CS120: Computer Networks

Lecture 18. Congestion Control 2

Haoxian Chen

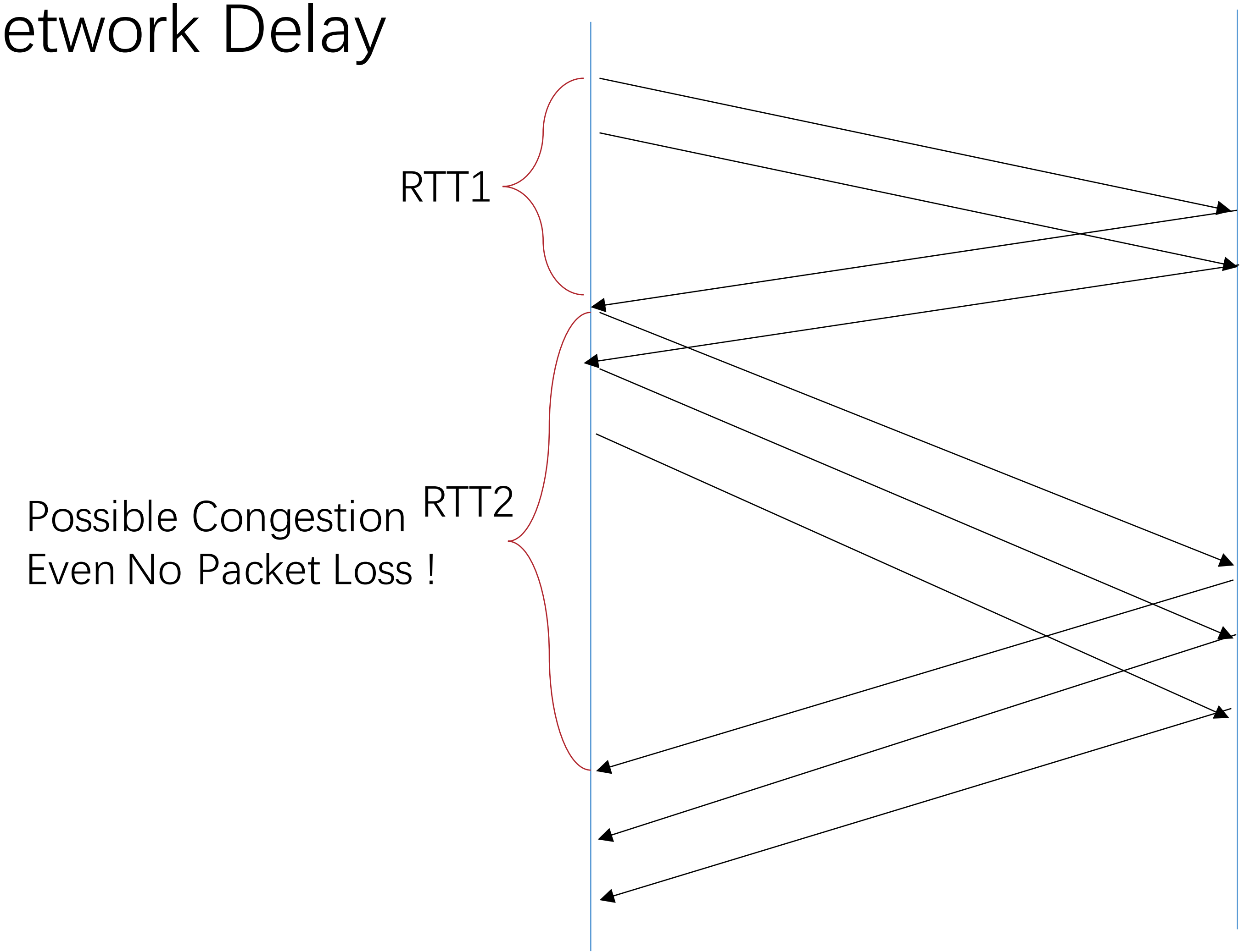
Slides adopted from: Zhice Yang

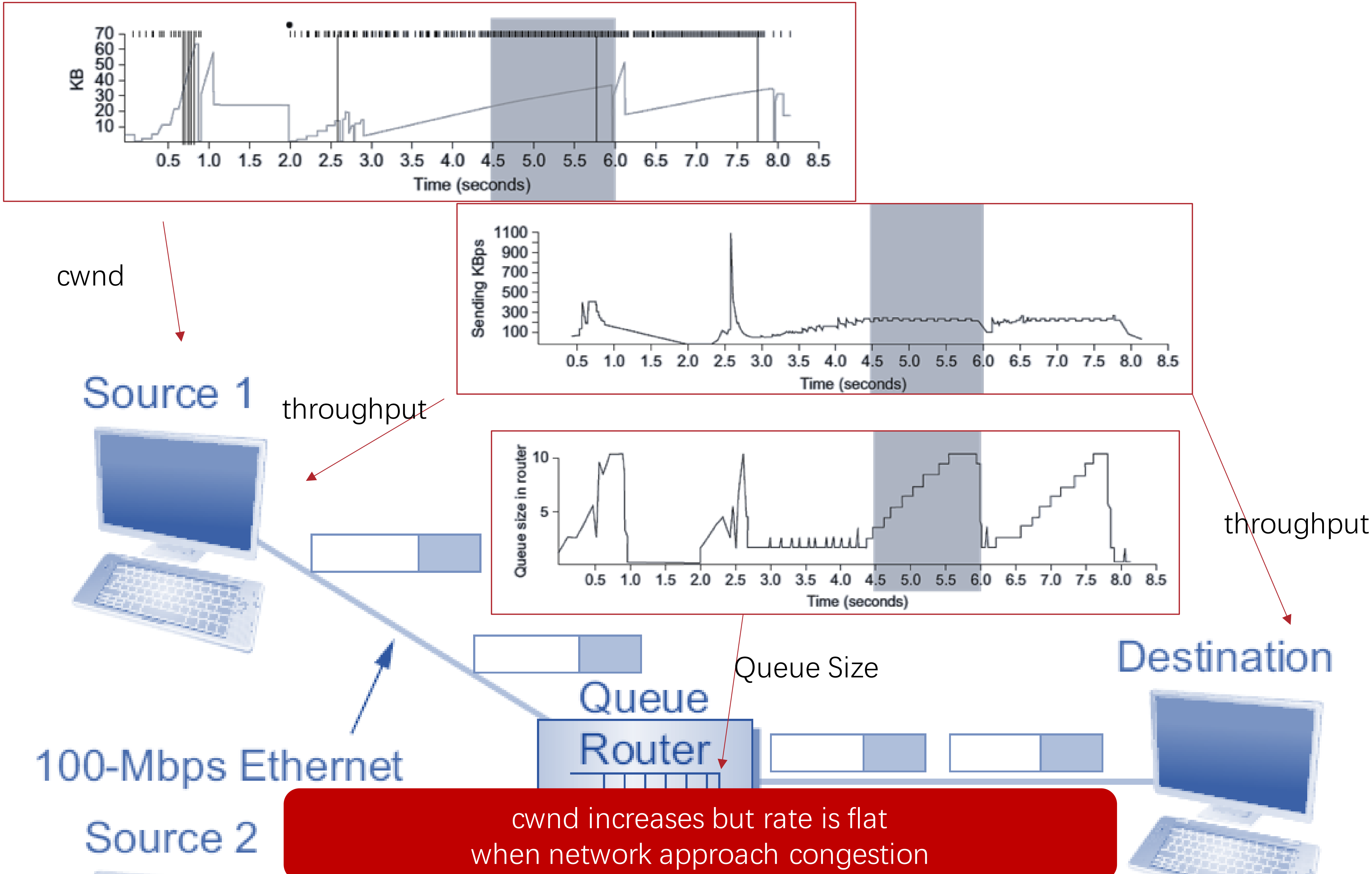
Congestion Control

- Host-based Congestion Control
  - Packet Loss
  - ➢ Delay
- Router-based Congestion Control
  - Queuing

