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: RTP, SIP
7.5 network support for multimedia

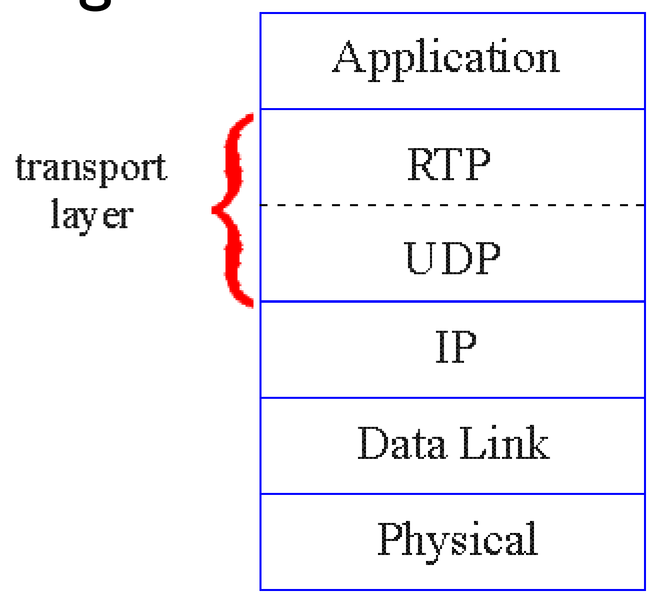
Real-Time Protocol (RTP)

- 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

RTP header

payload type	sequence number	time stamp	Synchronization Source ID	Miscellaneous fields

payload type (7 bits): indicates type of encoding currently being used. If sender changes encoding during call, 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 # (16 bits): increment by one for each RTP packet sent

- detect packet loss, restore packet sequence

RTP header

payload type	sequence number	time stamp	Synchronization Source ID	Miscellaneous fields

- *timestamp field (32 bits long):* sampling instant of first byte in this RTP data packet
  - for audio, timestamp clock increments by one for each sampling period (e.g., each 125 usecs for 8 KHz sampling clock)
  - if application generates chunks of 160 encoded samples, timestamp increases by 160 for each RTP packet when source is active. Timestamp clock continues to increase at constant rate when source is inactive.
- *SSRC field (32 bits long):* identifies source of RTP stream. Each stream in RTP session has distinct SSRC

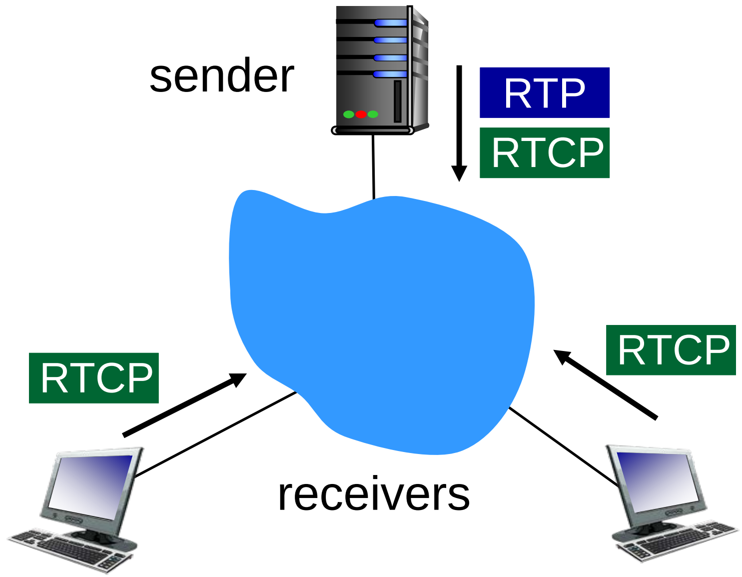
RTSP/RTP programming assignment

- build a server that encapsulates stored video frames into RTP packets
  - grab video frame, add RTP headers, create UDP segments, send segments to UDP socket
  - include seq numbers and time stamps
  - client RTP provided for you
- also write client side of RTSP
  - issue play/pause commands
  - server RTSP provided for you

Real-Time Control Protocol (RTCP)

- works in conjunction with RTP
- each participant in RTP session periodically sends RTCP control packets to all other participants
- each RTCP packet contains sender and/or receiver reports
  - report statistics useful to application: # packets sent, # packets lost, interarrival jitter
- feedback used to control performance
  - sender may modify its transmissions based on feedback

RTCP: multiple multicast senders



- each RTP session: typically a single multicast address; all RTP /RTCP packets belonging to session use multicast address
- RTP, RTCP packets distinguished from each other via distinct port numbers
- to limit traffic, each participant reduces RTCP traffic as number of conference participants increases

RTCP: packet types

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

source description packets:

- e-mail address of sender, sender's name, SSRC of associated RTP stream
- provide mapping between the SSRC and the user/host name

RTCP: stream synchronization

- RTCP can synchronize different media streams within a RTP session
- e.g., videoconferencing app: each sender generates one RTP stream for video, one for audio.
- timestamps in RTP packets tied to the video, audio sampling clocks
  - *not* tied to wall-clock time
- each RTCP sender-report packet contains (for most recently generated packet in associated RTP stream):
  - timestamp of RTP packet
  - wall-clock time for when packet was created
- receivers uses association to synchronize playout of audio, video

RTCP: bandwidth scaling

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

example : one sender, sending video at 2 Mbps

- RTCP attempts to limit RTCP traffic to 100 Kbps
- RTCP gives 75% of rate to receivers; remaining 25% to sender

- 75 kbps is equally shared among receivers:
  - with R receivers, each receiver gets to send RTCP traffic at 75/R kbps.
- sender gets to send RTCP traffic at 25 kbps.
- participant determines RTCP packet transmission period by calculating avg RTCP packet size (across entire session) and dividing by allocated rate

SIP: Session Initiation Protocol [RFC 3261]

