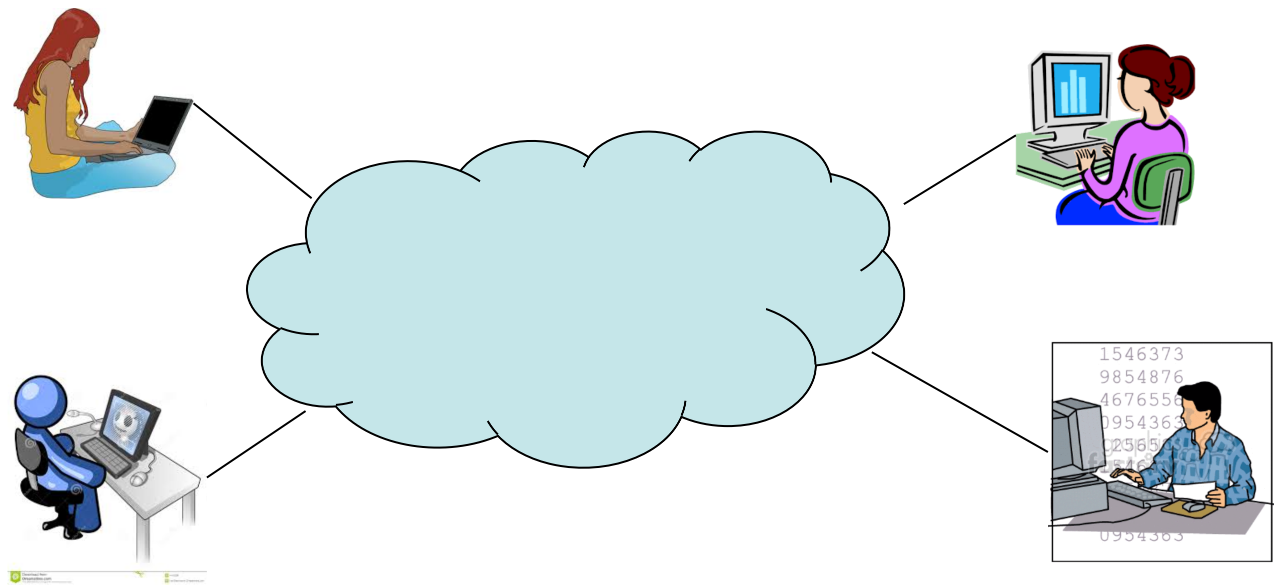
CS 356: Computer Network Architectures
Lecture 21: Queuing Disciplines and Congestion Avoidance
Chap. 6.4 and related papers

Xiaowei Yang
xwy@cs.duke.edu

Overview

- Resource allocation framework
- Congestion Avoidance
- Queuing Disciplines

Resource allocation



- A fundamental question of networking: who gets to send at what speed?

Resource allocation vs Congestion control

- Resource allocation: The process by which network elements try to meet the competing demands that applications have for network resources
  - Bandwidth and buffer space

- Congestion control: efforts made only by network nodes to prevent or respond to overload conditions

Network Model

- Packet switched
- Connectionless flows
  - Flow: a sequence o packets sent between a source host and a destination host
- Service model
  - Best-effort
  - Quality of Service

Design Space for resource allocation

- Router-centric vs. Host-centric

- Reservation-based vs. Feedback-based

- Window-based vs. Rate-based

Evaluation criteria

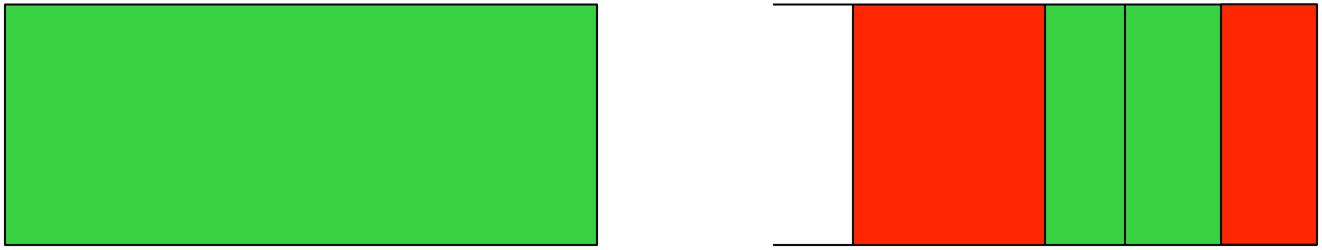
- Performance and Fairness
  - Performance: high throughput, low latency
    - Power = throughput/Delay
  - Fairness: Chiu-Jain fairness index

$$F(x) = \frac{(\Sigma x_i)^2}{n(\Sigma x_i^2)}.$$

Queuing disciplines for resource allocation

Queuing mechanisms

- Router-enforced resource allocation
  - Scheduling policy: which packet gets sent
  - Drop policy: which packet gets dropped
- Default
  - First come first serve (FIFO) with DropTail



Limitations of FIFO with DropTail

- Unfair
  - TCP flows reduce sending rates
  - Non-TCP may not

Priority queuing

- Mark packets with priority bits
- Multiple FIFO queues, each for one priority
- Transmit packets out of highest priority queues

- Limitation: may starve low priority packets
  - Users cannot set their priority bits
  - Routing messages get high priority

Fair Queuing

Fair Queuing Motivation

- End-to-end congestion control + FIFO queue has limitations
  - What if sources mis-behave?

