# File Sharing

# File sharing

- Purpose: find a file and transfer it from multiple other users
- Service characteristics:
  - Search for file
  - Identify which users have which pieces
  - Transfer pieces and put it together
- Performance:
  - Loss – not ok
  - Delay – very flexible, often hours
  - Throughput – higher is better, but flexible

# File sharing: search

- Location?
  - e.g. tracker
- Client-server or peer-to-peer?
- Two functions:
  - Search
    - usually client-server
  - File transfer
    - peer-to-peer

# File sharing: search

client-server followed by peer-to-peer, e.g. gnutella

- server: put peer in contact with peers
- peer-to-peer: file transfer

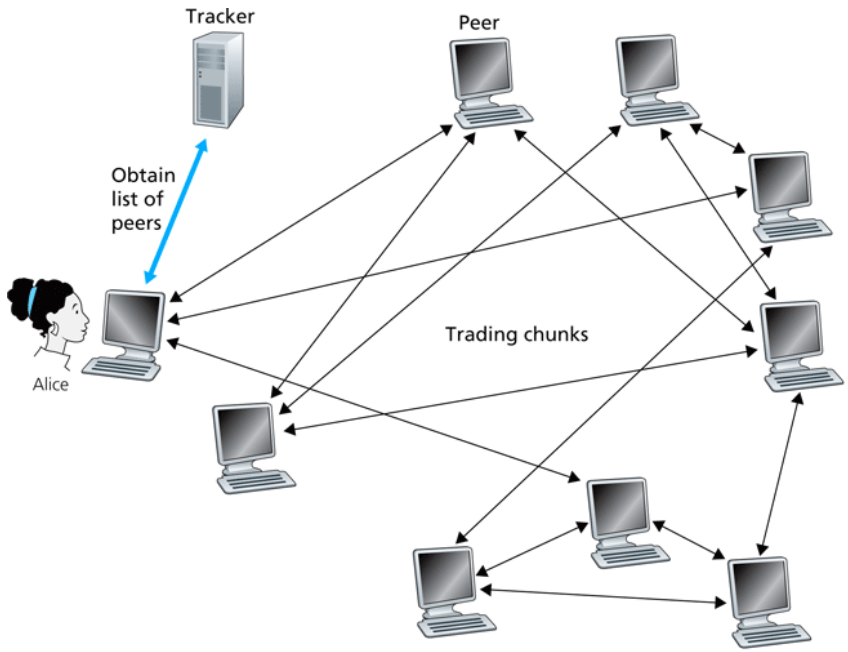



**Figure 2.26** ♦ File distribution with BitTorrent

# File sharing: Method & Connection Management

- Method
  - e.g. bittorrent, …
- Connection Management
  - often uses a large number of unregistered ports
  - peers determined by protocol
  - connections set up directly to peers

# File sharing: action management

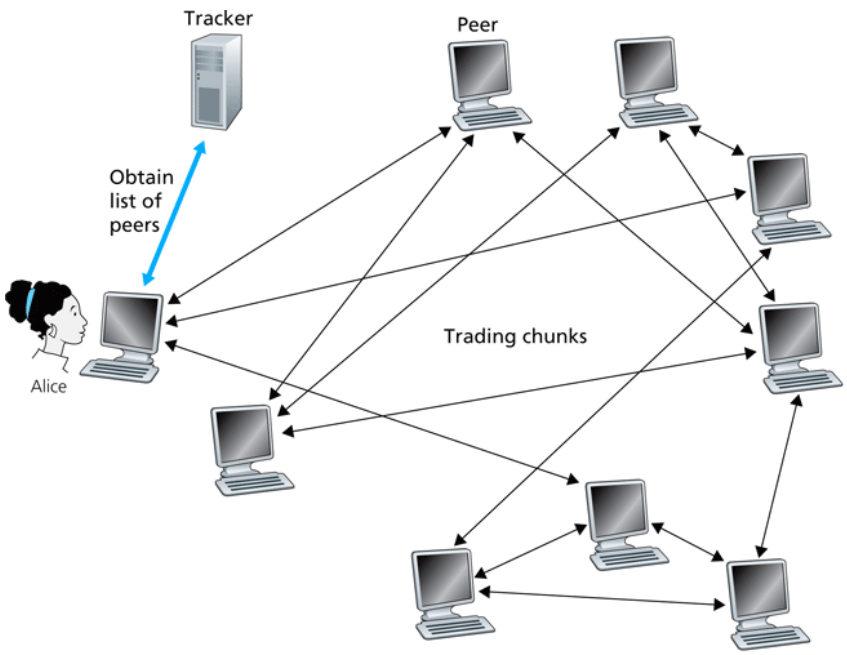



**Figure 2.26** ♦ File distribution with BitTorrent

Kurose

BitTorrent:

- File split into "chunks"
- Client side:
  - Request missing chunks directly from other peers, via TCP
- Server side:
  - Listen for and service requests for chunks you have, via TCP

# File sharing: action management

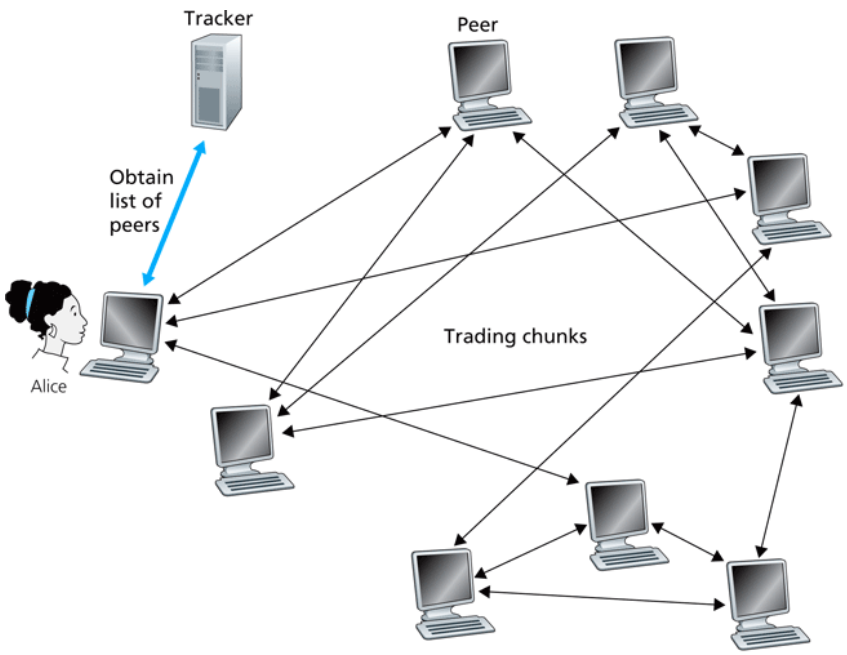



**Figure 2.26** ♦ File distribution with BitTorrent

Kurose

BitTorrent program:

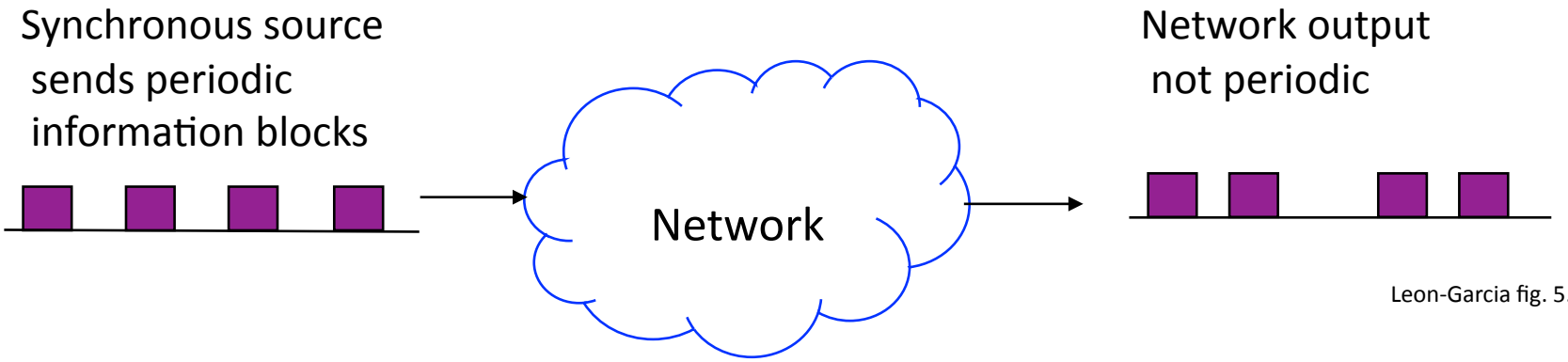
- Client algorithm determines which chunk to request first, e.g. rarest first
- Server algorithm determines rate, e.g. number connections, max rate

