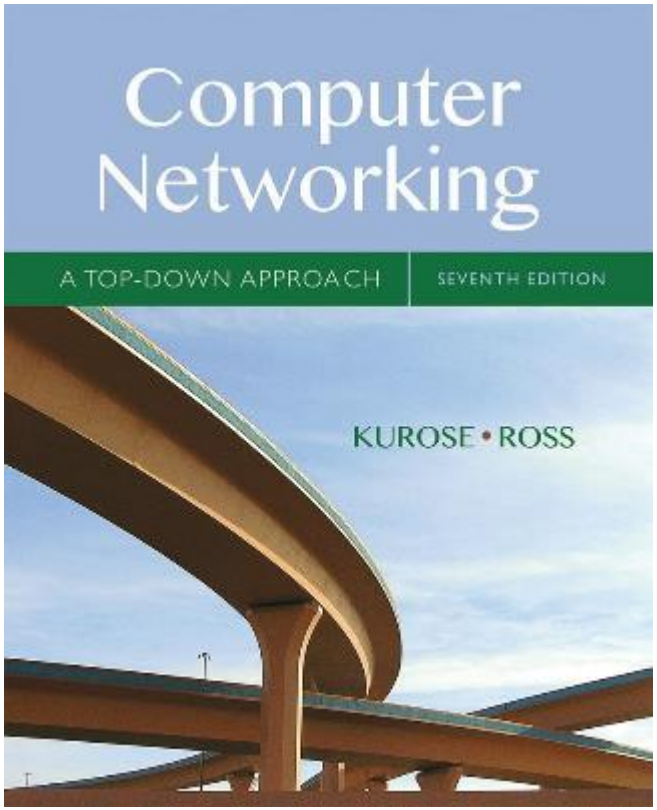
Τεχνολογίες Διαδικτύου 2024-25 (DIT299)

Δρ. Ειρήνη Λιώτου

eliotou@hua.gr

3/12/2024

Chapter 9 Multimedia Networking



Computer Networking: A Top Down Approach

7th edition
Jim Kurose, Keith Ross
Pearson/Addison Wesley
April 2016

A note on the use of these Powerpoint 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

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: RTP, SIP

9.5 network support for multimedia

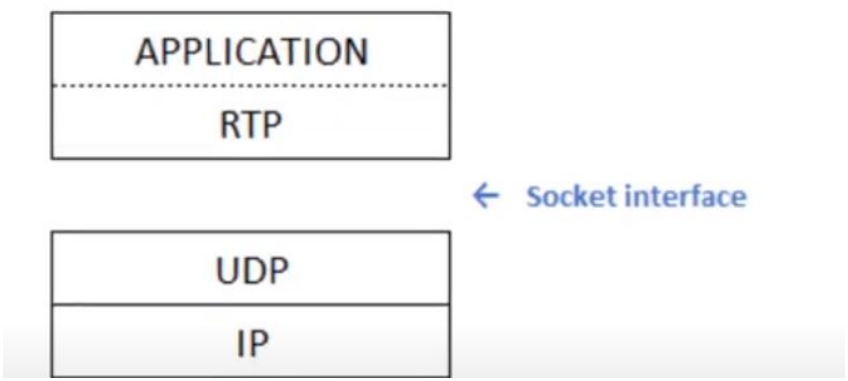
Real-Time Protocol (RTP)

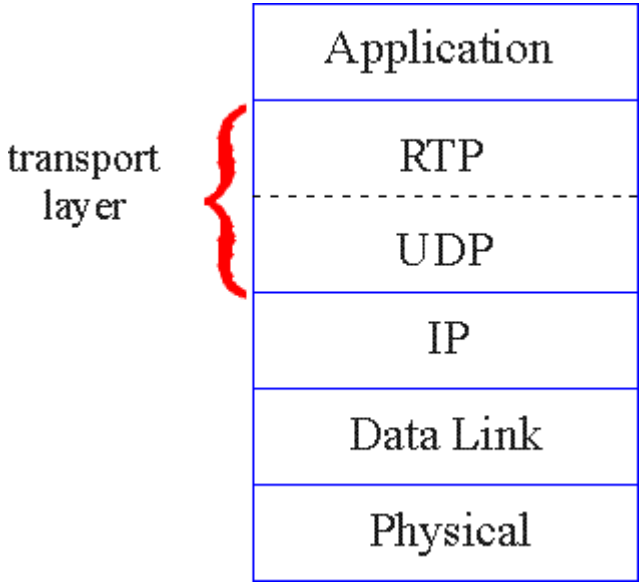
- RTP specifies packet structure for packets carrying audio, video data
- RFC 3550
- RTP packet provides
  - payload type identification
  - packet sequence numbering
  - time stamping
- RTP runs in end systems
- RTP packets encapsulated in UDP segments
- interoperability: if two VoIP applications run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

RTP runs on top of UDP

RTP libraries provide transport-layer interface that extends UDP:

- 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 (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

- *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

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


Zoiper
account@sip.example.com
Find a contact...
calls
John Smith
00:15:31
Calling
Contacts
Recent
All
Online
Favorites
John Smith
Jeff Groves
15
Multiparty Chat
9
Casey Stoner
3
Catherin Stone
Andrea Stella
Marco Minneman
Mute
Speaker
Keypad
Statistics
Record
Video
Hold
Transfer
Add call
Incoming call
John Smith
john.smith
account@sip.example.com
00:15:31