- Approach 2:
  - Fair Queuing: a queuing algorithm that aims to "fairly" allocate buffer, bandwidth, latency among competing users

Outline

- What is fair?
- Weighted Fair Queuing
- Deficit Round Robin

What is fair?

- Fair to whom?
  - Source, Receiver, Process
  - Flow / conversation: Src and Dst pair
    - Flow is considered the best tradeoff

- Maximize fairness index?
  - Fairness = $(\Sigma x_i)^2/n(\Sigma x_i^2)$ $0<\text{fairness}<1$

- What if a flow uses a long path?

- Tricky, no satisfactory solution, policy vs mechanism

One definition: Max-min fairness

- Many fair queuing algorithms aim to achieve this definition of fairness
- Informally
  - Allocate user with “small” demand what it wants, evenly divide unused resources to “big” users
- Formally
  - 1. No user receives more than its request
  - 2. No other allocation satisfies 1 and has a higher minimum allocation
    - Users that have higher requests and share the same bottleneck link have equal shares
  - Remove the minimal user and reduce the total resource accordingly, 2 recursively holds

Max-min example

- Assume sources 1..n, with resource demands X1..Xn in an ascending order
- Assume channel capacity C.
  - Give C/n to X1; if this is more than X1 wants, divide excess (C/n - X1) to other sources: each gets C/n + (C/n - X1)/(n-1)
  - If this is larger than what X2 wants, repeat process

- A numerical example
  - Bottleneck link bandwidth 10Mbps
  - Three users: r1: 1Mbps, r2: 6Mbps, r3: 8Mbps
  - Allocation results:
    - r1: 1Mbps, r2=r3: 4.5Mbps

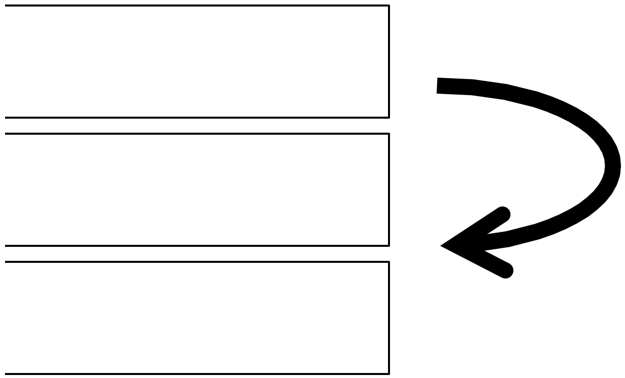
Design of weighted fair queuing

- Resources managed by a queuing algorithm
  - Bandwidth: Which packets get transmitted
  - Promptness: When do packets get transmitted
  - Buffer: Which packets are discarded
  - Examples: FIFO
    - The order of arrival determines all three quantities

Design goals

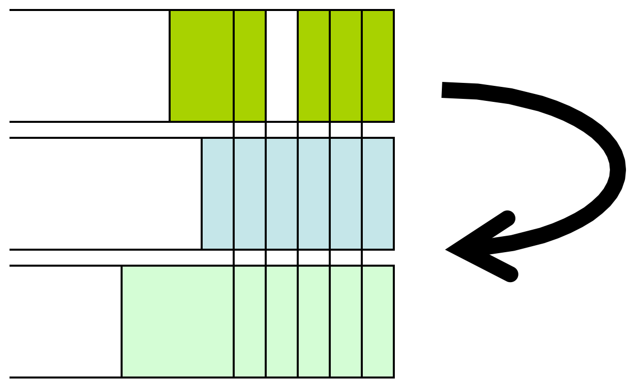
- Max-min fair
- Work conserving: link’s not idle if there is work to do
- Isolate misbehaving sources
- Has some control over promptness
  - E.g., lower delay for sources using less than their full share of bandwidth
  - Continuity
    - On Average does not depend discontinuously on a packet’s time of arrival
    - Not blocked if no packet arrives

A simple fair queuing algorithm



- Nagle's proposal: separate queues for packets from each individual source
- Different queues are serviced in a round-robin manner
- Limitations
  - Is it fair?
  - What if a packet arrives right after one departs?

Implementing max-min Fairness

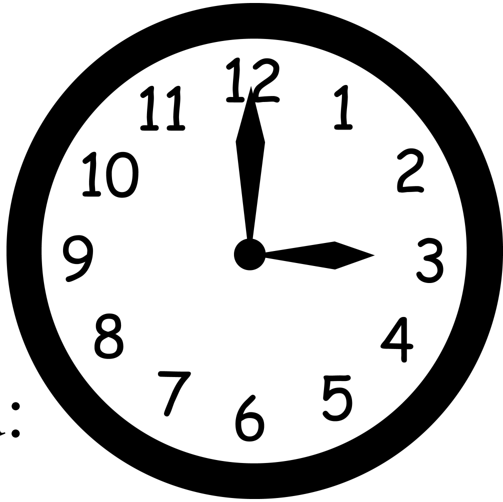


- Generalized processor sharing
  - Fluid fairness
  - Bitwise round robin among all queues

- WFQ:
  - Emulate this reference system in a packetized system
  - Challenges: bits are bundled into packets. Simple round robin scheduling does not emulate bit- by-bit round robin

