CE693: Adv. Computer Networking

L-5 Fair Queuing

Acknowledgments: Lecture slides are from the graduate level Computer Networks course thought by Srinivasan Seshan at CMU. When slides are obtained from other sources, a a reference will be noted on the bottom of that slide. A full list of references is provided on the last slide.

Overview

- TCP and queues
- Queuing disciplines
- RED
- Fair-queuing
- Core-stateless FQ
- XCP

Fairness Goals

- Allocate resources fairly
- Isolate ill-behaved users
  - Router does not send explicit feedback to source
  - Still needs e2e congestion control
- Still achieve statistical muxing
  - One flow can fill entire pipe if no contenders
  - Work conserving → scheduler never idles link if it has a packet

What is Fairness?

- At what granularity?
  - Flows, connections, domains?
- What if users have different RTTs/links/etc.
  - Should it share a link fairly or be TCP fair?
- Maximize fairness index?
  - Fairness = $(\Sigma x_i)^2 / n(\Sigma x_i^2)$ $0 < \text{fairness} < 1$
- Basically a tough question to answer – typically design mechanisms instead of policy
  - User = arbitrary granularity

Max-min Fairness

- Allocate user with “small” demand what it wants, evenly divide unused resources to “big” users
- Formally:
    - Resources allocated in terms of increasing demand
    - No source gets resource share larger than its demand
    - Sources with unsatisfied demands get equal share of resource

Max-min Fairness Example

- Assume sources 1..n, with resource demands X1..Xn in 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

Implementing max-min Fairness

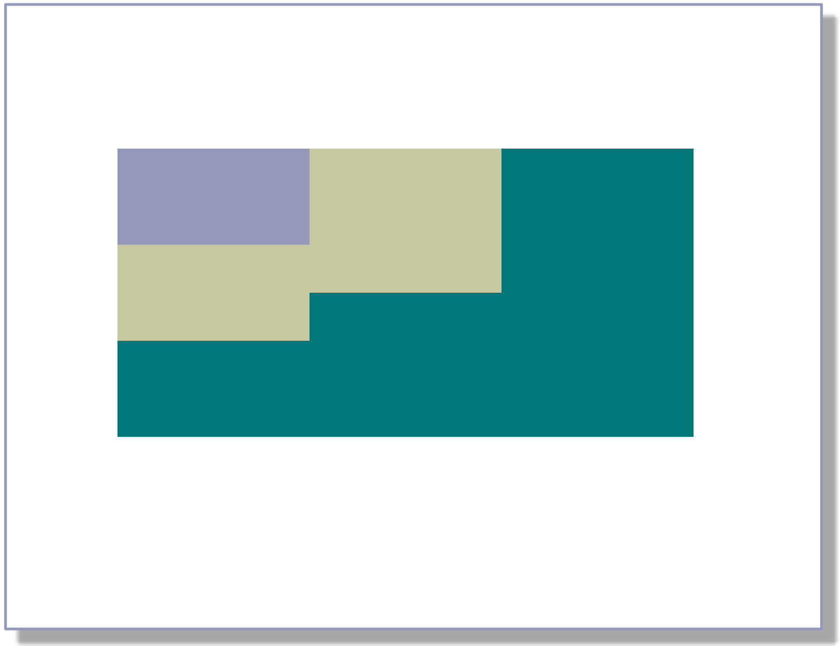
- Generalized processor sharing
  - Fluid fairness
  - Bitwise round robin among all queues
- Why not simple round robin?
  - Variable packet length → can get more service by sending bigger packets
  - Unfair instantaneous service rate
    - What if arrive just before/after packet departs?

Bit-by-bit RR

- Single flow: clock ticks when a bit is transmitted. For packet i:
  - P_i = length, A_i = arrival time, S_i = begin transmit time, F_i = finish transmit time
  - $F_i = S_i + P_i = \max(F_{i-1}, A_i) + P_i$
- Multiple flows: clock ticks when a bit from all active flows is transmitted → round number
  - Can calculate F_i for each packet if number of flows is known at all times
    - This can be complicated

