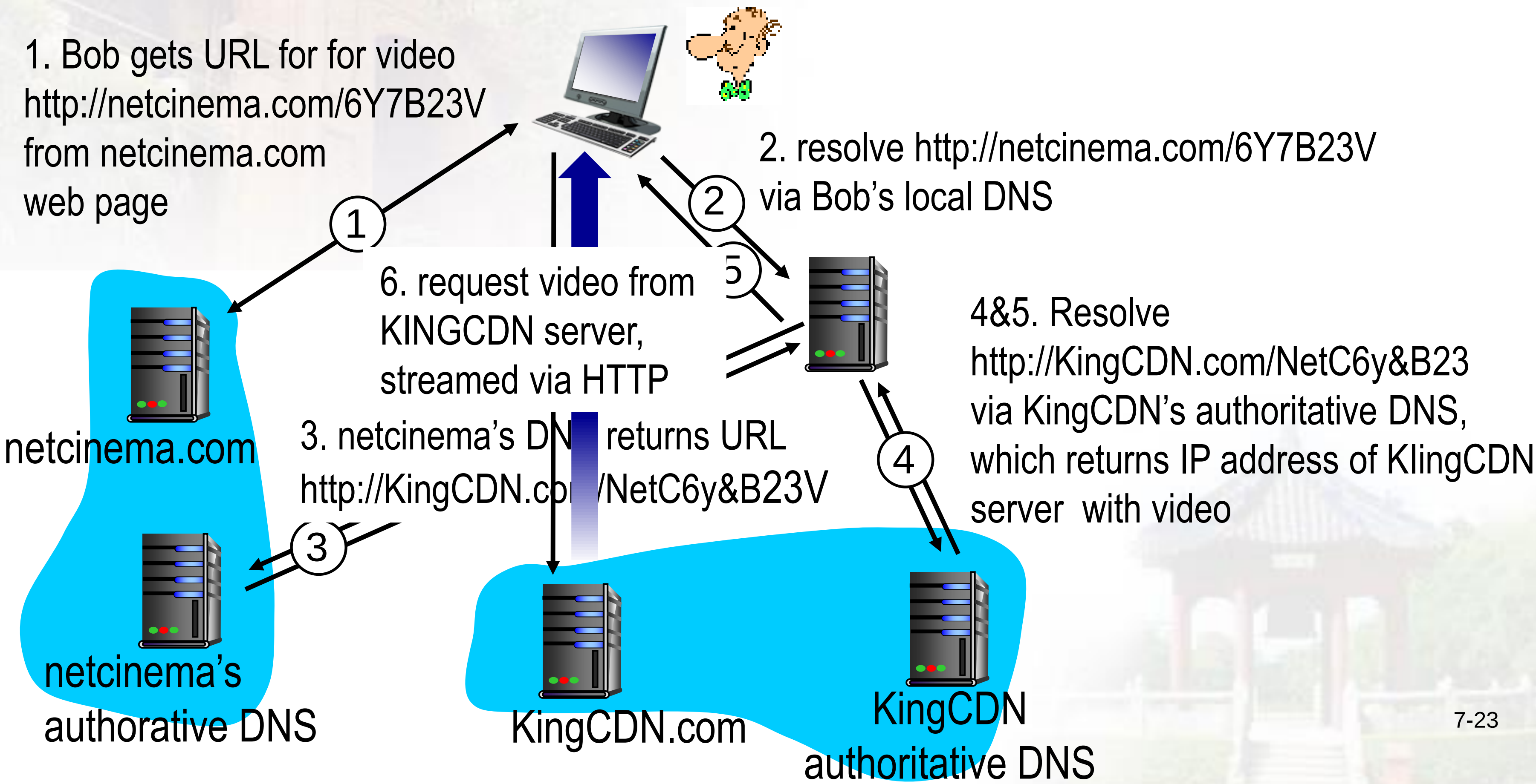
- ***challenge:*** **how to stream content (selected from millions of videos) to hundreds of thousands of simultaneous users?**

- ***option 2:*** **store/serve multiple copies of videos at multiple geographically distributed sites *(CDN)***
  - ***enter deep:*** **push CDN servers deep into many access networks**
    - **close to users**
    - **used by Akamai, 1700 locations**
  - ***bring home:*** **smaller number (10's) of larger clusters in POPs near (but not within) access networks**
    - **used by Limelight**

CDN: “simple” content access scenario

Bob (client) requests video http://netcinema.com/6Y7B23V

- video stored in CDN at http://KingCDN.com/NetC6y&B23V



CDN cluster selection strategy

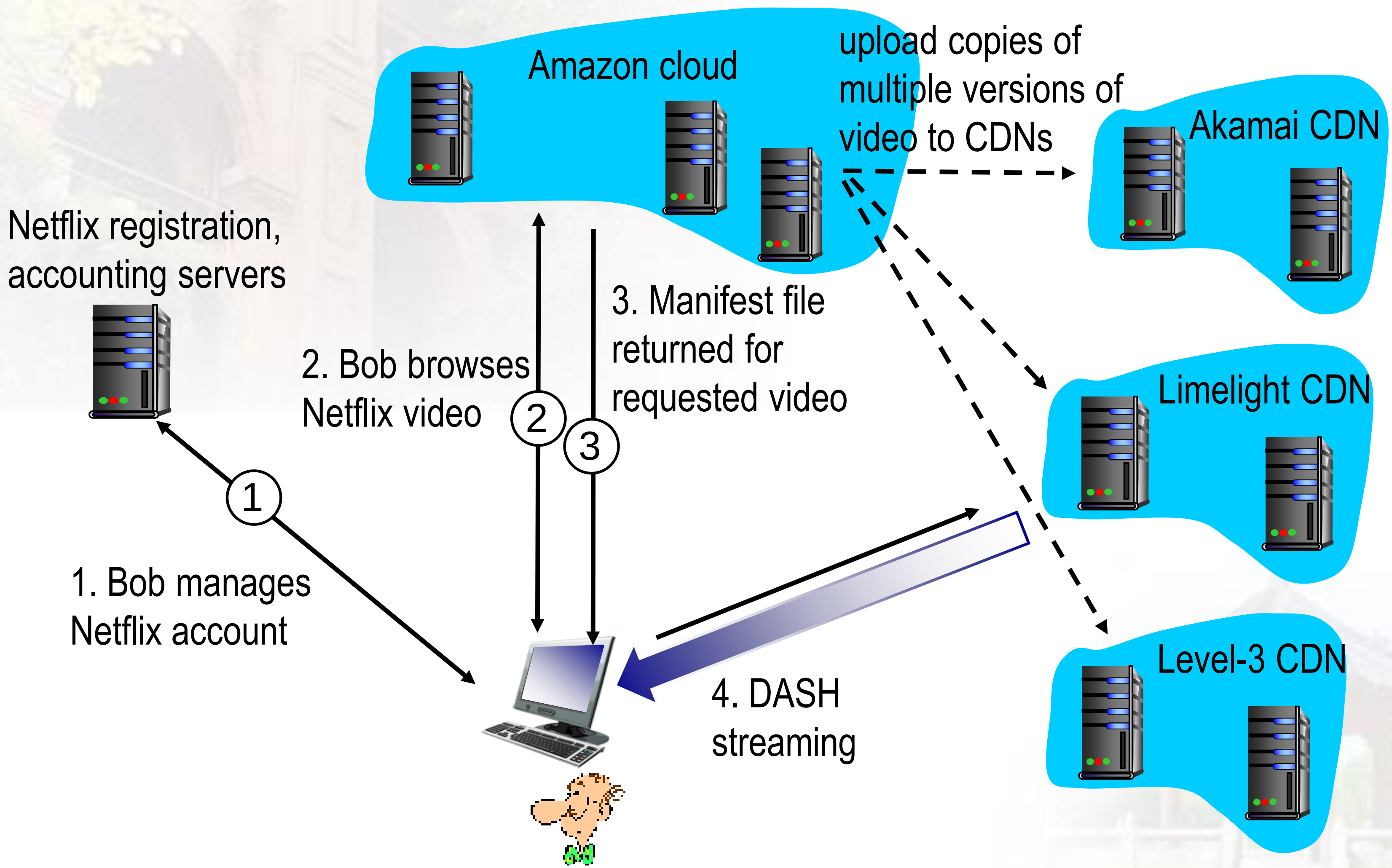
- *challenge:* how does CDN DNS select "good" CDN node to stream to client
  - pick CDN node geographically closest to client
  - pick CDN node with shortest delay (or min # hops) to client (CDN nodes periodically ping access ISPs, reporting results to CDN DNS)
  - IP anycast

- *alternative:* let *client* decide - give client a list of several CDN servers
  - client pings servers, picks "best"
  - Netflix approach

Case study: Netflix

- 30% downstream US traffic in 2011
- 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 endodings) in cloud
    - upload versions from cloud to CDNs
    - Cloud hosts Netflix web pages for user browsing
  - *three* 3rd party CDNs host/stream Netflix content: Akamai, Limelight, Level-3