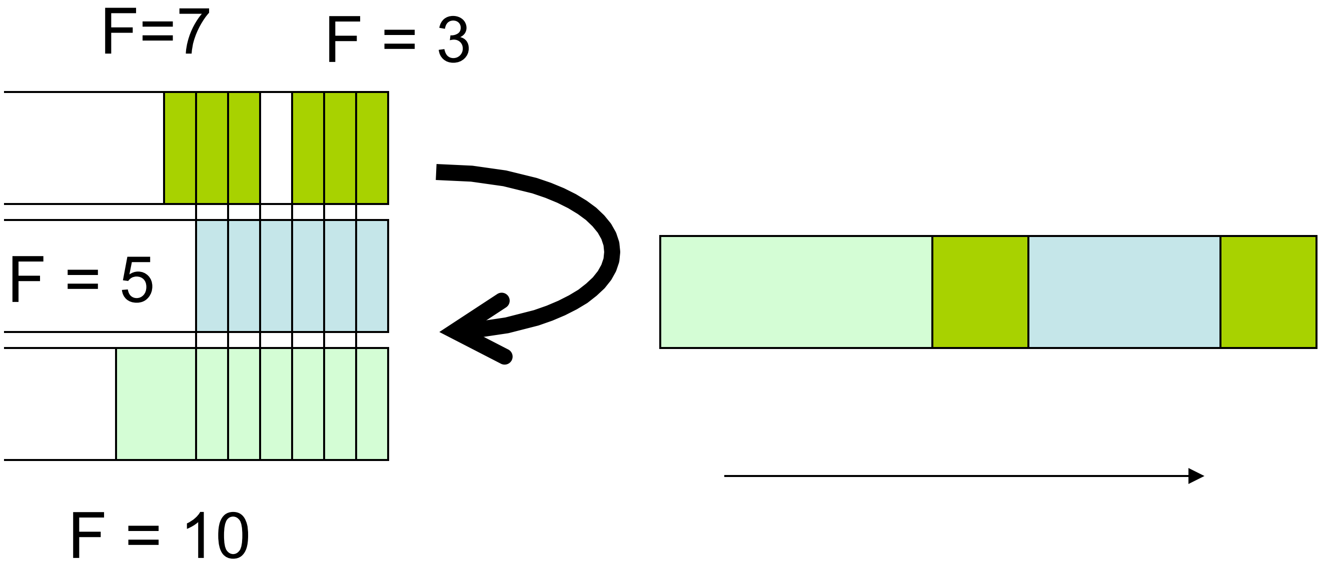
Emulating Bit-by-Bit round robin

- Define a virtual clock: the round number R(t) as the number of rounds made in a bit-by-bit round-robin service discipline up to time t



- A packet with size P whose first bit serviced at round $R(t_0)$ will finish at round:
  - $R(t) = R(t_0) + P$
- Schedule which packet gets serviced based on the finish round number

Example



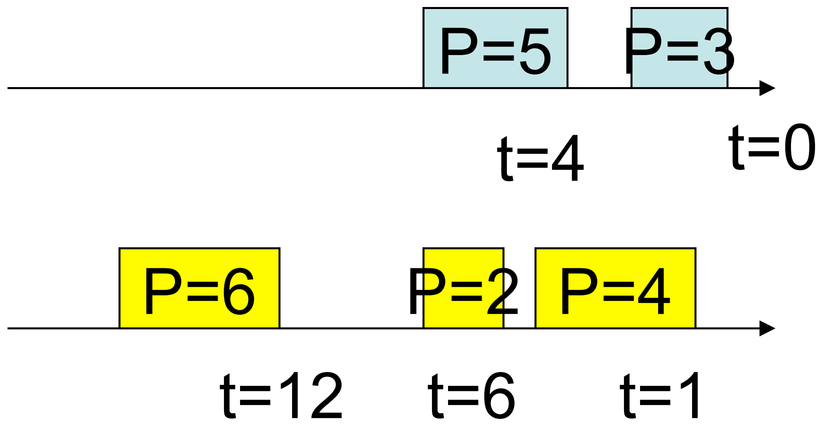
Compute finish times

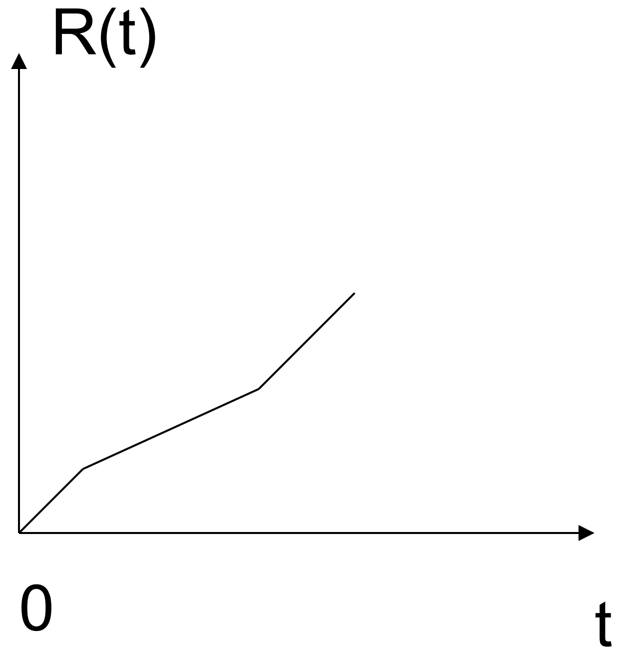
- Arrival time of packet i from flow α: t_i^α
- Pacet size: P_i^α
- S_i^α be the round number when the packet starts service
- F_i^α be the finish round number
- $F_i^\alpha = S_i^\alpha + P_i^\alpha$
- $S_i^\alpha = \text{Max}\ (F_{i-1}^\alpha, R(t_i^\alpha))$