# Multimedia

# Streaming

- Purpose: 1 way transmission of audio or video
- Service characteristics:
  - Constant bit rate (unless compressed)
  - Duration = mins to hours
- Performance:
  - Loss – small amount ok
  - Delay – seconds, firm once stream started
  - Throughput – fixed

# Streaming




Leon-Garcia fig. 5.27

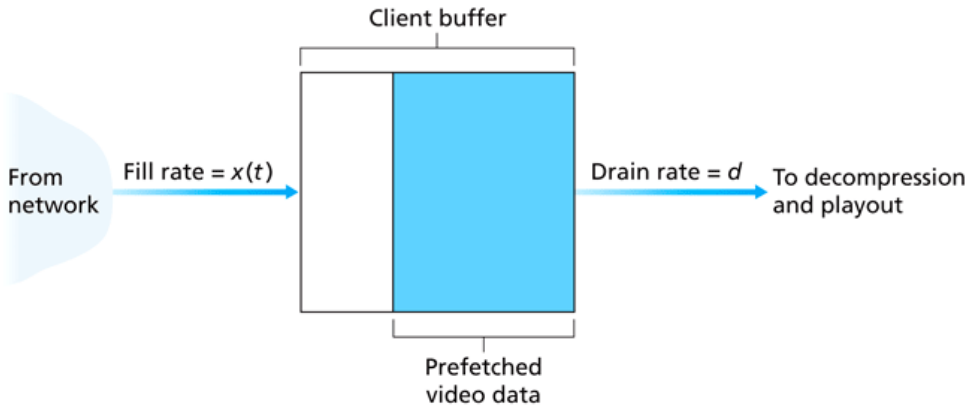



**Figure 7.3** ♦ Client buffer being filled at rate $x(t)$ and drained at rate $d$

Kurose

# Voice or Video over IP

- Purpose: 2 way interactive transmission of voice or video
- Service characteristics:
  - Constant bit rate (unless compressed)
  - Duration = mins to hours
- Performance:
  - Loss – small amount ok
  - Delay – a few tenths of a second
  - Throughput – fixed

# Voice or Video over IP

- Crude version: similar to streaming
- Better version: give priority to these packets over packets such as email or web browsing

Location

# Location

- Location identifier (application method independent)
  - Part of a Uniform Resource Locator (URL)
    - commonly a domain name or IP address
- Search
  - Give provider identifier
  - Return domain name or IP address
- Caching
  - May redirect you to a location closer to you

# Location

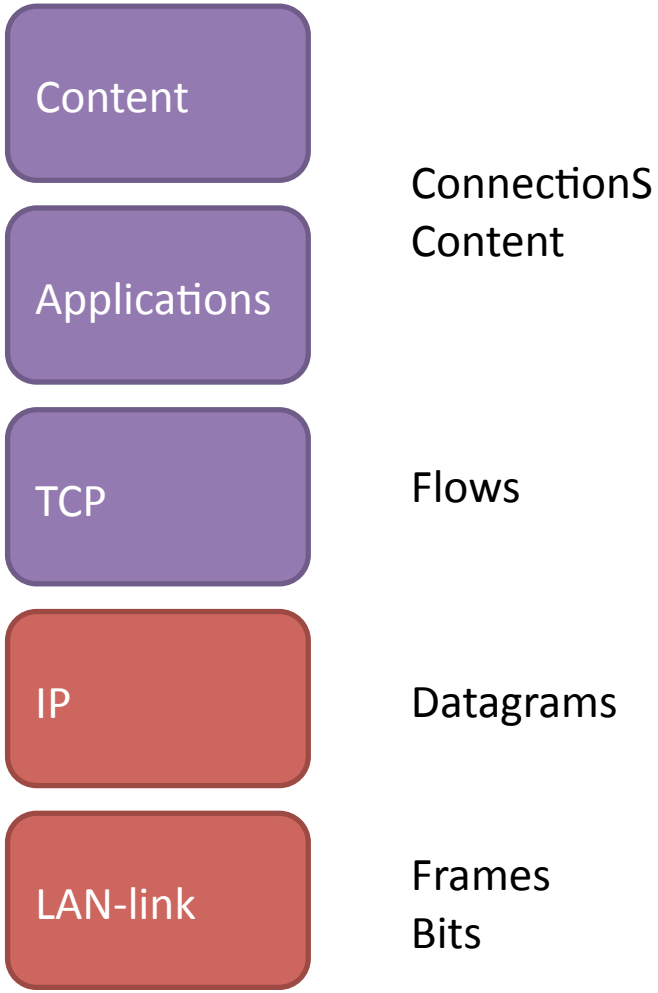
- Source location(s)
  - client/server or peer-to-peer?
- Destination location(s)
  - stationary or mobile?

# Location Model Examples

- http:
  - Distribution of popular webservers
- Email:
  - Distribution of email source/destination pairs
- Streaming:
  - Distribution of popular streaming servers
- VoIP:
  - Distribution of who is calling whom
  - Plus perhaps mobility

# Connections

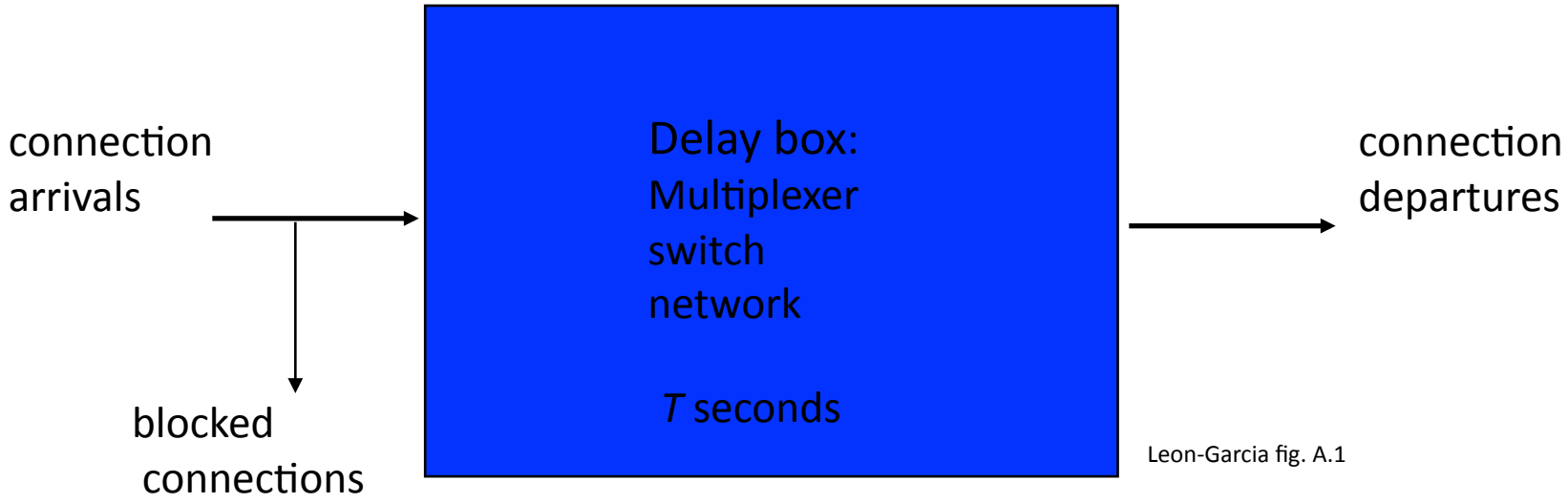
# Traffic characterization by layer



# Connection Management

- Connection Initiation
  - Connection access control
    - Server may block a connection
  - Server balancing
    - Server may redirect a connection
- Connection Termination
  - When?