Avoid Congestion

- Feedback: Network Delay





cwnd increases but rate is flat
when network approach congestion

TCP Vegas

Idea: use changes in throughput rate as congestion signal

- **ExpectedRate**: rate without congestions.
- **ActualRate**

When ActualRate << ExpectedRate: decrease congestion window.

When ActualRate is close to ExpectedRate: increase congestion window.

Keep full utilization

TCP Vegas Algorithm

- BaseRTT: the reference for increases in RTTs
  - The minimum RTT
    - If RTT < BaseRTT
      - BaseRTT = RTT
- ExpectRate:
  - CongestionWindow/BaseRTT
- ActualRate:
  - ActualRTT: according to timestamps
  - ActualRate=CongestionWindow/ActualRTT

TCP Vegas Algorithm

- Source compares ActualRate with ExpectRate
  - if ExpectRate - ActualRate < α
    - cwnd++
  - if α <ExpectRate - ActualRate < β
    - cwnd = cwnd
  - if β <ExpectRate - ActualRate
    - cwnd --

TCP Vegas

Top, congestion window; bottom, expected (colored line) and actual (black line) throughput. The shaded area is the region between the α and β threshold

TCP Vegas Algorithm

Why linear decrease?

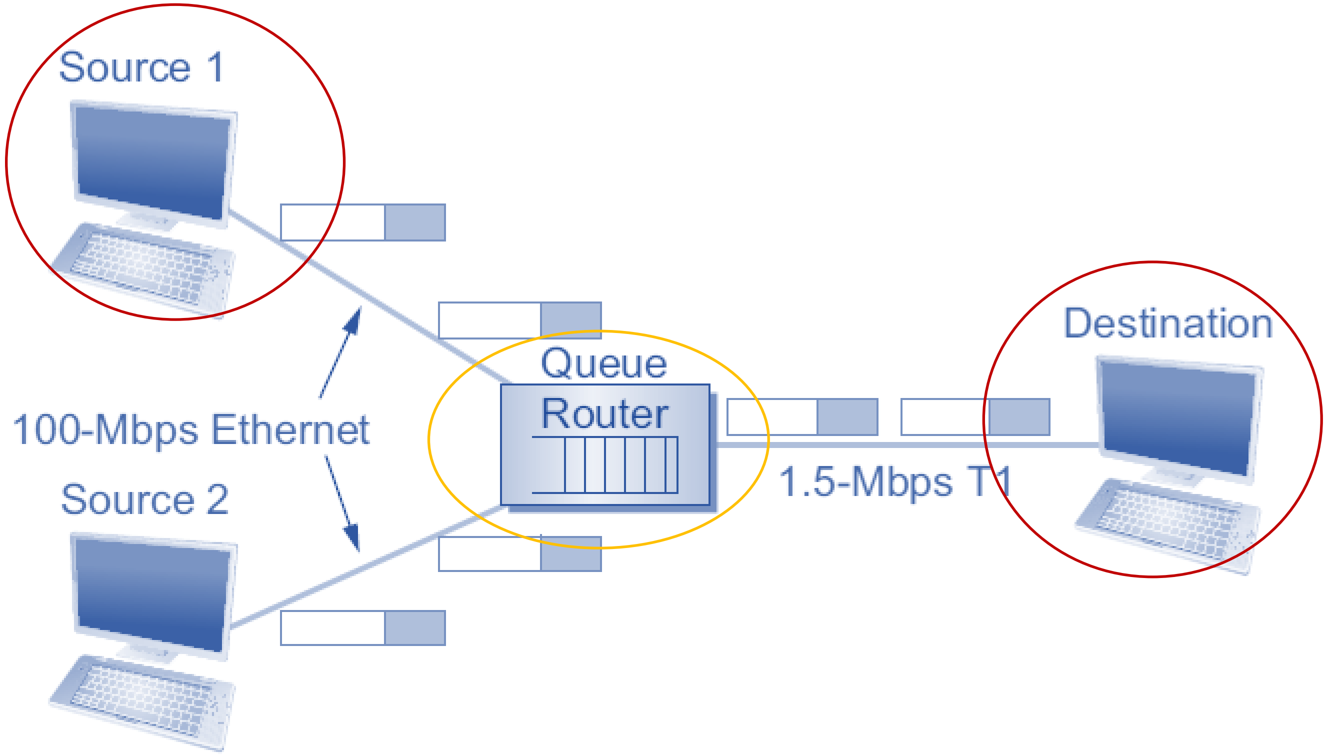
- at odds with TCP's multiplicative decrease?

TCP Vegas still use multiplicative decrease on timeout.

- Use linear decrease before potential congestion.

Two Places to Handle Network Congestion

- End hosts
- ➢Routers

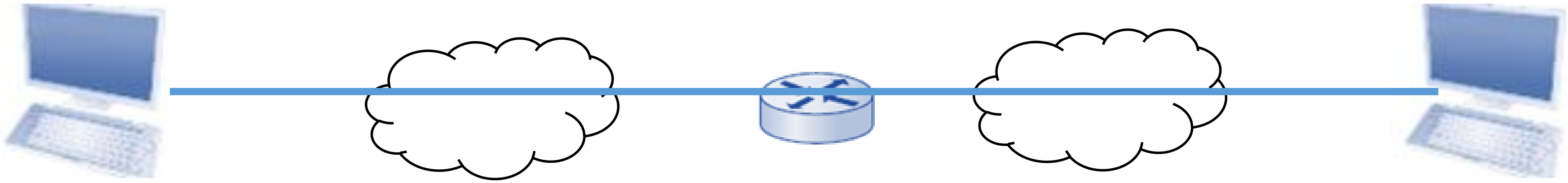


Congestion Control

- Host-based Congestion Control
  - Packet Loss
  - Delay
- Router-based Congestion Control
  - ➢Queuing

Congestion Control in Routers

- Congestion Control Approaches
  - Control congestion if (and when) is occurs: reactive
  - Pre-allocate resources so at to avoid congestion: proactive
- Resources in Network
  - Bandwidth
  - Router Queue Buffer
- Two Places of Congestion Control Implementation
  - <u>Hosts</u> at the edges of the network (transport protocol)
  - <u>Routers</u> inside the network (queuing discipline)