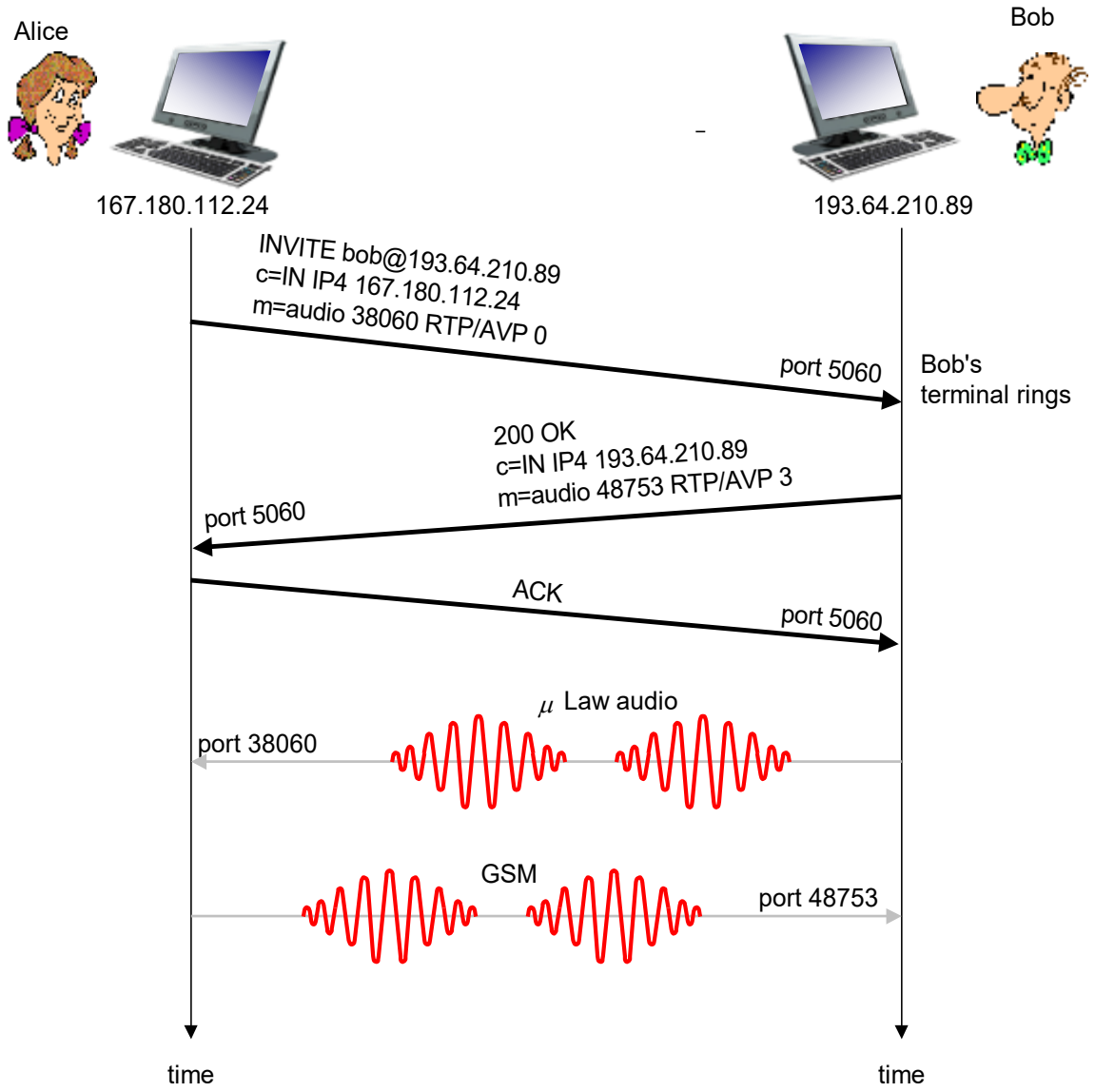

Yealink


Search ...
Colin Edwards
Christina Smith
Andrew Benson
Manuel Gonzales
Christopher Smith
Chad Lee
Colin Edwards
Mobile SIP Account 1
00:07
Video call
Add call
Call statistics
Call record
Transfer call
Search ...
A
Adam Parr
Andrew Benson
Anonymous
B
Barbara Johanson
C
Christina Smith
Colin Edwards
Colin Stevens
D
David Grey

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 identifier
- 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"

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
- Call-ID is unique for every call
- Session Description Protocol (SDP)

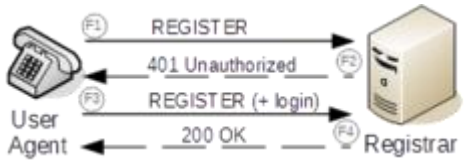
- Here we don't know Bob's IP address
  - intermediate SIP servers needed
- Alice sends, receives SIP messages using SIP default port 5060
- Alice specifies in header that SIP client sends, receives SIP messages over UDP