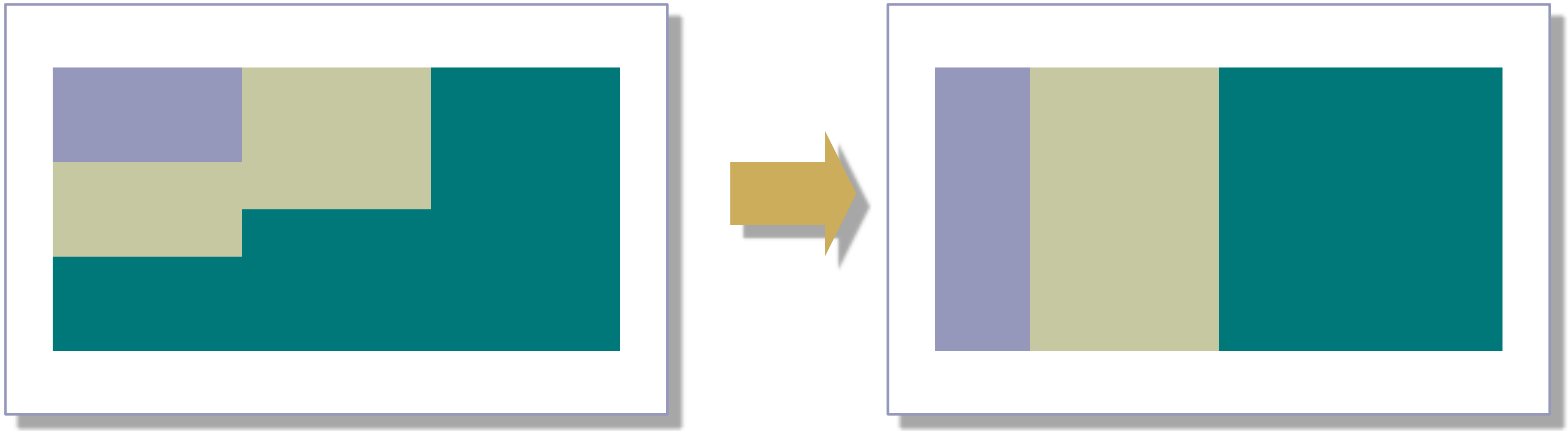
Bit-by-bit RR Illustration

- Not feasible to interleave bits on real networks
  - FQ simulates bit-by-bit RR