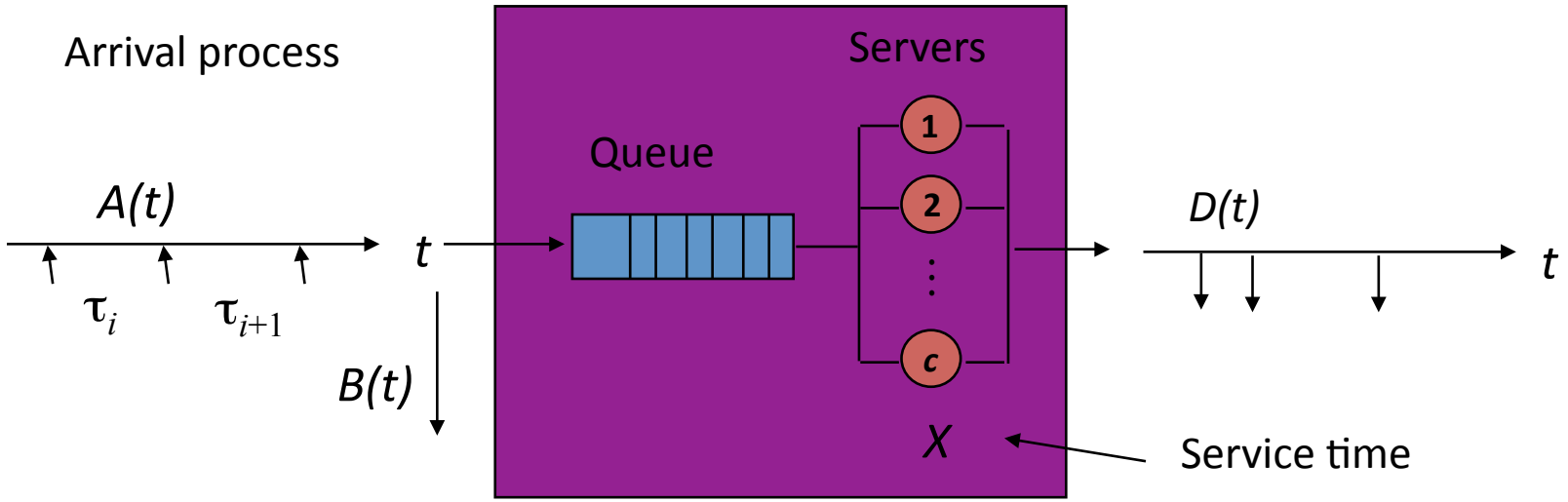
# Connection models




Leon-Garcia fig. A.1

- Arrivals
- Duration
- Connection access control

# Connection Queuing Model




Leon-Garcia fig. A.6

# Connection arrivals

- Exponential arrivals
  - when users start connections at times that are
    - independent of other users
    - independent of the user's other connections
    - same probability of starting a connection at every time
- Arrival Rate
  - $\lambda = 1/E[\tau]$
  - likely slowly changing hour by hour, day by day

# Blocking

- Connection Access Control (CAC)
  - System may block a connection
    - if not enough resources
    - or if enough resources, but they can be used better
  - System may queue connection requests
    - connection buffer
  - System may redirect connection requests
    - e.g. server balancing, content distribution networks

# Connection duration

- Communication
  - Exponential durations
    - when users end connections at times that are
      - independent of other users
      - independent of the user's other connections
      - same probability of ending a connection at every time
- Service Rate
  - $\mu = 1/E[X]$
  - likely dependent on type of communication, not on time of day

# Connection duration

- Content
  - Probably not an Exponential duration
  - Duration = file size / throughput
- File size distribution
  - Depends on type of content
  - Often a heavy-tailed distribution
    - e.g. if $X \sim \text{Exp}(\mu)$, then $e^X$ has a Pareto distribution,
  - Better to look at $P(X>x)$ than $P(X=x)$

# Connection duration

- Service rate
  - Throughput
  - Depends on lower layers, including
    - Application: number of parallel connections
    - Transport: effect of flow and congestion control
    - Network: effect of packet scheduling & dropping
    - Link: effect of multiple access

# Queue notation

*Arrival Process / Service Time / Servers / Max Occupancy*

| Interarrival times $\tau$ | Service times $X$ | 1 server | $K$ customers |
|---|---|---|---|
| M = exponential | M = exponential | $c$ servers | |
| D = deterministic | D = deterministic G = general | infinite | unspecified if unlimited |
| G = general | | | |

Arrival Rate:

$\lambda = 1/E[\tau]$

Service Rate:

$\mu = 1/E[X]$

Leon-Garcia fig. A.7

# Connection model examples

- http:
  - Arrivals
    - "M"
    - with queuing
    - possibly with server balancing or content distribution
  - Duration
    - depends on file size distribution
    - and on throughput
    - and on user impatience (termination)
    - and on persistent vs. non-persistent
    - and on number of parallel connections

# Connection model examples

- Email:
  - Arrivals
    - "M"
    - with queuing
  - Duration
    - depends on file size distribution
    - and on throughput
  - But really multiple queues
    - Client
    - Server(s)
    - Queuing network?

# Connection model examples

- Streaming:
  - Arrivals
    - "M"
    - with queuing
    - likely with server balancing or content distribution
  - Duration
    - depends on file distribution (but not necessarily size)
    - and on user behavior (termination)

# Connection model examples

- VoIP:
  - Arrivals
    - "M"
    - with no queuing
    - possibly with blocking
  - Duration
    - "M"
    - and on network behavior (termination)

# Action Management

- During connection, do whatever the application needs to do!
- Application layer at source and destination coordinate through messages
- State: information about current status of the application instance
  - May be maintained by client, server, and/or peer application
  - May be maintained in a file

# Connection Action models

- Actions taken by each side to manage an ongoing connection
- Likely modeled by Finite state machines

# Action Model Examples

- http:
  - File requests
- Email:
  - Upload
  - Download
- Streaming:
  - Pause
  - Rewind or fast forward
  - Rebuffer
  - Change encoding rate
- VoIP:
  - Change encoding rate
  - Handoff

# Voice Over IP

# VoIP

- Real time application
- Multimedia streams
- Both directions

# QoS Requirements

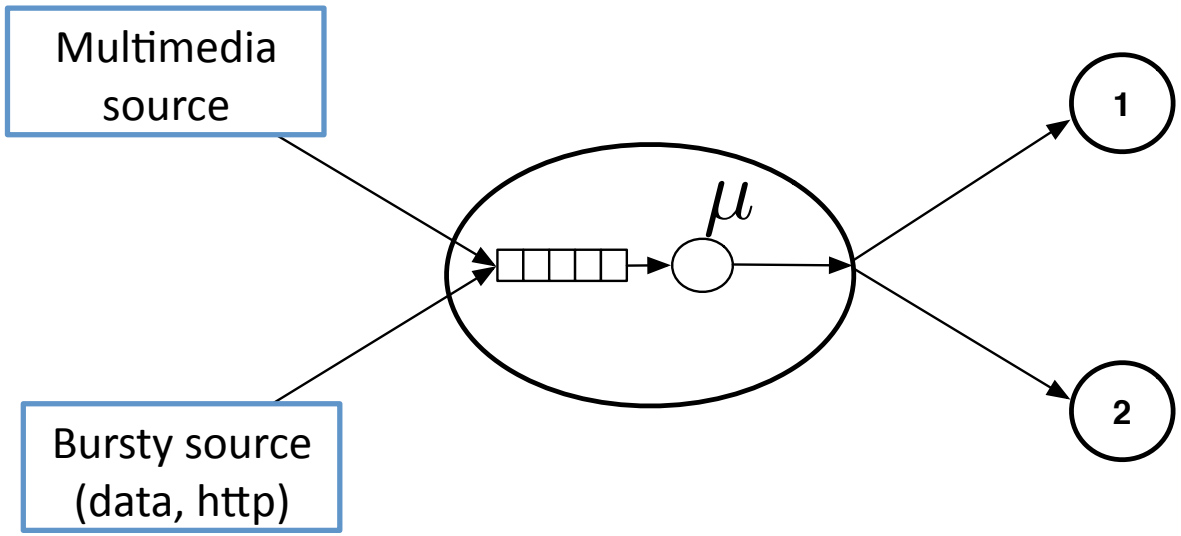
- Packet loss
  - Up to 20% is tolerated
  - Packet losses
    - Buffer overflow
    - Link layer
    - Delay
  - UDP vs TCP
    - Reliability (retransmissions)
    - Delay
    - Buffer starvation
    - Delay variations

# QoS Requirements

- End-to-End delay
  - Sum of all the
    - Transmission
    - Propagation
    - Processing
    - Queueing delays
  - Up to 400ms tolerable

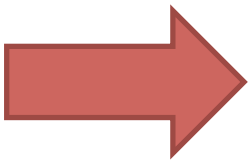
# QoS Requirements

- Jitter
  - Packets generated periodically
  - Delay variations at the receiver
    - E.g., queue conditions



# Best effort

- Individual pkt end-to-end performance is random
- Possibly large variations
- Average may vary over time



Voice over IP?