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



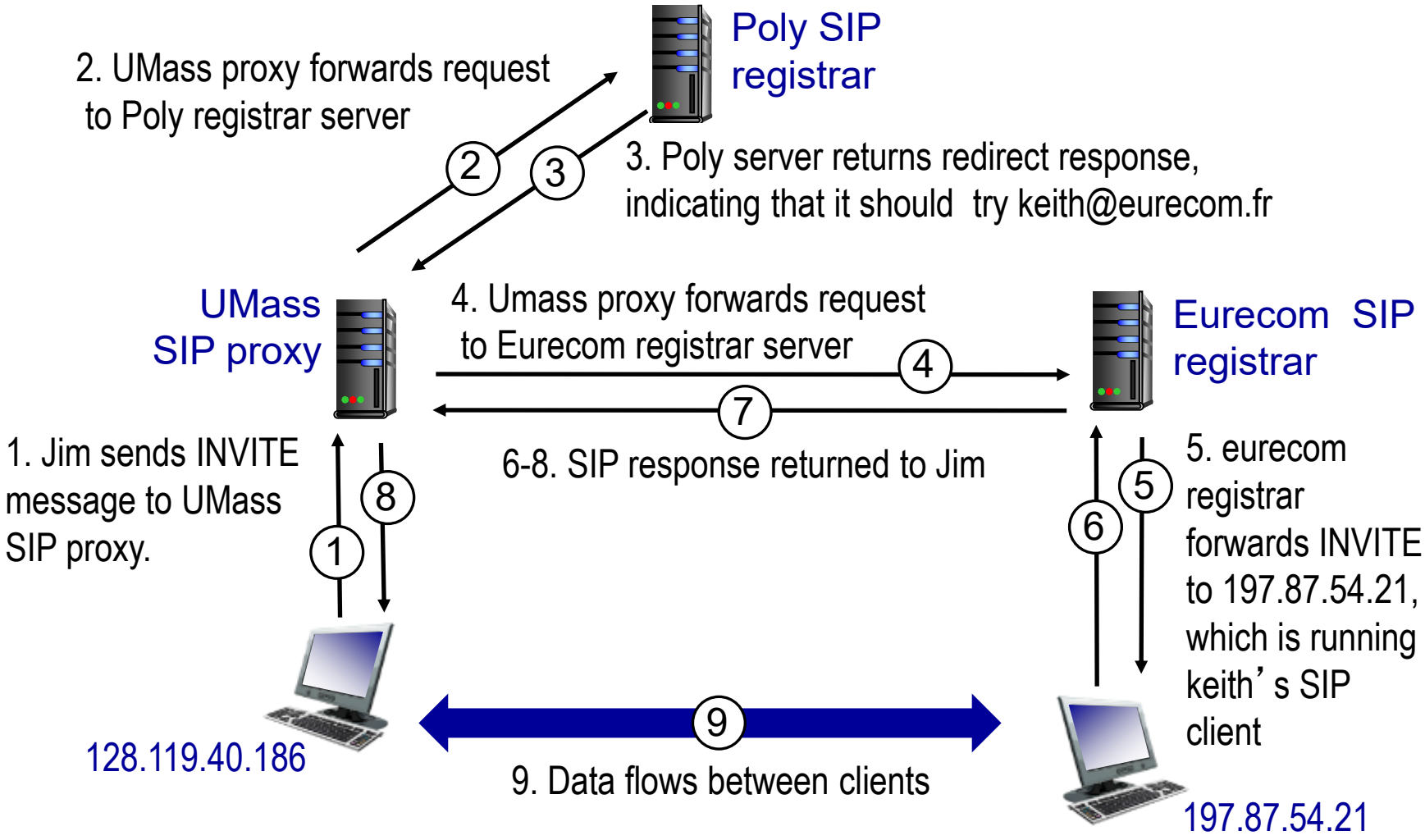
register message:

```
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



SIP messages

SIP requests

Request name	Description
REGISTER	Register the URI listed in the To-header field with a location server and associates it with the network address given in a *Contact* header field.
INVITE	Initiate a dialog for establishing a call. The request is sent by a user agent client to a user agent server.
ACK	Confirm that an entity has received a final response to an INVITE request.
BYE	Signal termination of a dialog and end a call.
CANCEL	Cancel any pending request.
UPDATE	Modify the state of a session without changing the state of the dialog.
REFER	Ask recipient to issue a request for the purpose of call transfer.
PRACK	Provisional acknowledgement.
SUBSCRIBE	Initiates a subscription for notification of events from a notifier.
NOTIFY	Inform a subscriber of notifications of a new event.
PUBLISH	Publish an event to a notification server.
MESSAGE	Deliver a text message.
INFO	Send mid-session information that does not modify the session state.
OPTIONS	Query the capabilities of an endpoint.

SIP messages

Responses are sent by the user agent server indicating the result of a received request. Several classes of responses are recognized, determined by the numerical range of result codes:

- 1xx: Provisional responses to requests indicate the request was valid and is being processed.
- 2xx: Successful completion of the request. As a response to an INVITE, it indicates a call is established. The most common code is 200, which is an unqualified success report.
- 3xx: Call redirection is needed for completion of the request. The request must be completed with a new destination.
- 4xx: The request cannot be completed at the server for a variety of reasons, including bad request syntax (code 400).
- 5xx: The server failed to fulfill an apparently valid request, including server internal errors (code 500).
- 6xx: The request cannot be fulfilled at any server. It indicates a global failure, including call rejection by the destination.

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:
  - *Bandwidth provisioning:* how much bandwidth is "enough?"
  - *Network dimensioning:* how to design a network topology

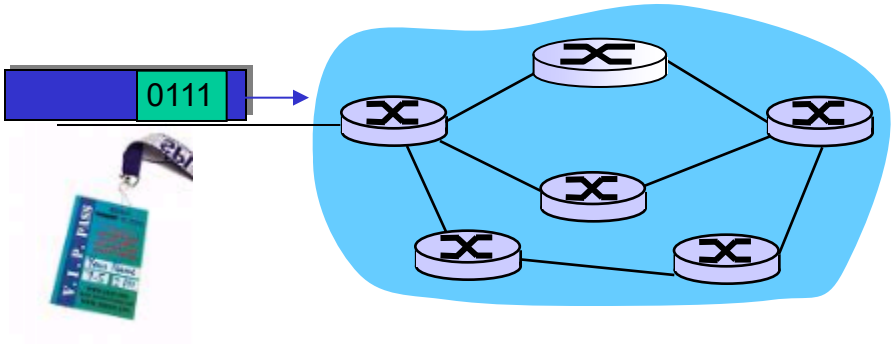
Dimensioning best effort networks

- Models of traffic demand between network end points. Models may need to be specified at both the call level and at the packet level.
- Well-defined performance requirements. For example, a performance requirement for supporting delay-sensitive traffic.
- Models to predict end-to-end performance for a given workload model, and techniques to find a minimal cost bandwidth allocation that will result in all user requirements being met.