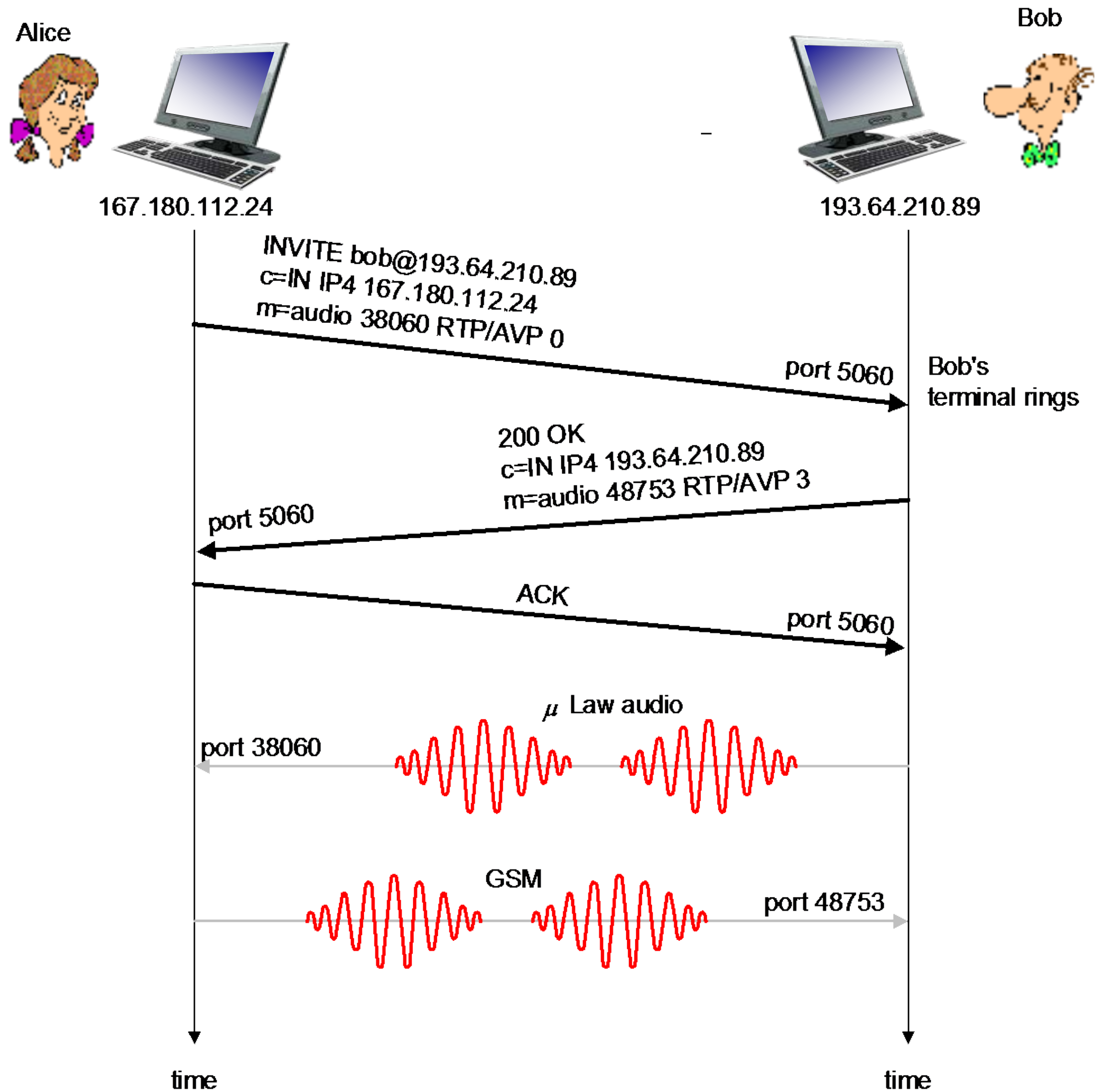
long-term vision:

- 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（助记的）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

Setting up a call (more)

- codec negotiation:
  - suppose Bob doesn't have PCM μlaw encoder
  - Bob will instead reply with 606 Not Acceptable Reply, listing his encoders. Alice can then send new INVITE message, advertising different encoder
- rejecting a call
  - Bob can reject with replies "busy," "gone," "payment required," "forbidden"
- media can be sent over RTP or some other protocol

Example of SIP message

```
INVITE sip:bob@domain.com SIP/2.0
Via: SIP/2.0/UDP 167.180.112.24
From: sip:alice@hereway.com
To: sip:bob@domain.com
Call-ID: a2e3a@pigeon.hereway.com
Content-Type: application/sdp
Content-Length: 885

c=IN IP4 167.180.112.24
m=audio 38060 RTP/AVP 0
```

Notes:

- HTTP message syntax
- sdp = session description protocol
- Call-ID is unique for every call

- Here we don't know Bob's IP address
  - intermediate SIP servers needed
- Alice sends, receives SIP messages using SIP default port 5060
- Alice specifies in header that SIP client sends, receives SIP messages over UDP

Name translation, user location

- caller wants to call callee, but only has callee's name or e-mail address.
- need to get IP address of callee's current host:
  - user moves around
  - DHCP protocol
  - user has different IP devices (PC, smartphone, car device)
- result can be based on:
  - time of day (work, home)
  - caller (don't want boss to call you at home)
  - status of callee (calls sent to voicemail when callee is already talking to someone)

SIP registrar

- one function of SIP server: *registrar*
- when Bob starts SIP client, client sends SIP REGISTER message to Bob's registrar server

