# Lecture 13

## Avoiding buffer overflow

- TCP needs pkt loss to discover congestion (caused by buffer overflow)
- alternative to avoid overflow:
  - use RTT to measure average queue length (TCP Vegas)
  - randomly drop pkts before buffer overflows (RED)
  - explicit congestion notification

## RED

- Router observes average queue length to detect congestion
- To notify end-hosts, router randomly drops pkt already before overflow occurs.
- RED is an example of Active Queue Management (AQM):
  - drop some pkts (or mark them as ECN) to let sources reduce sending rate already before buffers overflow

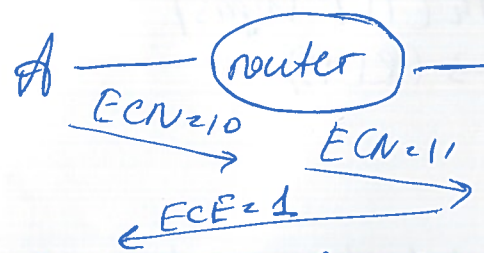
## Explicit Congestion Notification

- Idea:
  - routers tell end-hosts explicitly that there is congestion, rather than having the end-hosts discover the congestion implicitly.
- Possible approaches:
  - Congested router sends back a ECN to source
  - Congested router modifies a ECN bit in pkt, and let dest notify source → used in Internet
  - Define special pkts that are sent by src to probe congestion level at routers

## ECN for IP

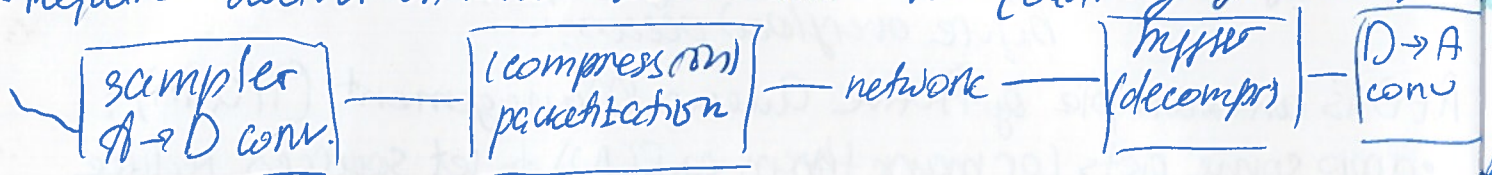
- uses 2 (unused) bits in IP header
- 00 – transport layer is not ECN capable
  01 – TL is ECN capable
  10 – TL is ECN capable
  11 – ECN capable and congestion has been experienced
- uncongested router leaves these bits unchanged
- congested router drops pkt if TL is not-ECN capable or sets bits to 11 if TL is ECN capable

- ECN at the TL
  - Router marks congestion in pkts flowing from A to B
  - However, A must reduce its sending rate
  - Thus, TL must echo the congestion-experienced notification back to A



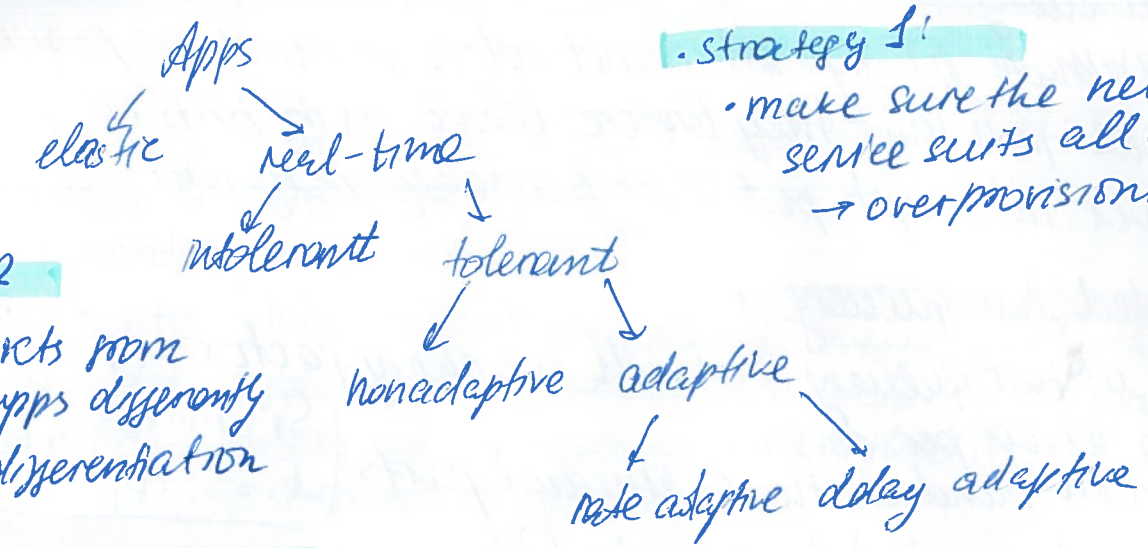

- In TCP: designate one (unused) bit as the ECN-echo bit (ECE)