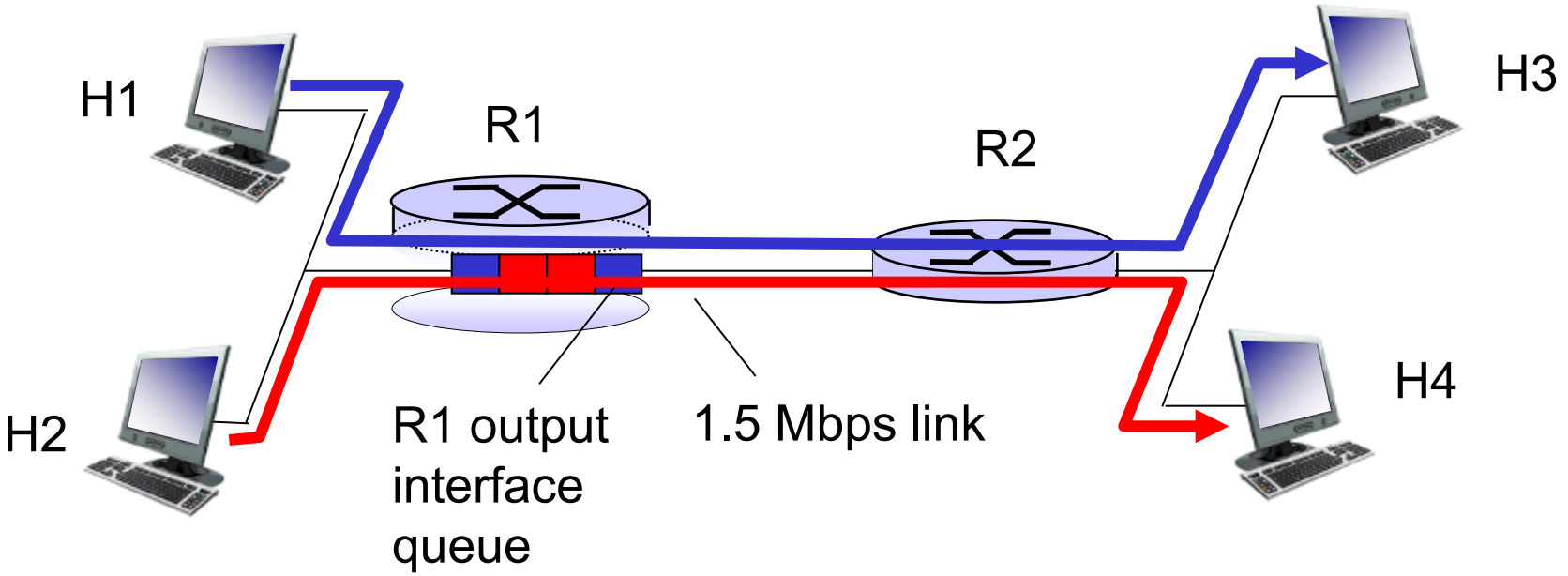
Providing multiple classes of service

- 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

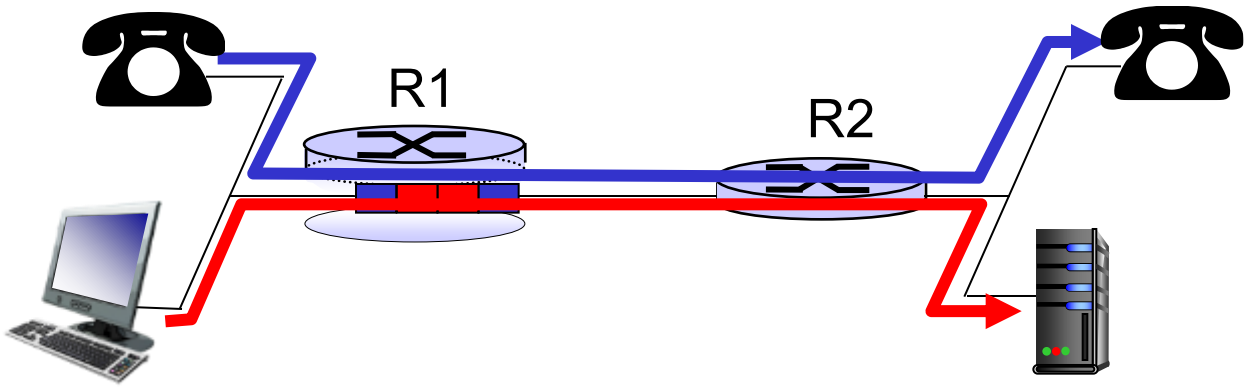


Multiple classes of service: scenario



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

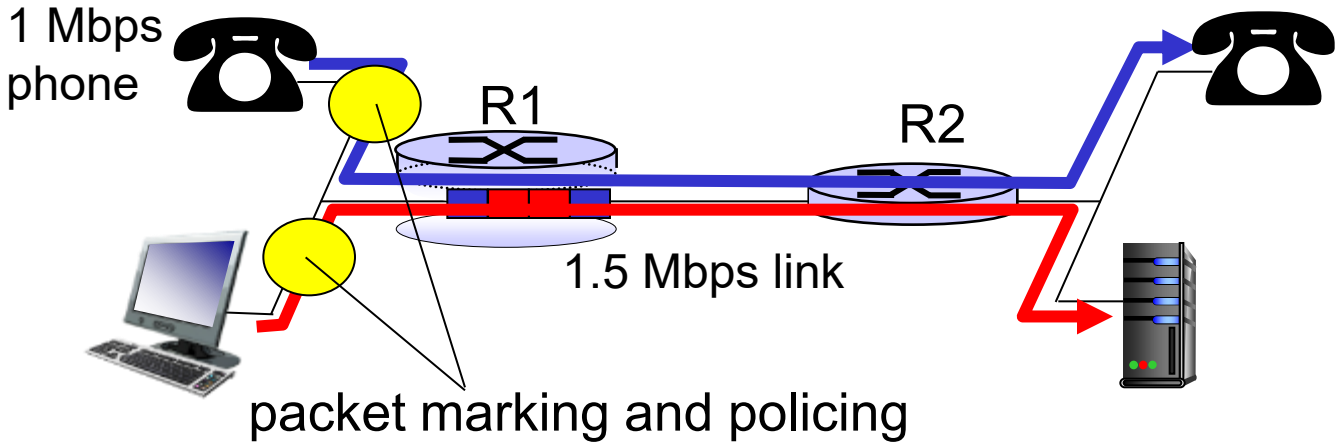


Principle 1

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

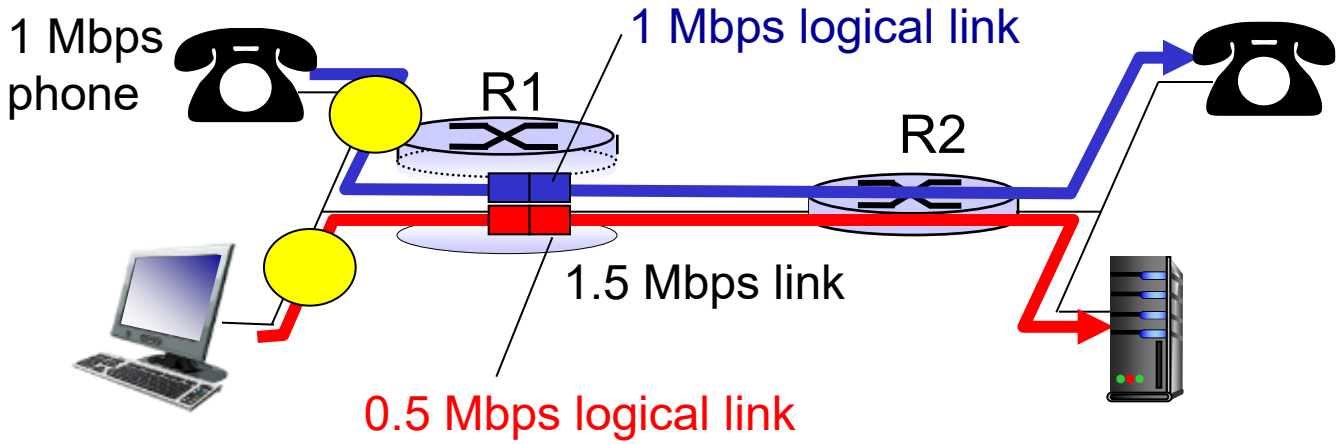


Principle 2

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 a flow doesn't use its allocation

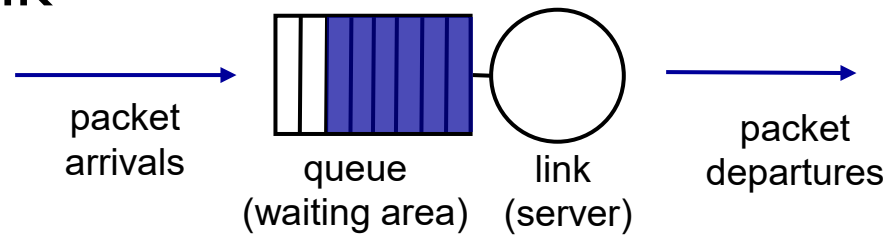


Principle 3

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



  - FCFS: first come first served
  - simply multi-class priority
  - weighted fair queueing (WFQ)
  - …

Policing mechanisms

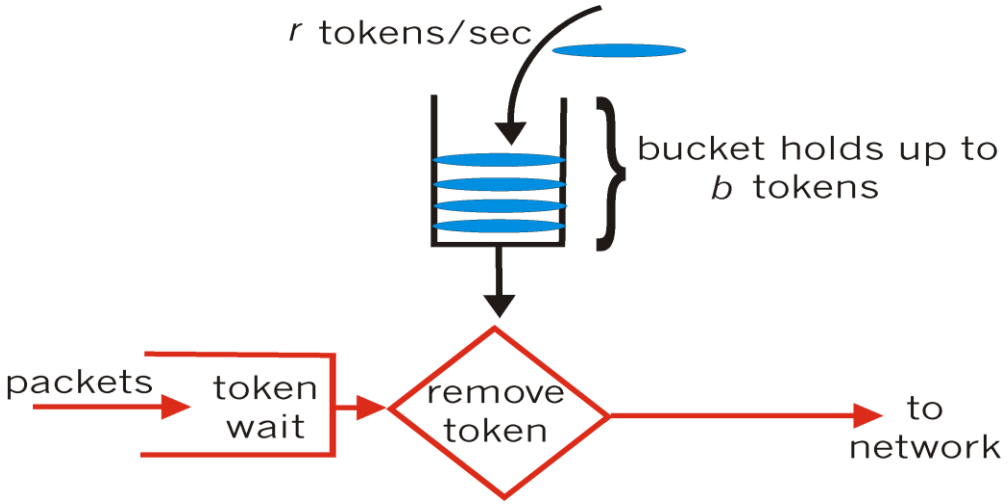
goal: limit traffic to not exceed declared parameters

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 pkts per sec peak rate
- *(max.) burst size:* max number of pkts sent consecutively (with no intervening idle)

Policing mechanisms: implementation

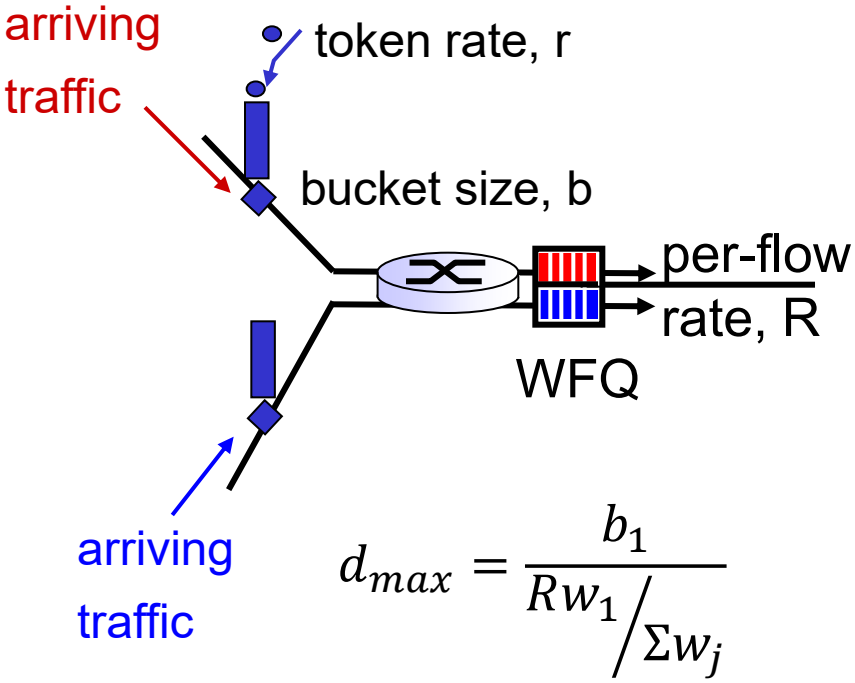
token bucket: limit input to specified *burst size* and *average rate*



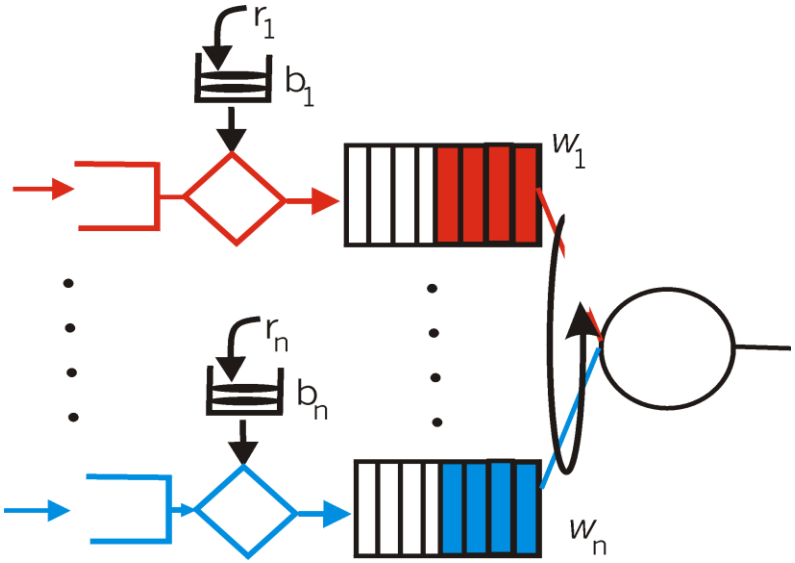
- bucket can hold b tokens
- tokens generated at rate *r token/sec* unless bucket full