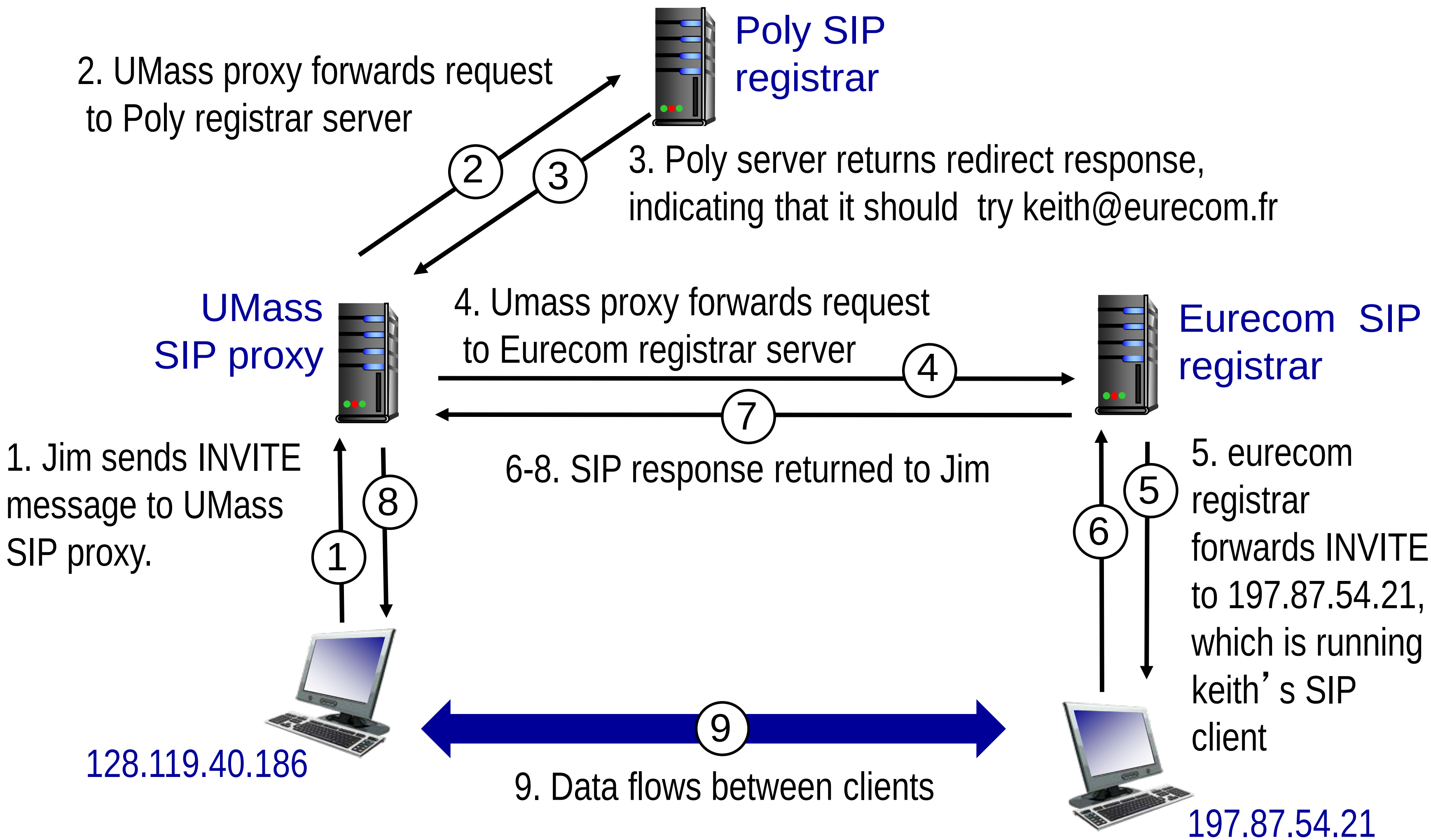
register message:

```
REGISTER sip:domain.com SIP/2.0
Via: SIP/2.0/UDP 193.64.210.89
From: sip:bob@domain.com
To: sip:bob@domain.com
Expires: 3600
```

SIP proxy

- another function of SIP server: *proxy*
- Alice sends invite message to her proxy server
  - contains address sip:bob@domain.com
  - proxy responsible for routing SIP messages to callee, possibly through multiple proxies
- Bob sends response back through same set of SIP proxies
- proxy returns Bob's SIP response message to Alice
  - contains Bob's IP address
- SIP proxy analogous to local DNS server plus TCP setup

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



Comparison with H.323

- H.323: another signaling protocol for real-time, interactive multimedia
- H.323: complete, vertically integrated suite of protocols for multimedia conferencing: signaling, registration, admission control, transport, codecs
- SIP: single component. Works with RTP, but does not mandate it. Can be combined with other protocols, services
- H.323 comes from the ITU (telephony)
- SIP comes from IETF: borrows much of its concepts from HTTP
  - SIP has Web flavor; H.323 has telephony flavor
- SIP uses KISS principle: Keep It Simple Stupid

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

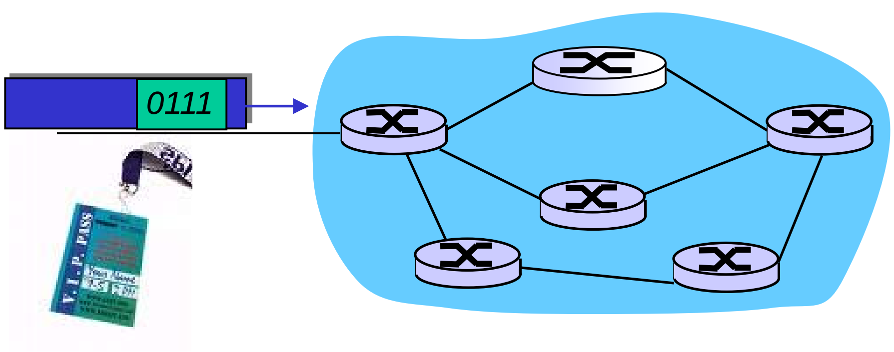
Dimensioning best effort networks

- *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)