- Real-time applications
  - Require "deliver on time" assurances: QoS (Quality of Service)




- Example audio application:
  - Sampler's A/D converter produce one byte 8000 times/sec
  - Samples need to be played back again at 8000 times/sec
  - Network delays pkts by variable amounts
  - Network may drop pkts
- In Internet:
  - IP (so is UDP) provides best effort:
    - no delivery guarantees
    - no delay guarantees
  - TCP uses retransmissions to provide:
    - delivery guarantees
    - possibly very long delays
  - TCP is unsuitable: retransmissions would arrive too late
    → use UDP and need to handle:
      - jitter (var of delay)
      - pkt loss

QoS




- strategy 1:
  - make sure the network service suits all apps
    → overprovisioning
- strategy 2
  - treat pkts from different apps differently
    → QoS differentiation

scheduling discipline
- resource: link capacity
- decision: which pkt to send next

drop policy:
- resource: buffer space
- decision: when to drop which pkt in the queue

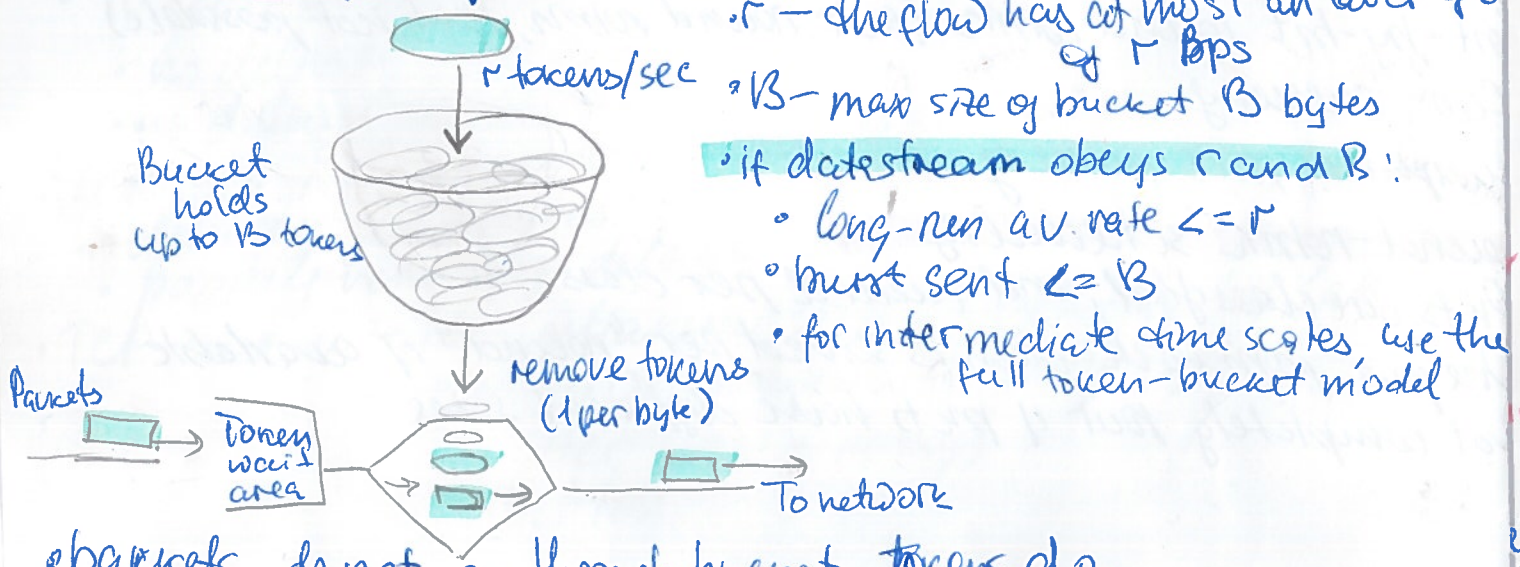
scheduling
- FIFO = first in, first out
- priority (pkts are classified; lower class is served if there are none in higher class waiting)
- round robin
- bit-by-bit round robin (fair round robin, but not possible)
- fair queuing
- weighted fair queuing

round-robin scheduling
- Pkts are classified, and queued per class
- One pkt from each class is served per "round" if available
- Not completely fair if pkts have different sizes