# Jitter – Countermeasure

- Timestamp
  - Generation time
- Delayed playout
  - (Most of the) Packets arrive before playout time
  - Introduces delay
  - Packets after playout time are discarded

# Jitter – Countermeasure

- Delayed playout
  - Delay is a random variable
  - Variations due to network conditions
  - Min delay with loss constraint
  - Fixed playout
    - t + q
    - Generation + delay + max variations
    - Large variations = large delays
  - Adaptive playout
    - Talk spur = delay re-estimated
    - Fixed during talk spur
    - Recent measured delays used for estimation

# Jitter – Countermeasure

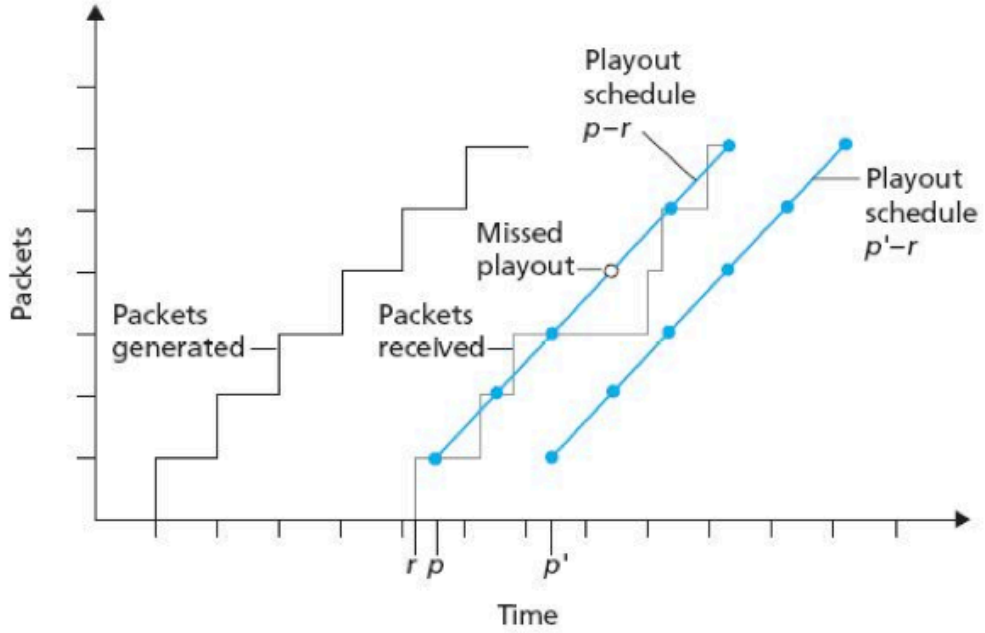



**Figure 7.7** ♦ Packet loss for different fixed playout delays

Kurose-Ross 7.7

# Packet Loss Recovery

- TCP
  - Retransmission
  - Delay!
- Recovery
  - FEC
  - Interleaving
  - No Additional RTT

# FEC

- Redundancy
  - +1 pkt every N
  - N small
    - Larger generation rate
    - Better recovery
  - Delay: wait for the entire group
- Low rate stream
  - Low-quality/Low-bitrate stream appended

# FEC

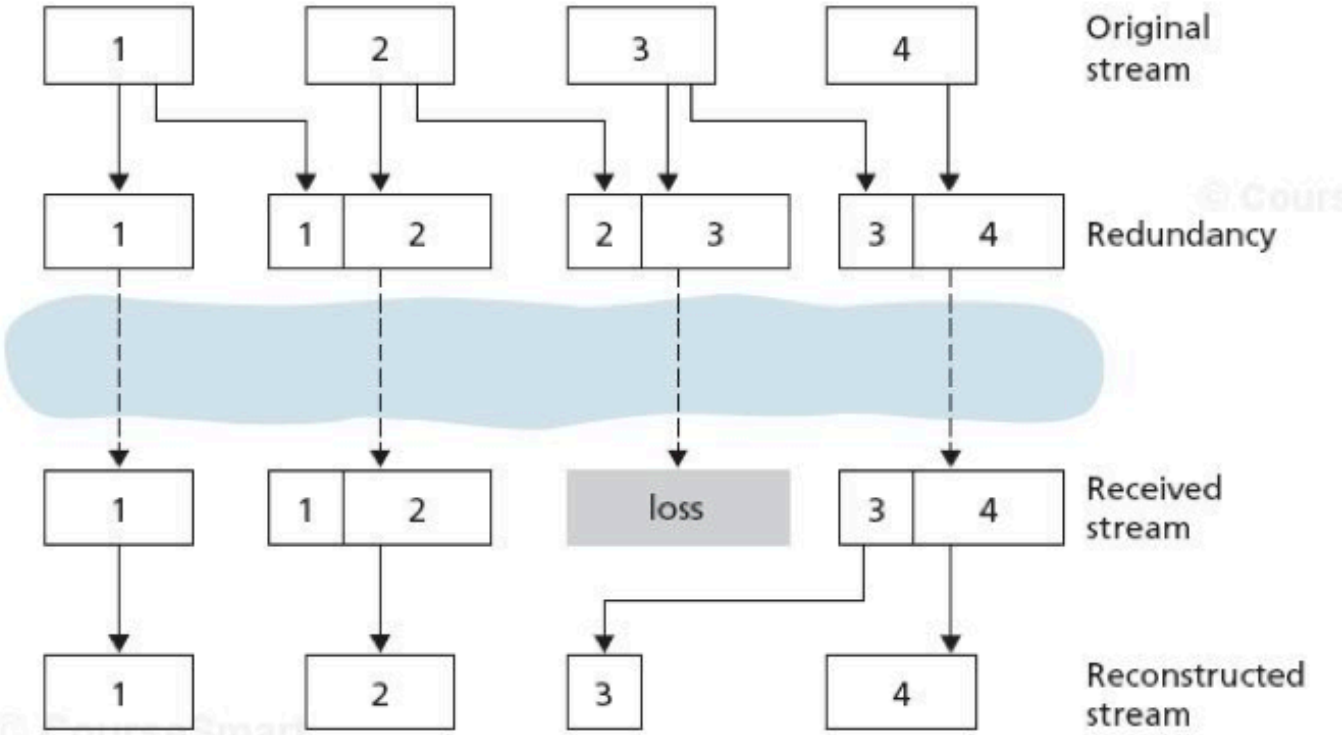



**Figure 7.8** ♦ Piggybacking lower-quality redundant information

Kurose-Ross 7.8

# Interleaving

- Samples are resequenced
  - Adjacent samples assigned to different chunks
- Packet loss mitigated
  - Avoids gaps
- Increased latency
- Same bandwidth