Queuing Discipline

- Why ?
  - Queuing discipline in routers determines how packets are transmitted (or dropped)
- Router Resource
  - Bandwidth
    - Which packets get transmitted
  - Queue Buffer
    - Which packets get discarded

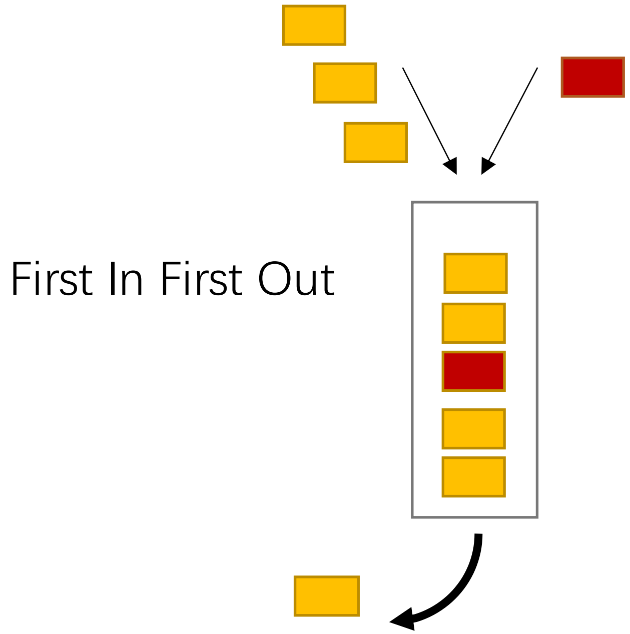
Queuing Discipline

- First-In-First-Out (FIFO)

Drop Policy
Simplest one: Tail
Drop if FIFO is full

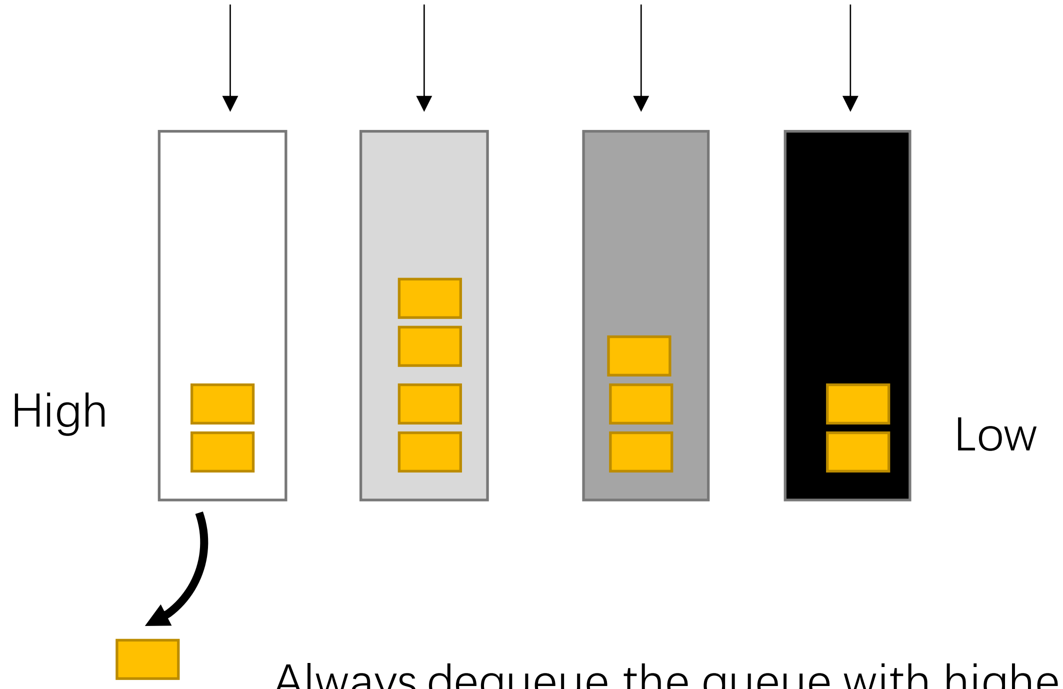
Queuing Discipline

- Problems in FIFO
  - Too simple to provide resource allocation polices (to avoid congestion)
  - Hard to enforce every network source/flow to follow the same behavior
    - eg. yellow src does not follow congestion control (UDP), can occupy more network source



Queuing Discipline

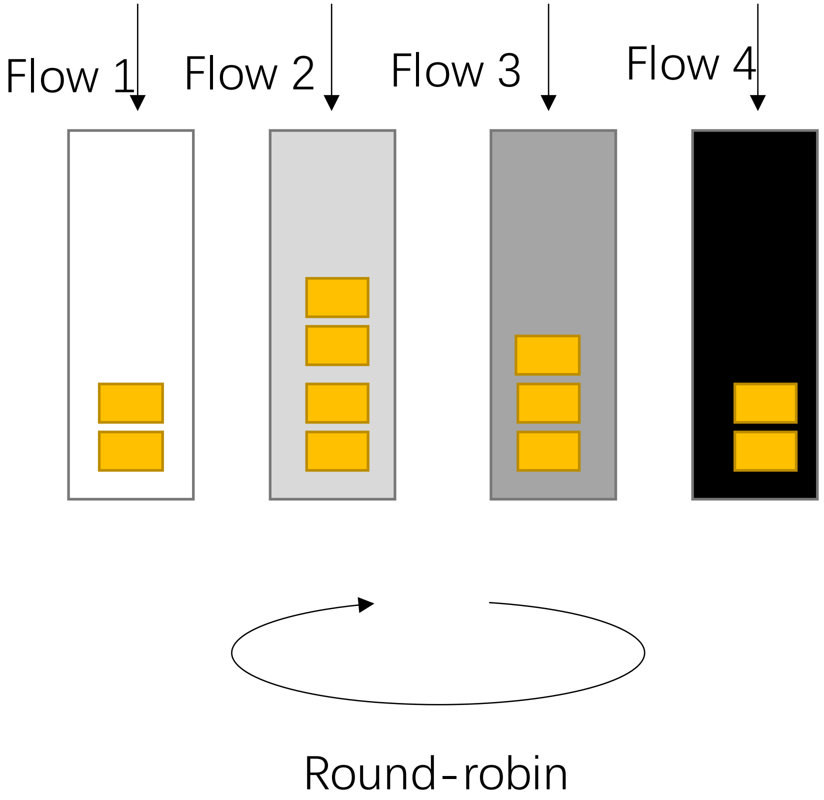
- First-In-First-Out (FIFO) with Priority



Always dequeue the queue with higher priority unless it is empty
Problem: starvation