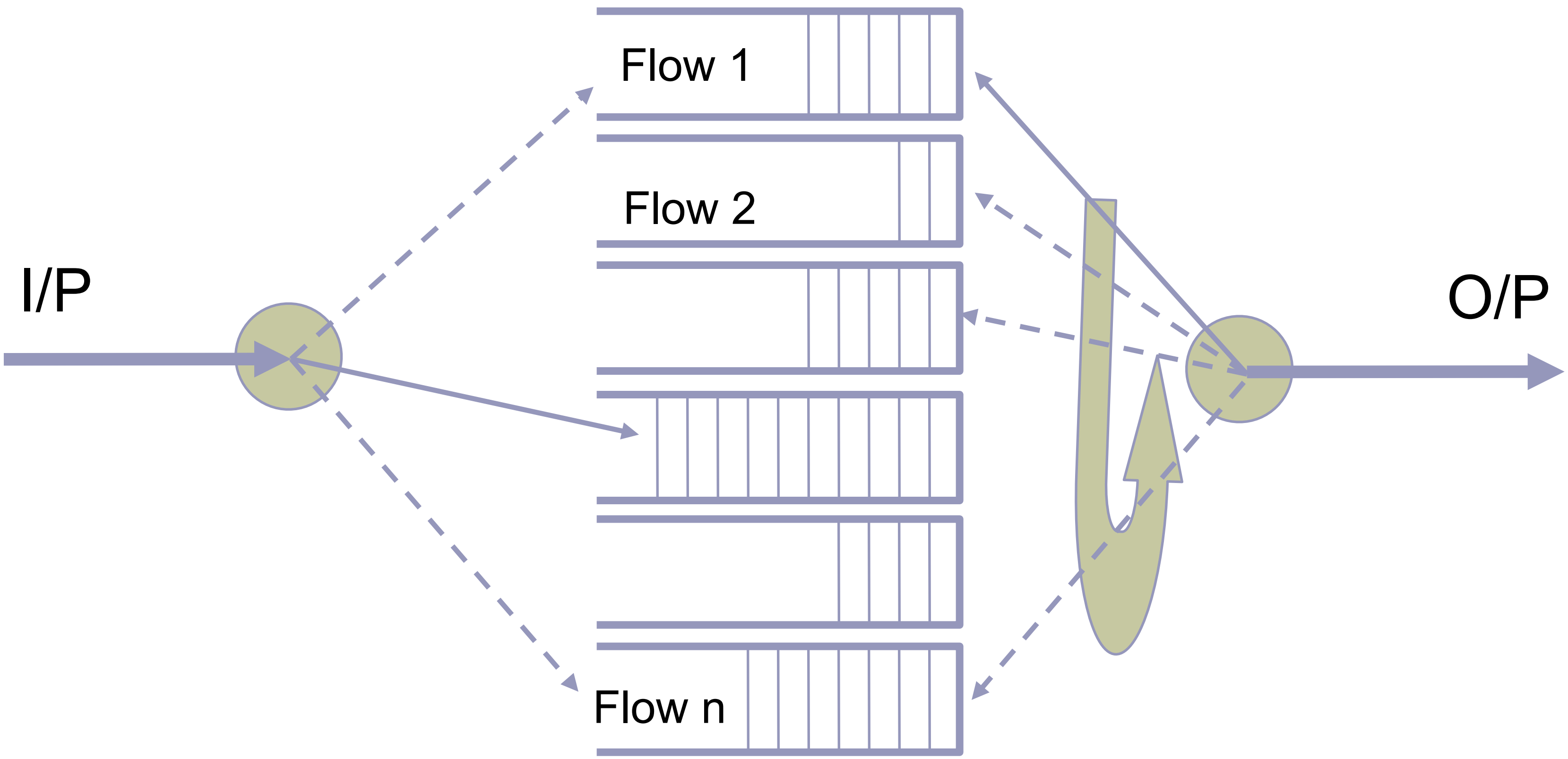


Fair Queuing

- Mapping bit-by-bit schedule onto packet transmission schedule
- Transmit packet with the lowest F_i at any given time
  - How do you compute F_i?

