# Multimedia networking: outline

9.1 multimedia networking applications
9.2 streaming *stored* video
9.3 voice-over-IP
9.4 protocols for *real-time* conversational applications: SIP – Skip RTP, RTCP
9.5 network support for multimedia

# SIP: Session Initiation Protocol [RFC 3261]

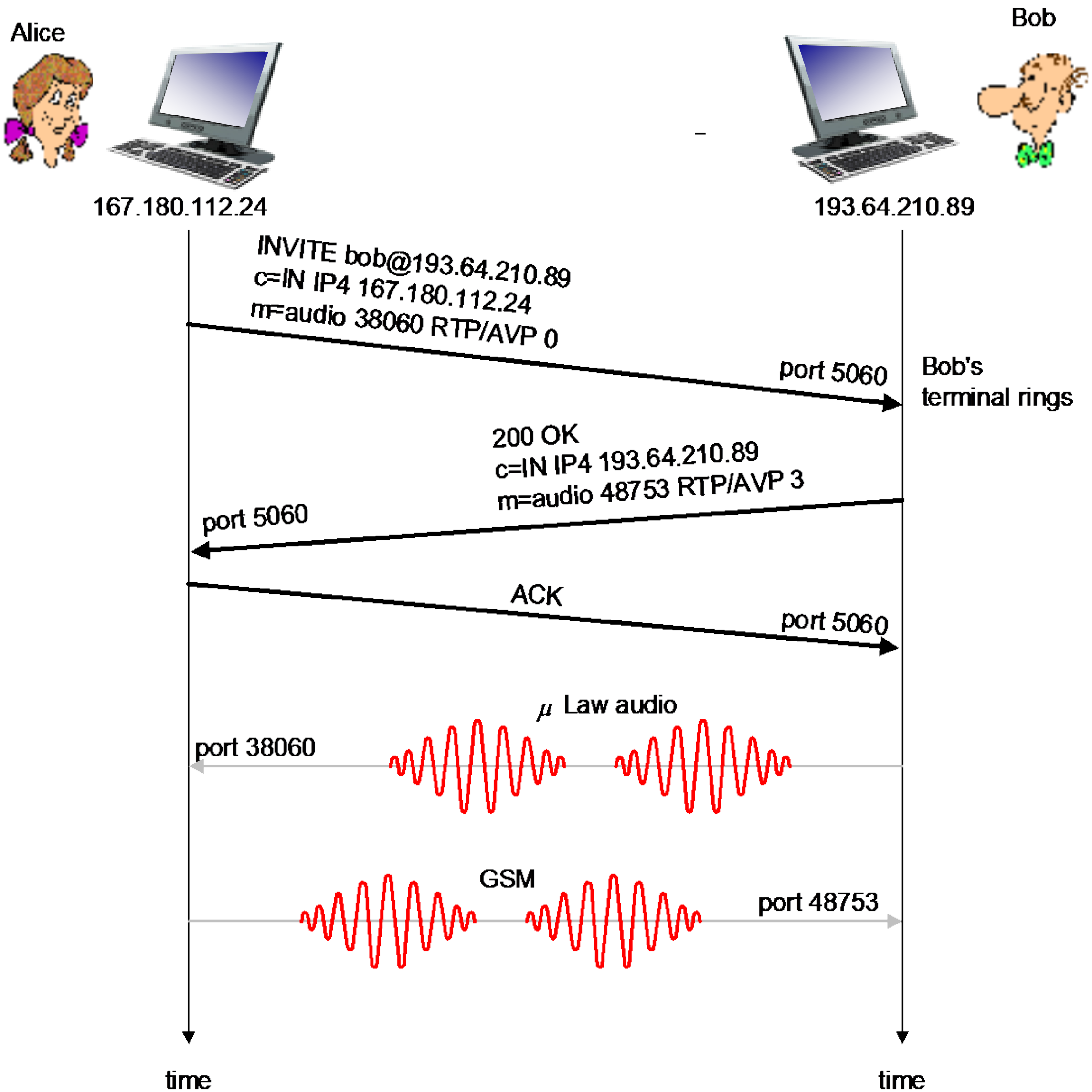
*long-term vision:*

- 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