Compute R(t) can be complicated

- Single flow: clock ticks when a bit is transmitted. For packet i:
  - Round number $\leq$ Arrival time A_i
  - $F_i = S_i + P_i = \max(F_{i-1}, A_i) + P_i$
- Multiple flows: clock ticks when a bit from all active flows is transmitted
  - When the number of active flows vary, clock ticks at different speed: $\partial R / \partial t = {}^{1}/N_{ac}(t)$

An example



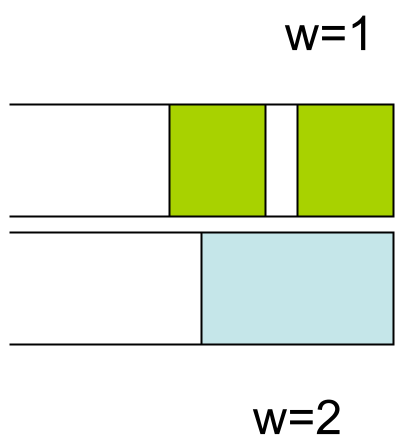


- Two flows, unit link speed 1 bit per second

Delay Allocation

- Reduce delay for flows using less than fair share
  - Advance finish times for sources whose queues drain temporarily
- Schedule based on B_i instead of F_i
  - $F_i = P_i + \max(F_{i-1}, A_i) \rightarrow B_i = P_i + \max(F_{i-1}, A_i - \delta)$
  - If $A_i < F_{i-1}$, conversation is active and δ has no effect
  - If $A_i > F_{i-1}$, conversation is inactive and δ determines how much history to take into account
    - Infrequent senders do better when history is used
  - When $\delta = 0$, no effect
  - When δ = infinity, an infrequent sender preempts other senders

Weighted Fair Queuing



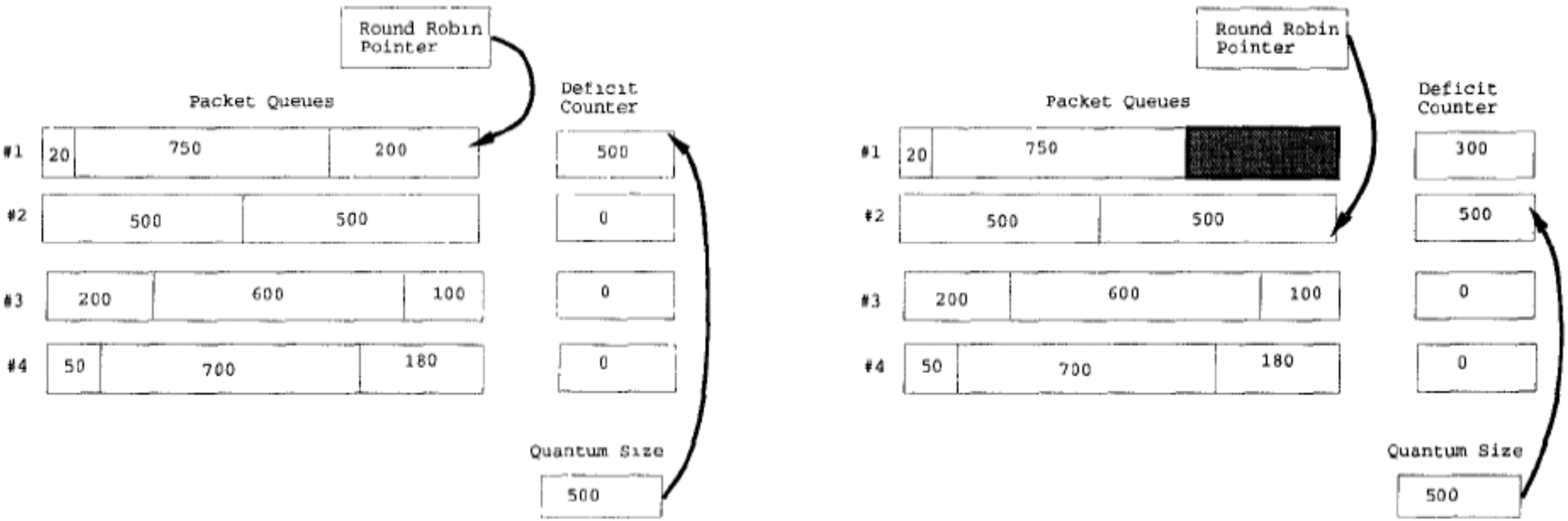
- Different queues get different weights
  - Take w_i amount of bits from a queue in each round
  - $F_i = S_i + P_i / w_i$

Stochastic Fair Queuing

- Goal: fixed number of queues rather than various number of queues
  - Compute a hash on each packet
  - Instead of per-flow queue have a queue per hash bin
  - Queues serviced in round-robin fashion
  - Memory allocation across all queues
  - When no free buffers, drop packet from longest queue
- Limitations
  - An aggressive flow steals traffic from other flows in the same hash
  - Has problems with packet size unfairness

Deficit Round Robin

- O(1) rather than O(log Q)
- Each queue is allowed to send Q bytes per round
- If Q bytes are not sent (because packet is too large) deficit counter of queue keeps track of unused portion
- If queue is empty, deficit counter is reset to 0
- Similar behavior as FQ but computationally simpler