# Interleaving

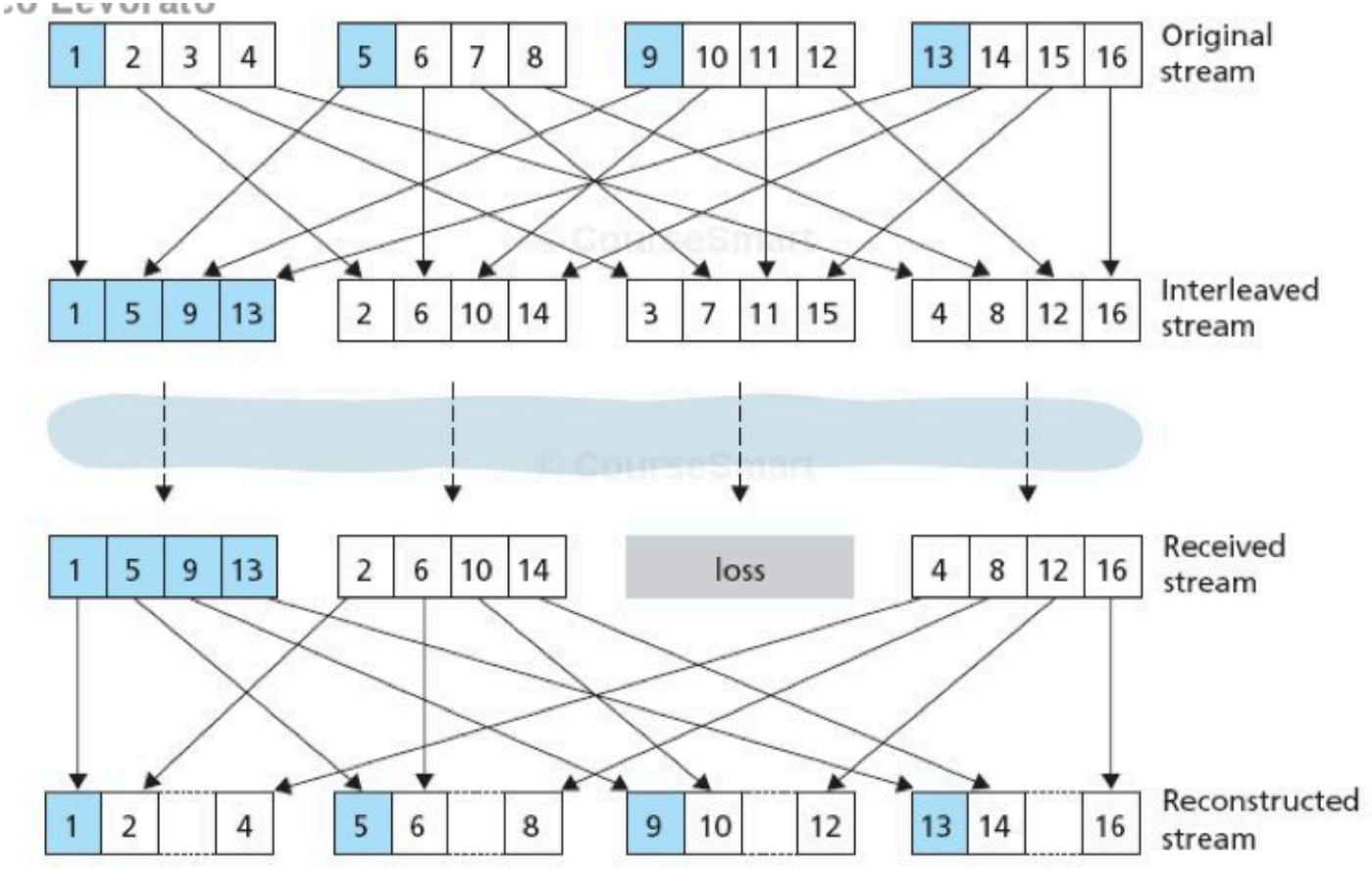



**Figure 7.9** ♦ Sending interleaved audio

Kurose-Ross 7.9

# Real Time Protocol (RTP)

- UDP
  - RTP – UDP - IP
- RTP
  - Independent RTP stream per source
  - Video/audio payload + header

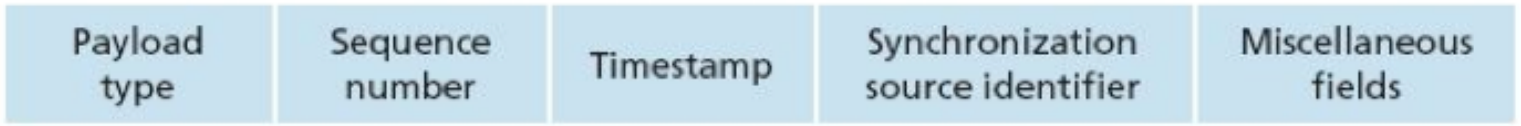



**Figure 7.11** ♦ RTP header fields

Kurose-Ross 7.11

# Real Time Protocol (RTP)

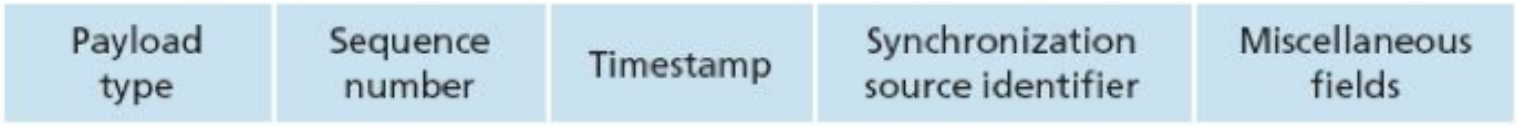



**Figure 7.11** ♦ RTP header fields

Kurose-Ross 7.11

- Payload type
  - Encoding
- Sequence number
  - Re-sequencing
  - Packet recovery
- Timestamp
  - Playout control
- Synch. Source ID
  - identification

# Supporting Multimedia Applications

# Supporting Multimedia

- Network dimensioning
  - Enough bandwidth to support QoS
- Differentiated service
  - Hierarchy of priorities
- Per-connection Guarantees
  - End-to-end resource reservation

# Dimensioning

- Avoid congestion
  - Links have enough bandwidth
  - No loss, small delay, small jitter etc.
- No changes to best-effort model
- End-to-end
  - Multiple ISP – cooperation
- How much is enough?
  - Traffic demand
  - Performance requirements
  - End-to-end performance prediction

# Multiple Service Classes

- Multimedia/priority first, then the others
  - Priority per class and not per user/stream
  - Improved service
  - Avoids congestion

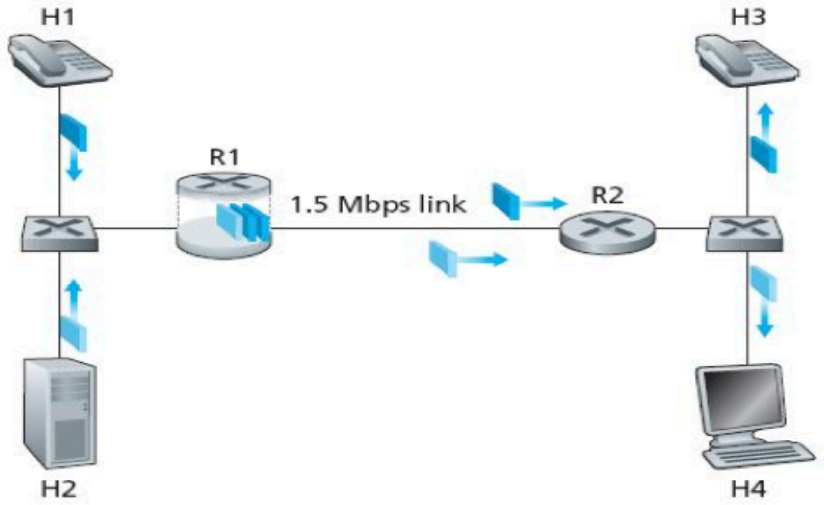



**Figure 7.14** ♦ Competing audio and HTTP applications

Kurose-Ross 7.14

# Multiple Service Classes

- Operations
  - Packet marking
  - Router processing
  - End-to-end

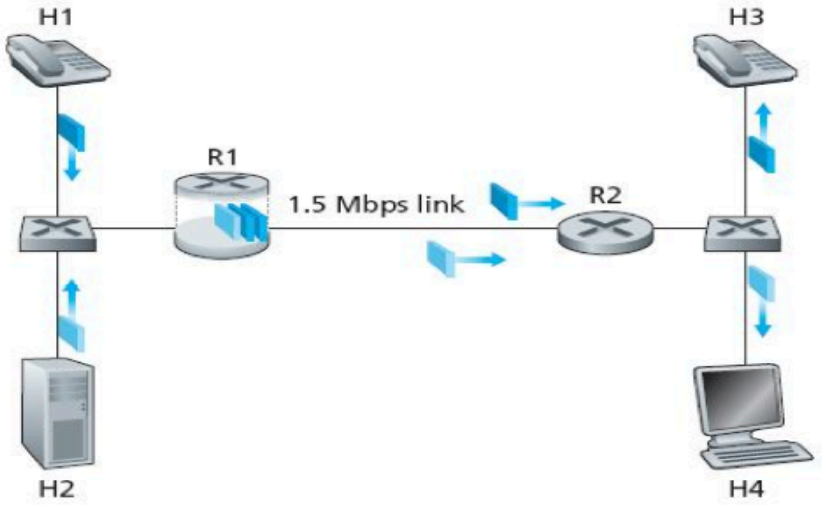



**Figure 7.14** ♦ Competing audio and HTTP applications

Kurose-Ross 7.14

# Multiple Service Classes

- Issues
  - Too many prioritized streams or too much prioritized traffic
  - Congestion of low-priority traffic
- Solutions
  - Policing
    - Traffic control (router)
    - Drop/delay packets
  - Fixed allocation
    - Link level scheduling
- Efficiency?