Case study: Netflix



Multimedia networking: outline

7.1 multimedia networking applications

7.2 streaming *stored* video

7.3 voice-over-IP

7.4 protocols for *real-time* conversational applications

7.5 network support for multimedia

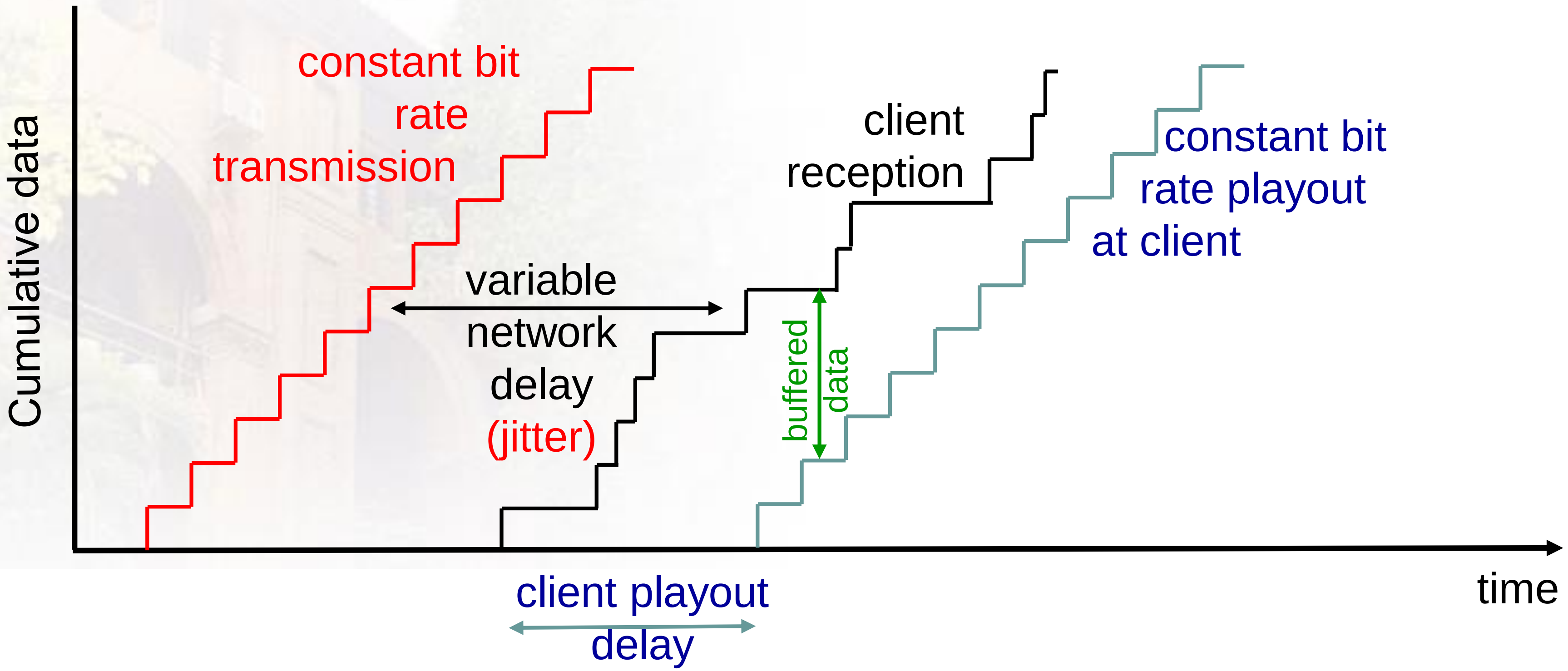
Voice-over-IP (VoIP)

- *VoIP end-end-delay requirement*: needed to maintain "conversational" aspect
  - higher delays noticeable, impair interactivity
  - < 150 msec: good
  - > 400 msec bad
  - includes application-level (packetization,playout), network delays
- *session initialization:* how does callee advertise IP address, port number, encoding algorithms?
- *value-added services:* call forwarding, screening, recording
- *emergency services:* 911

VoIP: packet loss, delay

- ***network loss:*** **IP datagram lost due to network congestion (router buffer overflow)**
- ***delay loss:*** **IP datagram arrives too late for playout at receiver**
  - **delays: processing, queueing in network; end-system (sender, receiver) delays**
  - **typical maximum tolerable delay: 400 ms**
- ***loss tolerance:*** **depending on voice encoding, loss concealment, packet loss rates between 1% and 10% can be tolerated**

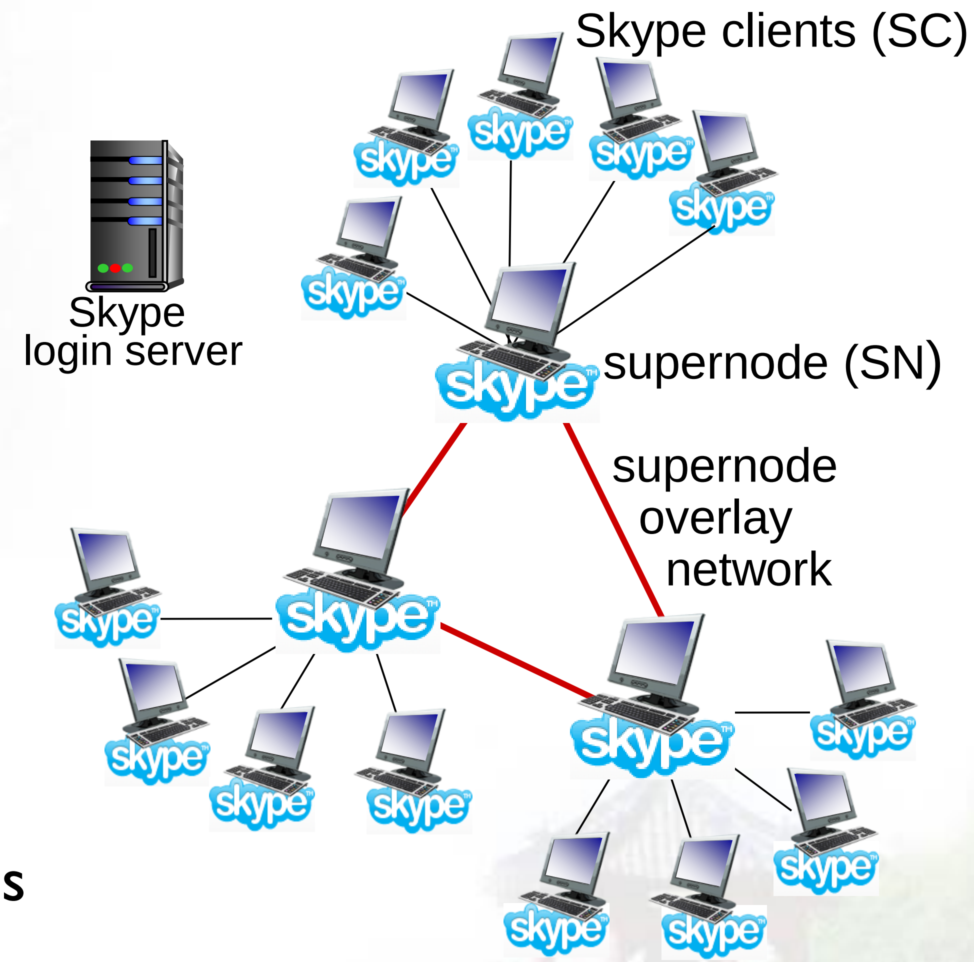
Delay jitter



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

Voice-over-IP: Skype

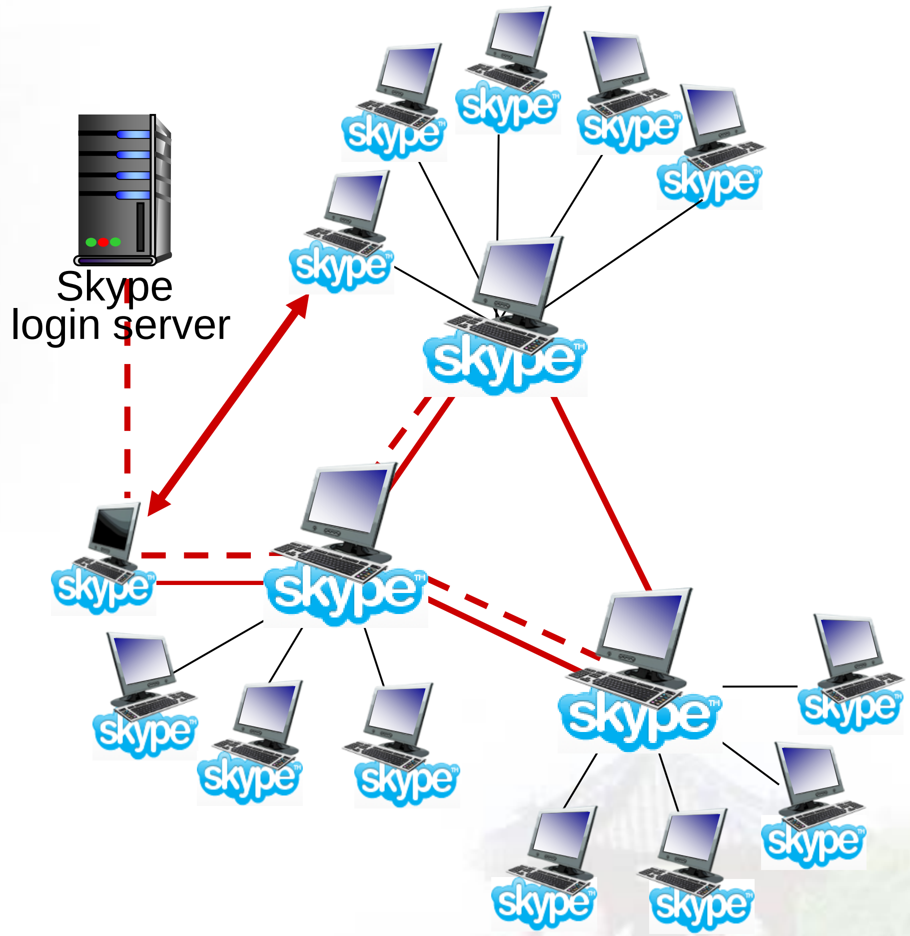
- **proprietary application-layer protocol (inferred via reverse engineering)**
  - **encrypted msgs**
- **P2P components:**
  - clients: skype peers connect directly to each other for VoIP call
  - super nodes (SN): skype peers with special functions
  - overlay network: among SNs to locate SCs
    - login server