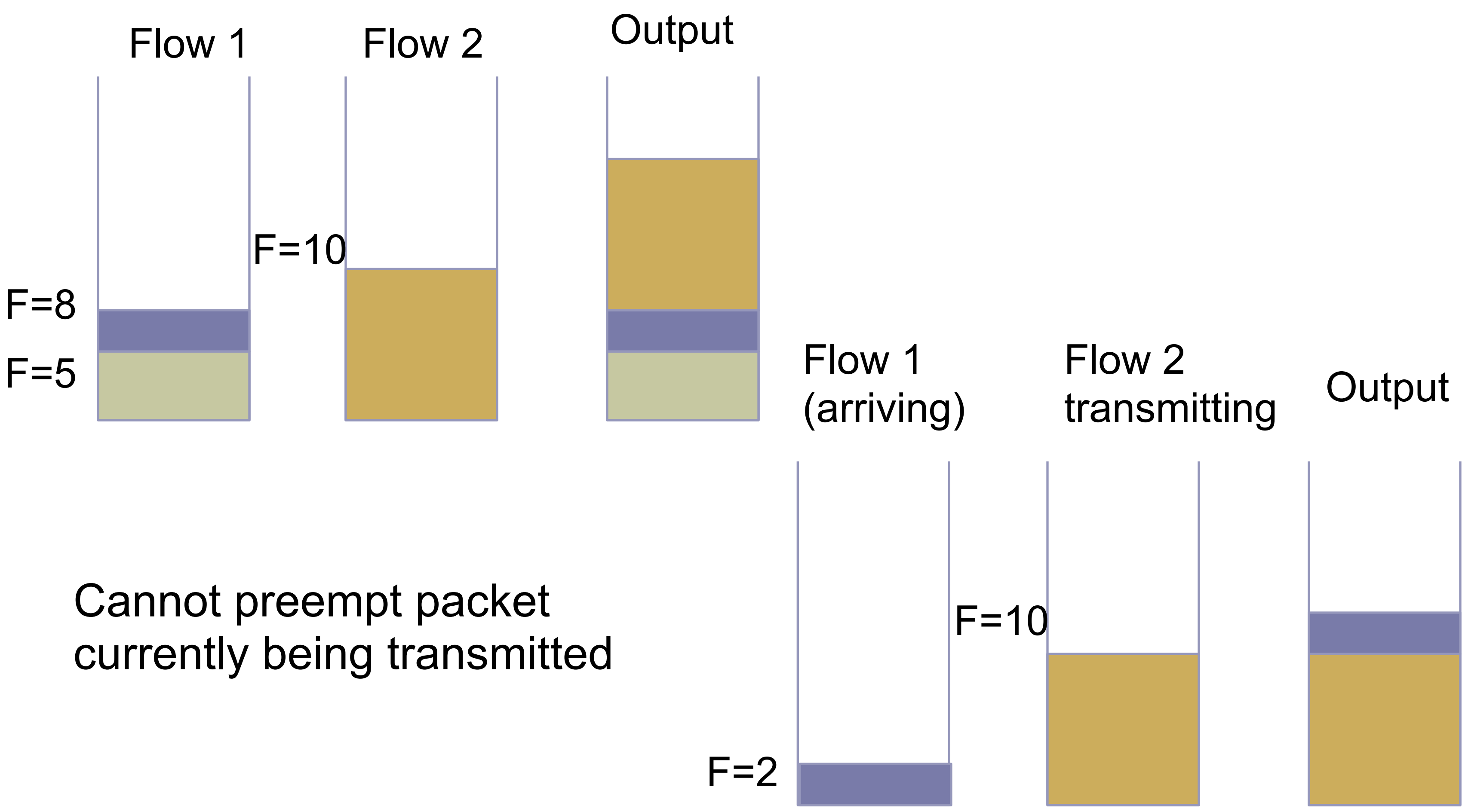


FQ Illustration



Variation: Weighted Fair Queuing (WFQ)

Bit-by-bit RR Example



Cannot preempt packet currently being transmitted

Fair Queuing Tradeoffs

- FQ can control congestion by monitoring flows
  - Non-adaptive flows can still be a problem – why?
- Complex state
  - Must keep queue per flow
    - Hard in routers with many flows (e.g., backbone routers)
    - Flow aggregation is a possibility (e.g. do fairness per domain)
- Complex computation
  - Classification into flows may be hard
  - Must keep queues sorted by finish times
  - Finish times change whenever the flow count changes

Overview

- TCP and queues
- Queuing disciplines
- RED
- Fair-queuing
- Core-stateless FQ
- XCP

Core-Stateless Fair Queuing

- Key problem with FQ is core routers
  - Must maintain state for 1000’s of flows
  - Must update state at Gbps line speeds
- CSFQ (Core-Stateless FQ) objectives
  - Edge routers should do complex tasks since they have fewer flows
  - Core routers can do simple tasks
    - No per-flow state/processing → this means that core routers can only decide on dropping packets not on order of processing
    - Can only provide max-min bandwidth fairness not delay allocation

Core-Stateless Fair Queuing

- Edge routers keep state about flows and do computation when packet arrives
- DPS (Dynamic Packet State)
  - Edge routers label packets with the result of state lookup and computation
- Core routers use DPS and local measurements to control processing of packets

