Lecture 21: Buffering & Scheduling

CSE 123: Computer Networks
Aaron Schulman
Lecture 21 Overview

- Router buffer Management
  - FIFO
  - RED

- Router traffic Policing/Scheduling
Key Router Challenges

- **Buffer management**: which packet to drop when?
  - We only have finite-length queues
- **Scheduling**: which packet to transmit next?
FIFO + drop-tail
  - Simplest choice
  - Used widely in the Internet

FIFO (first-in-first-out)
  - Implies single class of traffic

Drop-tail
  - Arriving packets get dropped when queue is full regardless of flow or importance

Important distinction:
  - FIFO: scheduling discipline
  - Drop-tail: drop policy
Leaves responsibility of preventing congestion completely to the edges
  - Transport protocols will need to make sure queues in routers are not so full that they drop, (called congestion control, later class)

Does not separate between different flows
  - Packet can be dropped regardless of what TCP or UDP flow it is part of

No policing: send more packets → get more service

Synchronization: end hosts react to same problems
Active Queue Management

- Design active router queue management to aid in reducing the congestion at routers

- Why?
  - Router has unified view of queuing behavior
  - Routers see actual queue occupancy (distinguish queue delay and propagation delay).
  - Routers can see if congestion is transient
Design Objectives

- Keep throughput high and delay low
  - High power (throughput/delay)

- Accommodate bursts

- Queue size should reflect ability to accept bursts rather than steady-state queuing

- Help with transport protocol (e.g., TCP) performance with minimal hardware changes in router
Random Early Detection (RED)

- Detect incipient congestion

- **Assume** hosts respond to lost packets
  - We know they will retransmit based on losses
  - Soon we will see *losses will also make them slow down!*

- Avoid window synchronization
  - Randomly mark packets

- Do not bias against bursty traffic
RED Algorithm

- Maintain running average of queue length in router
  - If $\text{avg} < \text{min}_{\text{th}}$, do nothing
    - Low queuing, send packets through
  - If $\text{avg} > \text{max}_{\text{th}}$, drop packet
    - Protection from misbehaving sources
  - Else drop/mark packet in a manner proportional to queue length
    - Dropping notifies sources of incipient congestion
    - Q: Dropping vs Marking (Explicit Congestion Notification). Why drop to notify of congestion rather than mark in header?
RED Operation

Max thresh

Min thresh

Average Queue Length

P(drop)

1.0

max_P

min_th max_th

Avg queue length
So far we’ve done **traffic policing**
- Limit the rate of flows regardless of the load in the network

In general, need **scheduling**
- Dynamically allocate resources when multiple flows compete
- Give each “flow” (or src/destination pair) own queue (at least theoretically)

**Weighted fair queuing**
- Proportional-share scheduling
- Schedule round-robins among queues in proportion to some weight parameter
Example with contending hosts

1 UDP (10 Mbps) and 31 TCPs sharing 10 Mbps
(Note: TCP will throttle due to congestion control)
UDP vs. TCP w/FIFO
TCP vs. UDP w/Fair Queuing

Throughput (Mbps)

Flow Number

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32

FQ
(Weighted) Fair Queuing

Flow 1
Flow 2
Flow n

I/P

O/P

CSE 123 – Lecture 21: Buffering & Scheduling
Fair Queuing

- Maintain a queue for each flow
  - What is a flow?
    - We approximate this with the 5 tuple
      - \((IP \text{ src, IP } \text{ dst, port src, port dst, protocol})\)

- Implements **max-min fairness**: each flow receives
  \(\min(r_i, f)\), where
  - \(r_i\) – flow arrival rate
  - \(f\) – link fair rate (see next slide)

- **Weighted Fair Queuing** (WFQ) – associate a weight with each flow to divvy bandwidth up non-equally
If link congested, compute $f$ such that

$$\sum_i \min(r_i, f) = C$$

- $f = 4$:
  - $\min(8, 4) = 4$
  - $\min(6, 4) = 4$
  - $\min(2, 4) = 2$
Weighted Fair Queuing

- Associate a weight $w_i$ with each flow $i$
- If link congested, compute $f$ such that

$$\sum_i \min(r_i, f \times w_i) = C$$

Flow $i$ is guaranteed to be allocated a rate $\geq w_i \cdot C / (\sum_k w_k)$

If $\sum_k w_k \leq C$, flow $i$ is guaranteed to be allocated a rate $\geq w_i$
WFQ Scheduling: Fluid Flow

- Simplification: Flows can be served one bit at a time

- WFQ can be implemented using bit-by-bit weighted round robin
  - During each round from each flow that has data to send, send a number of bits equal to the flow’s weight
Fluid Flow Example

- Red flow has packets backlogged between time 0 and 10
- Other flows have packets continuously backlogged
- All packets have the same size
Packet (real) system can’t do bit-by-bit splitting: packet transmission cannot be preempted. Why?

Issue: flows can get more bandwidth by sending bigger packets

Solution: serve packets in the order in which they would have finished being transmitted in the fluid flow system
Packet-Based Example

Service in fluid flow system

- Select the first packet that finishes in the fluid flow system

Packet system

time
For next time…

- Stopping congestion at hosts instead of routers
  
  Congestion control!
  - Read Ch. 6.3 in P&D

- Project 3 Due Wednesday!