# 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 506
- 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

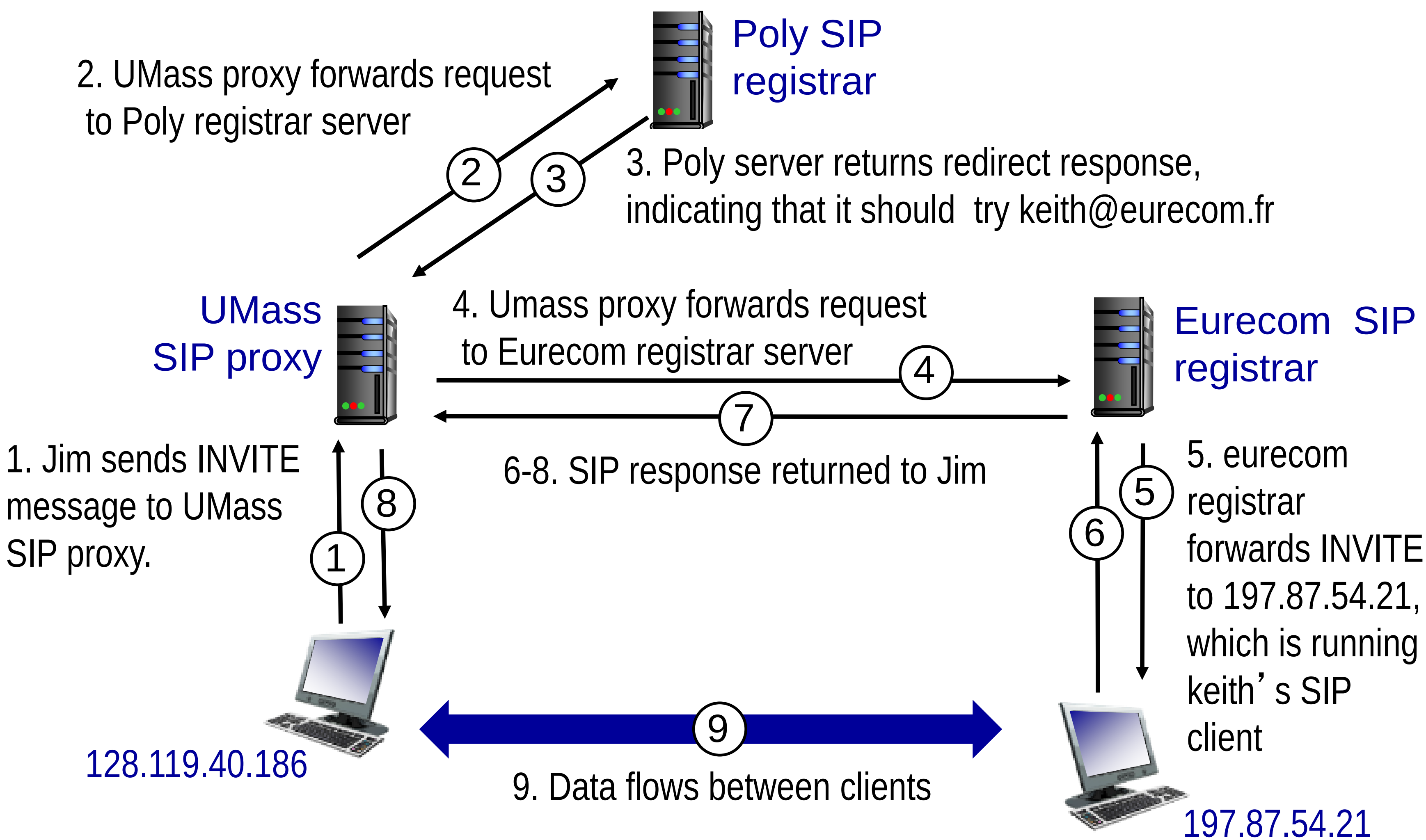
register message:

```
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



# Multimedia networking: outline

9.1 multimedia networking applications
9.2 streaming *stored* video
9.3 voice-over-IP
9.4 protocols for *real-time* conversational applications
9.5 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 |

