Fundamentals of Computer Networks
ECE 478/578

Lecture #22: Resource Allocation and Congestion Control
Instructor: Loukas Lazos
Dept of Electrical and Computer Engineering
University of Arizona

Congestion Control and Resource Allocation

Goal of resource allocation

Dedicate some portions of available resources (link bandwidth, buffer space) for a flow

Problem: When to say no? and to whom?

Goal of congestion control

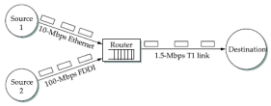
Efforts made by network to present or respond to overload conditions and restore the system to a stable state

Example: Refuse connections, drop packets, reduce sending rates, deflect traffic

2

Networking Model

Packet-switched networks



Connectionless flows

No reservation of network resources at the start of a session

Flows can be defined at several granularities (process to process, source-destination, etc.)

Service model

Best effort: All packets are treated in exactly the same manner

3

Taxonomy of Resource Allocation

Router-centric vs. host-centric

- Router-centric: Routers decide when to schedule, which packets drop, inform the rate to hosts
- Host-centric: End-hosts observe the network conditions
- Router-centric and host-centric are not mutually exclusive

Reservation-based vs. feedback-based

- Reservation-based: Resource reserved in advance, flow is dropped if resources are unavailable
- Feedback-based: Source transmits packets without reserving any capacity
- Feedback may be explicit or implicit

Window-based vs. rate-based

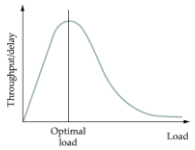
- Window-based: Use the same window employed for reliable transmission
- Rate-based: Rate is explicitly controlled by the network or receiver

4

Evaluation Metrics

Performance

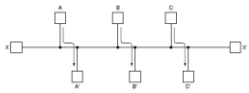
- Throughput and Delay (end-to-end)
- Network must be "stable"
- Avoid "congestion collapse"
- Does higher throughput mean lower delay?



Fairness

- What is fair share of bandwidth?
- Fairness index proposed by Raj Jain
- x_i: Bandwidth of flow *I*

$$f(x_1, x_2, \ldots, x_n) = \frac{(\sum_{i=1}^{n} x_i)^2}{n \sum_{i=1}^{n} x_i^2}$$



Example 1: Total BW = C, one node receives C, others 0, f = 1/n

Example 2: Total BW = C, each node receives 1, C nodes total, f = 1

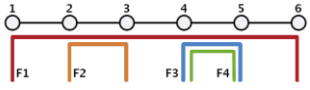
5

Other Notions of Fairness

Max-min fairness

- Resources are allocated in flows
- An allocation is "max-min fair" if and only an increase of any rate of a flow occurs at the expense of a flow that is already receiving lesser bandwidth
- It is not allowed to decrease the share of smaller flows

Computation of max-min fair rate

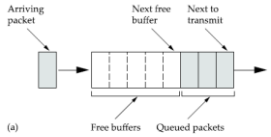


- Start increasing the rate of all flows progressively until you fill a bottleneck
- The rate of all flows passing through the bottleneck stops increasing
- Example: all links have one unit of BW.
- All flows increase to 0.25 when link 4 - 5 saturates. No further increase is achieved

6

Queuing Principles

FIFO: Simple queuing model
- First come first served
- Drop packets if buffers is full (tail drop)
- FIFO: scheduling policy
- Tail drop: drop policy

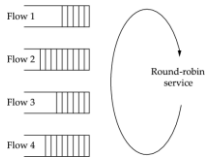


Problem
- All flows are treated the same
- Fast rate flow fills up buffer
- Slow rate may not get service

7

Fair Queuing

Attempt to neutralize the advantage of high rate flows
- Maintain a queue for each flow
- Service the queues in a round-robin (RR) fashion
- Drop packets when a queue is full



How about packet sizes?
- Not all flows use the same packet size
- Per bit RR could ensure fairness

8

Fair Queuing

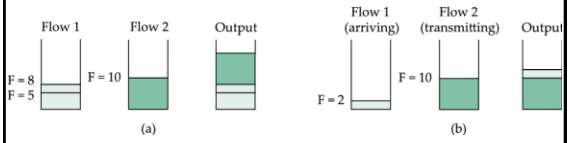
Bit-by-bit round robin
- Pi: length of packet i, Si start of transmission of *i*, F_i: end of transmission

$$F_i = S_i + P_i$$

When does transmission of packet i start? A_i: arrival of *i*

$$S_i = \max\{F_i\text{-}1, A_i\}, \qquad F_i = \max\{F_i\text{-}1, A_i\} + P_i$$

Next packet to transmit. One with smallest F_i



9

TCP Congestion Control

Additive Increase/Multiplicative decrease strategy

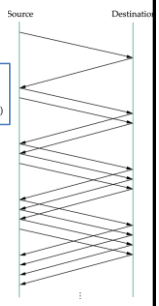
```
CW= CongestionWindow
MaxWindow = min(CW, AdvertisedWindow)
EfWindow = MaxWindow - (LastByteSent - LastByteACKed)
```

If packet loss, `CW = CW/2`

If a packet is ACKed

```
Increment = MSS x (MSS/CW)
CW = CW + increment
```

After CW packets are ACKed CW is increased by MSS (1 packet)

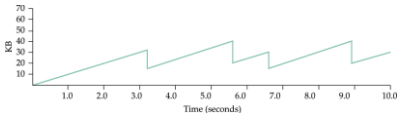


10

Performance

Sawtooth behavior

Plots the evolution of CW as a function of time



Source reduces the rate much faster than increasing the rate

AIMD is necessary for achieving stable operation

Intuitive reason: Larger window size is worse than smaller window size

11

Slow Start Mechanism

AIMD is employed when operating near optimal rate

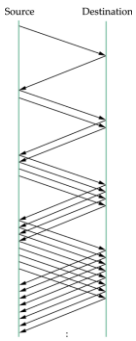
- Ramping up to the optimal rate may take a long time
- Assume that the optimal window size is 100 x MSS
- Requires 99 RTTs to get to optimal window size

Slow start phase

- For every ACK received, increment CongestionWindow by MSS
- Results in doubling the window size every RTT
- Why is this exponential growth of window size referred to as slow start?