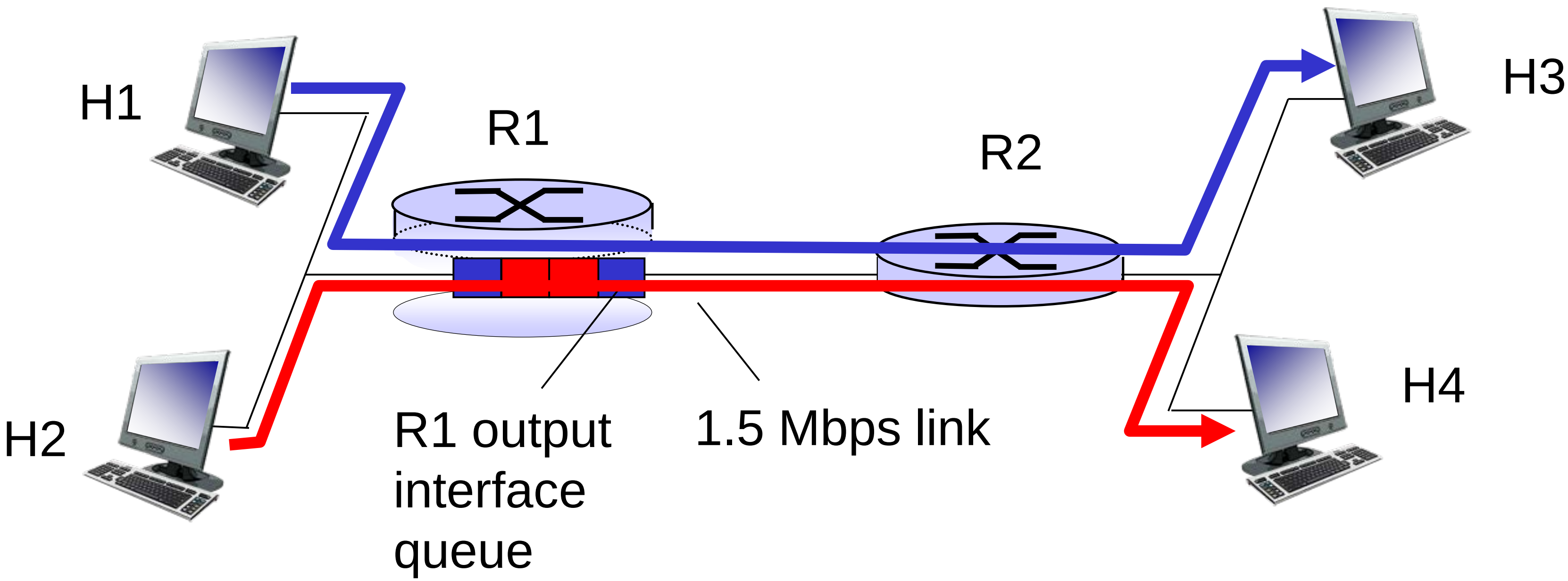
Providing multiple classes of service

- 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

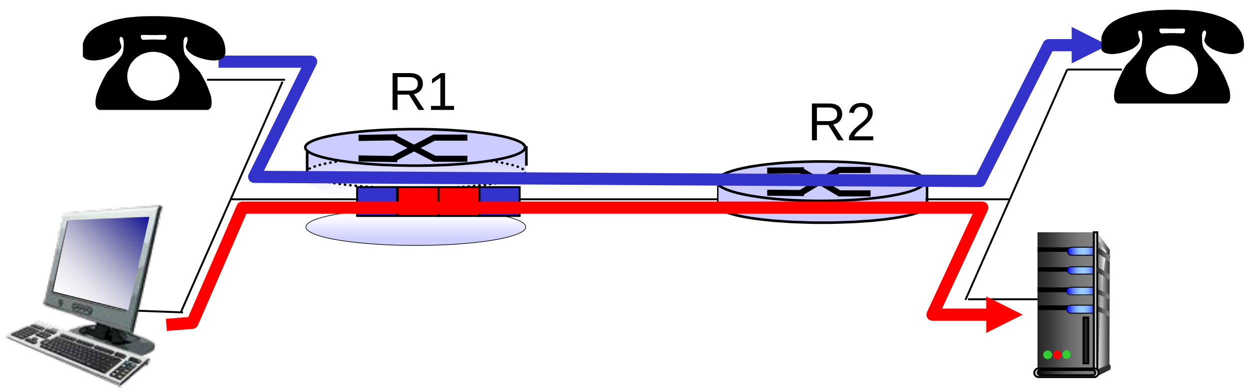


Multiple classes of service: scenario



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

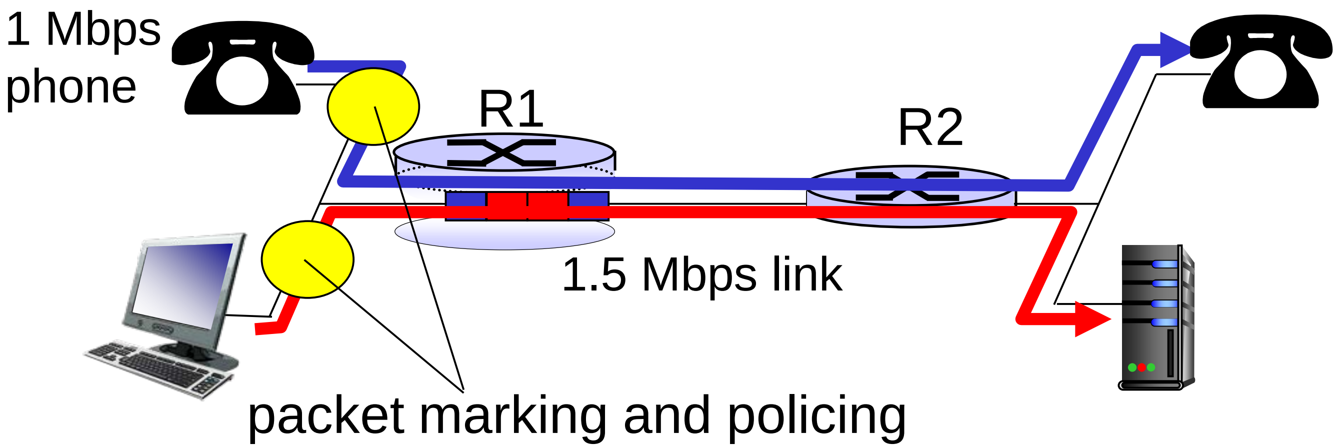


Principle 1

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

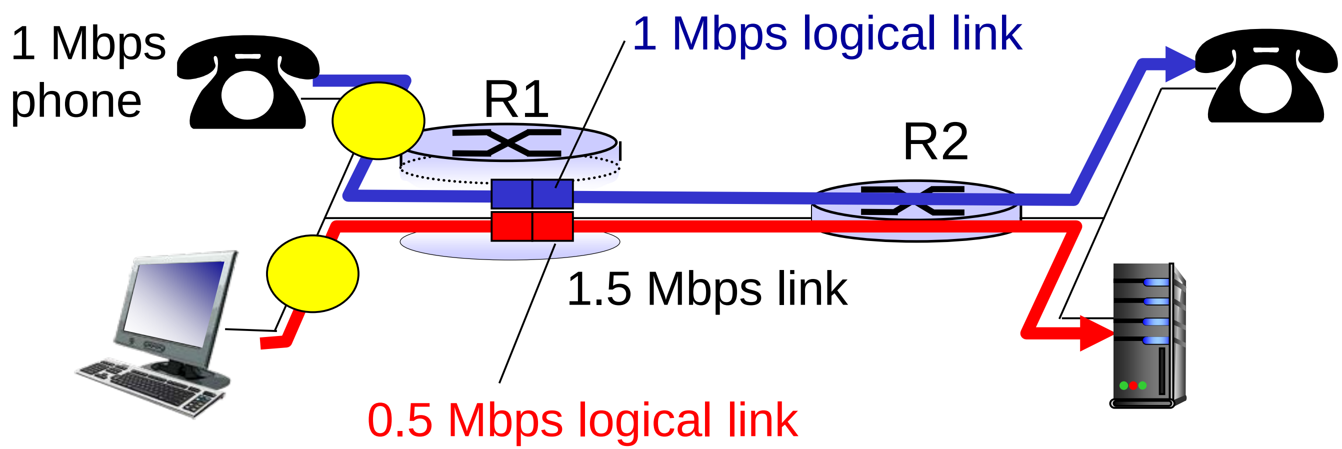


Principle 2

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

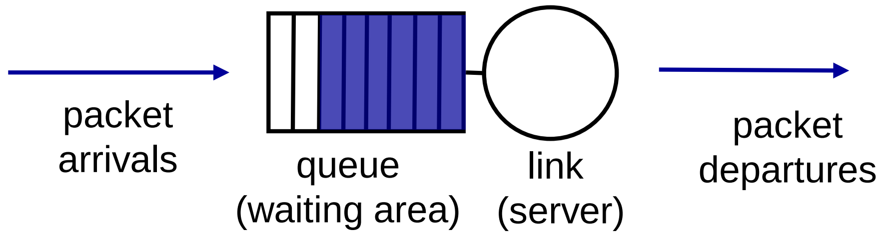


Principle 3

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

Scheduling and policing mechanisms

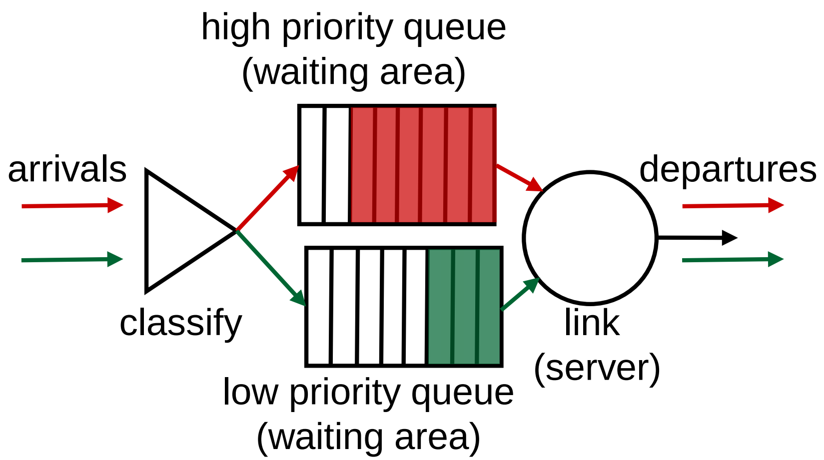
- *scheduling:* choose next packet to send on link