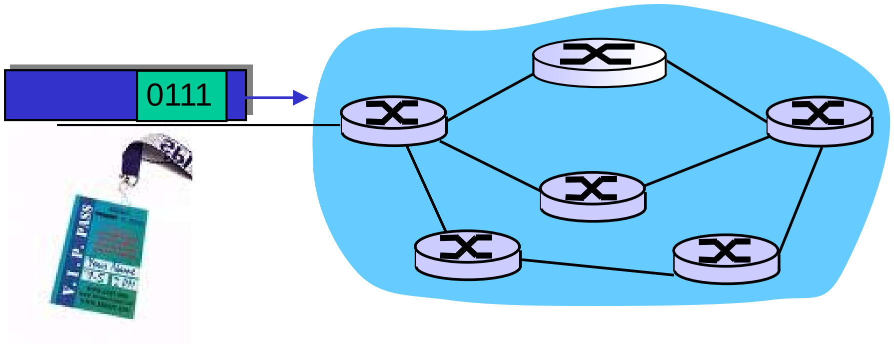
# Dimensioning best effort networks

- *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)

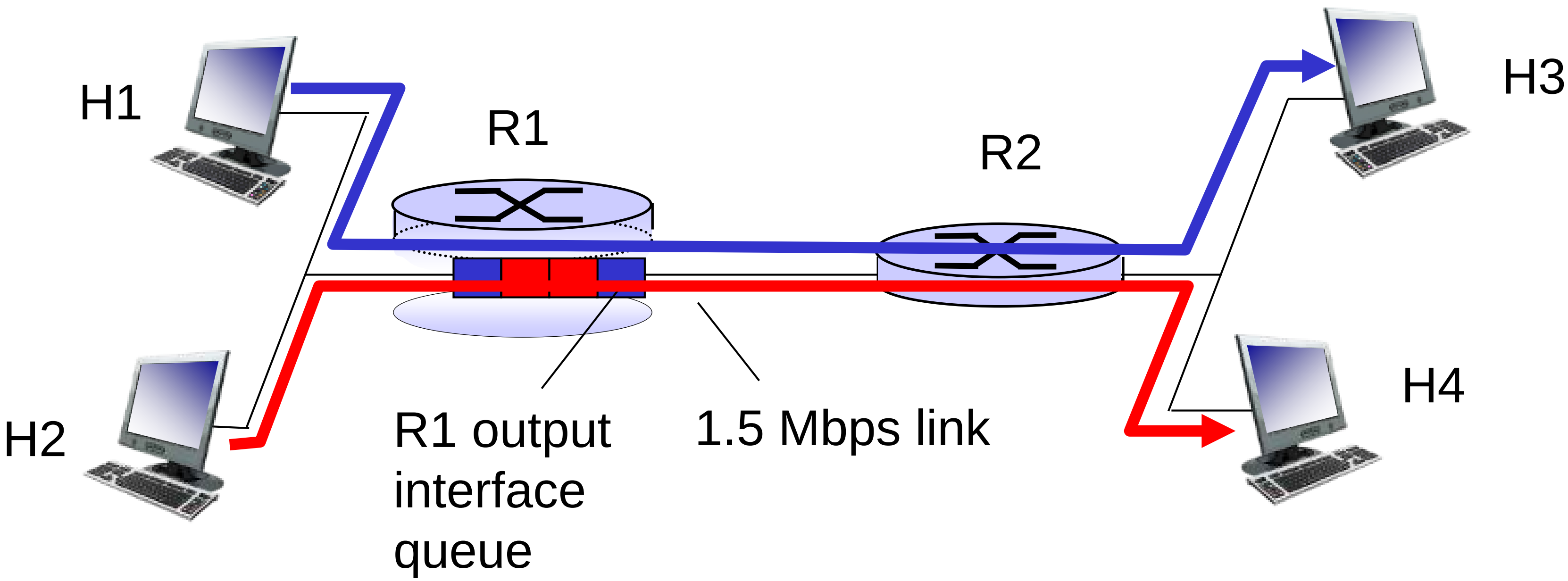
# Providing multiple classes of service

- 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

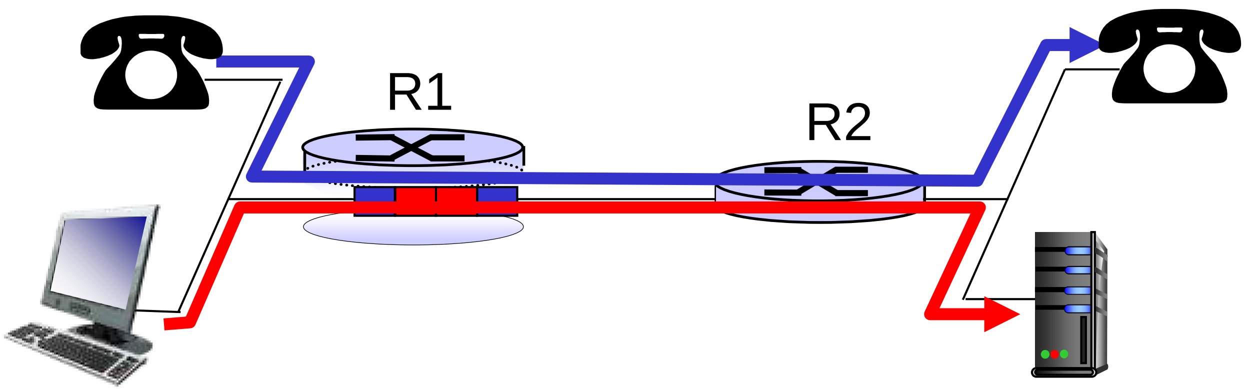


# Multiple classes of service: scenario



# 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

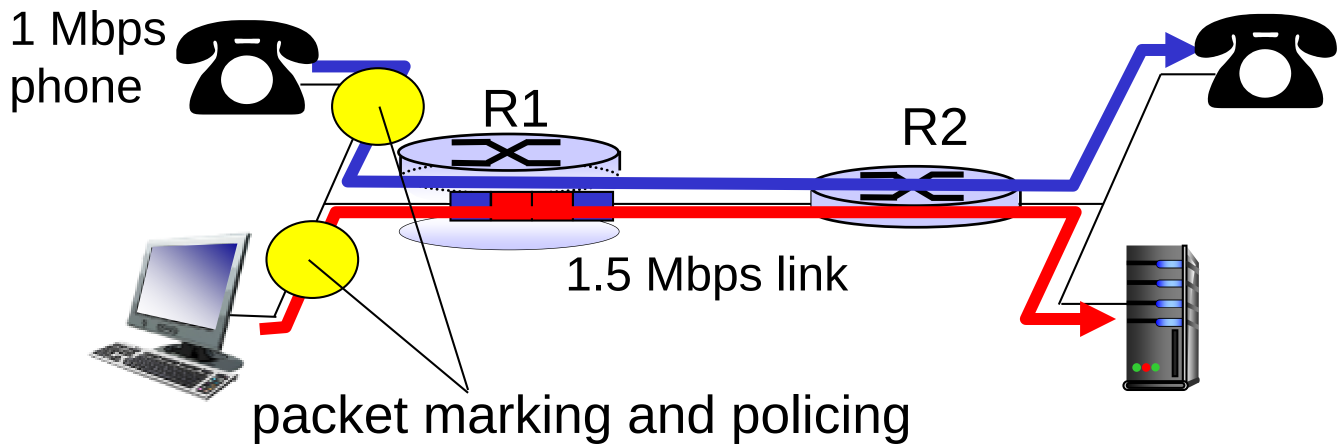



