CS 268: Computer Networking

L-6 Router Congestion Control

TCP & Routers

- RED
- XCP
- Assigned reading
  - [FJ93] Random Early Detection Gateways for Congestion Avoidance
  - [KHR02] Congestion Control for High Bandwidth-Delay Product Networks
Overview

- Queuing Disciplines
- RED
- RED Alternatives
- XCP

Queuing Disciplines

- Each router must implement some queuing discipline
- Queuing allocates both bandwidth and buffer space:
  - Bandwidth: which packet to serve (transmit) next
  - Buffer space: which packet to drop next (when required)
- Queuing also affects latency
Packet Drop Dimensions

- Aggregation
- Per-connection state
- Single class
- Class-based queuing
- Drop position
  - Head
  - Tail
  - Random location
- Early drop
- Overflow drop

Typical Internet Queuing

- 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
FIFO + Drop-tail Problems

- Leaves responsibility of congestion control to edges (e.g., TCP)
- Does not separate between different flows
- No policing: send more packets → get more service
- Synchronization: end hosts react to same events

Active Queue Management

- Design active router queue management to aid congestion control
- Why?
  - Routers can distinguish between propagation and persistent queuing delays
  - Routers can decide on transient congestion, based on workload
Active Queue Designs

- Modify both router and hosts
  - DECbit: congestion bit in packet header
- Modify router, hosts use TCP
  - Fair queuing
    - Per-connection buffer allocation
  - RED (Random Early Detection)
    - Drop packet or set bit in packet header as soon as congestion is starting

Overview

- Queuing Disciplines
- RED
- RED Alternatives
- XCP
Internet Problems

- Full queues
  - Routers are forced to have large queues to maintain high utilizations
  - TCP detects congestion from loss
    - Forces network to have long standing queues in steady-state
- Lock-out problem
  - Drop-tail routers treat bursty traffic poorly
  - Traffic gets synchronized easily → allows a few flows to monopolize the queue space

Design Objectives

- Keep throughput high and delay low
- Accommodate bursts
- Queue size should reflect ability to accept bursts rather than steady-state queuing
- Improve TCP performance with minimal hardware changes
Lock-out Problem

- Random drop
  - Packet arriving when queue is full causes some random packet to be dropped
- Drop front
  - On full queue, drop packet at head of queue
- Random drop and drop front solve the lock-out problem but not the full-queues problem

Full Queues Problem

- Drop packets before queue becomes full (early drop)
- Intuition: notify senders of incipient congestion
  - Example: early random drop (ERD):
    - If queue length > drop level, drop each new packet with fixed probability $p$
    - Does not control misbehaving users
Random Early Detection (RED)

- Detect incipient congestion, allow bursts
- Keep power (throughput/delay) high
  - Keep average queue size low
  - Assume hosts respond to lost packets
- Avoid window synchronization
  - Randomly mark packets
- Avoid bias against bursty traffic
- Some protection against ill-behaved users

RED Algorithm

- Maintain running average of queue length
- If avgq < min_{th} do nothing
  - Low queuing, send packets through
- If avgq > max_{th}, drop packet
  - Protection from misbehaving sources
- Else mark packet in a manner proportional to queue length
  - Notify sources of incipient congestion
RED Operation

![Diagram showing RED operation with min threshold, max threshold, average queue length, and P(drop) functions.]

RED Algorithm

- Maintain running average of queue length
  - Byte mode vs. packet mode – why?
- For each packet arrival
  - Calculate average queue size (avg)
  - If $\min_{th} \leq \text{avgq} < \max_{th}$
    - Calculate probability $P_a$
    - With probability $P_a$
      - Mark the arriving packet
    - Else if $\max_{th} \leq \text{avg}$
      - Mark the arriving packet
Queue Estimation

- Standard EWMA: \( \text{avgq} = (1-w_q) \text{avgq} + w_q \text{qlen} \)
  - Special fix for idle periods – why?
- Upper bound on \( w_q \) depends on \( \text{min}_\text{th} \)
  - Want to ignore transient congestion
  - Can calculate the queue average if a burst arrives
    - Set \( w_q \) such that certain burst size does not exceed \( \text{min}_\text{th} \)
- Lower bound on \( w_q \) to detect congestion relatively quickly
- Typical \( w_q = 0.002 \)

Thresholds

- \( \text{min}_\text{th} \) determined by the utilization requirement
  - Tradeoff between queuing delay and utilization
- Relationship between \( \text{max}_\text{th} \) and \( \text{min}_\text{th} \)
  - Want to ensure that feedback has enough time to make difference in load
  - Depends on average queue increase in one RTT
  - Paper suggest ratio of two
    - Current rule of thumb is factor of three
Packet Marking

• Marking probability based on queue length
  • \( P_b = \max_p(\text{avgq} - \text{min}_{th}) / (\max_{th} - \text{min}_{th}) \)
• Just marking based on \( P_b \) can lead to clustered marking
  • Could result in synchronization
  • Better to bias \( P_b \) by history of unmarked packets
    • \( P_a = P_b / (1 - \text{count} \cdot P_b) \)

Packet Marking

• \( \max_p \) is reflective of typical loss rates
• Paper uses 0.02
  • 0.1 is more realistic value
• If network needs marking of 20-30% then need to buy a better link!
• Gentle variant of RED (recommended)
  • Vary drop rate from \( \max_p \) to 1 as the avgq varies from \( \max_{th} \) to \( 2 \cdot \max_{th} \)
  • More robust to setting of \( \max_{th} \) and \( \max_p \)
Extending RED for Flow Isolation