- *FIFO (first in first out) scheduling:* send in order of arrival to queue
  - real-world example?
  - *discard policy:* if packet arrives to full queue: who to discard?
    - *tail drop:* drop arriving packet
    - *priority:* drop/remove on priority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

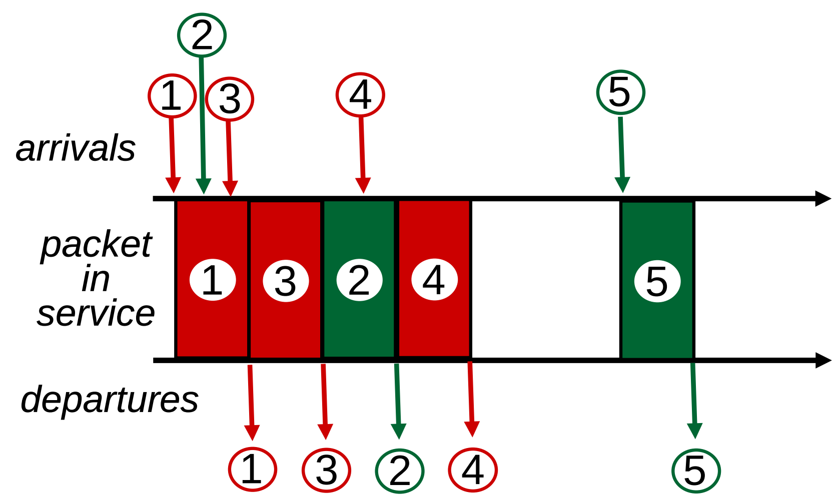


Scheduling policies: priority

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?

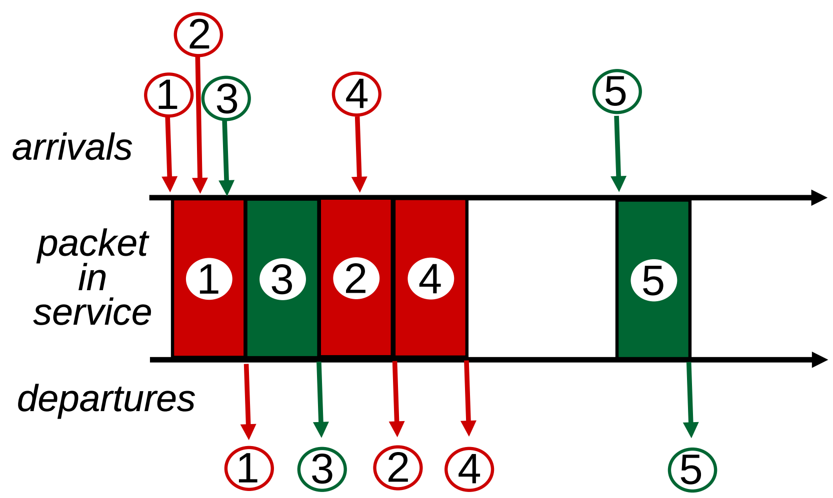




Scheduling policies: still more

Round Robin (RR) scheduling:

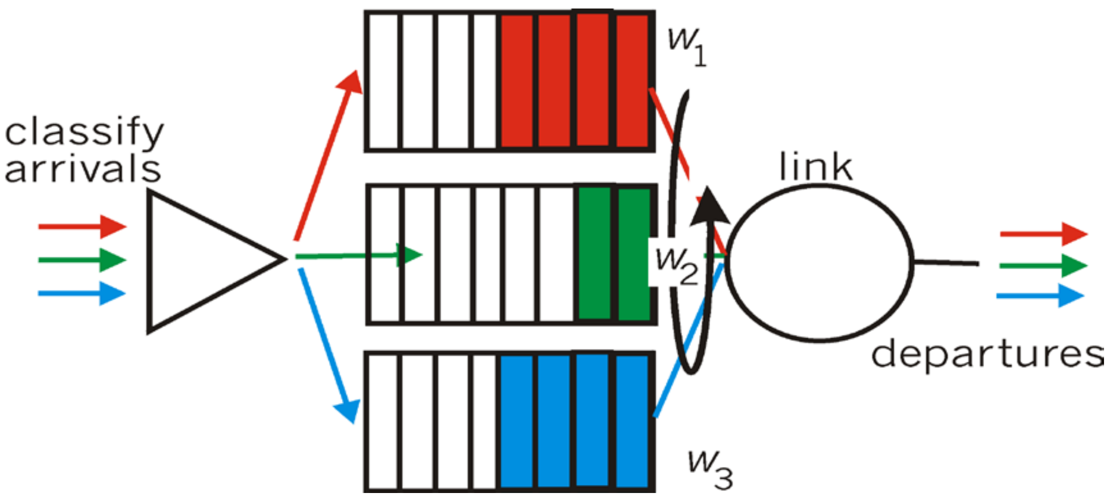
- multiple classes
- cyclically scan class queues, sending one complete packet from each class (if available)
- real world example?