P2P voice-over-IP: skype

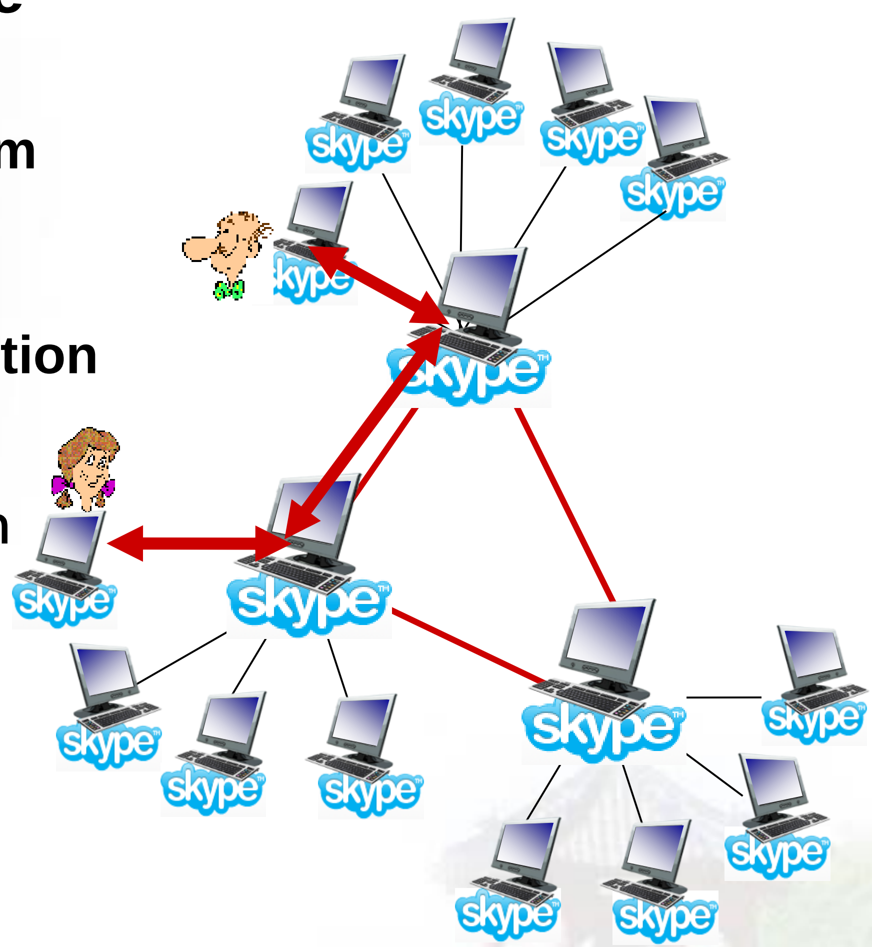
skype client operation:

1. joins skype network by contacting SN (IP address cached) using TCP
2. logs-in (usename, password) to centralized skype login server
3. obtains IP address for callee from SN, SN overlay
   - or client buddy list
4. initiate call directly to callee



Skype: peers as relays

- ***problem:*** **both Alice, Bob are behind "NATs"**
  - **NAT prevents outside peer from initiating connection to insider peer**
  - **inside peer *can* initiate connection to outside**



- relay solution: Alice, Bob maintain open connection to their SNs
  - Alice signals her SN to connect to Bob
  - Alice's SN connects to Bob's SN
  - Bob's SN connects to Bob over open connection Bob initially initiated to his SN

Multimedia networking: outline

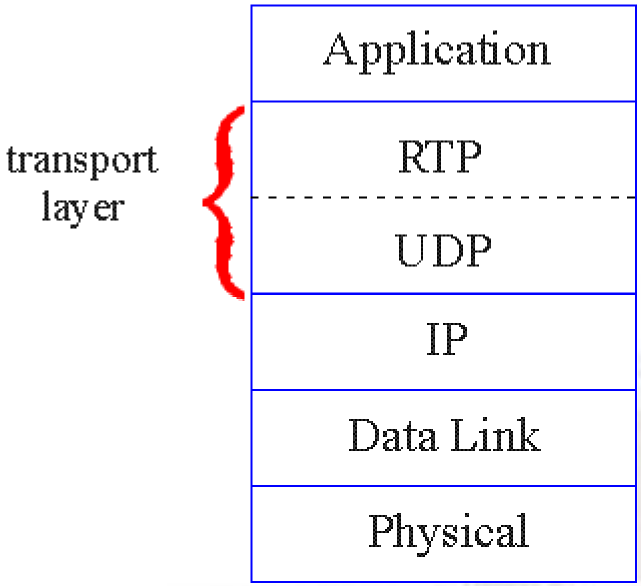
Real-Time Protocol (RTP)

- RTP specifies packet structure for packets carrying audio, video data
- RFC 3550
- RTP packet provides
  - payload type identification
  - packet sequence numbering
  - time stamping
- RTP runs in end systems
- RTP packets encapsulated in UDP segments
- interoperability: if two VoIP applications run RTP, they may be able to work together

RTP runs on top of UDP

RTP libraries provide transport-layer interface that extends UDP:

- port numbers, IP addresses
- payload type identification
- packet sequence numbering
- time-stamping



RTP example

example: sending 64 kbps PCM-encoded voice over RTP

- application collects encoded data in chunks, e.g., every 20 msec = 160 bytes in a chunk
- audio chunk + RTP header form RTP packet, which is encapsulated in UDP segment

- RTP header indicates type of audio encoding in each packet
  - sender can change encoding during conference
- RTP header also contains sequence numbers, timestamps

RTP and QoS

- RTP does *not* provide any mechanism to ensure timely data delivery or other QoS guarantees
- RTP encapsulation only seen at end systems (*not* by intermediate routers)
  - routers provide best-effort service, making no special effort to ensure that RTP packets arrive at destination in timely matter

SIP: Session Initiation Protocol [RFC 3261]

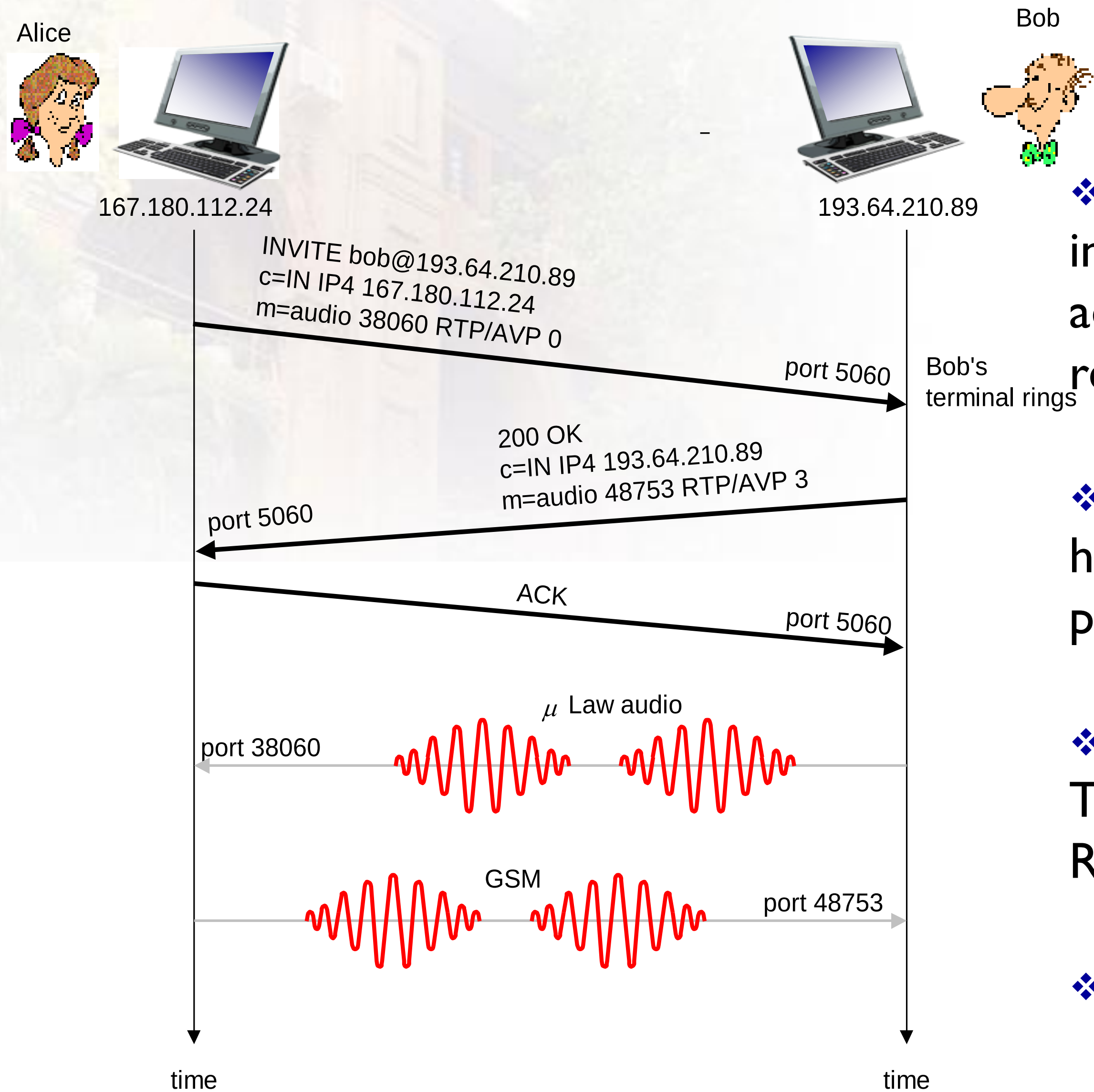
long-term vision:

- **all telephone calls, video conference calls take place over Internet**
- **people identified by names or e-mail addresses, rather than by phone numbers**
- **can reach callee *(if callee so desires),* no matter where callee roams, no matter what IP device callee is currently using**

SIP services

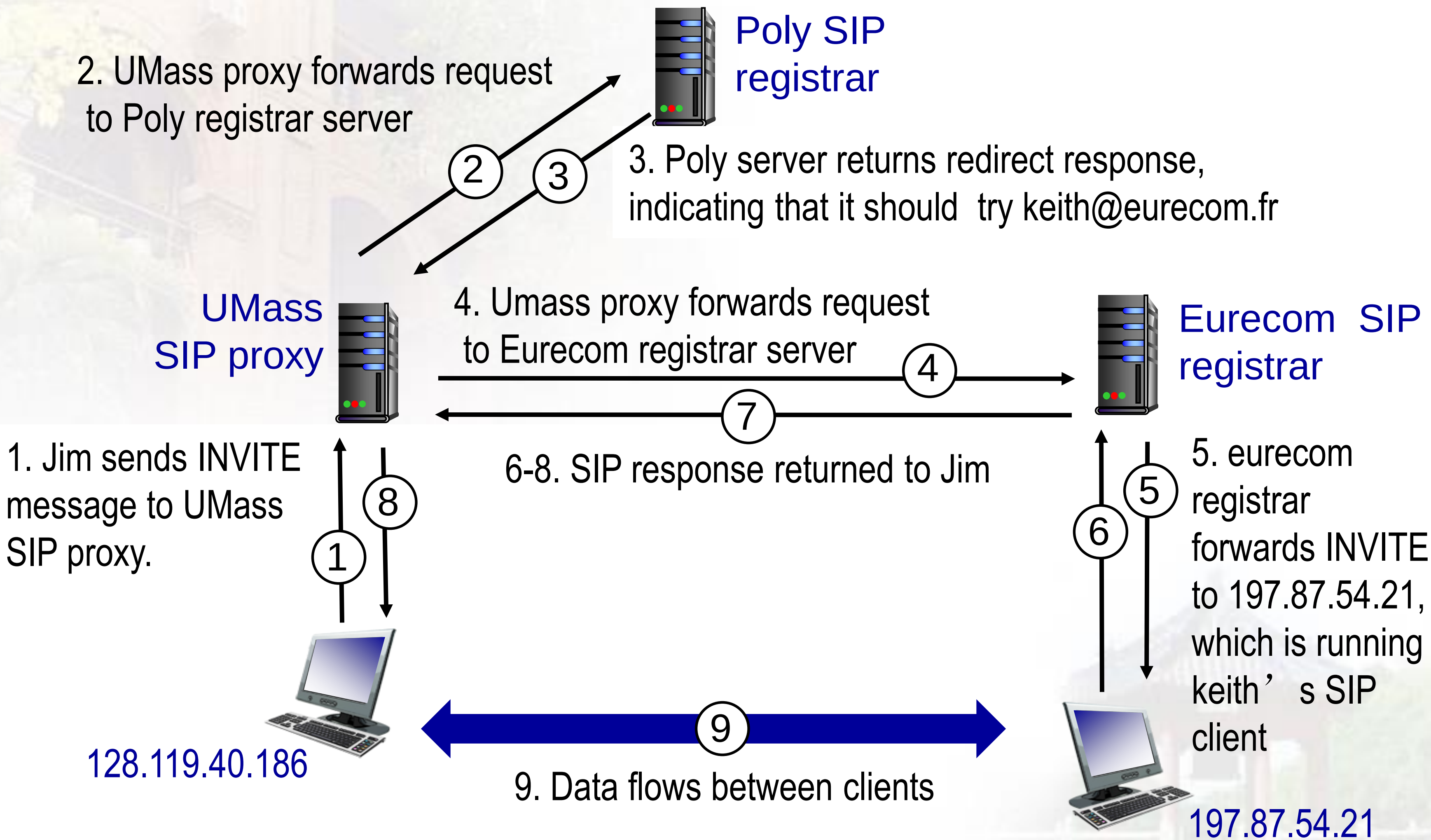
- SIP provides mechanisms for call setup:
  - for caller to let callee know she wants to establish a call
  - so caller, callee can agree on media type, encoding
  - to end call
- determine current IP address of callee:
  - maps mnemonic identifier to current IP address
- call management:
  - add new media streams during call
  - change encoding during call
  - invite others
  - transfer, hold calls

Example: setting up call to known IP address



- Alice's SIP invite message indicates her port number, IP address, encoding she prefers to receive (PCM μlaw)
- Bob's 200 OK message indicates his port number, IP address, preferred encoding (GSM)
- SIP messages can be sent over TCP or UDP; here sent over RTP/UDP
- default SIP port number is 5060

SIP example: jim@umass.edu calls keith@poly.edu



Multimedia networking: outline

7.1 multimedia networking applications

7.2 streaming stored video

7.3 voice-over-IP

7.4 protocols for real-time conversational applications

7.5 network support for multimedia

Network support for multimedia

Approach	Granularity	Guarantee	Mechanisms	Complex	Deployed?
Making best of best effort service	All traffic treated equally	None or soft	No network support (all at application)	low	everywhere
Differentiated service	Traffic "class"	None of soft	Packet market, scheduling, policing.	med	some
Per-connection QoS	Per-connection flow	Soft or hard after flow admitted	Packet market, scheduling, policing, call admission	high	little to none

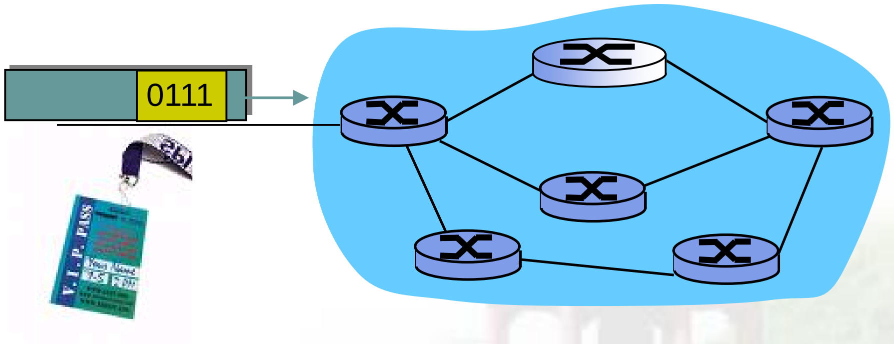
Dimensioning best effort networks

- *approach:* deploy enough link capacity so that congestion doesn't occur, multimedia traffic flows without delay or loss
  - low complexity of network mechanisms (use current "best effort" network)
  - high bandwidth costs
- challenges:
  - *network dimensioning:* how much bandwidth is "enough?"
  - *estimating network traffic demand:* needed to determine how much bandwidth is "enough" (for that much traffic)