• Problem: what to do with non-cooperative flows?
• Fair queuing achieves isolation using per-flow state – expensive at backbone routers
  • How can we isolate unresponsive flows without per-flow state?
• RED penalty box
  • Monitor history for packet drops, identify flows that use disproportionate bandwidth
  • Isolate and punish those flows

Overview

• Queuing Disciplines
• RED
• RED Alternatives
• XCP
FRED

- Fair Random Early Drop (Sigcomm, 1997)
- Maintain per flow state only for active flows (ones having packets in the buffer)
- $\min_q$ and $\max_q \rightarrow$ min and max number of buffers a flow is allowed occupy
- $\text{avgcq} = \text{average buffers per flow}$
- Strike count of number of times flow has exceeded $\max_q$

FRED – Fragile Flows

- Flows that send little data and want to avoid loss
- $\min_q$ is meant to protect these
- What should $\min_q$ be?
  - When large number of flows $\rightarrow$ 2-4 packets
    - Needed for TCP behavior
  - When small number of flows $\rightarrow$ increase to $\text{avgcq}$
FRED

- Non-adaptive flows
  - Flows with high strike count are not allowed more than avgcq buffers
  - Allows adaptive flows to occasionally burst to \( \text{max}_q \) but repeated attempts incur penalty

CHOKe

- CHOse and Keep/Kill (Infocom 2000)
  - Existing schemes to penalize unresponsive flows (FRED/penalty box) introduce additional complexity
  - Simple, stateless scheme
- During congested periods
  - Compare new packet with random pkt in queue
  - If from same flow, drop both
  - If not, use RED to decide fate of new packet
CHOKe

- Can improve behavior by selecting more than one comparison packet
  - Needed when more than one misbehaving flow
- Does not completely solve problem
  - Aggressive flows are punished but not limited to fair share
  - Not good for low degree of multiplexing → why?

Stochastic Fair Blue

- Same objective as RED Penalty Box
  - Identify and penalize misbehaving flows
- Create L hashes with N bins each
  - Each bin keeps track of separate marking rate ($p_m$)
  - Rate is updated using standard technique and a bin size
  - Flow uses minimum $p_m$ of all L bins it belongs to
  - Non-misbehaving flows hopefully belong to at least one bin without a bad flow
    - Large numbers of bad flows may cause false positives
Stochastic Fair Blue

• False positives can continuously penalize same flow

• Solution: moving hash function over time
  • Bad flow no longer shares bin with same flows
  • Is history reset \(\rightarrow\) does bad flow get to make trouble until detected again?
    • No, can perform hash warmup in background

Overview

• Queuing Disciplines
• RED
• RED Alternatives
• XCP
How does XCP Work?

Feedback:
Round Trip Time + 0.1 packet
Congestion Window
Feedback = + 0.1 packet
Congestion Header

How does XCP Work?

Round Trip Time
Congestion Window
Feedback = - 0.3 packet
How does XCP Work?

- Congestion Window = Congestion Window + Feedback

- XCP extends ECN and CSFQ

- Routers compute feedback without any per-flow state

How Does an XCP Router Compute the Feedback?

1. **Congestion Controller**
   - Algorithm:
     - Aggregate traffic changes by $\Delta$
     - $\Delta \sim$ Spare Bandwidth
     - $\Delta \sim$ Queue Size
     - So, $\Delta = \alpha \cdot d_{avg} \cdot$ Spare - $\beta \cdot$ Queue

2. **Fairness Controller**
   - Algorithm:
     - If $\Delta > 0 \Rightarrow$ Divide $\Delta$ equally between flows
     - If $\Delta < 0 \Rightarrow$ Divide $\Delta$ between flows proportionally to their current rates
Getting the devil out of the details …

**Congestion Controller**

\[ \Delta = \alpha \cdot d_{avg} \cdot \text{Spare} - \beta \cdot \text{Queue} \]

**Theorem:** System converges to optimal utilization (i.e., stable) for any link bandwidth, delay, number of sources if:

\[ 0 < \alpha < \frac{\pi}{4\sqrt{2}} \quad \text{and} \quad \beta = \alpha^2 \cdot \frac{\sqrt{2}}{2} \]

**Fairness Controller**

**Algorithm:**
- If \( \Delta > 0 \) ⇒ Divide \( \Delta \) equally between flows
- If \( \Delta < 0 \) ⇒ Divide \( \Delta \) between flows proportionally to their current rates

Need to estimate number of flows \( N \)

\[ N = \sum_{pkts} n \times \left( \frac{Cwnd_{pkts}}{RTT_{pkts}} \right) \]

\( RTT_{pkts} \): Round Trip Time in header

No Parameter Tuning

No Per-Flow State

---

**Lessons**

- TCP alternatives
  - TCP being used in new/unexpected ways
  - Key changes needed
- Routers
  - FIFO, drop-tail interacts poorly with TCP
  - Various schemes to desynchronize flows and control loss rate
- Fair-queuing
  - Clean resource allocation to flows
  - Complex packet classification and scheduling
- Core-stateless FQ & XCP
  - Coarse-grain fairness
  - Carrying packet state can reduce complexity
Discussion

- XCP
  - Misbehaving routers
  - Deployment (and incentives)
- RED
  - Parameter setting