**Principle 1**

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

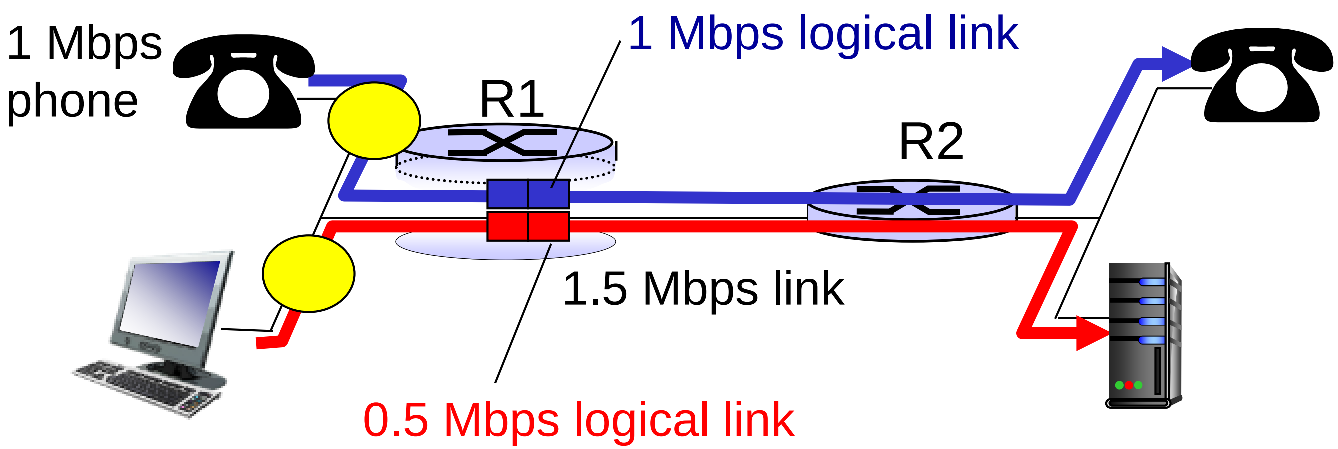



**Principle 2**

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

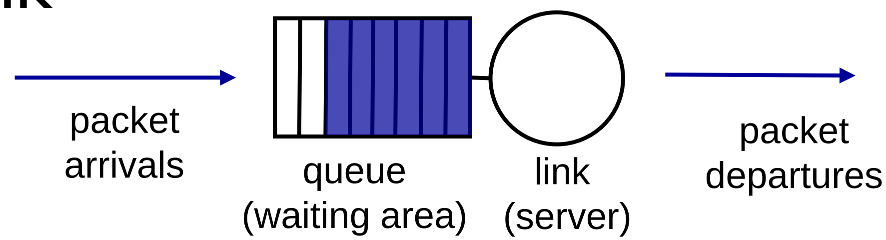



**Principle 3**

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

# Scheduling and policing mechanisms

- *packet scheduling:* choose next queued packet to send on outgoing link