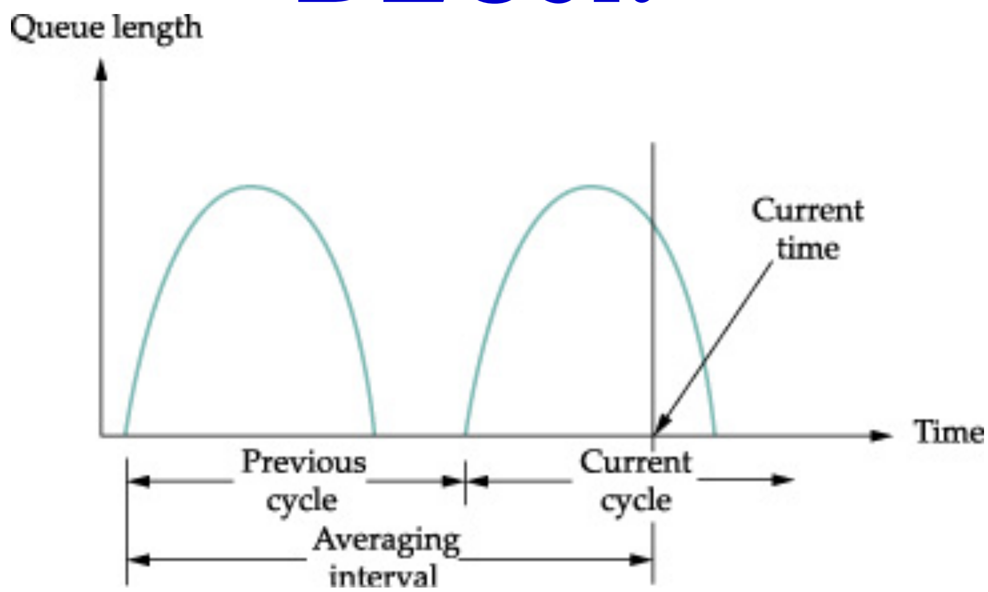


- Unused quantum is saved for the next round to offset packet size unfairness

Design space for resource allocation

- Router+host joint control
  - Router: Early signaling of congestion
  - Host: react to congestion signals
  - Case studies: DECbit, Random Early Detection

DECbit

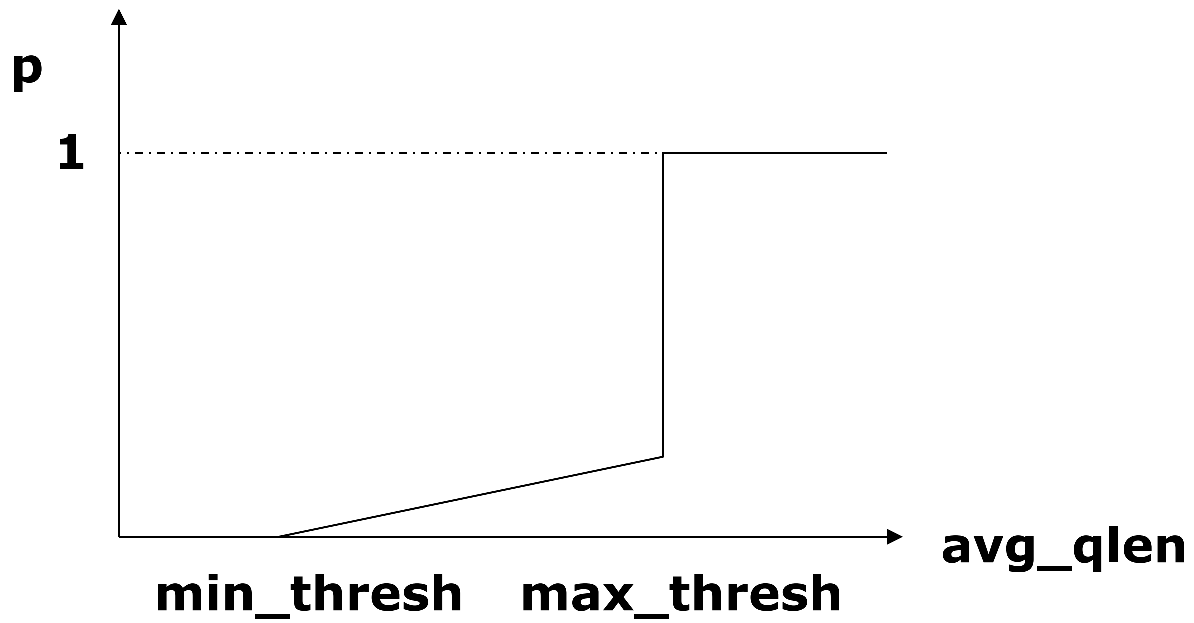


- Add a congestion bit to a packet header
- A router sets the bit if its average queue length is non-zero
  - Queue length is measured over a busy+idle interval
- If less than 50% of packets in one window do not have the bit set
  - A host increases its congest window by 1 packet
- Otherwise
  - Decreases by 0.875
- AIMD