- *over interval of length t: number of packets admitted less than or equal to (r t + b)*

Policing and QoS guarantees

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



$$d_{max} = \frac{b_1}{Rw_1 / \Sigma w_j}$$



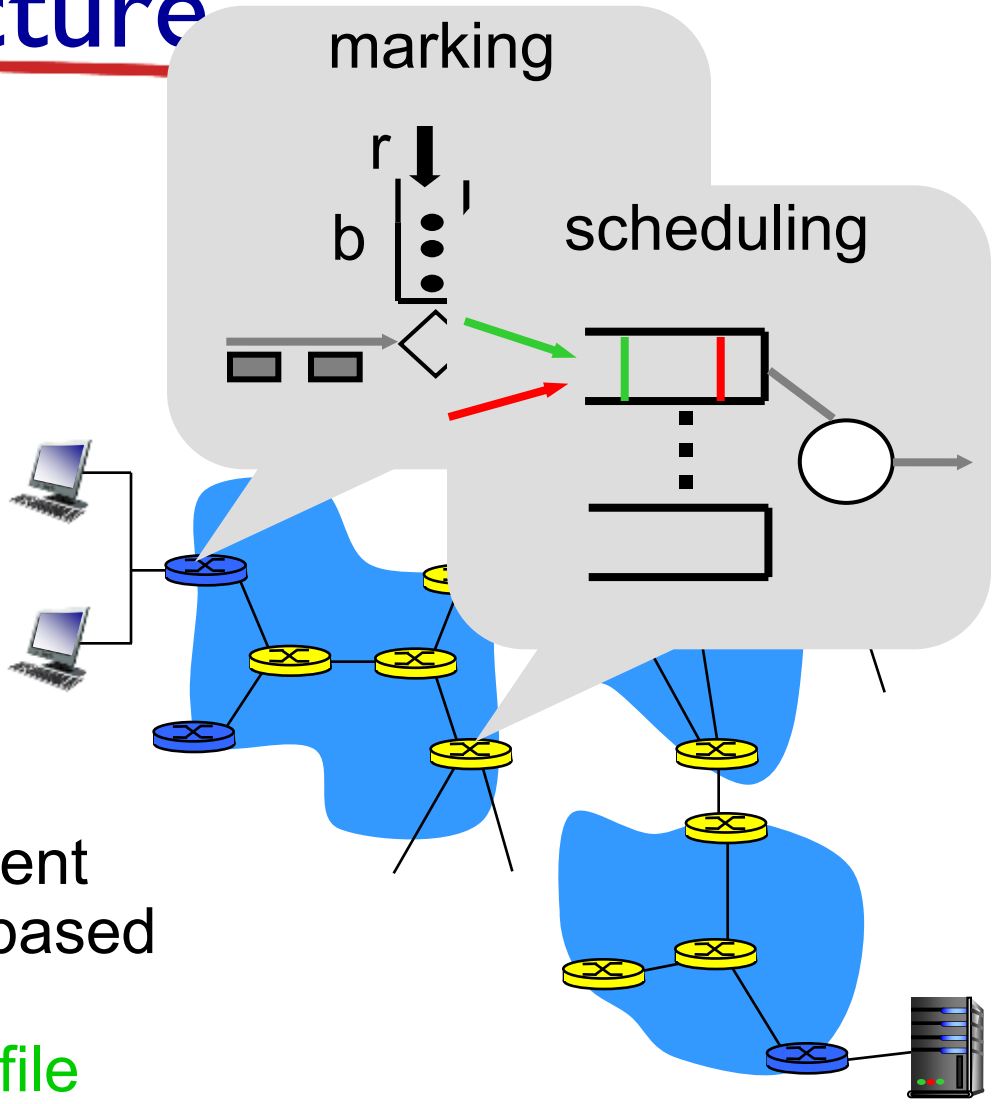
Differentiated services

- want "qualitative" service classes
  - "behaves like a wire"
  - relative service distinction: Platinum, Gold, Silver
- Rather than differentiating network traffic based on the requirements of an individual flow, DiffServ operates on the principle of traffic classification, placing each data packet into one of a limited number of traffic classes
- Each router on the network is then configured to differentiate traffic based on its class. Each traffic class can be managed differently, ensuring preferential treatment for higher-priority traffic on the network
- *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

Diffserv architecture

edge router:

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

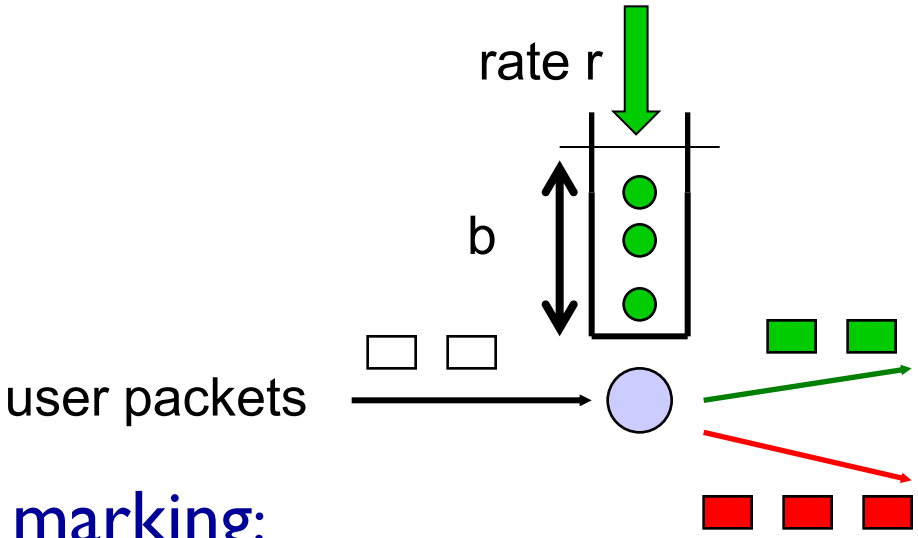


core router:

- 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

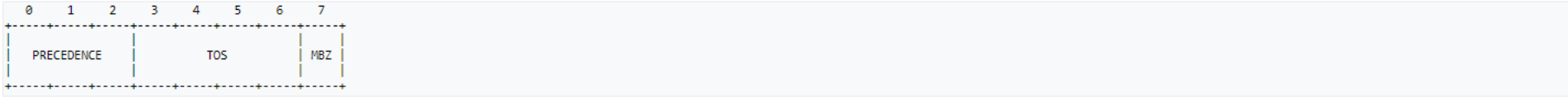
Diffserv packet marking: details

- 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

Type of Service (ToS) [edit]