Threshold for slow start phase

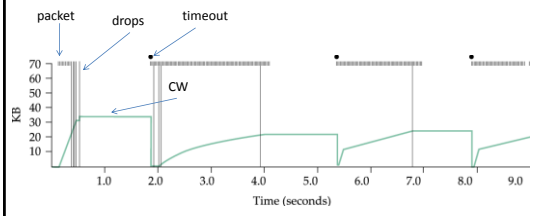
- ssthresh: Continue slow start until congestion window reaches this threshold
- Beyond this, additive increase is employed



12

Performance of Slow Start

Evolution of CW with time



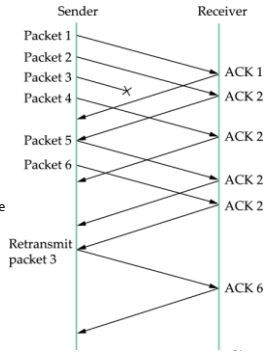
13

Fast Retransmit and fast Recovery

Fast retransmit
- Waiting for timeout may be expensive
- Arrival of duplicate ACKs indicate out-of-order packet arrival
- If three duplicate acknowledgments are received, retransmit
- Only the first unacknowledged packet

Entering fast retransmit implies congestion
- Delivery of out-of-order packets implies the congestion may not be serious
- Don't let the CW to 0 and restart
- Instead, reduce CW to half and resume additive increase

Slow start is used only at the beginning or when coarse timeout occurs



Congestion Avoidance

TCP causes congestion and then backs off
- Alternative is to try avoid congestion in the first place
- TCP always tries to probe for more bandwidth
- TCP requires packet drops to estimate bandwidth on a link

Goal is to predict the congestion and take early precautions

Congestion avoidance mechanisms
- DEC-bit scheme
- Random early detection (RED)
- Source-based congestion avoidance

15

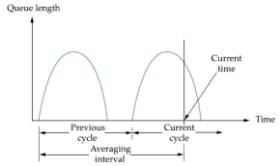
DECbit – Router part

Developed on the Digital Network Architecture (DNA)

- A connectionless network with a connection oriented transport protocol

Split the responsibility between routers and end hosts

- DEC-bit: A bit to determine early congestion is added to every packet
- DECbit is set to 1 in a packet if the average queue length at the router is greater or equal to 1
- Queue length is counted over last busy period + idle + current busy period



16

DECbit – Source part

How does source adjust the rate?

- Source maintains a congestion window, similar to TCP
- Observes how many packets has the congestion bit set to 1 in the last window worth of packets
- If less than 50% of the ACKs have the DEC-bit set, then increase the window by 1 packet
- Otherwise, set the window to 0.875 times the original value
- 50% was chosen based on analysis - corresponds to peak value of the power curve
- Additive increase and multiplicative decrease makes the mechanism stable

Also referred to as Explicit Congestion Notification

17

Random Early Detection (RED)

Every router monitors its queue length

- Unlike DEC-bit, implicit notification by dropping packets
- Designed to be used in conjunction with TCP
- Drop packets “early” to notify end hosts, hence adjust window sooner

How to drop packets?

- Drop packets according to a drop probability whenever queue length is above drop level

Algorithmic details

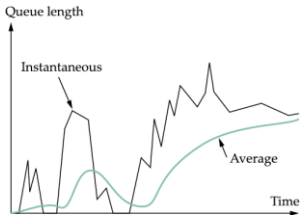
- Compute an average queue length using a weighted running average
- $0 < w < 1$

```
AvgLen = (1 – w ) x AvgLen + (w x SampleLen)
```

18

Why use Running Average

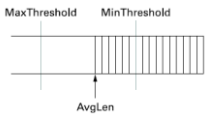
Instantaneous queue length is influenced by burstiness of internet traffic



19

RED Algorithm

Two queue thresholds



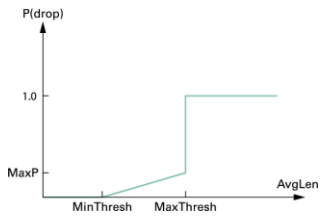

```
if AvgLen <= MinThreshold
    queue the packet

if MinThreshold < AvgLen < MaxThreshold
    calculate probability P
    drop the arriving packet with probability P

if MaxThreshold <= AvgLen
    drop the arriving packet
```

20

Calculation of Drop Probability

$$TempP = \frac{MaxP \times (AvgLen - MinThreshold)}{(MaxThreshold - MinThreshold)}$$

$$P = TempP / (1 - count \times TempP)$$



21

Source-based Congestion Avoidance

General idea

- Watch for some sign from the network and if nothing is done, congestion will occur

Example

- Measurable increase in RTT for each successive packet sent
- For every two RTT, check if current RTT is greater than average of minimum & maximum RTT
  - If so, reduce the congestion window by one-eighth
- Once in every two RTT, compute: (CurrentWindow - OldWindow) x (CurrentRTT - OldRTT)
  - If the result is positive, decrease window by one-eighth
  - Otherwise, increase the window size by one packet
- Compare throughput obtained with that obtained when window size was one packet less
  - If the difference is less than one half of the throughput when only one packet was in transit, reduce window

22