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

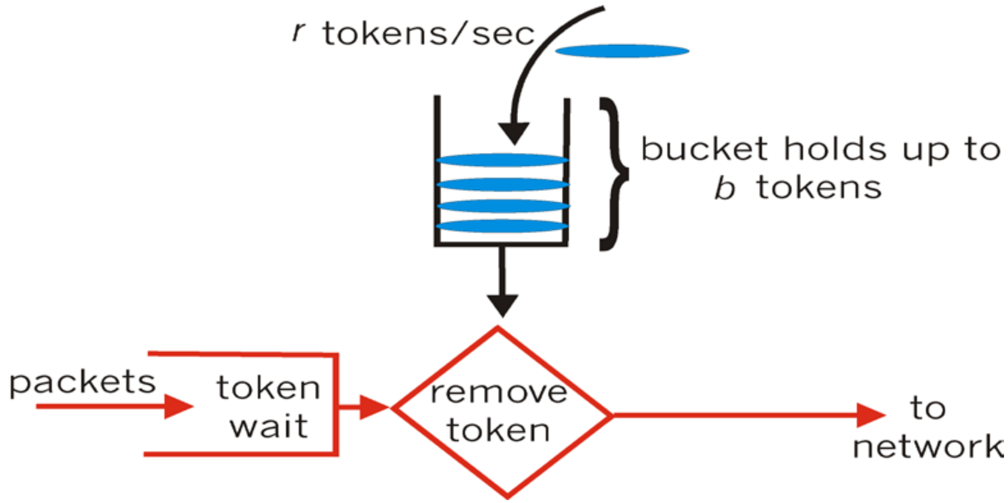
goal: limit traffic to not exceed declared parameters

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

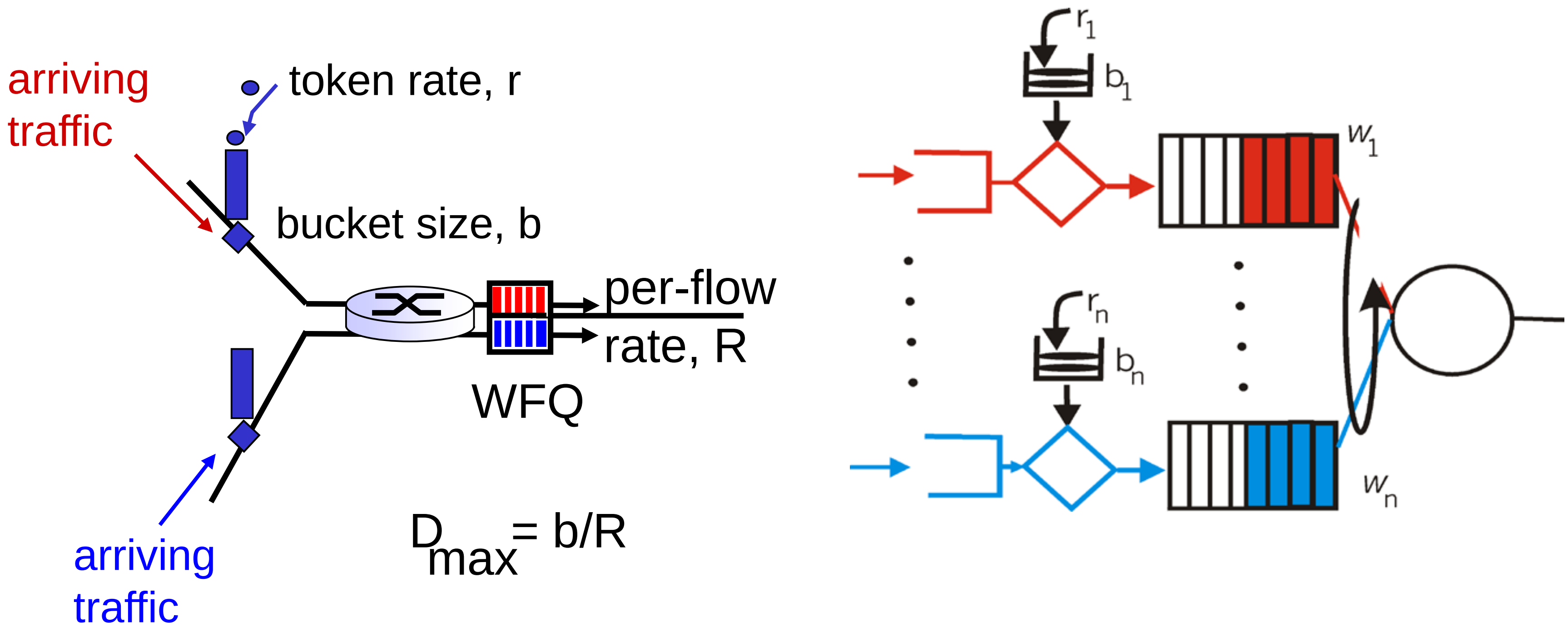
token bucket: limit input to specified *burst size* and *average rate*