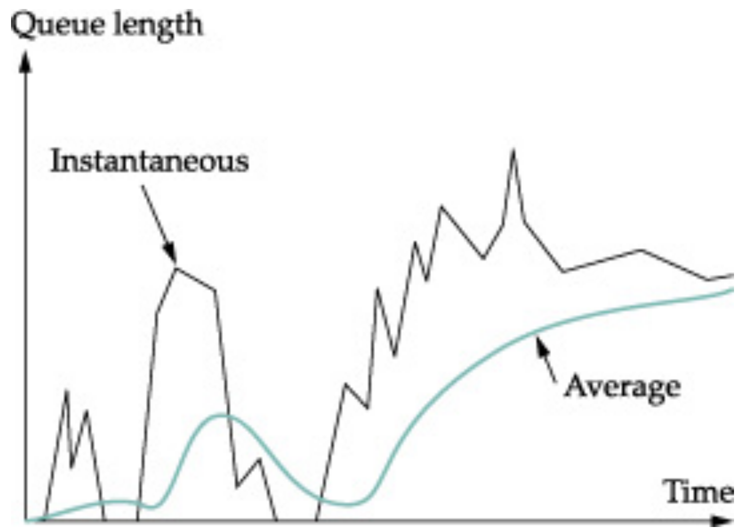
Random Early Detection

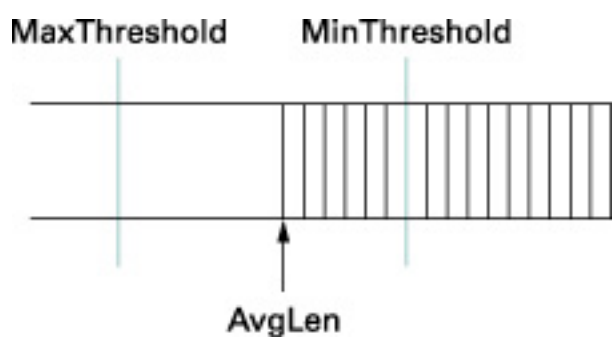
- Random early detection (Floyd93)
  - Goal: operate at the “knee”
  - Problem: very hard to tune (why)
- RED is generalized by Active Queue Managment (AQM)
- A router measures average queue length using exponential weighted averaging algorithm:
  - AvgLen = (1-Weight) * AvgLen + Weight * SampleQueueLen

RED algorithm

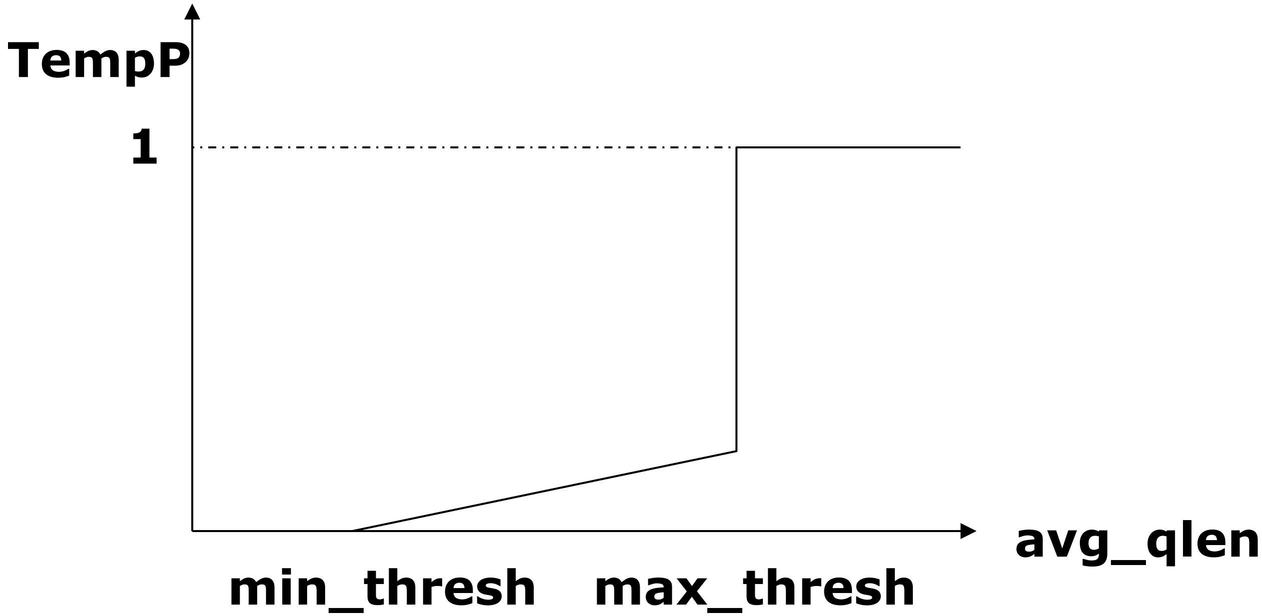


- If AvgLen ≤ MinThreshold
  - Enqueue packet
- If MinThreshold < AvgLen < MaxThreshold
  - Calculate dropping probability P
  - Drop the arriving packet with probability P
- If MaxThreshold ≤ AvgLen
  - Drop the arriving packet