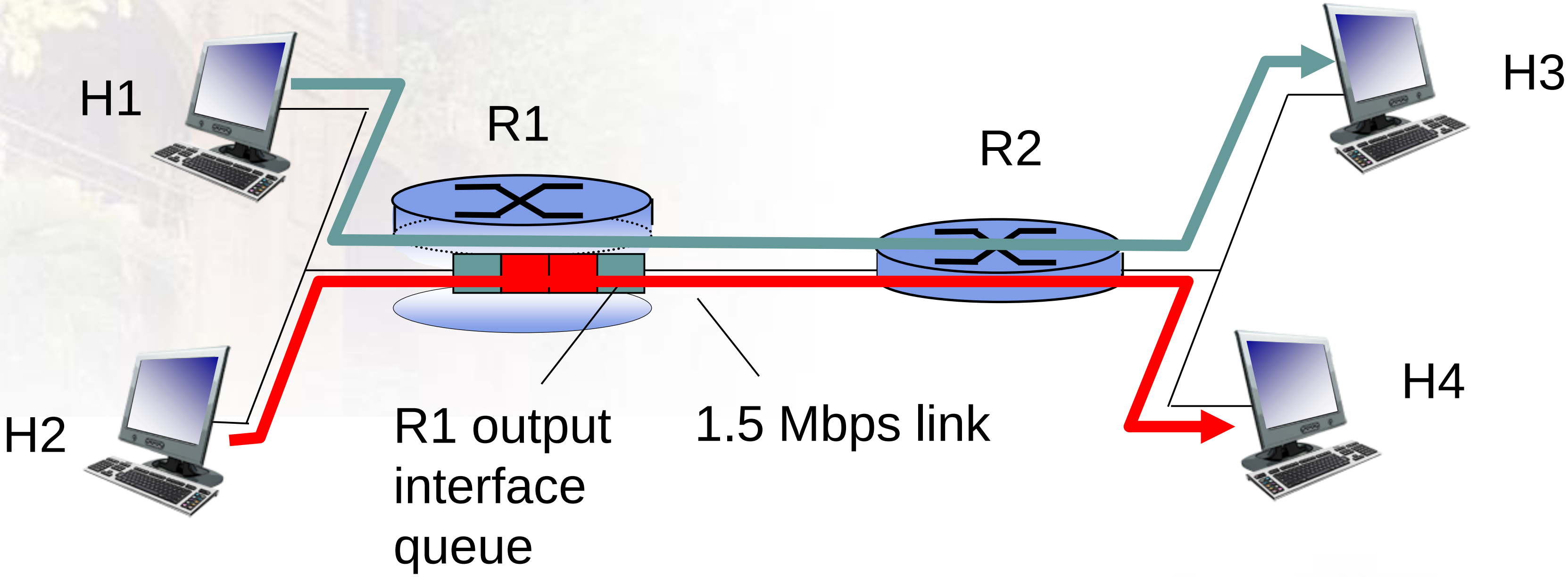
Providing multiple classes of service

- **thus far: making the best of best effort service**
  - **one-size fits all service model**
- **alternative: multiple classes of service**
  - **partition traffic into classes**
  - **network treats different classes of traffic differently (analogy: VIP service versus regular service)**

- granularity: differential service among multiple classes, not among individual connections
- history: ToS bits

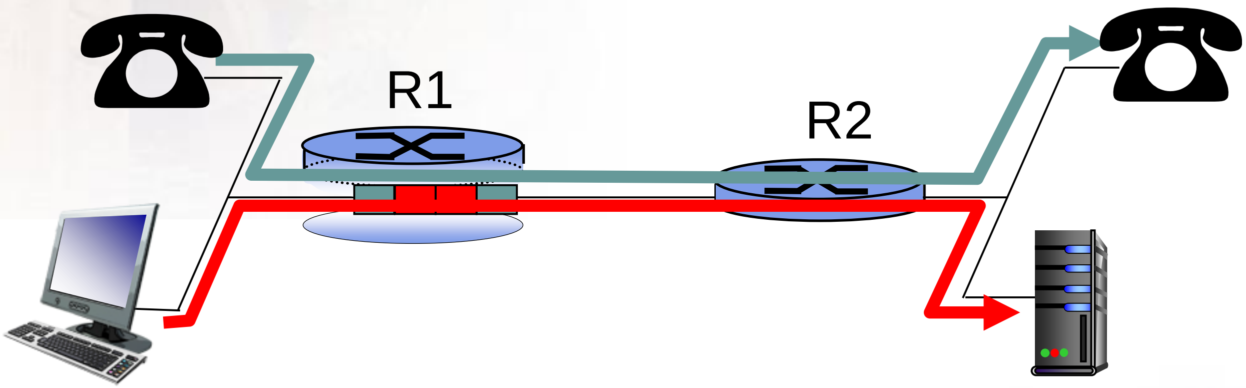


Multiple classes of service: scenario



Scenario 1: mixed HTTP and VoIP

- example: 1Mbps VoIP, HTTP share 1.5 Mbps link.
  - HTTP bursts can congest router, cause audio loss
  - want to give priority to audio over HTTP

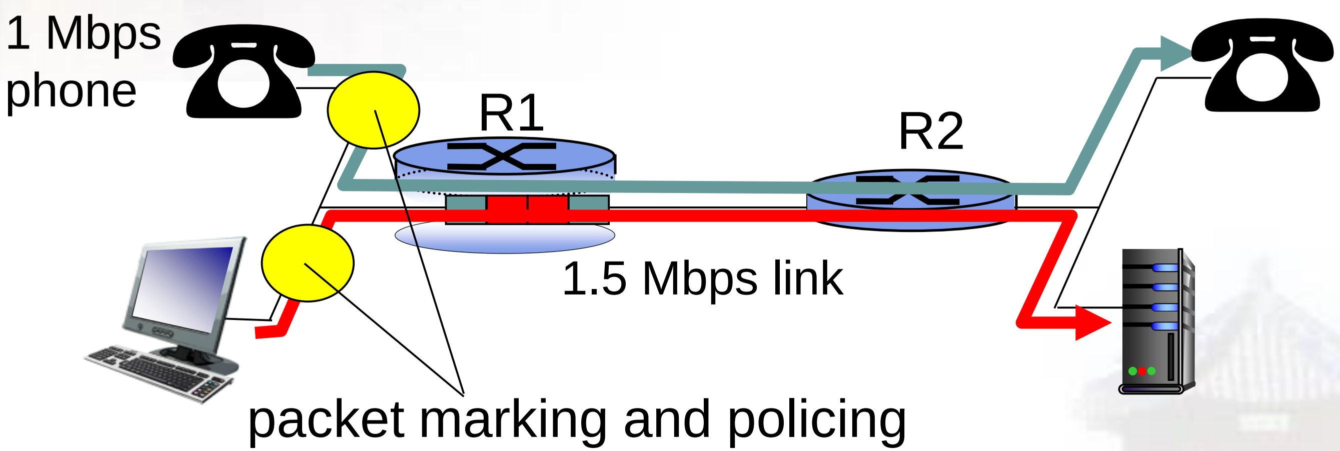


Principle 1

packet marking needed for router to distinguish between different classes; and new router policy to treat packets accordingly

Principles for QOS guarantees (more)

- what if applications misbehave (VoIP sends higher than declared rate)
  - policing: force source adherence to bandwidth allocations
- ***marking***, ***policing*** **at network edge**

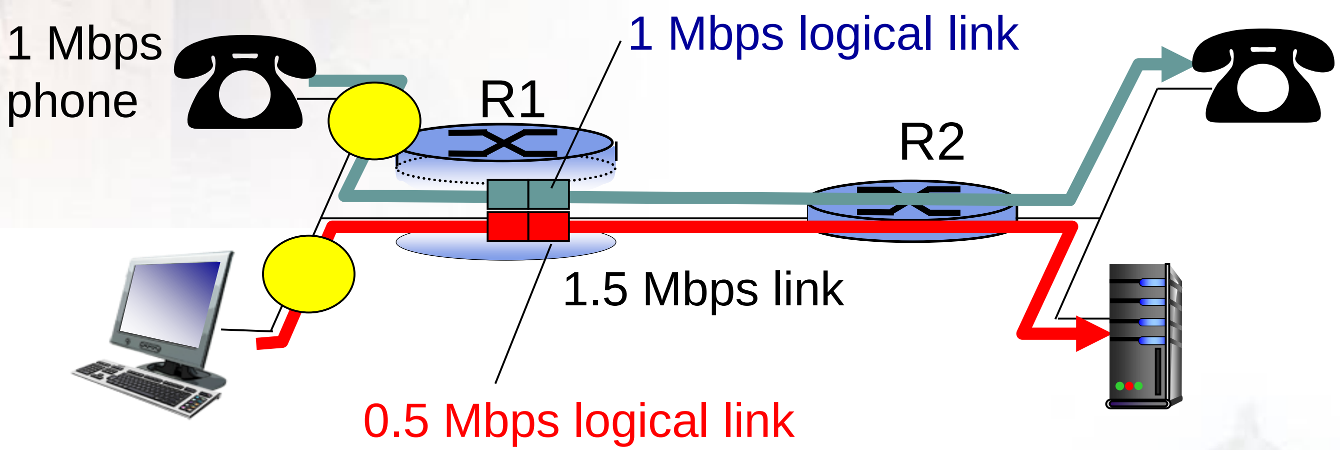


Principle 2

provide protection (isolation) for one class from others

Principles for QOS guarantees (more)

- allocating *fixed* (non-sharable) bandwidth to flow: *inefficient* use of bandwidth if flows doesn't use its allocation