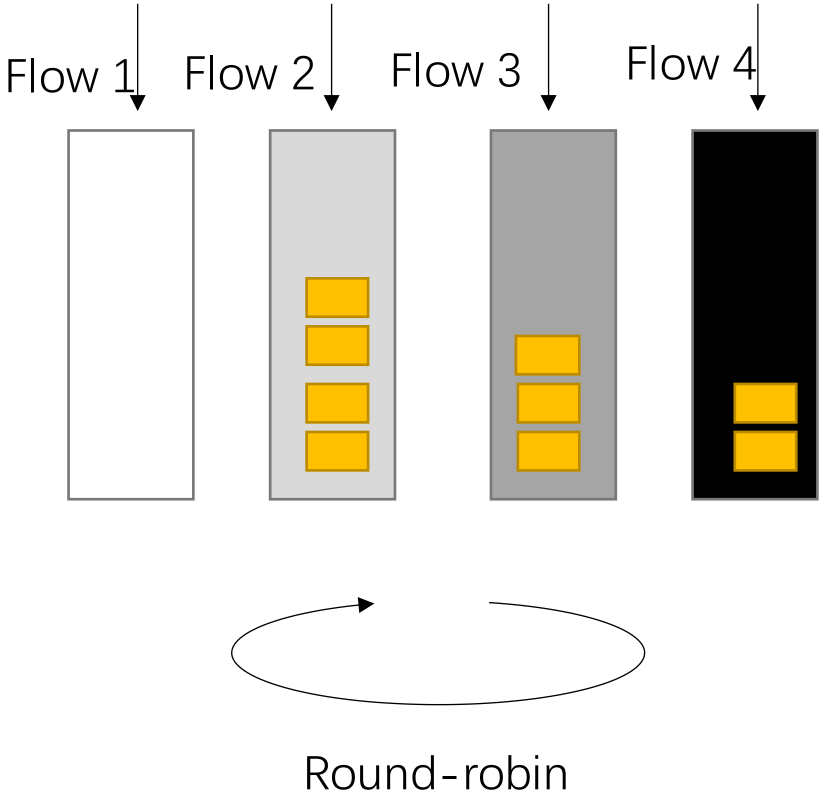
Queuing Discipline

- Fair Queuing (FQ)
  - Each flow gets 1/4 output bandwidth



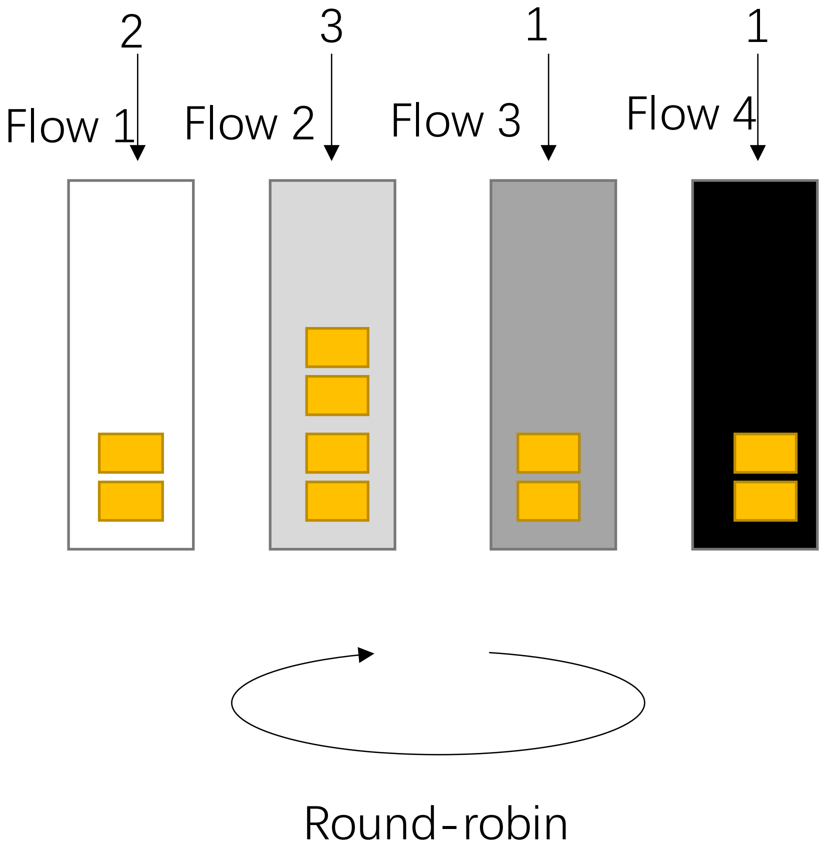
Queuing Discipline

- Fair Queuing (FQ)
  - Each flow gets 1/3 output bandwidth



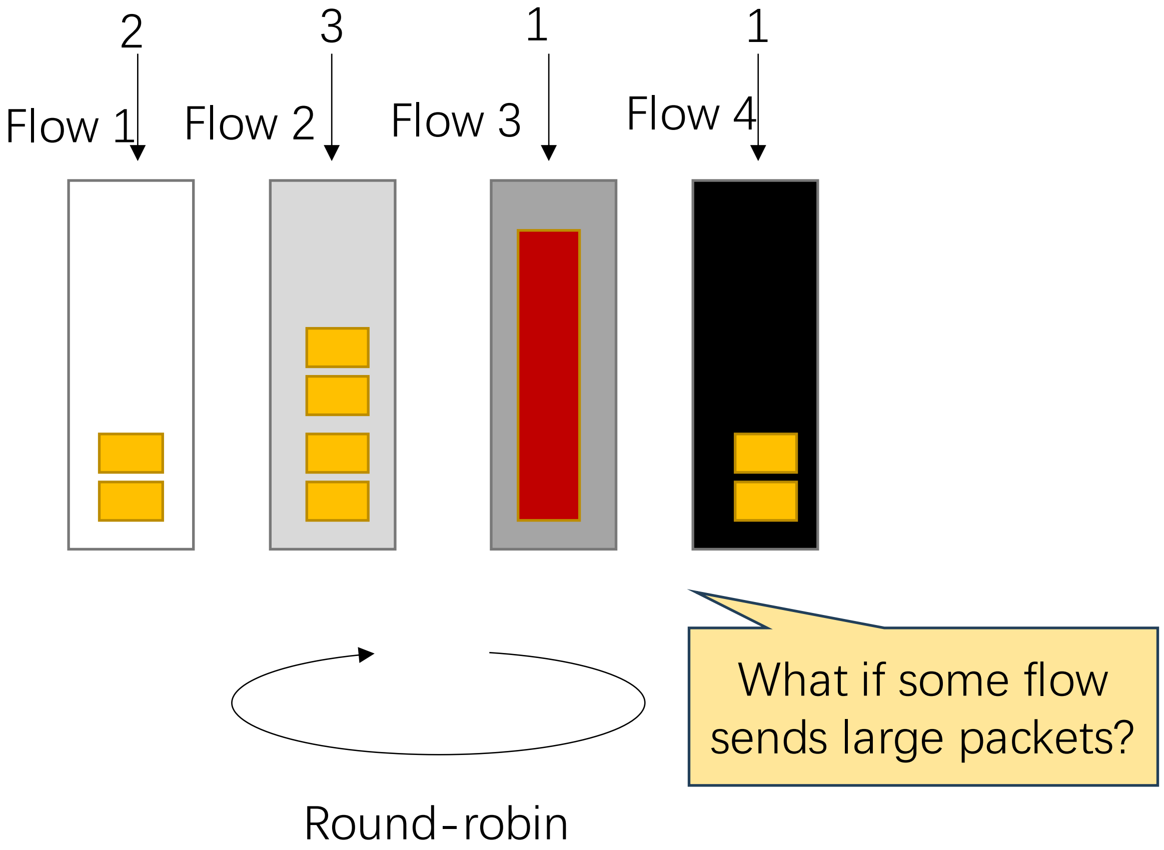
Queuing Discipline

- Weighted Fair Queuing (FQ)
  - Flows with higher weight get more output bandwidth (2/7, 3/7, 1/7, 1/7)



