15-744: Computer Networking

L-6 TCP & Routers

TCP & Routers

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

2

Overview

- Queuing Disciplines
- RED
- RED Alternatives
- XCP

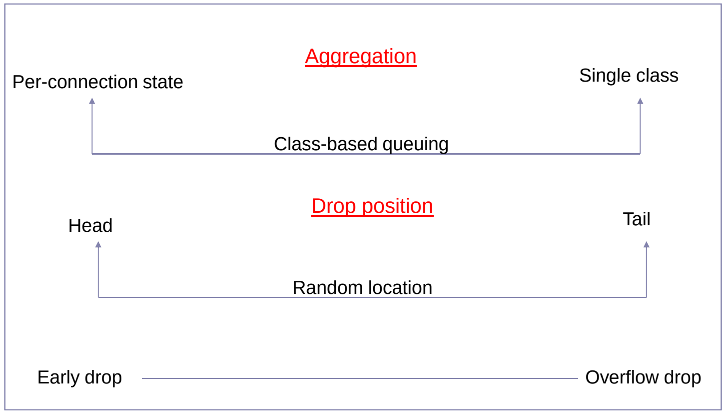
3

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

4

Packet Drop Dimensions



5

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

6

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

7

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

8

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 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 qlen > drop level, drop each new packet with fixed probability *p*
    - Does not control misbehaving users

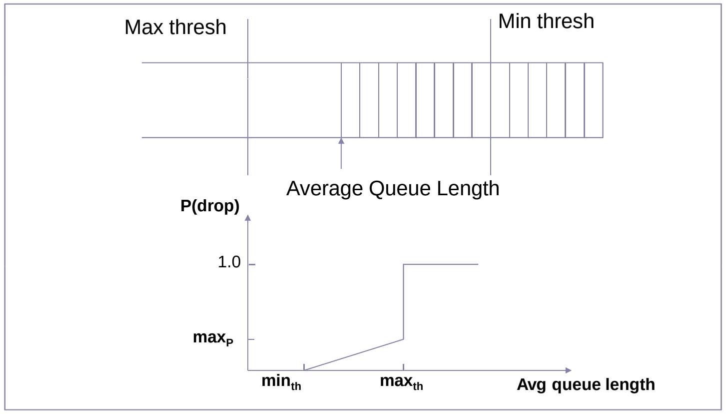
Random Early Detection (RED)

- 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



17

RED Algorithm

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

18

Queue Estimation

- Standard EWMA: $avgq = (1\text{-}w_q)\ avgq + w_q qlen$
  - Special fix for idle periods – why?
- Upper bound on w_q depends on min_{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 min_{th}
- Lower bound on w_q to detect congestion relatively quickly
- Typical $w_q = 0.002$

19

Thresholds

- min_{th} determined by the utilization requirement
  - Tradeoff between queuing delay and utilization
- Relationship between max_{th} and min_{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 2
    - Current rule of thumb is factor of 3

20

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* 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 → min and max number of buffers a flow is allowed occupy
- avgcq = 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 → 2-4 packets
    - Needed for TCP behavior
  - When small number of flows → increase to avgcq

26

FRED

- Non-adaptive flows
  - Flows with high strike count are not allowed more than avgcq buffers
  - Allows adaptive flows to occasionally burst to max_q but repeated attempts incur penalty

27

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

28

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?

29

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

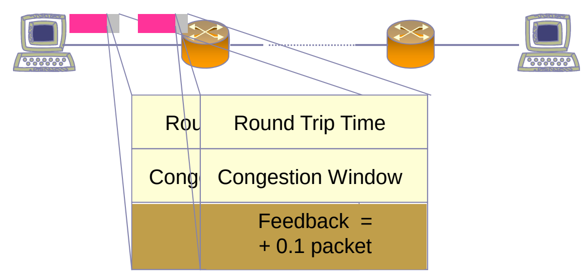
Stochastic Fair Blue

- 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 → does bad flow get to make trouble until detected again?
    - No, can perform hash warmup in background

Overview

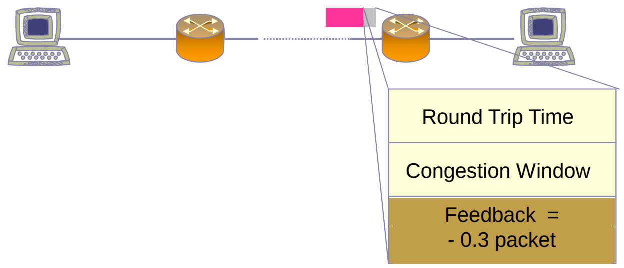
- Queuing Disciplines
- RED
- RED Alternatives
- XCP

How does XCP Work?



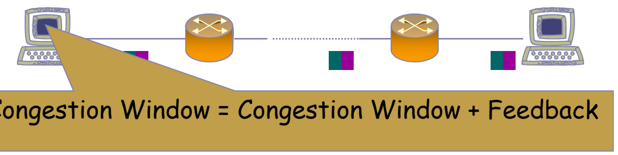
Congestion Header

How does XCP Work?



34

How does XCP Work?

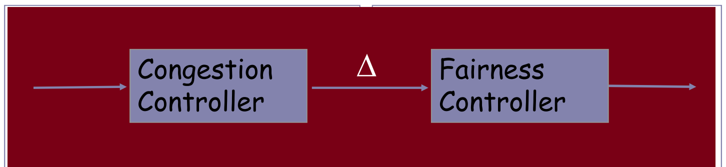


XCP extends ECN and CSFQ

Routers compute feedback without any per-flow state

35

How Does an XCP Router Compute the Feedback?



queue

MIMD

Algorithm:

Aggregate traffic changes by Δ

Δ ~ Spare Bandwidth

Δ ~ - Queue Size

So, $\Delta = \alpha\, d_{avg}\, Spare - \beta\, Queue$

Congestion He…

AIMD

Algorithm:

If Δ > 0 ⇒ Divide Δ equally between flows

If Δ < 0 ⇒ Divide Δ between flows proportionally to their current rates

36

Getting the devil out of the details …

Congestion Controller

$\Delta = \alpha\, d_{avg}\, Spare - \beta\, Queue$

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

$$0 < \alpha < \frac{\pi}{4\sqrt{2}} \quad and \quad \beta = \alpha^2\sqrt{2}$$

No Parameter Tuning

Fairness Controller

Algorithm:

If Δ > 0 ⇒ Divide Δ equally between flows

If Δ < 0 ⇒ Divide Δ between flows proportionally to their current rates

Need to estimate number of flows N

$$N = \sum_{pkts\ in\ T} \frac{1}{T \times (Cwnd_{pkt} / RTT_{pkt})}$$

RTT_{pkt}: Round Trip Time in header

No Per-Flow State

37

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