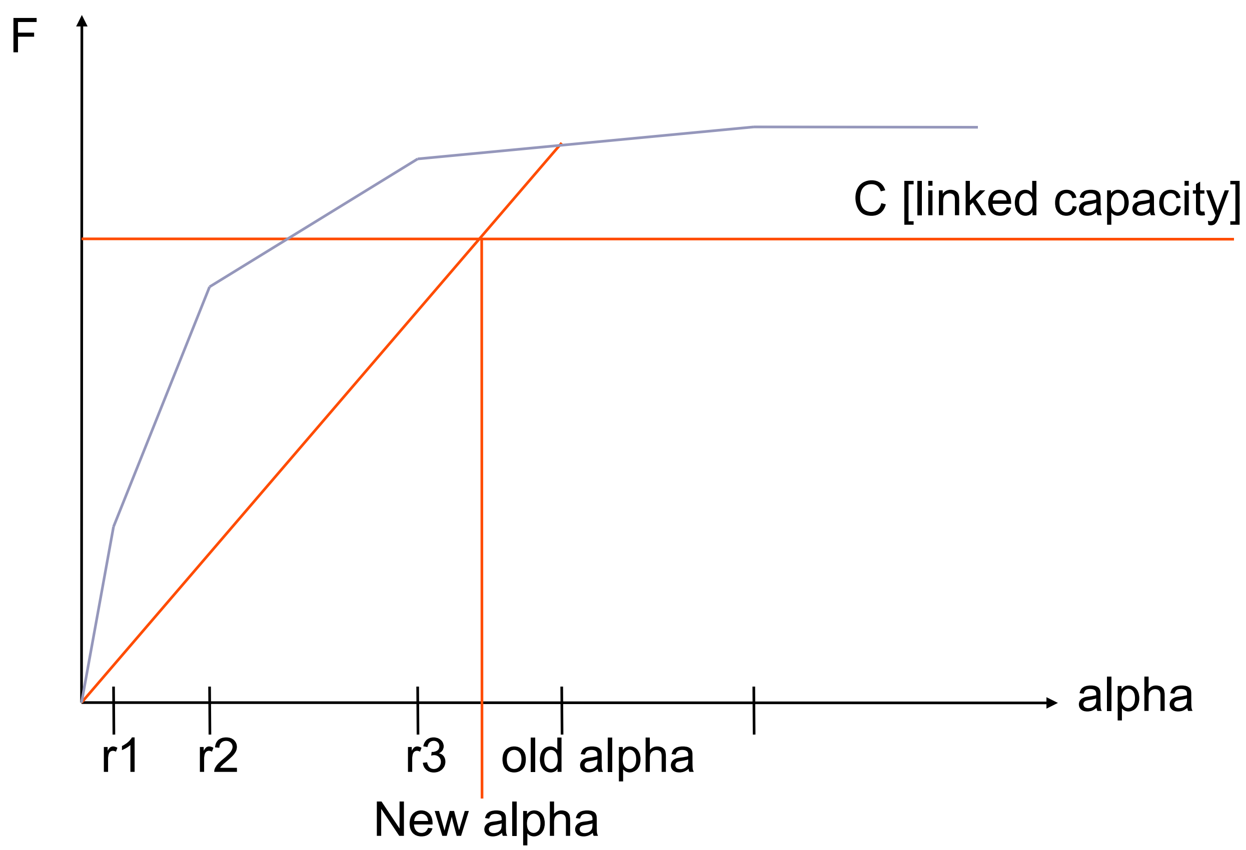
Edge Router Behavior

- Monitor each flow i to measure its arrival rate (r_i)
  - EWMA of rate
  - Non-constant EWMA constant
    - $e^{-T/K}$ where T = current interarrival, K = constant
    - Helps adapt to different packet sizes and arrival patterns
- Rate is attached to each packet

Core Router Behavior

- Keep track of fair share rate α
  - Increasing α does not increase load (F) by N * α
  - $F(\alpha) = \Sigma_i \min(r_i, \alpha)$ → what does this look like?
  - Periodically update α
  - Keep track of current arrival rate
    - Only update α if entire period was congested or uncongested
- Drop probability for packet = $\max(1 - \alpha/r, 0)$

F vs. Alpha



Estimating Fair Share

- Need $F(\alpha)$ = capacity = C
  - Can't keep map of $F(\alpha)$ values → would require per flow state
  - Since $F(\alpha)$ is concave, piecewise-linear
    - $F(0) = 0$ and $F(\alpha)$ = current accepted rate = F_c
    - $F(\alpha) = F_c / \alpha$
    - $F(\alpha_{new}) = C$ → $\alpha_{new} = \alpha_{old} * C/F_c$
- What if a mistake was made?
  - Forced into dropping packets due to buffer capacity
  - When queue overflows α is decreased slightly