- previously covered in Chapter 4:
  - FCFS: first come first served
  - simply multi-class priority
  - round robin
  - weighted fair queueing (WFQ)

# Policing mechanisms

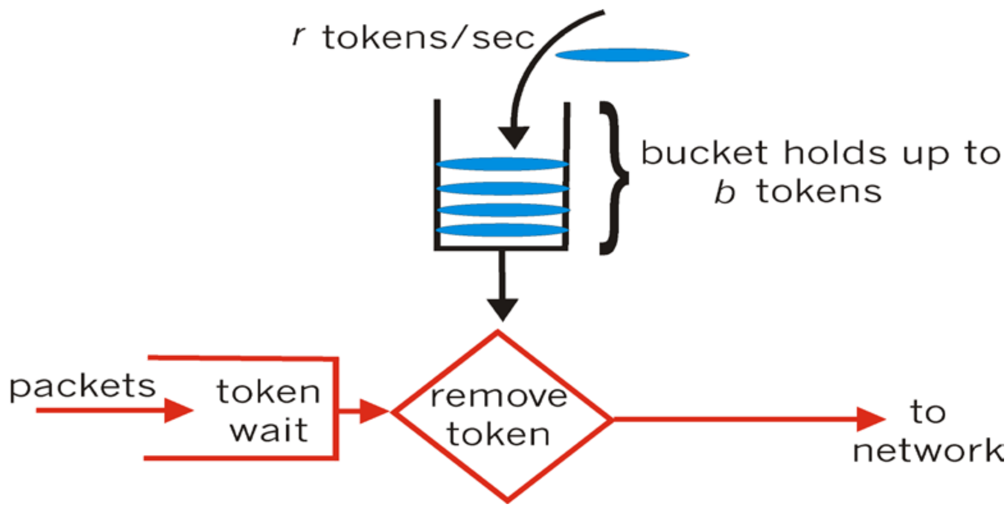
*goal:* limit traffic to not exceed declared parameters

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

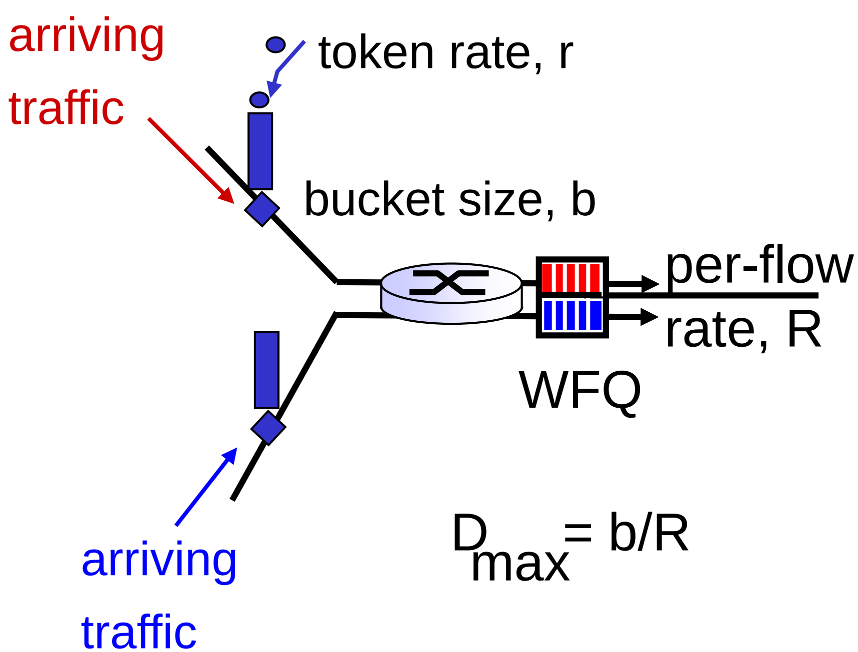
*token bucket:* limit input to specified *burst size* and *average rate*