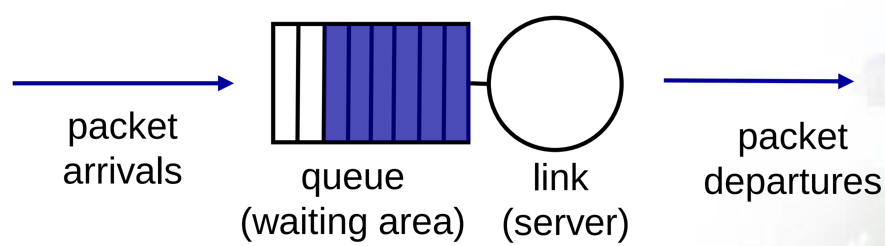


Principle 3

while providing isolation, it is desirable to use resources as efficiently as possible

Scheduling and policing mechanisms

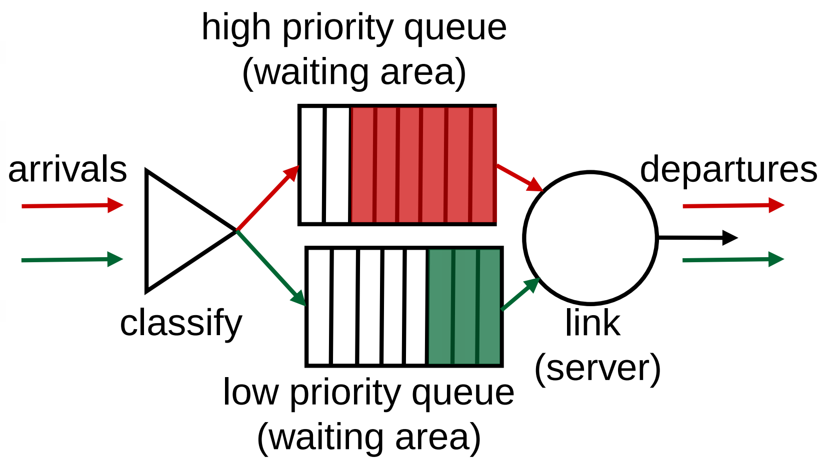
- ***scheduling:*** **choose next packet to send on link**
- ***FIFO (first in first out) scheduling:*** **send in order of arrival to queue**
  - **real-world example?**
  - ***discard policy:*** **if packet arrives to full queue: who to discard?**
    - ***tail drop:*** **drop arriving packet**
    - ***priority:*** **drop/remove on priority basis**
    - ***random:*** **drop/remove randomly**

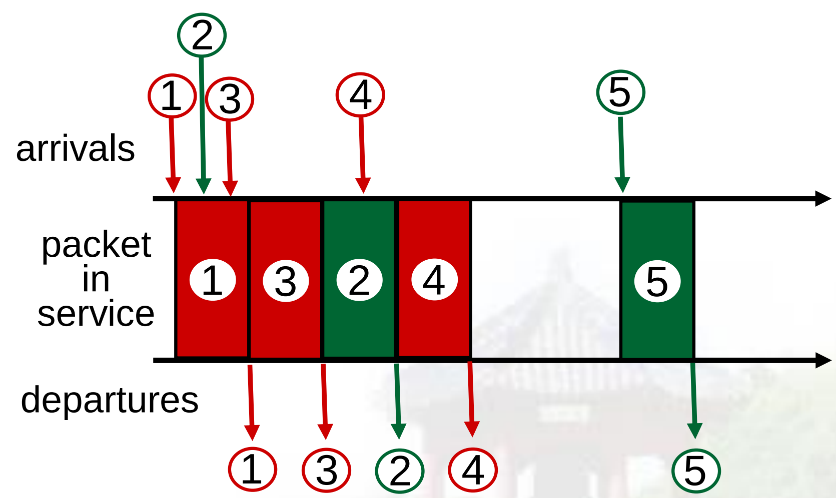


Scheduling policies: priority

priority scheduling: send highest priority queued packet

- multiple *classes*, with different priorities
  - class may depend on marking or other header info, e.g. IP source/dest, port numbers, etc.
  - real world example?

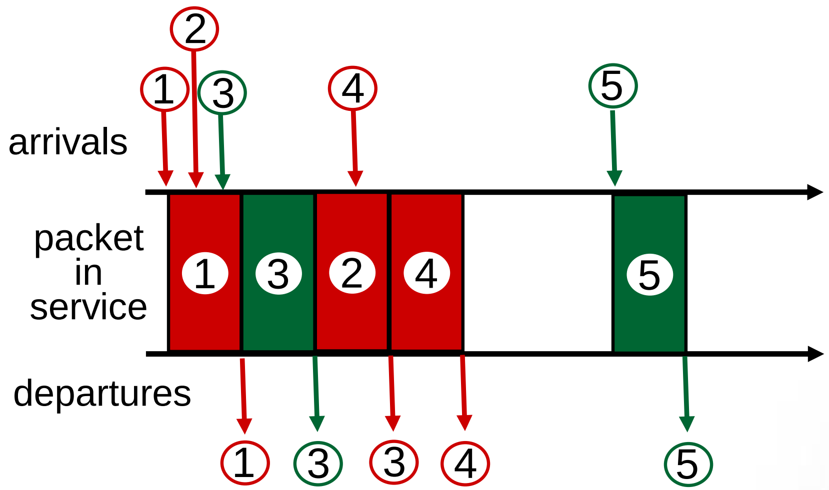




Scheduling policies: still more

Round Robin (RR) scheduling:

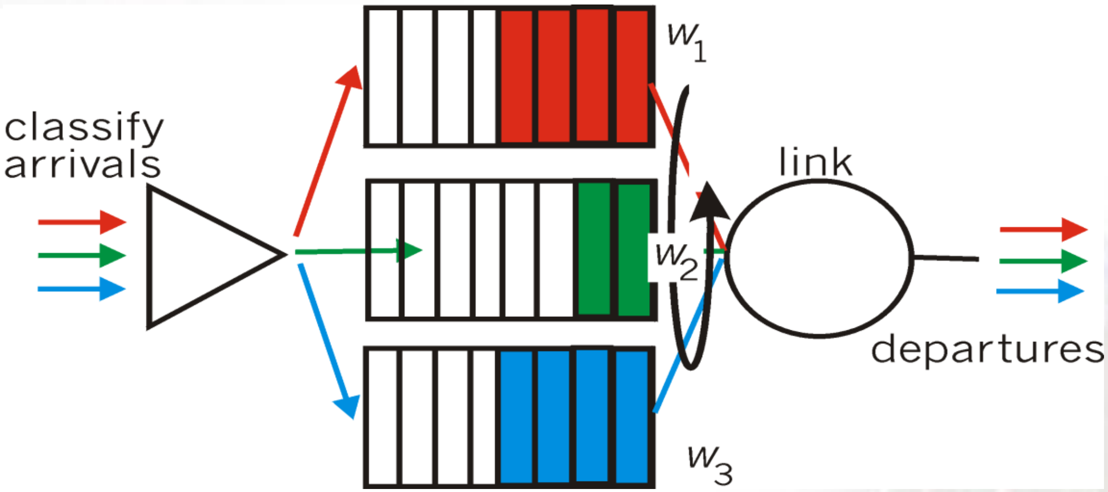
- multiple classes
- cyclically scan class queues, sending one complete packet from each class (if available)
- real world example?



Scheduling policies: still more

Weighted Fair Queuing (WFQ):

- **generalized Round Robin**
- **each class gets weighted amount of service in each cycle**
- **real-world example?**



Policing mechanisms

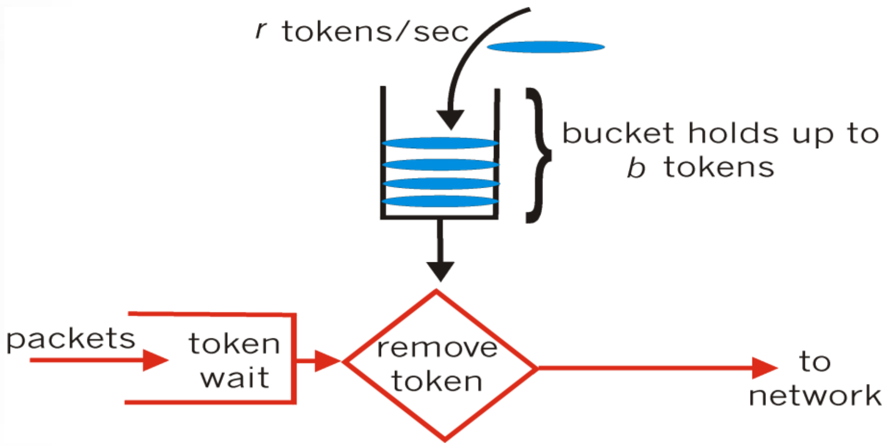
goal: **limit traffic to not exceed declared parameters**

Three common-used criteria:

- ***(long term) average rate:*** **how many pkts can be sent per unit time (in the long run)**
  - **crucial question: what is the interval length: 100 packets per sec or 6000 packets per min have same average!**
- ***peak rate:*** **e.g., 6000 pkts per min (ppm) avg.; 1500 ppm peak rate**
- ***(max.) burst size:*** **max number of pkts sent consecutively (with no intervening idle)**