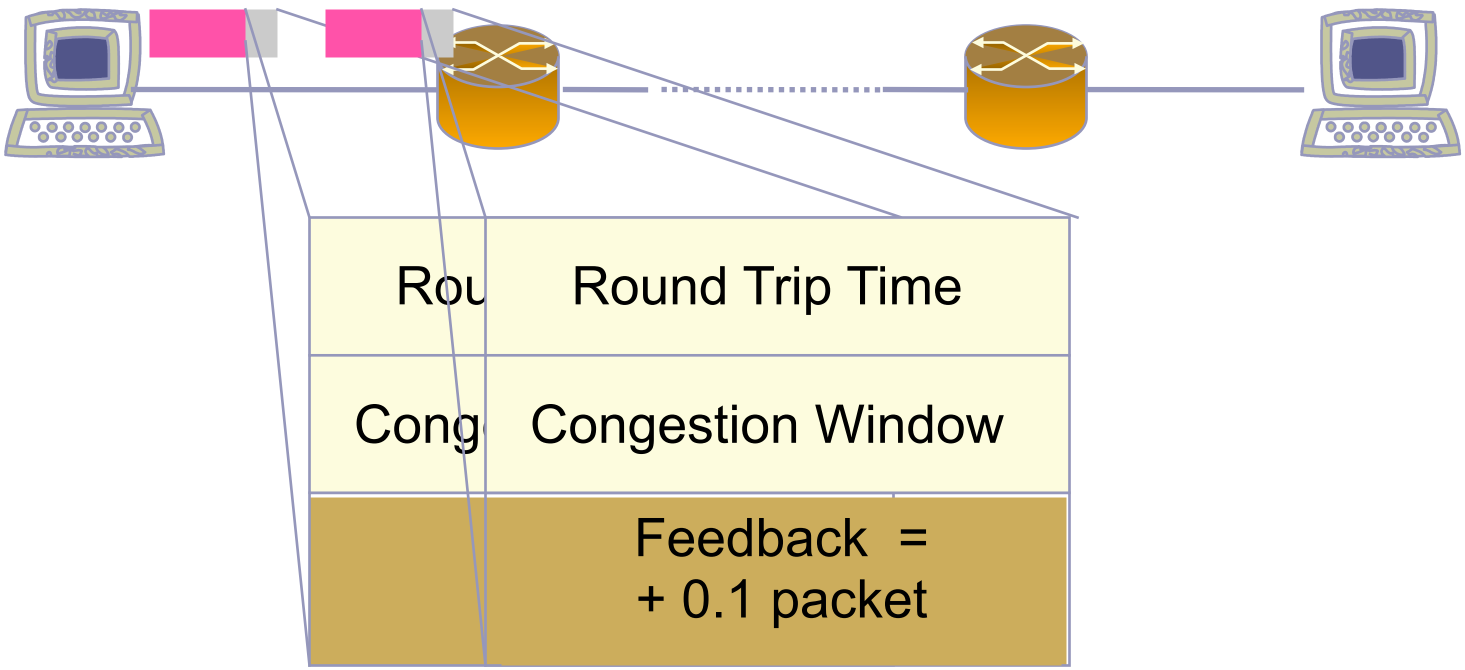
Other Issues

- Punishing fire-hoses – why?
  - Easy to keep track of in a FQ scheme
- What are the real edges in such a scheme?
  - Must trust edges to mark traffic accurately
  - Could do some statistical sampling to see if edge was marking accurately

Overview

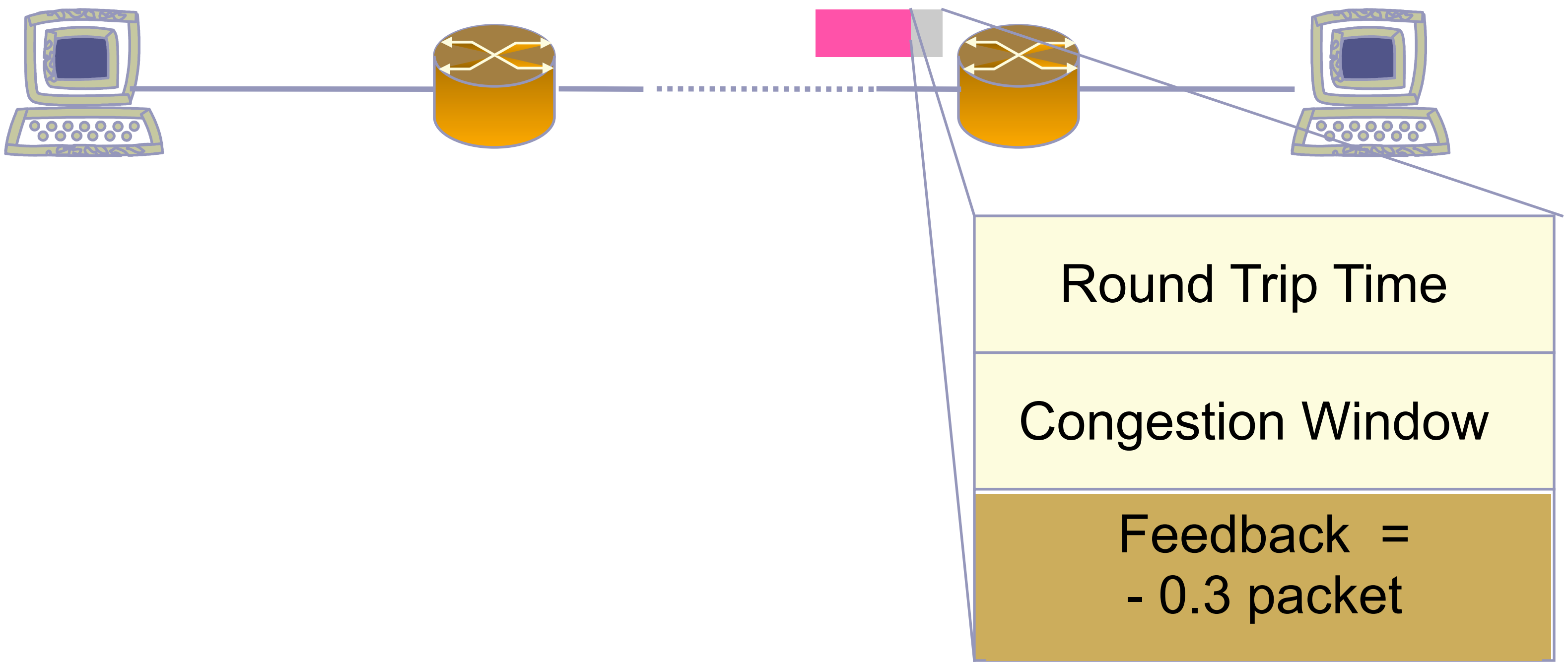
- TCP and queues
- Queuing disciplines
- RED
- Fair-queuing
- Core-stateless FQ
- XCP