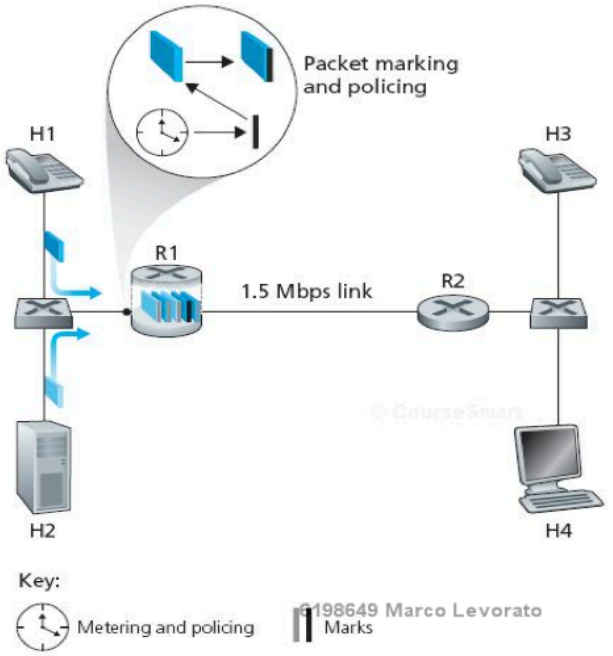
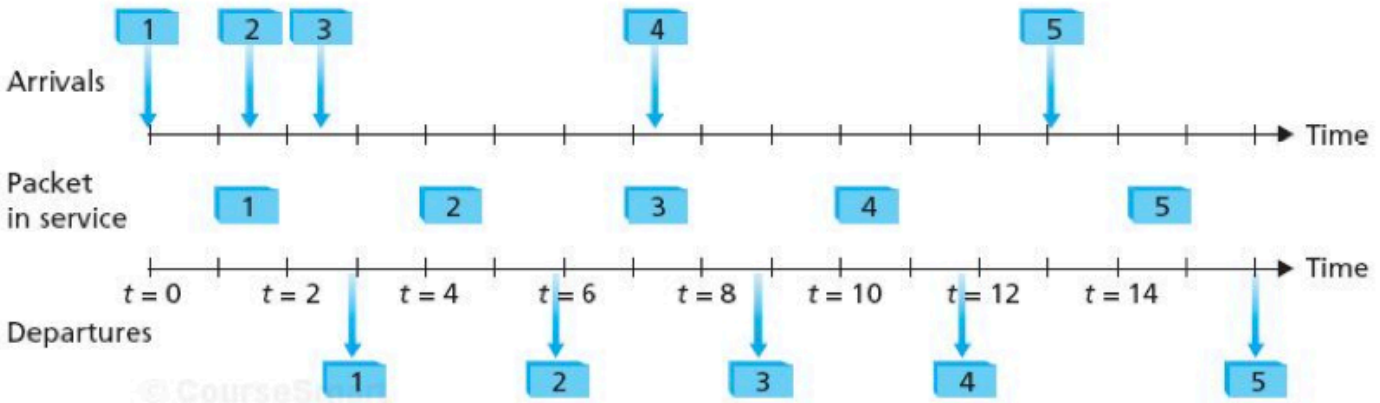



**Figure 7.15** ♦ Policing (and marking) the audio and HTTP traffic classes

Kurose-Ross 7.15

# Scheduling

FIFO




Kurose-Ross 7.18/20

Priority

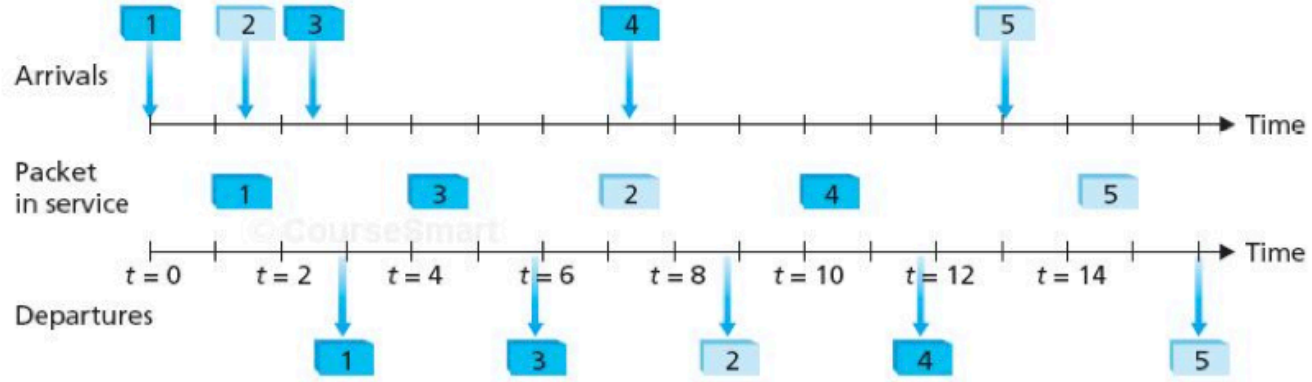
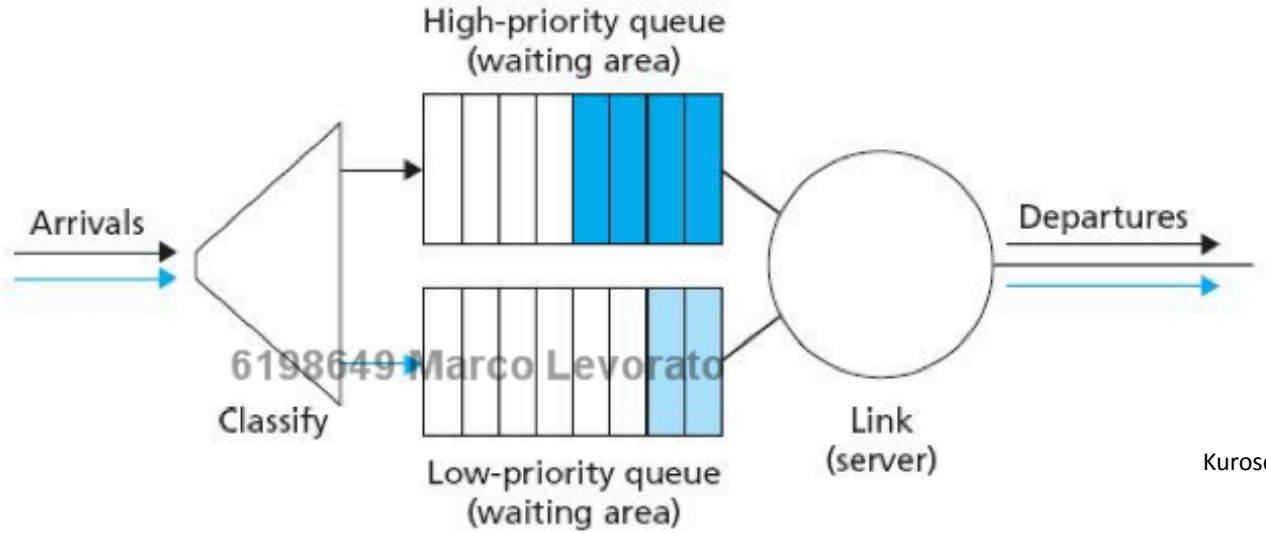