- 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!*



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

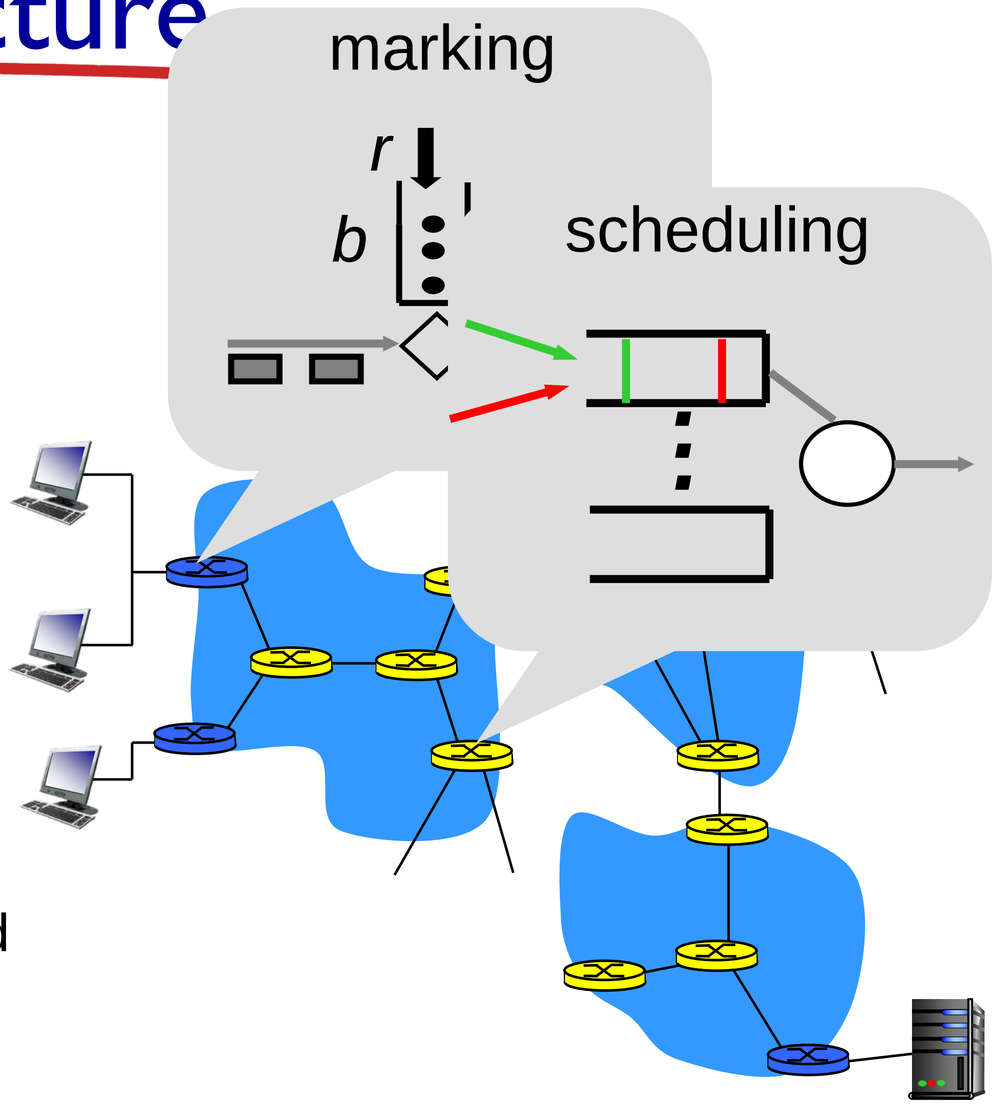
Diffserv architecture

edge router:

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

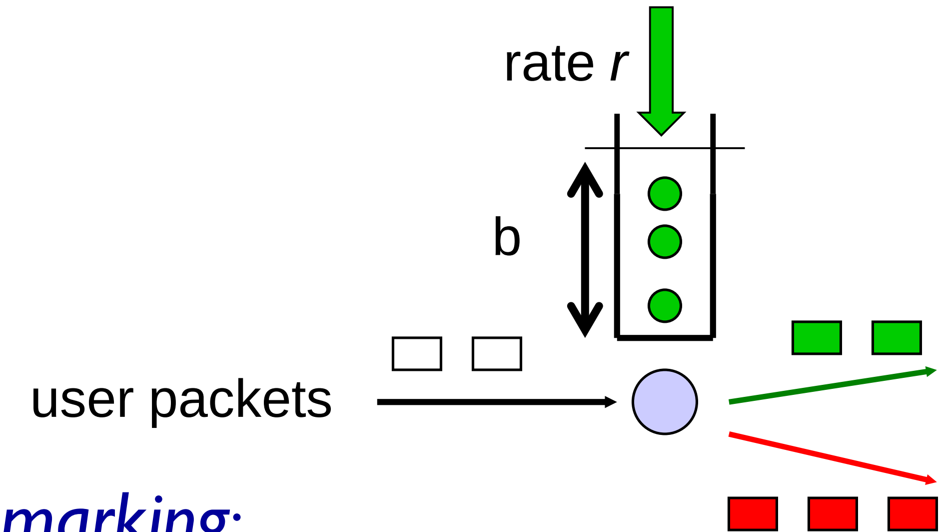
core router:

- *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

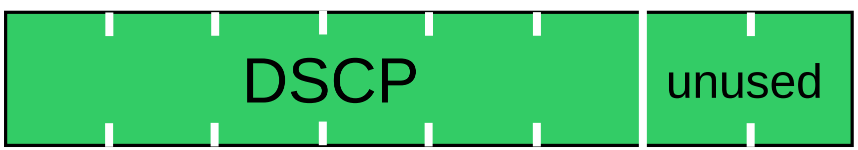


possible use of marking:

- 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

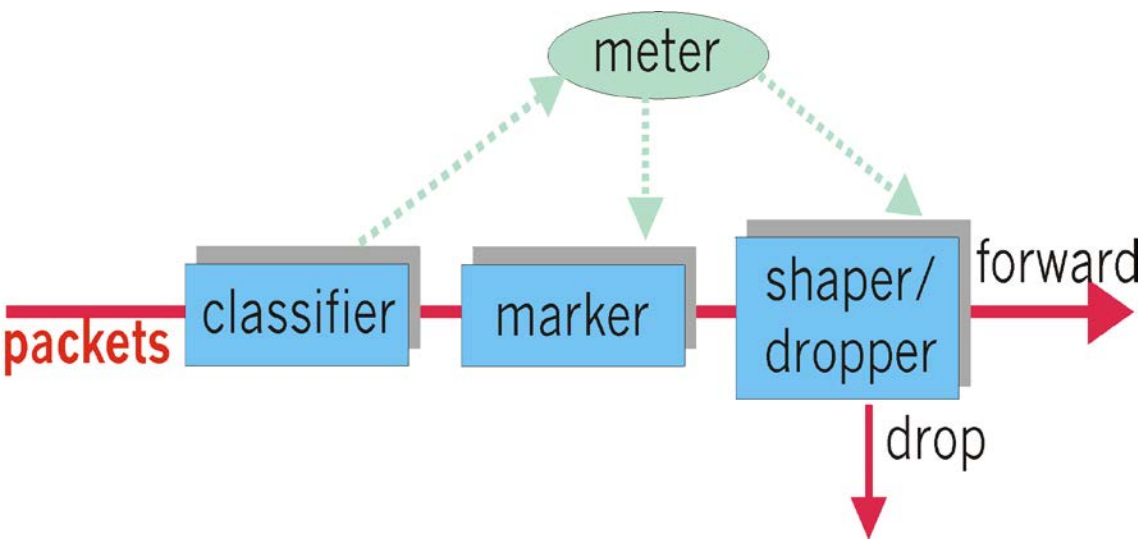
- packet is marked in the Type of Service (TOS) in IPv4, and Traffic Class in IPv6
- 6 bits used for Differentiated Service Code Point (DSCP)
  - determine PHB（Per-hop Behavior）that the packet will receive
  - 2 bits currently unused