How does XCP Work?

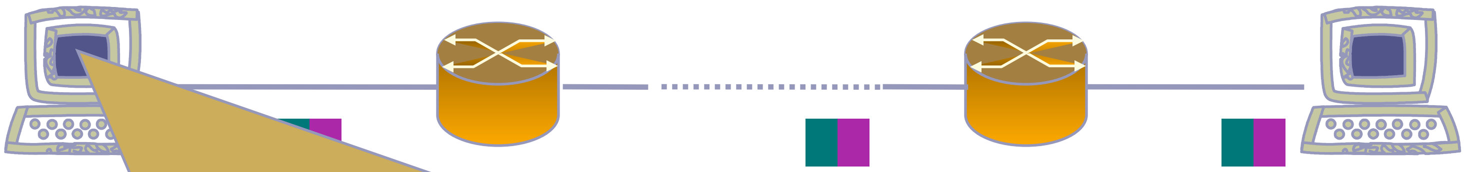


Congestion Header

How does XCP Work?



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?

Congestion Controller

Goal: Match[illegible] to link capacit[illegible]ueue

Looks at aggregate traffic & queue

MIMD

Algorithm:

Aggregate traffic changes by Δ

$\Delta \sim$ Spare Bandwidth

$\Delta \sim$ - Queue Size

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

Fairness Controller

Goal: [illegible]en flows[illegible] fairness

Looks at a flow's state in Congestion He[illegible]

AIMD

Algorithm:

If $\Delta > 0 \Rightarrow$ Divide Δ equally between flows

If $\Delta < 0 \Rightarrow$ Divide Δ between flows proportionally to their current rates

Getting the devil out of the details …

Congestion Controller

$\Delta = \alpha\, d_{avg}$ Spare - β 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 $\Delta > 0 \Rightarrow$ Divide Δ equally between flows

If $\Delta < 0 \Rightarrow$ 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

T : Counting Interval

Discussion

- RED
  - Parameter settings
- RED vs. FQ
  - How much do we need per flow tracking? At what cost?
- FQ vs. XCP/CSFQ
  - Is coarse-grained fairness sufficient?
  - Misbehaving routers/trusting the edge
  - Deployment (and incentives)
  - How painful is FQ
- XCP vs CSFQ
  - What are the key differences
- Granularity of fairness

Important Lessons

- How does TCP implement AIMD?
  - Sliding window, slow start & ack clocking
  - How to maintain ack clocking during loss recovery → fast recovery
- How does TCP fully utilize a link?
  - Role of router buffers
- TCP alternatives
  - TCP being used in new/unexpected ways
  - Key changes needed

Lessons

- Fairness and isolation in routers
  - Why is this hard?
  - What does it achieve – e.g. do we still need congestion control?
- Routers
  - FIFO, drop-tail interacts poorly with TCP
  - Various schemes to desynchronize flows and control loss rate (e.g. RED)
- 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