The ToS (Type of Service) byte inside the IP header can be used for prioritization of packets inside a network. The field was defined in the RFC 791 IP protocol specification published in September 1981.

The Type of Service octet consists of three fields (RFC 1349):



The first 3 bits of a ToS field indicate precedence.

- The 4th bit is used to signal if *low delay* is desired and required
- The 5th bit indicates if *high throughput* is desired.
- The 6th bit indicates if *high reliability* is desired.
- The 7th and 8th bits are reserved

Differentiated services fields (DS field) [edit]

Everyone was happy with good old simple Type of Service codes until RFC 2474 defined a Differentiated Services Field (DS Field) *using the IP protocols Type of Service byte* in December 1998.

The differentiated services code point (DSCP) values are defined by the first six bits of the DSCP/ToS byte. The last two bits can and are being used for ECN (Explicit Congestion Notification) as defined in RFC 3168.

DSCP defines several *traffic classes*. The primary DSCP classes are, per RFC 4594, and RFC 8622:

- Lower-Effort (LE)
- Default Forwarding (DF)
- Assured Forwarding (AF)
- Expedited Forwarding (EF)
- Class Selector (CS)

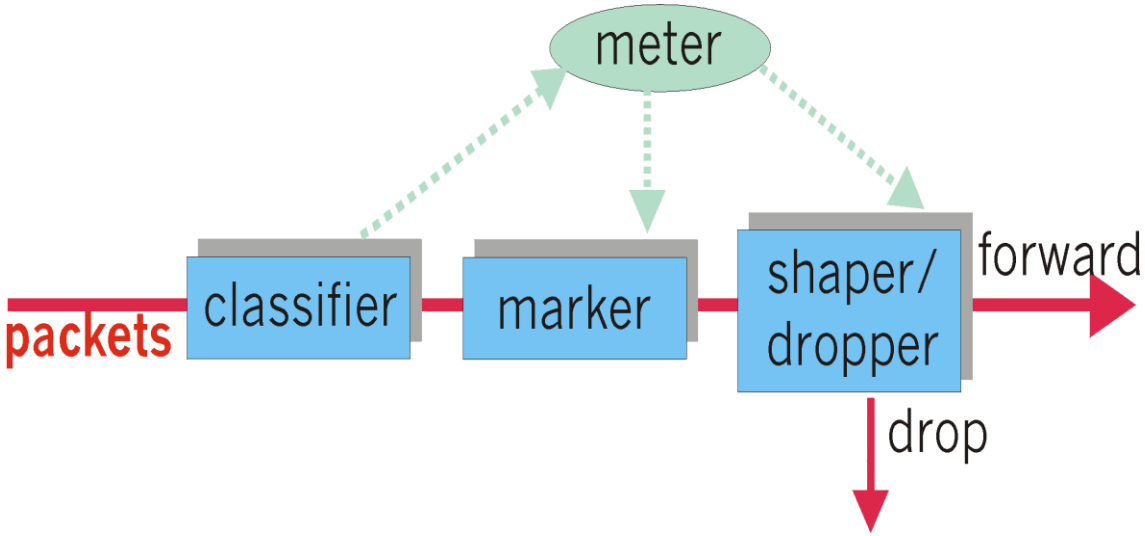
Higher numbered queues within the "Assured Forwarding" sub-classes have *lower* priority. (AF4**1** has a *higher* priority than AF4**2**). However, AF**2**1-AF**2**3 will have a higher priority than AF**1**1-AF**1**3.