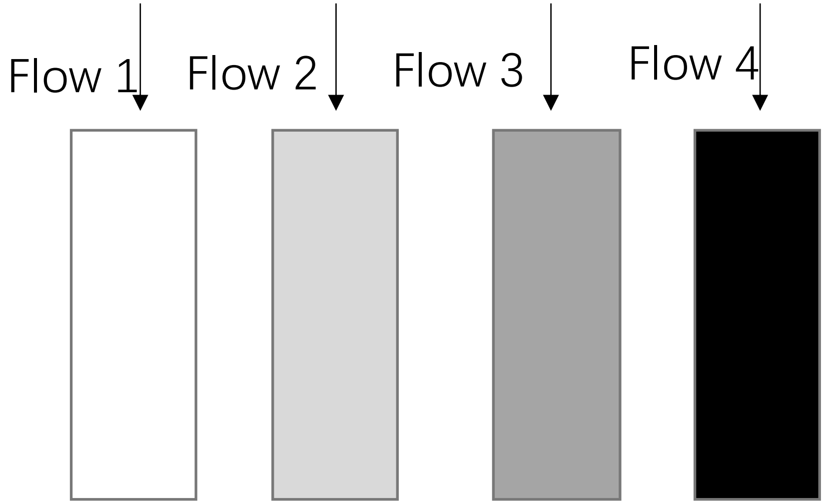
Queuing Discipline

- Weighted Fair Queuing (FQ)
  - Flows with higher weight get more output bandwidth (2/7, 3/7, 1/7, 1/7)



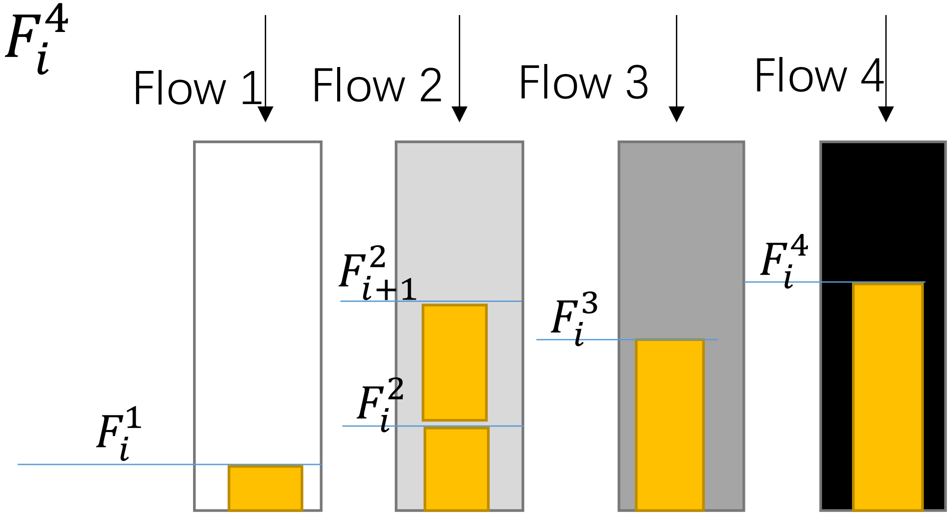
Queuing Discipline

- Bit-Level Fair Queuing (FQ)
  - Schedule according to earliest finish time
  - Finish time of packet i: $F_i = \max(F_{i-1}, A_i) + P_i$
  - P_i is the transmitting duration of packet i, A_i is the arriving time of packet i, F_{i-1} is the finish time of packet i-1 of the same flow



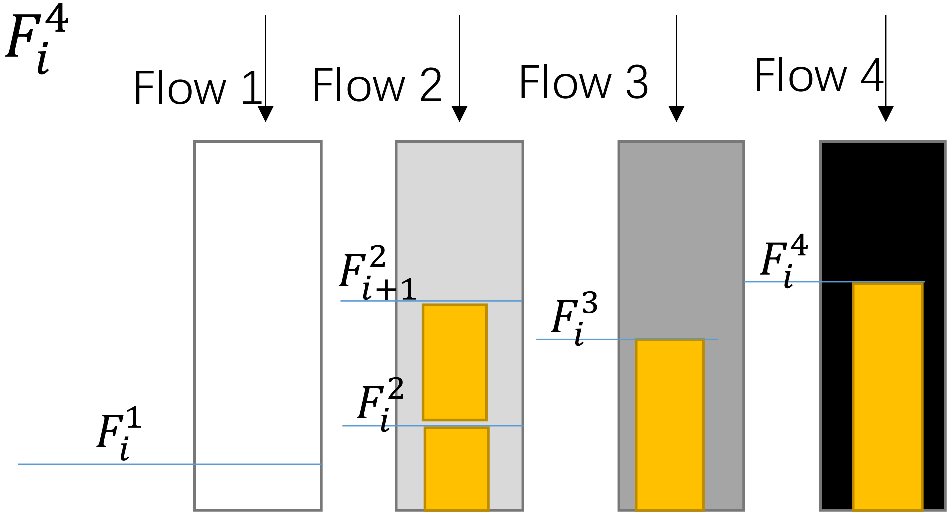
Queuing Discipline

- Bit-Level Fair Queuing (FQ)
  - <u>Schedule according to earliest finish time</u>
  - Finish time of packet i: $F_i = \max(F_{i-1}, A_i) + P_i$
  - P_i is the transmitting duration of packet i, A_i is the arriving time of packet i, F_{i-1} is the finish time of packet i-1 of the same flow
  - Finish time is virtually calculated in bit-level round robin, eg, $F_i^1 < F_i^2 < F_i^3 < F_{i+1}^2 < F_i^4$