**Figure 7.20** ♦ Operation of the priority queue

# Priority

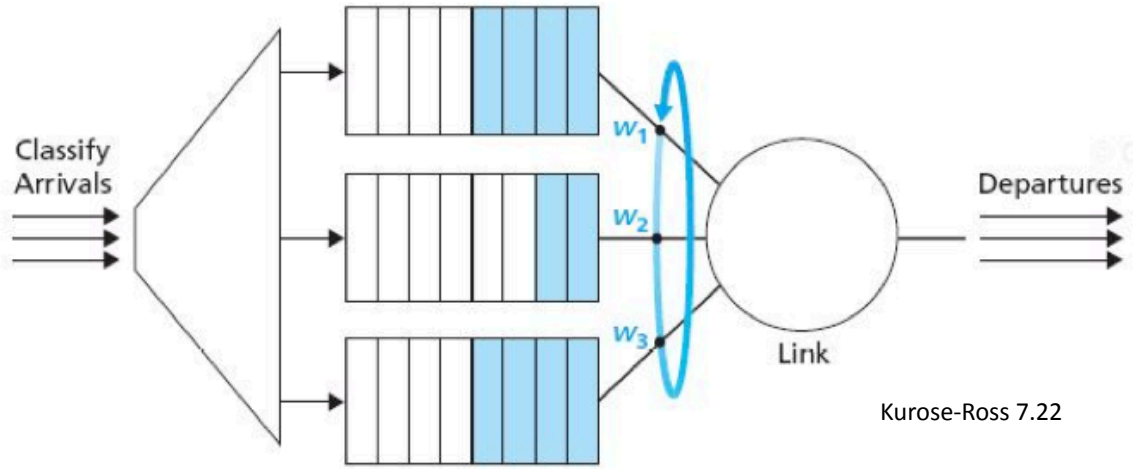



Kurose-Ross 7.19

**Figure 7.19** ♦ Priority queuing model

- Preemptive
  - Service is interrupted
- Non-preemptive
  - Service is not interrupted

# Weighted Round-Robin

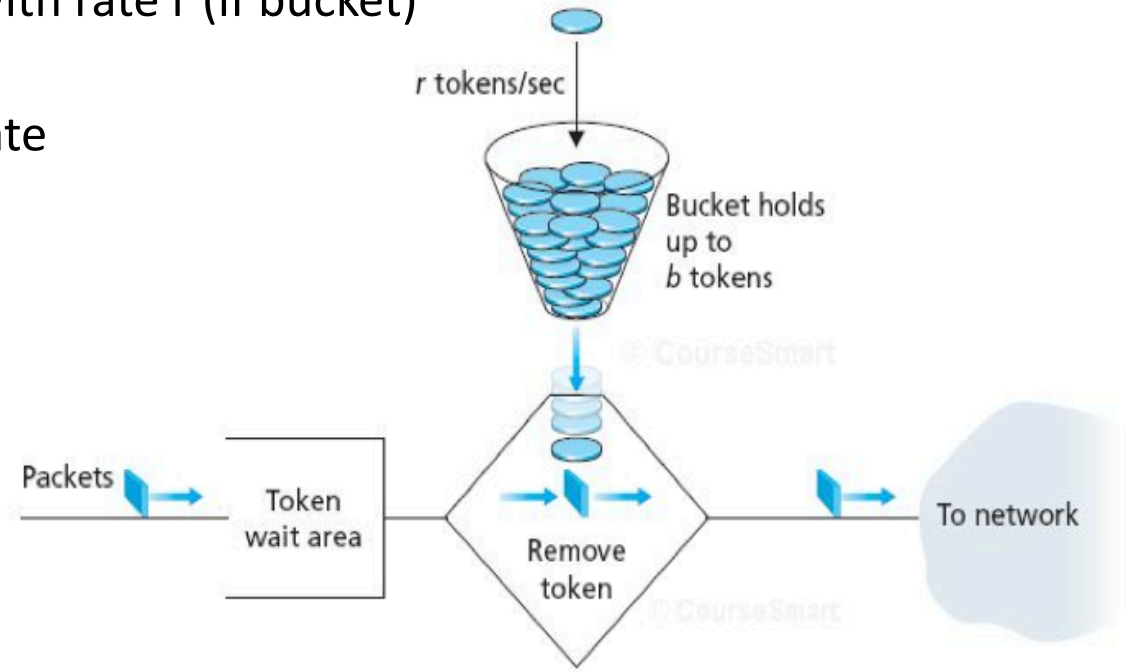



Kurose-Ross 7.22

- Round-robin
  - Classes with non-empty queue sequentially served
- Weighted fair queueing
  - Weight defines amount of time
  - Minimum fraction (due to empty queues) $\frac{w_i}{\sum_j w_j}$

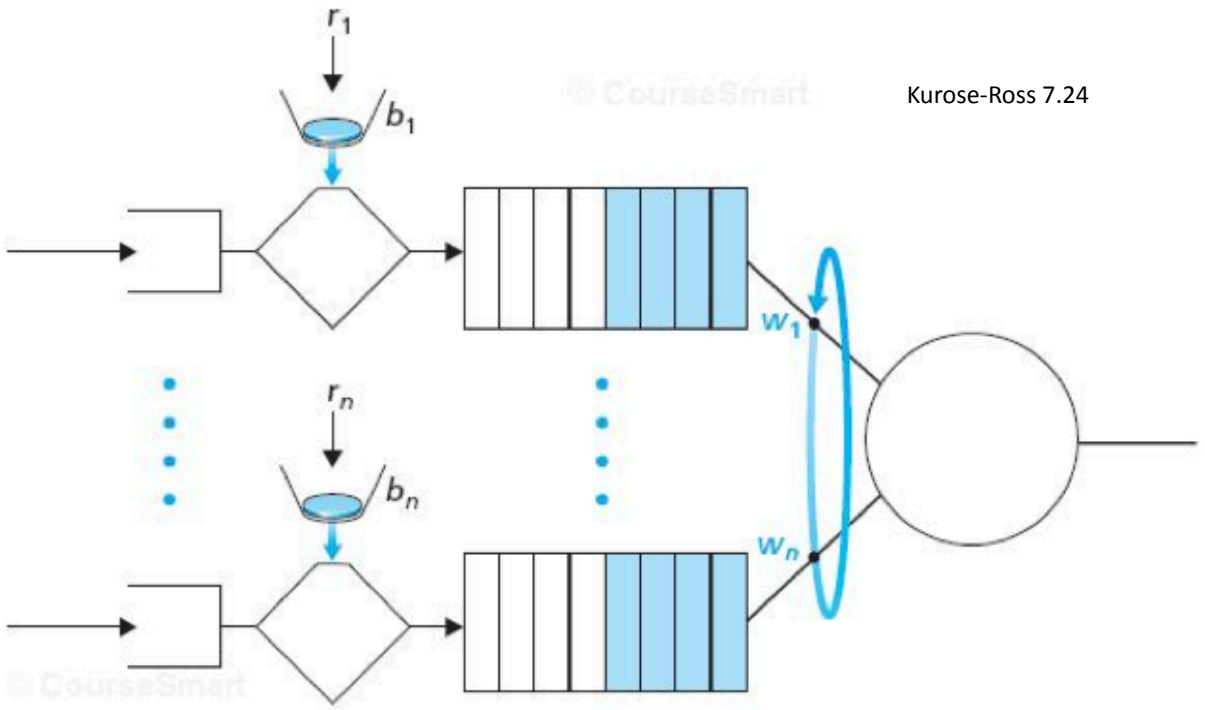
# The leaky-bucket

- Limited injection of traffic in the buffer(s)
  - B tokens
  - Tokens assigned to incoming packets
  - Tokens generated with rate r (if bucket)
- Policing
  - Average injection rate
  - Peak rate
  - Burst size
- Multiple buckets