Queue length
Instantaneous
Average
Time


MaxThreshold
MinThreshold
AvgLen

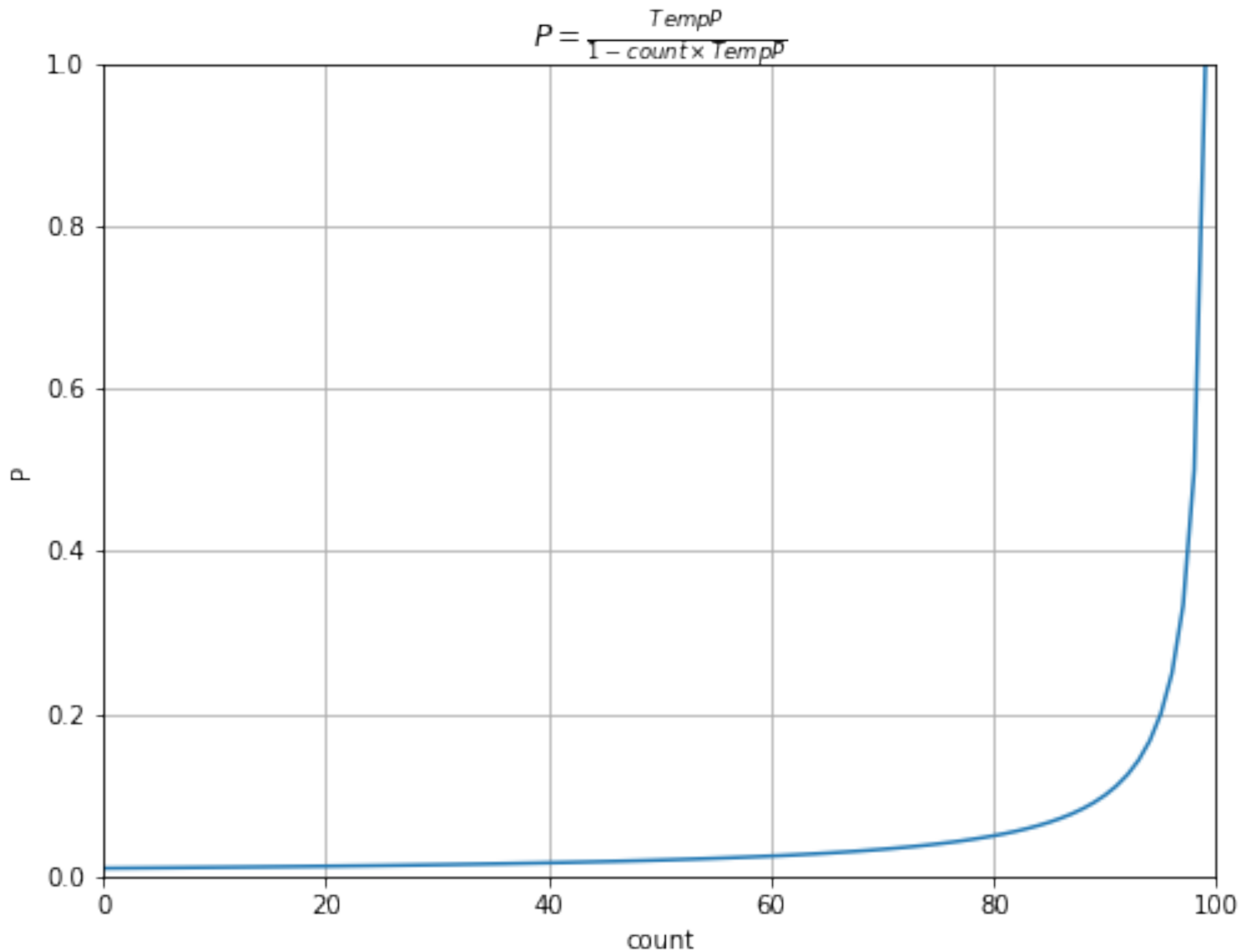
Even out packet drops



- TempP = MaxP x (AvgLen – Min)/(Max-Min)
- P = TempP / (1 – count * TempP)
- Count
  - keeps track of how many newly arriving packets have been queued when min < Avglen < max
  - It keeps drop evenly distributed over time, even if packets arrive in burst
  - Reset to zero after a drop

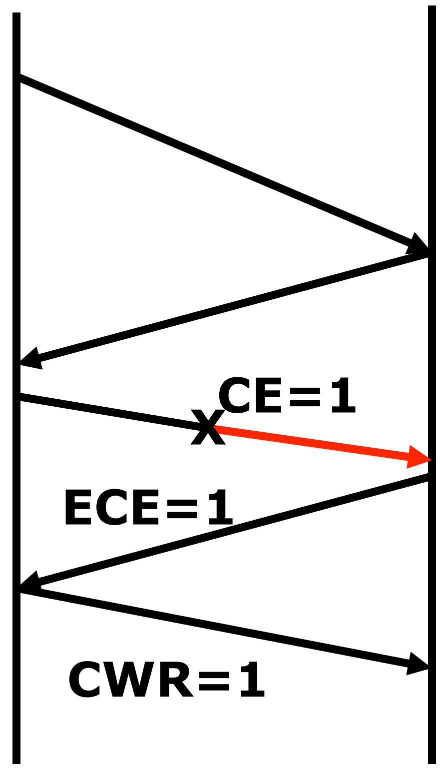
An example

- MaxP = 0.02
- AvgLen is half way between min and max thresholds
- TempP = 0.01
- A burst of 1000 packets arrive
- With TempP, 10 packets may be discarded uniformly randomly among the 1000 packets
- With P, they are likely to be more evenly spaced out, as P gradually increases if previous packets are not discarded


$P = \frac{TempP}{1 - count \times TempP}$
1.0
0.8
0.6
0.4
0.2
0.0
P
0
20
40
60
80
100
count

Explicit Congestion Notification

- A new IETF standard
- Two bits in IP header
  - 00: No ECN support
  - 01/10: ECN enabled transport
  - 11: Congestion experienced
- Two TCP flags
  - ECE: congestion experienced
  - CWR: cwnd reduced