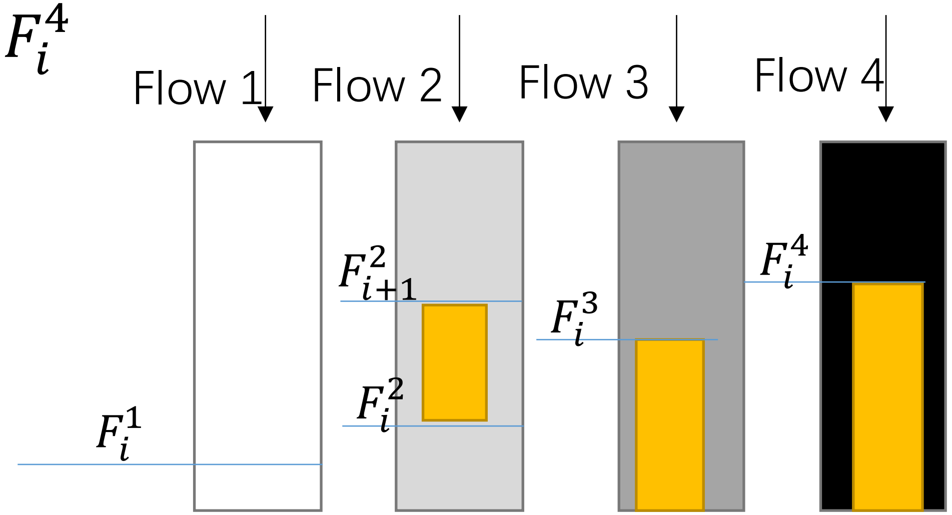
Queuing Discipline

- Bit-Level Fair Queuing (FQ)
  - Schedule according to earliest finish time
  - Finish time of packet i: $F_i = \max(F_{i-1}, A_i) + P_i$
  - P_i is the transmitting duration of packet i, A_i is the arriving time of packet i, F_{i-1} is the finish time of packet i-1 of the same flow
  - Finish time is virtually calculated in bit-level round robin, eg, $F_i^1 < F_i^2 < F_i^3 < F_{i+1}^2 < F_i^4$



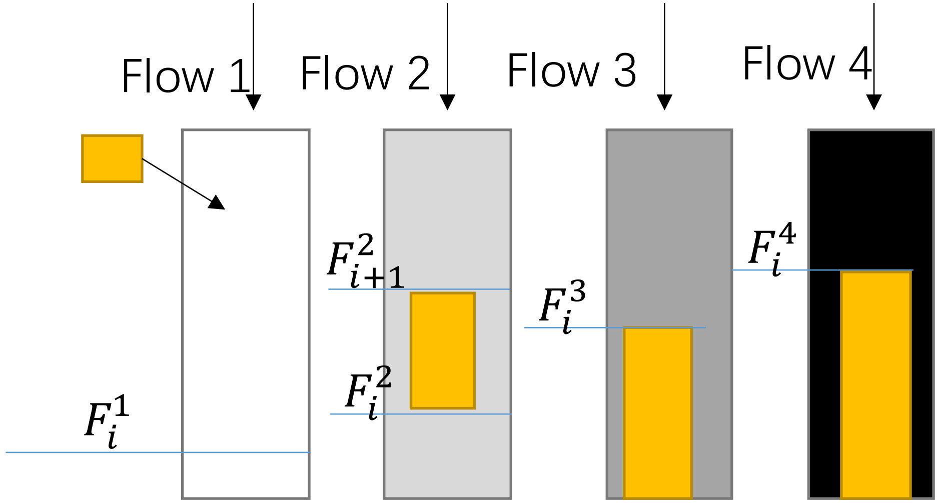
Queuing Discipline

- Bit-Level Fair Queuing (FQ)
  - Schedule according to earliest finish time
  - Finish time of packet i: $F_i = \max(F_{i-1}, A_i) + P_i$
  - P_i is the transmitting duration of packet i, A_i is the arriving time of packet i, F_{i-1} is the finish time of packet i-1 of the same flow
  - Finish time is virtually calculated in bit-level round robin, eg, $F_i^1 < F_i^2 < F_i^3 < F_{i+1}^2 < F_i^4$



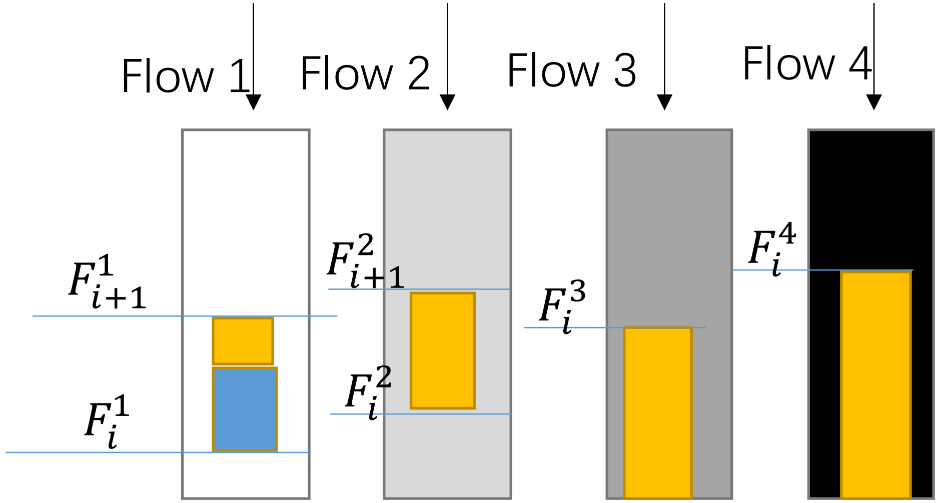
Queuing Discipline

- Bit-Level Fair Queuing (FQ)
  - Schedule according to earliest finish time
  - Finish time of packet i: $F_i = \max(F_{i-1}, A_i) + P_i$
  - P_i is the transmitting duration of packet i, A_i is the arriving time of packet i, F_{i-1} is the finish time of packet i-1 of the same flow
  - Case: when one queue is idle, idle time also count



Queuing Discipline

- Bit-Level Fair Queuing (FQ)
  - <u>Schedule according to earliest finish time</u>
  - Finish time of packet i: $F_i = \max(F_{i-1}, A_i) + P_i$
  - P_i is the transmitting duration of packet i, A_i is the arriving time of packet i, F_{i-1} is the finish time of packet i-1 of the same flow
  - Case: when one queue is idle, idle time also count