Kurose-Ross 7.23

# leaky-bucket + Weighted round robin



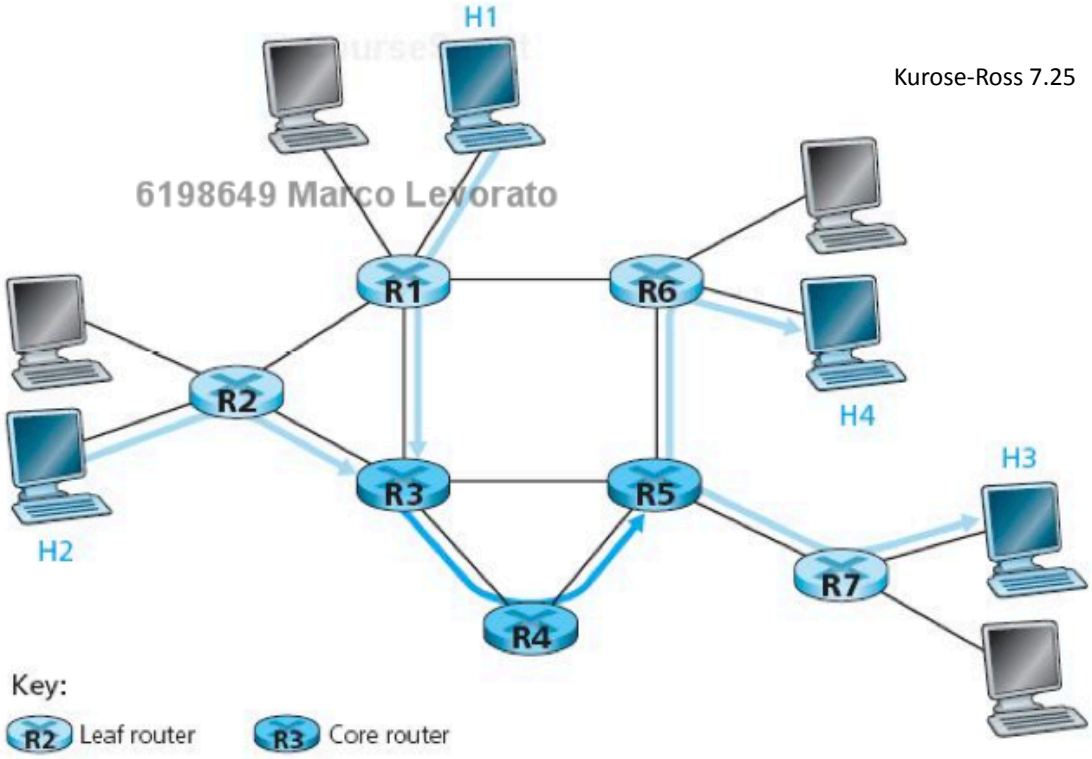

Kurose-Ross 7.24

$$\text{if} \quad r_1 < Rw_i / \sum_j w_j \quad \text{then max delay is} \quad d_{max} = \frac{b_1}{Rw_i / \sum_j w_j}$$

# DiffServ

- Supports service differentiation
- Edge functions
  - Packet classification/marking
  - Traffic conditioning
- Core functions
  - Per-hop behavior only function of the class

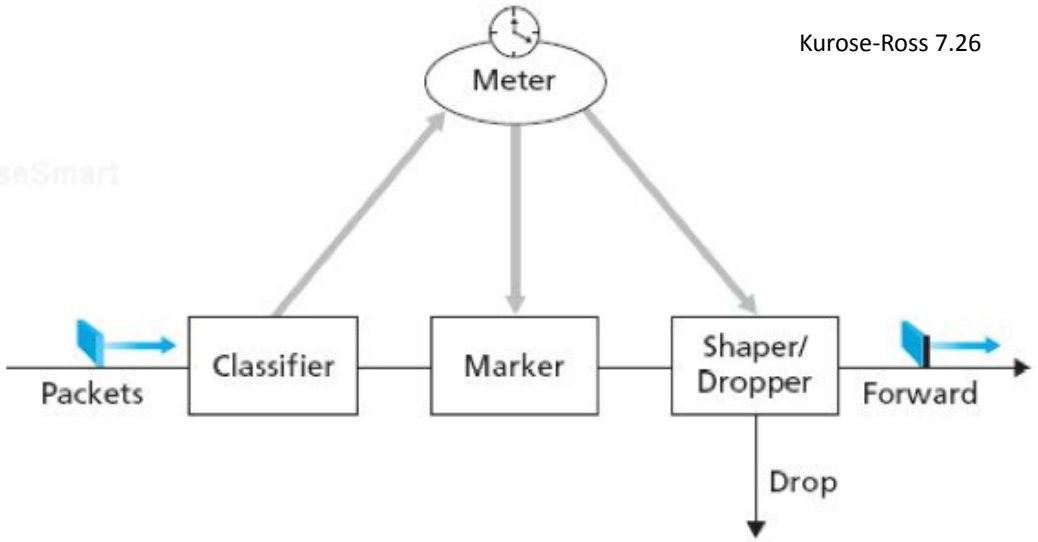
# DiffServ




Kurose-Ross 7.25

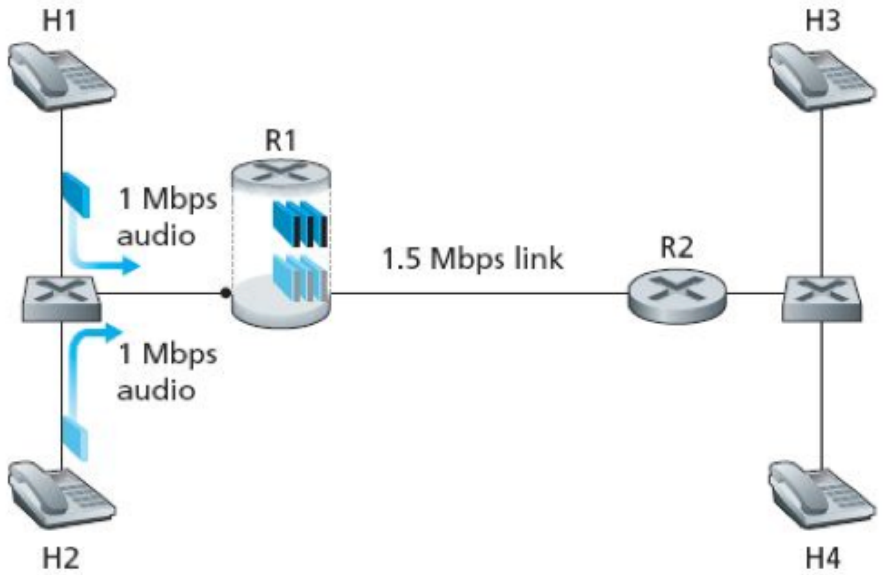
# Traffic conditioning

- Pre-negotiated characterization
- Leaky bucket



# Per-connection QoS

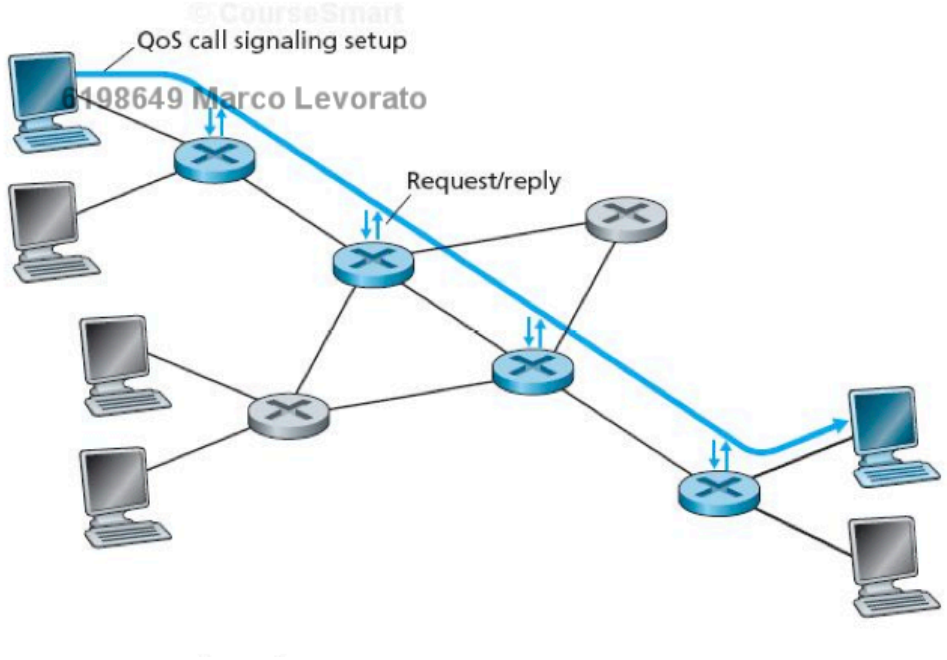
- End-to-end resource is pre-assigned
- QoS guarantees




Kurose-Ross 7.27

- Stream admission
- Avoid unusable flow

# Per-connection QoS




Kurose-Ross 7.27

- Stream admission procedure
- Setup signaling
- RSVP protocol

# Exercises!!