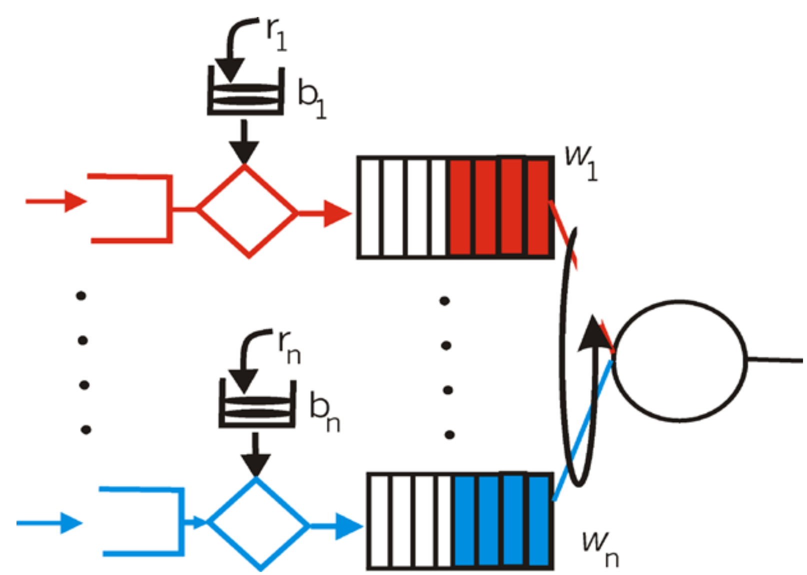

- bucket can hold b tokens
- tokens generated at rate *r token/sec* unless bucket full
- *over interval of length t: number of packets admitted less than or equal to (r t + b)*

# Policing and QoS guarantees

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




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



# DiffServ, IntServ

- Two proposals to add QoS on the internet
- Skip details