Congestion Control

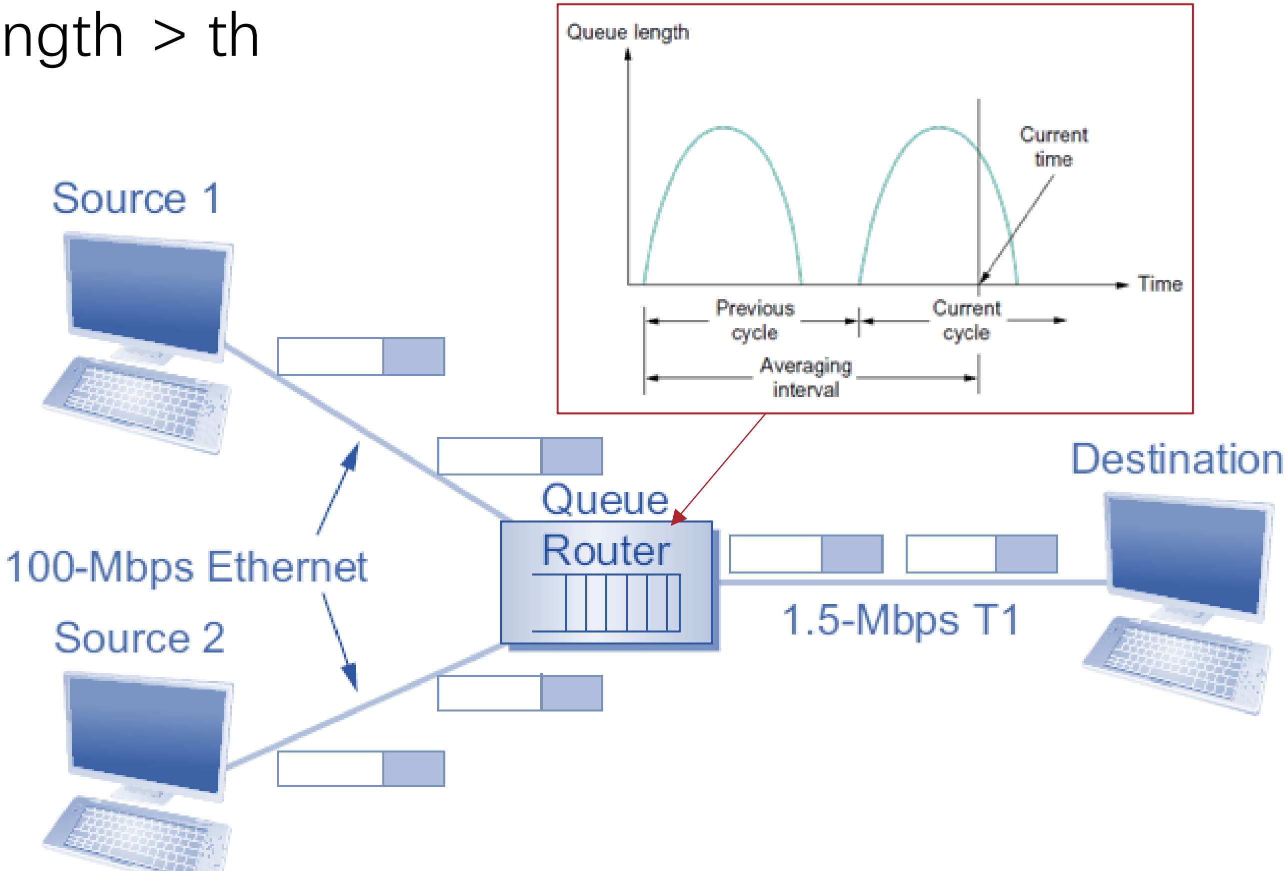
- Host-based Congestion Control
  - Packet Loss
  - Delay
- Router-based Congestion Control
  - Queuing
    - DECbit
    - Explicit congestion notification (ECN)
    - Random Early Detection (RED)

DECbit

- Developed for the Digital Network Architecture (DNA)
  - Before TCP/IP was "standardized"
- Idea: let routers explicitly indicate congestion
- Approach:
  - Router set congestion bit in passing packets if there is congestion
  - Destination echoes bit back to source through acks
  - Source adjusts cwnd according to congestion bit

DECbit

- Routers determine congestion
  - Monitor average queue length over last busy+idle cycle
  - If average queue length > th
    - set congestion bit

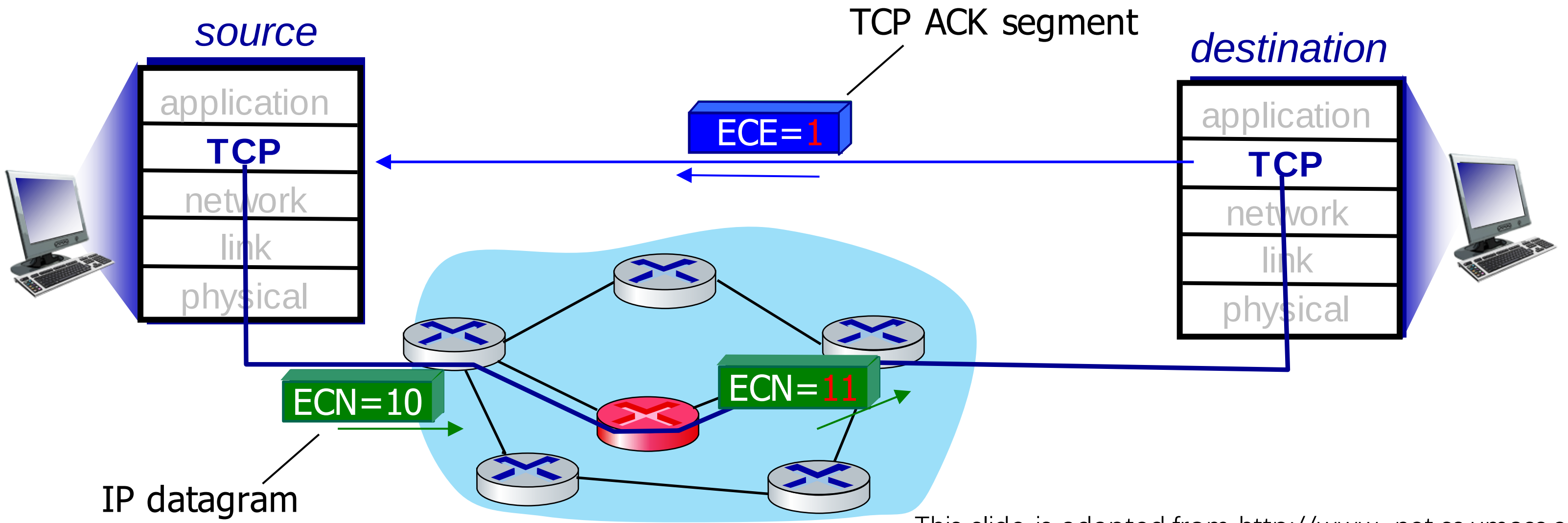


DECbit

- Source reacts to congestion bit
  - If < 50% of last window's packets had bit set
    - cwnd++
  - If > 50% of last window's packets had bit set
    - cwnd*0.875

Explicit congestion notification (ECN)