- Fair Queueing
  - approximate bit-by-bit round-robin as much as possible.
    - order pkts that they ~~would~~ leave as in bit-by-bit rr.
    - never interrupt pkt that is already being sent
- Weighted fair queueing
  - same as fair queueing, but with weighting factor (same weight)
  - # bits served per class
  - partition bandwidth in unequal parts

$$S_i = \max(F_{i-1}, A_i)$$
$$F_i = S_i + P_i$$

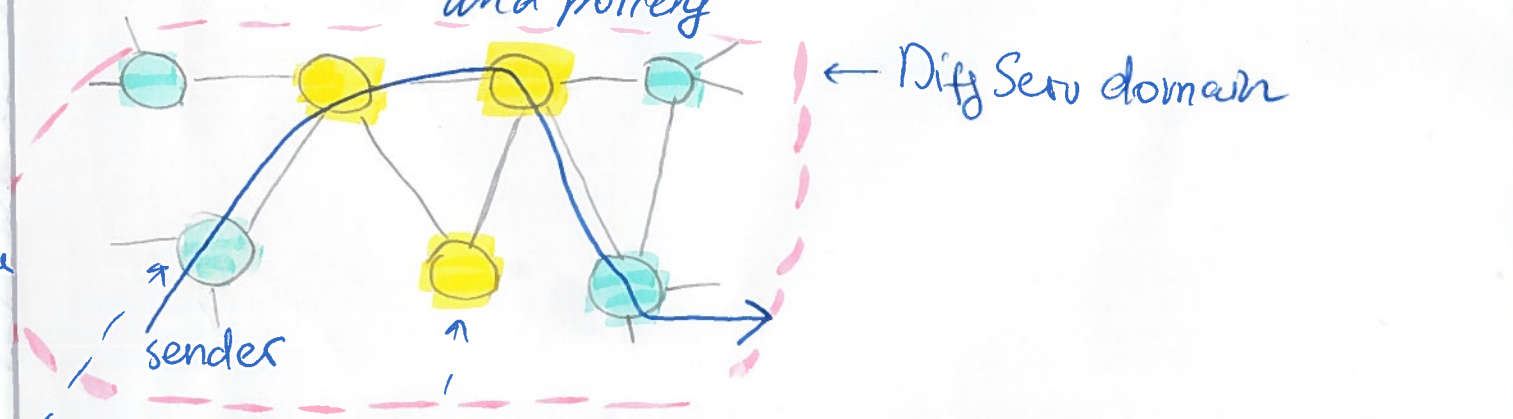
- Components needed to provide QoS
  - Queueing discipline
  - Pkt classification (which pkt which treatment)
  - Admission control
    - can network accept a new flow and provide QoS for old and new?
      - specification of requested service of the new flow
      - specification of the traffic characteristics of the new flow
  - Policing
    - do flows conform to their traffic specification?
- Token-bucket (leaky bucket) specification




  - r – the flow has at most an average of r bps
  - B – max size of bucket B bytes
  - if datastream obeys r and B:
    - long-run av. rate <= r
    - burst sent <= B
    - for intermediate time scales, use the full token-bucket model
  - packets do not go through bucket, tokens do.
  - it only describes how many pkts a system may send, when, but the system may use different method and never transmit more than allowed by this model.

QoS in Internet (3 approaches)

- Integrated Services
  - reserve capacity on every link for individual flows that need it
  - RSVP (Resource Reservation Protocol)
  - per flow reservations in routers
  - Admission control (token-bucket spec) prevents "overbooking" of the links
  - each router determines for every packet to which reservation it belongs (classification) and schedules it such that it gets the requested service
  - in theory:
    - guaranteed service – delay guarantees
    - controlled load service – flow can be given an experience of the network as if it is lightly loaded
  - scalability problems
- Differentiated Services
  - flows are assigned to a few classes, routers know only about classes
  - define a limited # traffic classes and specify for each class which service it can expect from router (PHB, "per hop behav")
  - classify pkts at the edge of the network, and mark their required PHB in the IP header
  - routers in interior of network handle (schedule) packets according to PHB; they do not need to do classifying and policing




es pkt classification (PHB in IP)
s traffic
anditions (incl. policing, dropping)
just forwards pkt based on PHB

- over provisioning
  - just provide more than enough bandwidth, and hope for the best

- Example PHB: "expedited forwarding"
  - rate at which EF is served >= configured rate R
  - over suitably defined interval
  - independent of the offered load of non-EF
  - can be defined mathematically
  - ↓ EF traffic → ↓ delay & pkt loss
  - Possibly implemented using WFQ
  - often used for VoIP traffic