Policing mechanisms: implementation

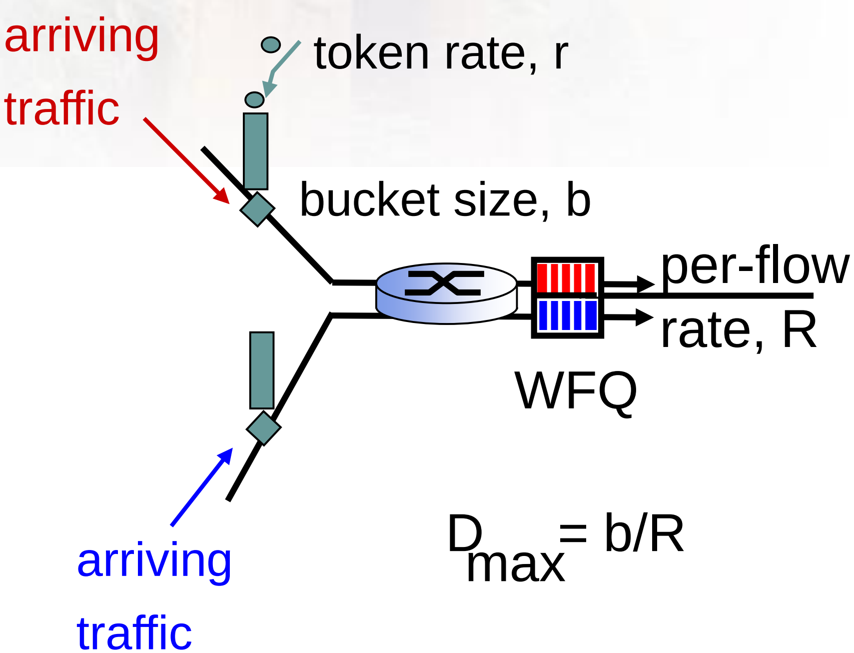
token bucket: limit input to specified ***burst size*** and ***average rate***



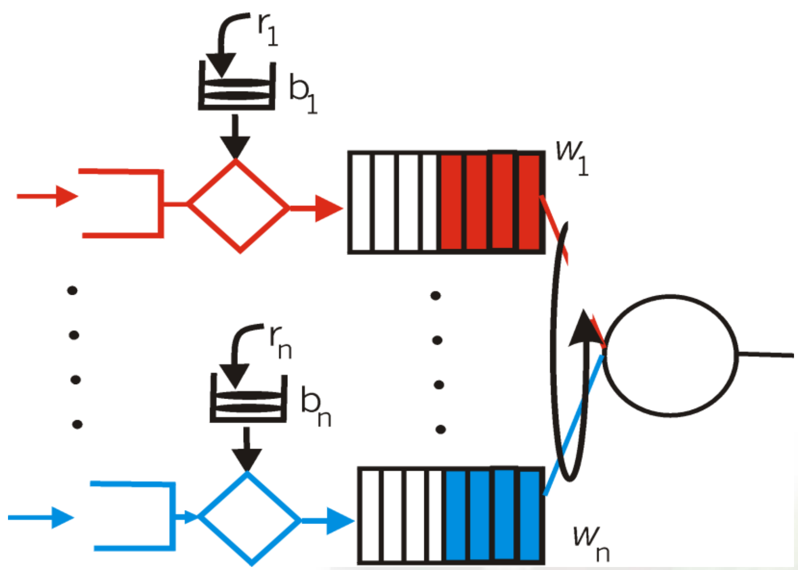
- bucket can hold b tokens
- tokens generated at rate *r token/sec* unless bucket full
- *over interval of length t: number of packets admitted less than or equal to (r t + b)*

Policing and QoS guarantees

- token bucket, WFQ combine to provide guaranteed upper bound on delay, i.e., ***QoS guarantee!***



$$D_{max} = b/R$$



Differentiated services

- want “qualitative” service classes
  - “behaves like a wire”
  - relative service distinction: Platinum, Gold, Silver
- *scalability:* simple functions in network core, relatively complex functions at edge routers (or hosts)
  - signaling, maintaining per-flow router state difficult with large number of flows
- don’t define service classes, provide functional components to build service classes

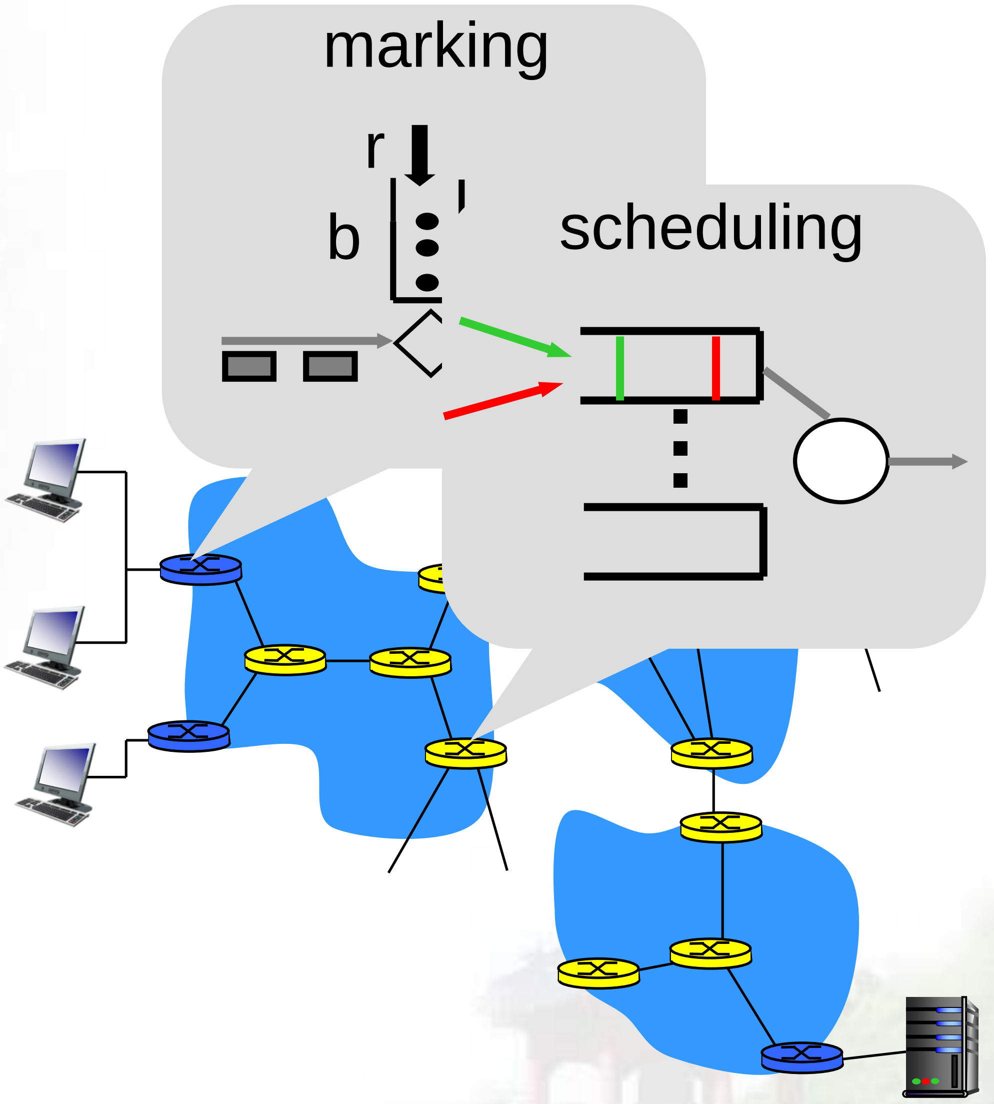
Diffserv architecture

edge router:

- per-flow traffic management
- marks packets as in-profile and out-profile

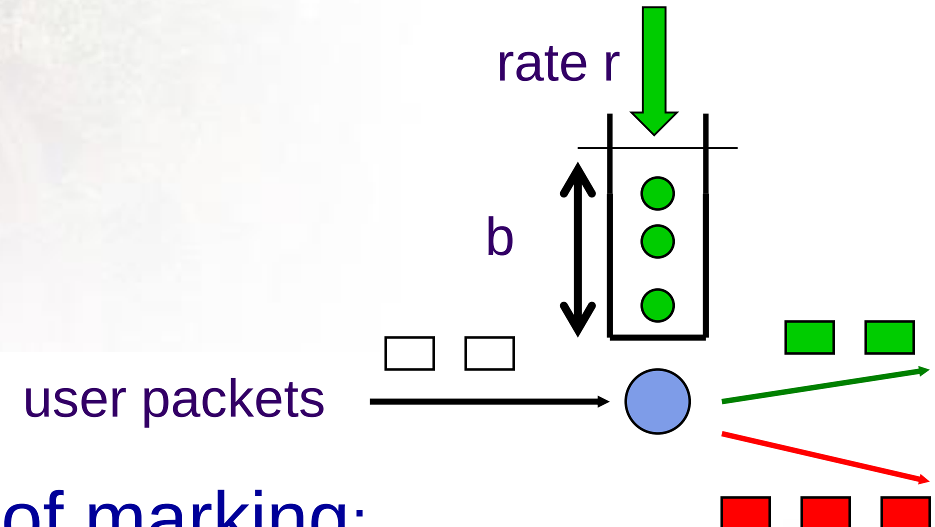
core router:

- per class traffic management
- buffering and scheduling based on marking at edge
- preference given to in-profile packets over out-of-profile packets



Edge-router packet marking

- profile: pre-negotiated rate r, bucket size b
- packet marking at edge based on per-flow profile

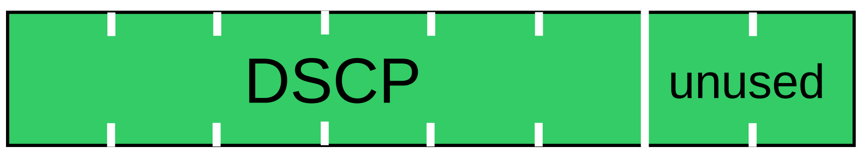


possible use of marking:

- class-based marking: packets of different classes marked differently
- intra-class marking: conforming portion of flow marked differently than non-conforming one

Diffserv packet marking: details

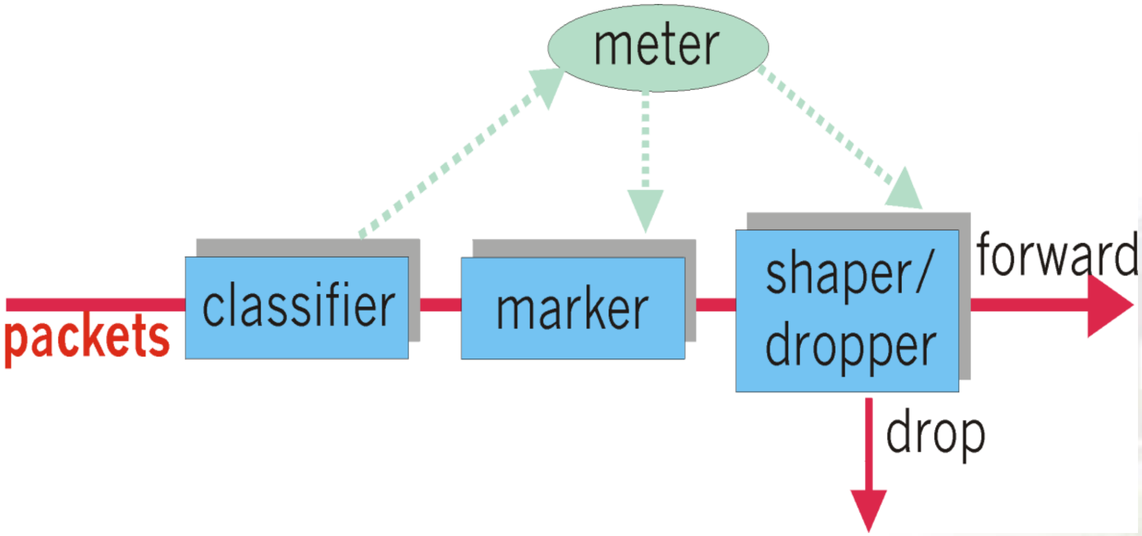
- packet is marked in the Type of Service (TOS) in IPv4, and Traffic Class in IPv6
- 6 bits used for Differentiated Service Code Point (DSCP)
  - determine PHB that the packet will receive
  - 2 bits currently unused



Classification, conditioning

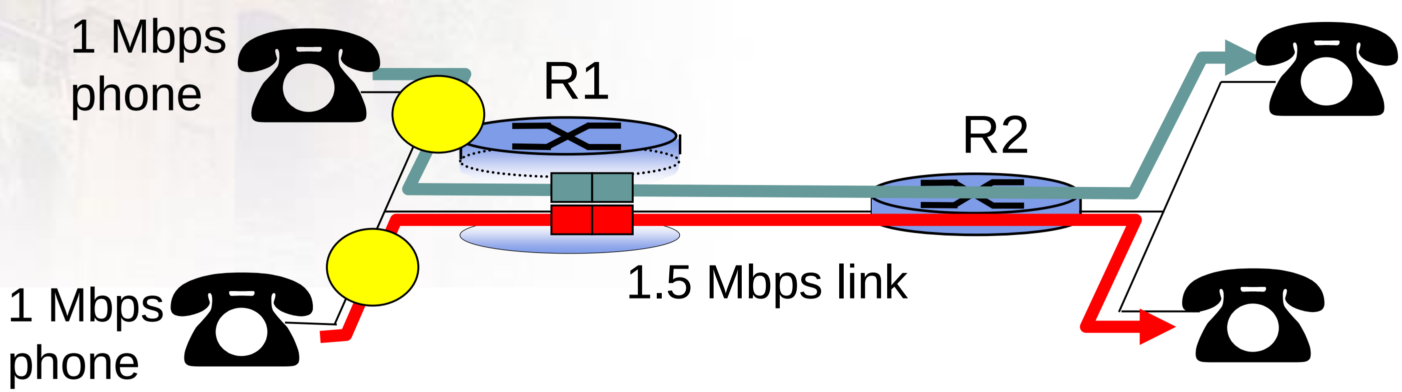
may be desirable to limit traffic injection rate of some class:

- user declares traffic profile (e.g., rate, burst size)
- traffic metered, shaped if non-conforming



Per-connection QOS guarantees

- ***basic fact of life:*** **can not support traffic demands beyond link capacity**

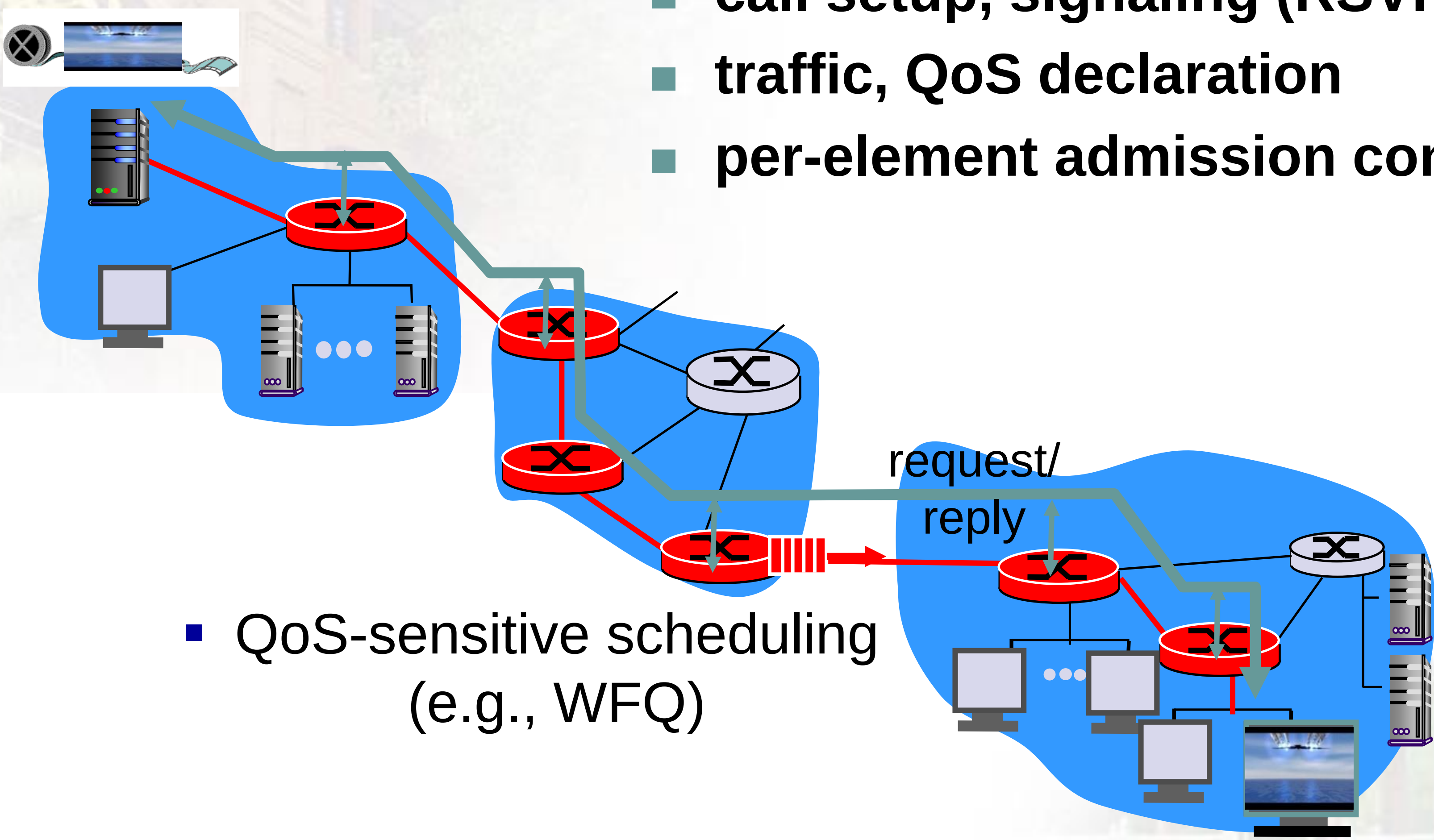


Principle 4

call admission: flow declares its needs, network may block call (e.g., busy signal) if it cannot meet needs

QoS guarantee scenario

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



- QoS-sensitive scheduling (e.g., WFQ)

The End of Chapter 7