Design Space for resource allocation

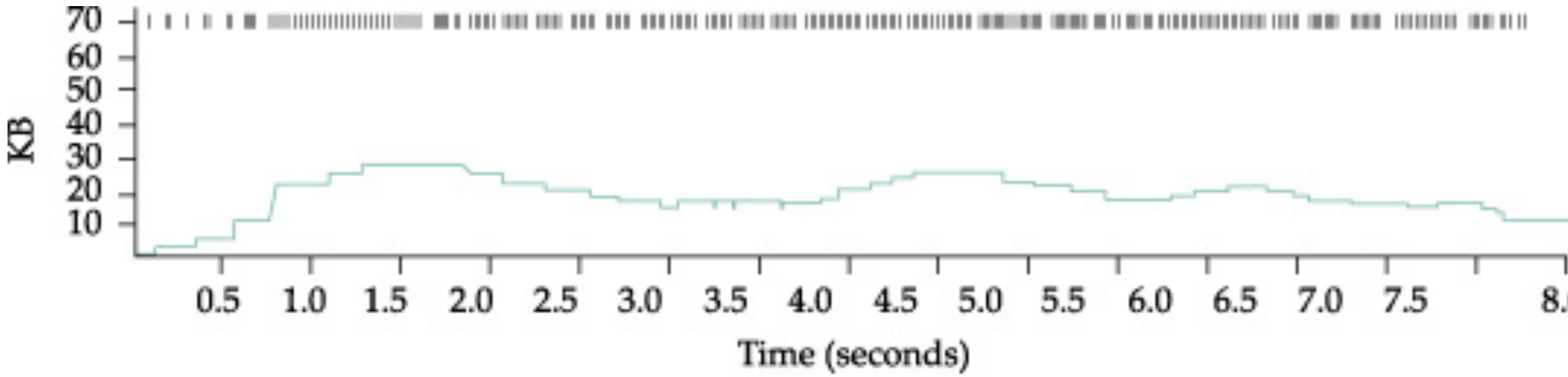
- Router-centric vs. Host-centric
  - Router-centric: fair queuing
  - Router/host joint design: Decbit, RED+ECN
  - A host-centric scheme: TCP vegas
- Reservation-based vs. Feedback-based
- Window-based vs. Rate-based

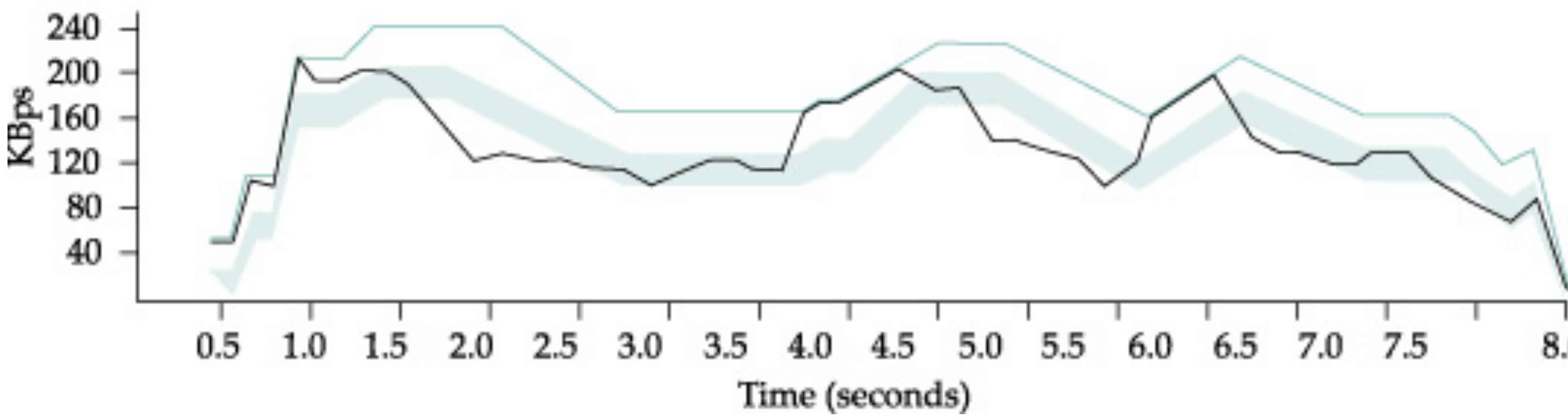
Source-based congestion avoidance

- TCP Vegas
  - Detect increases in queuing delay
  - Reduces sending rate

- Details
  - Record baseRTT (minimum seen)
  - Compute ExpectedRate = cwnd/BaseRTT
  - Diff = ExpectedRate - ActualRate
  - When Diff $< \alpha$, incr cwnd linearly, when Diff $> \beta$, decr cwnd linearly
    - When timeout occurs, decreases multiplicatively
    - $\alpha < \beta$

cwnd


KB
70
60
50
40
30
20
10
0.5 1.0 1.5 2.0 2.5 3.0 3.5 4.0 4.5 5.0 5.5 6.0 6.5 7.0 7.5 8.0
Time (seconds)


KBps
240
200
160
120
80
40
0.5 1.0 1.5 2.0 2.5 3.0 3.5 4.0 4.5 5.0 5.5 6.0 6.5 7.0 7.5 8.0
Time (seconds)

Summary

- Resource allocation for congestion avoidance
  - Queuing disciplines
  - RED+ECN, DECbit
    - RED is generalized by active queue management
  - Source-based congestion avoidance
    - TCP Vegas