The "Class Selector" values select class types, not priority.

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)

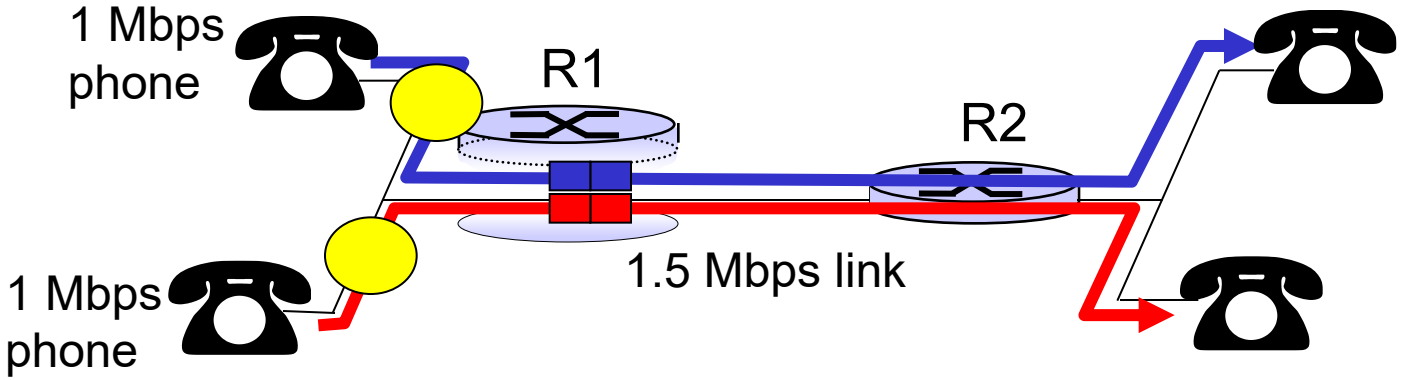
- PHB result in a different *observable (measurable)* forwarding performance behavior
- PHB does *not* specify what mechanisms to use to ensure required PHB performance behavior
- examples:
  - class A gets x% of outgoing link bandwidth over time intervals of a specified length
  - class A packets leave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

Forwarding PHB

- In theory, a network could have up to 64 different traffic classes using the 64 available DSCP values. In practice, :
- Default Forwarding (DF) PHB — which is typically best-effort traffic
- Expedited Forwarding (EF) PHB — dedicated to low-loss, low-latency traffic
- Assured Forwarding (AF) PHB — gives assurance of delivery under prescribed conditions
- Class Selector PHBs — which maintain backward compatibility with the IP precedence field.

Per-connection QoS guarantees

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

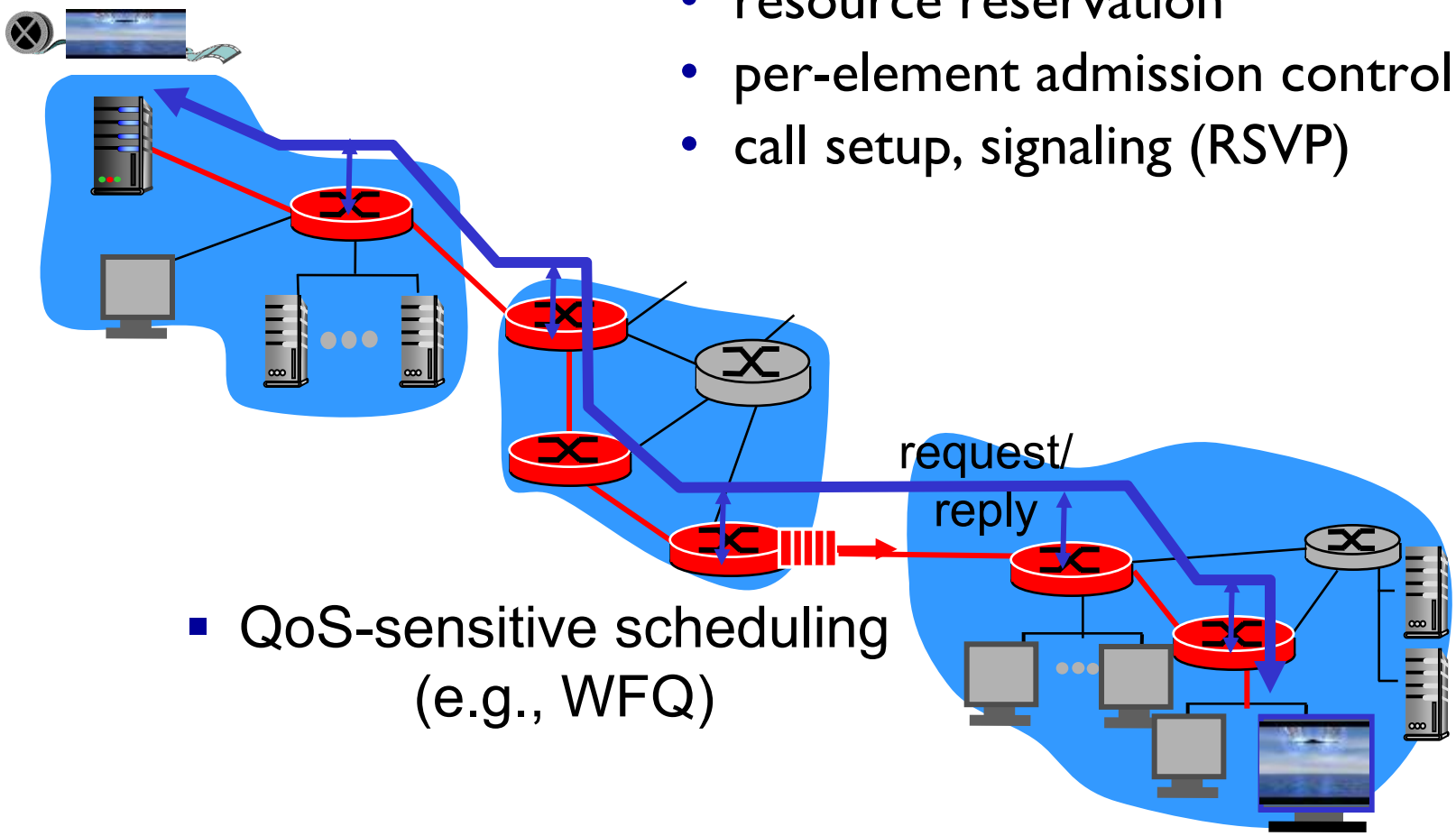


Principle 4

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

QoS guarantee scenario

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



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

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