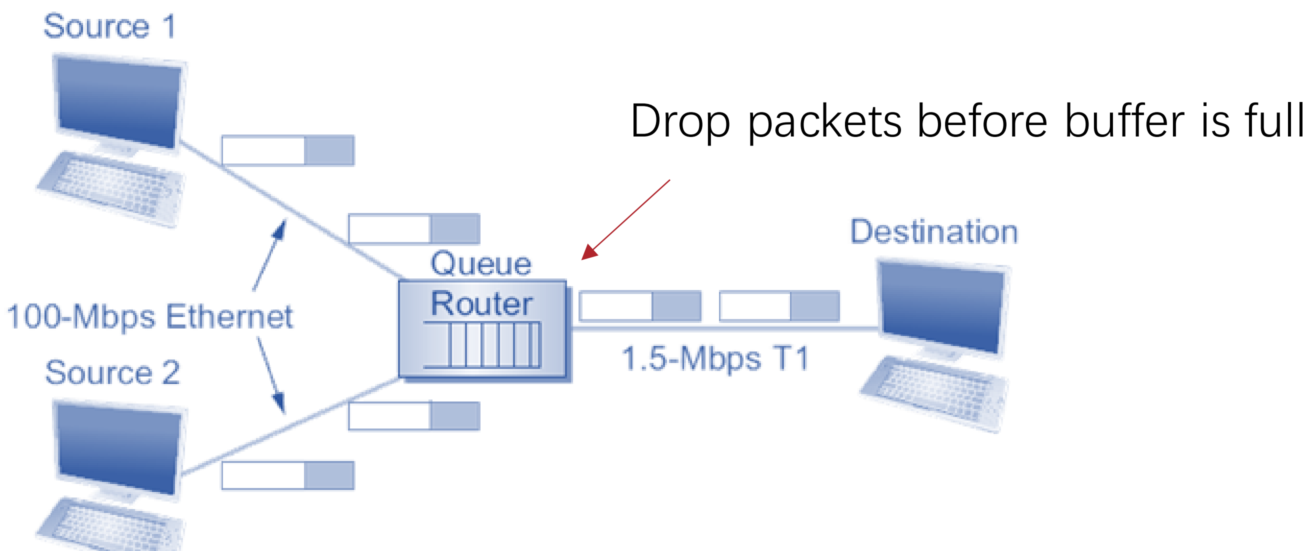
- RFC 3168
  - Two bits in IP header (ECN in IP's ToS field) marked by network router to indicate congestion
  - Mechanism is similar to DECbit
  - Destination sets ECE bit (in TCP Header) on ACK segment to notify sender of congestion



This slide is adapted from http://www-net.cs.umass.edu/kurose_ross/ppt.htm by Kurose Ross

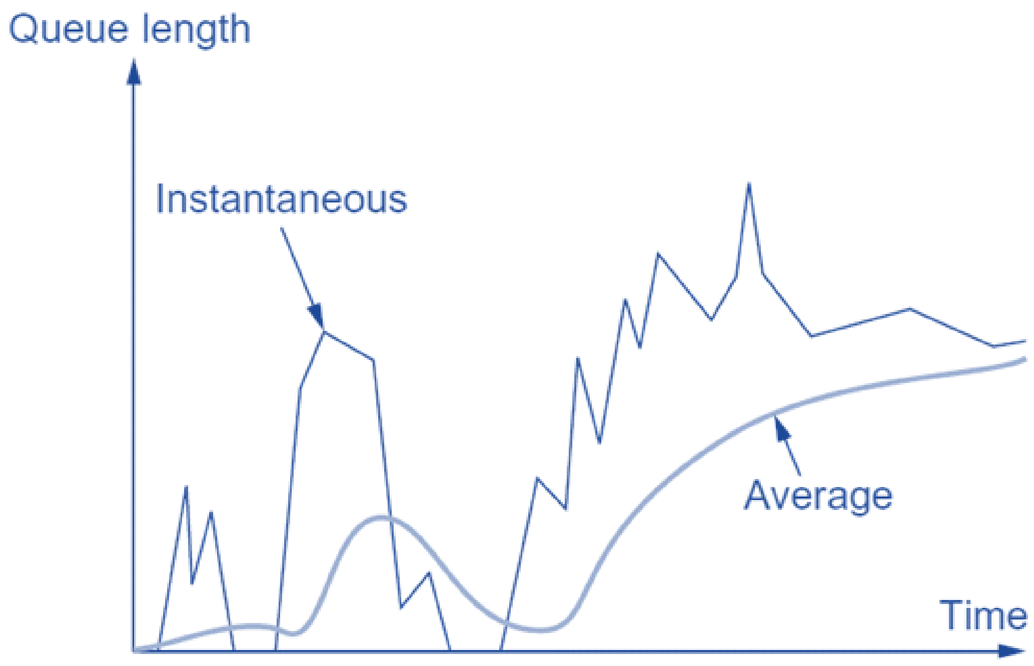
Random Early Detection (RED)

- Another way is to offload a part of congestion control to routers
- Idea: let routers implicitly indicate congestion
  - Router notices that the queue is getting backlogged
  - Router randomly drops packets to signal congestion
  - Source adjusts cwnd



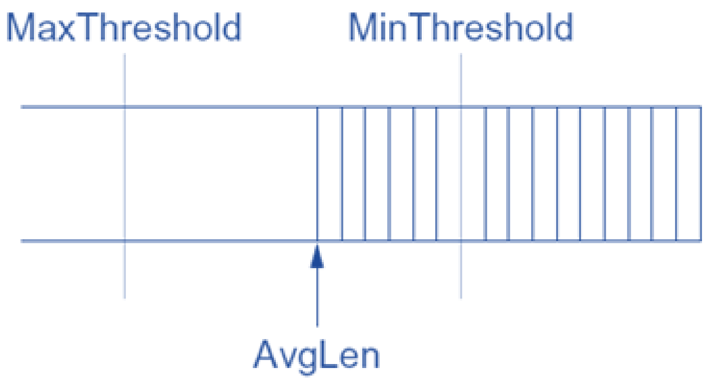
RED Algorithm

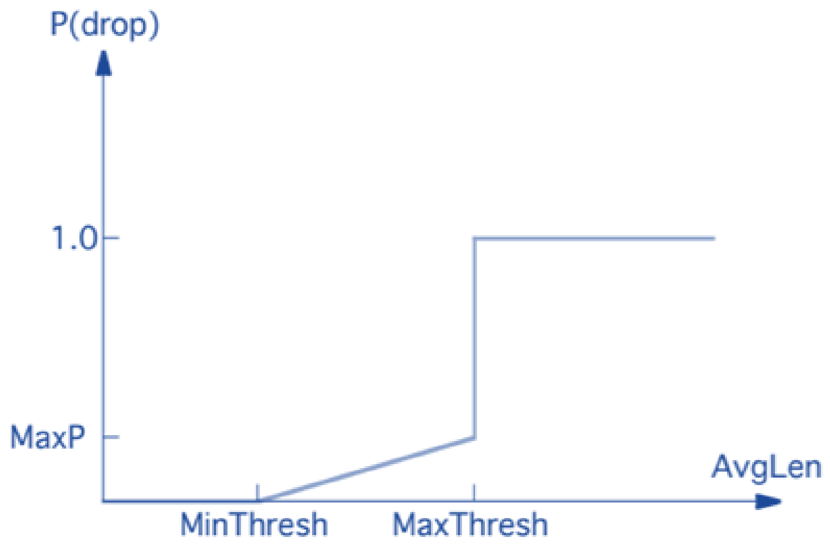
- Compute Average Queue Length
  - Moving average
    - AvgLen = (1 - Weight) * AvgLen + Weight * SampleLen



RED Algorithm

- Two queue length thresholds
  - if AvgLen <= MinThreshold
    - Enqueue the packet
  - if MinThreshold < AvgLen < MaxThreshold
    - Drop arriving packet with probability P
  - if MaxThreshold <= AvgLen then drop arriving packet





RED Algorithm

Computing probability P

- TempP = MaxP * (AvgLen - MinThreshold)/(MaxThreshold - MinThreshold)
- P = TempP/(1 - count * TempP)
  - Count: number of continuously queued packets without drop

Space out drops over time:
P increases as the time since last drop increases.

Reference

- Textbook 6.4