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



Forwarding Per-hop Behavior (PHB)

- PHB result in a different *observable (measurable)* forwarding performance behavior
- PHB does *not* specify what mechanisms to use to ensure required PHB performance behavior
- examples:
  - class A gets x% of outgoing link bandwidth over time intervals of a specified length
  - class A packets leave first before packets from class B

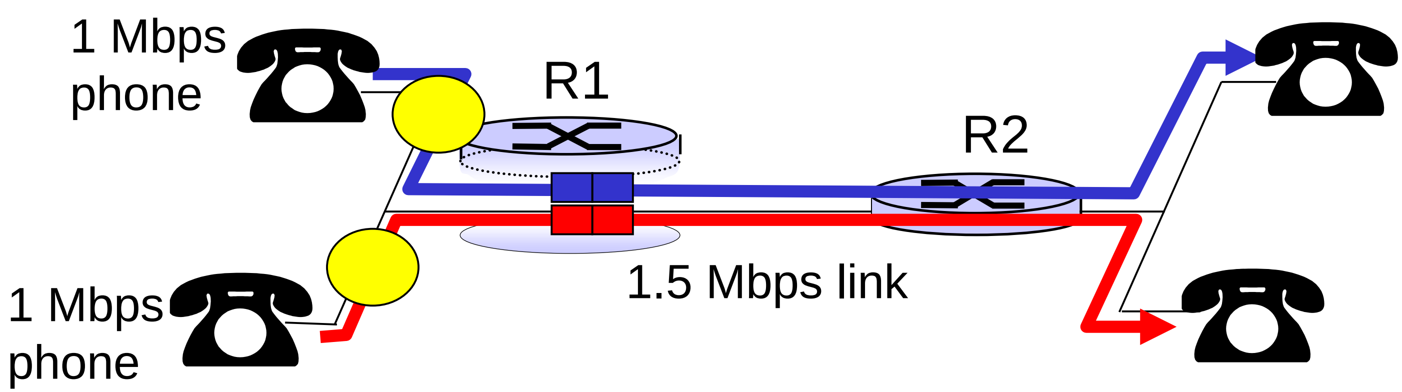
Forwarding PHB

PHBs proposed:

- *expedited forwarding:* pkt departure rate of a class equals or exceeds specified rate
  - logical link with a minimum guaranteed rate
- *assured forwarding:* 4 classes of traffic
  - each guaranteed minimum amount of bandwidth
  - each with three drop preference partitions

Per-connection QOS guarantees

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

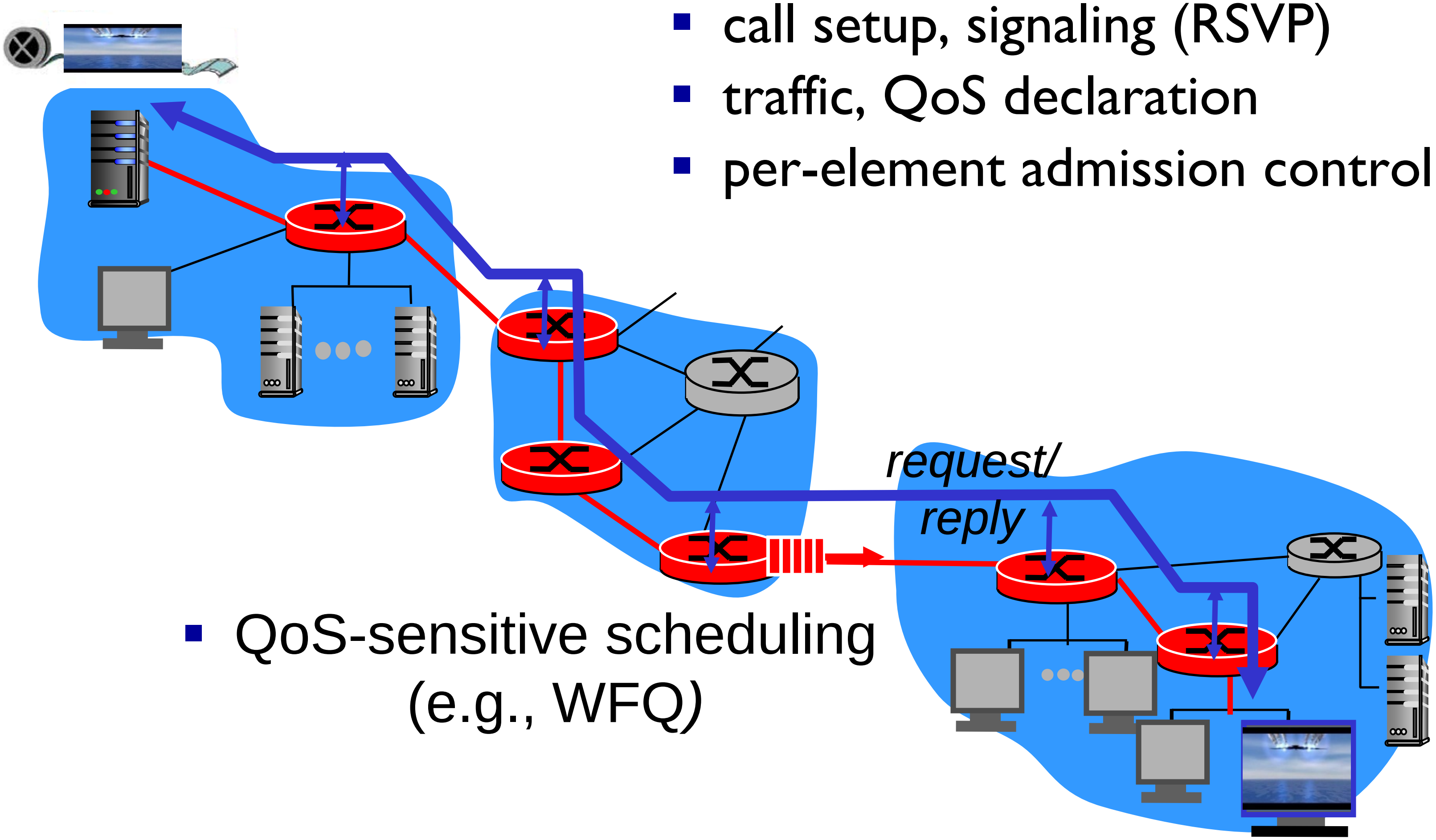


Principle 4

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)

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