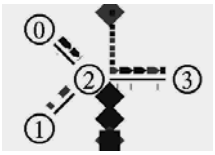
Internet Programming & Protocols Lecture 20

Active Queue Management (AQM)

ECN

XCP



Note: Vegas/Fast/Westwood need high precision timer. In ns, you want tcpTick_ to be 0.01 (ns default)

www.cs.utk.edu/~dunigan/ipp/

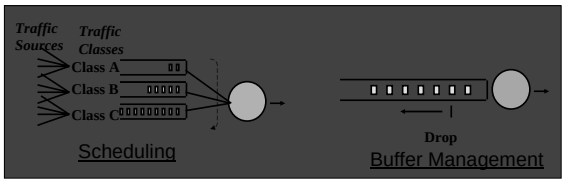
Active queue management

- Internet architecture assumes independence of end nodes from routers
  - Packets/flows can go through different routers
  - Transport layer and network layer are independent
- Routers do not guarantee service (best effort)
  - Transport protocols should recover from losses and adjust to varying RTT
- Router software should be simple
  - There are many end-nodes and few routers
  - Routers have large amounts of packets to process
- Routers are part of the problem, maybe make them part of the solution
- Current thinking is that maybe routers need to take a more active role in providing high throughput and fair service, and perhaps provide more explicit feedback to the transport protocol
  - Recall, old TCP had proposed ICMP source quench feedback from routers
  - DECnet and IBM SNA have explicit congestion feedback from routers

Queuing Disciplines

- Each router must implement some queuing discipline
- Queuing allocates bandwidth and buffer space:
  - Bandwidth: which packet to serve next (scheduling)
  - Buffer space: which packet to drop next (buff management)
- Queuing also affects latency
- There are more queuing disciplines than TCP flavors ☹

FIFO
RED
FRED
WFQ
CSFQ
DRR
...



Typical Internet Queuing

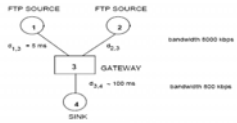
- FIFO + drop-tail
  - Simplest choice
  - Used widely in the Internet
- FIFO (first-in-first-out)
  - Implies single class of traffic
- Drop-tail
  - Arriving packets get dropped when queue is full regardless of flow or importance
- Important distinction:
  - FIFO: scheduling discipline
  - Drop-tail: drop (buffer management) policy

FIFO + Drop-tail Problems

- FIFO Issues: In a FIFO discipline, the service seen by a flow is *convoluted* with the arrivals of packets from all other flows!
  - No isolation between flows: full burden on e2e control
  - No policing: send more packets → get more service
- Drop-tail issues:
  - Routers are forced to have have large queues to maintain high utilizations
  - Larger buffers => larger steady state queues/delays
  - Bias against flows with long RTT
  - Synchronization: end hosts react to same events because packets tend to be lost in bursts (phase effects)
  - Lock-out: a side effect of burstiness and synchronization is that a few flows can monopolize queue space

Phase effects (floyd '92)

- Simulation of two flows (Newreno)
- Router with droptail queue
- Vary RTT of flow 2
- Throughput sensitive to RTT



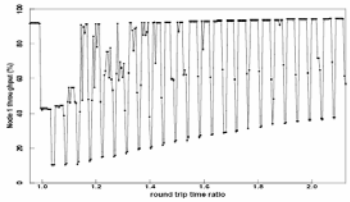
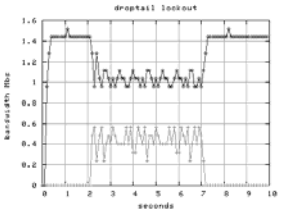


Figure 4: Node 1 throughput as a function of node 2's roundtrip time.

Droptail starvation (lock out)

- If there are adequate router buffers, two flows sharing a bottleneck link will each get a fair share
- If router queue is not big enough, packets are dropped, and with droptail queue, often one flow's packets tend to get into the queue first, and the other flow experiences most (all?) of the packet drops
- Chap 11, two flows sharing same bottleneck link
  - Green flow experiences ALL of the packet drops!



Queue Management Ideas

- Synchronization, lock-out:
  - Random drop: drop a randomly chosen packet
  - Drop front: drop packet from head of queue
- High steady-state queuing vs burstiness:
  - Early drop: Drop packets before queue full
  - Do not drop packets "too early" because queue may reflect only burstiness and not true overload
- Misbehaving vs Fragile flows:
  - Drop packets proportional to queue occupancy of flow
  - Try to protect fragile flows from packet loss (eg: color them or classify them on the fly)
- Drop packets vs mark packets:
  - Dropping packets interacts w/ reliability mechanisms
  - Mark packets: need to trust end-systems to respond!
    - ECN
    - XCP

Random drop

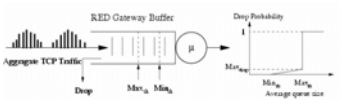
- Instead of droptail, randomly select a packet from queue to drop when queue becomes full
- Requires no state info in router
  - Simple (need random number generator)
  - No explicit flow id'ing
- Statistically, seems likely you will select packet from a flow with higher packet rate (more of its packets in the queue)
  - If arrivals were Poisson, any flow would be equally likely
  - But internet flows are correlated (packet trains), likely to pick higher rate flow
- '90 experiments showed random drop
  - Helped equal senders (reduced lock out / phase effects)
  - But did not help throughput or reduce packet drops ☹

Active Queue Management

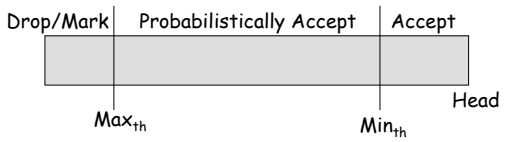
- **Proactively** Manage **Queues**
  - Drop packet before queue overflows
  - Small queues
- **Probabilistic Dropping**
  - Introduces randomization in network
- Early **Congestion Indication**
- Protect **TCP Flows**
  - CBR flows, selfish flows (non-responsive flows)
- **e.g. RED (and variants), REM**

Random Early Detection (RED)

- AQM scheme recommended by IETF
- Proposed by Floyd/Jacobson '93
- Router keeps track of average queue length
  - Exponential weighted (w) moving average (EWMA)
  - Accepts packets if queue lth < Min
  - Min < lth < Max randomly drop packet with linear probability distribution
  - Queue lth > Max drop packet
- Detect congestion early (but not too early …?)
- Four parameters: min, max, w, drop probability



Random Early Drop



avg: average queue length (EWMA)

- if avg < Min_{th} then queue packet
- if avg > Max_{th} then drop packet
- else, probabilistically drop/accept packet.

RED parameters

- Drop probability
  - Too small: won't prevent phase effects
  - Too big: decrease throughput
  - Dynamic value, function of number of connections, bandwidth, RTT
- Min/max
  - Typically max = 3 * min min =5 max = 15
  - Maybe different values for non-responsive traffic (UDP)
- Weight parameter, w
  - avgqlth(t) = (1-w)avgqlth(t-1) + wq(t) (q(t) current queue length)
  - w too small, average doesn't catch up with long range congestion
  - w too big, tracks instantaneous too closely
  - Many studies … 0.002?

RED and ns

- In your ns link commands replace DropTail with RED
  - $ns duplex-link $s3 $r1 10Mb 1ms **RED**
- You can tune various RED parameters

  Queue/RED set q_weight_ 0.002

  Queue/RED set thresh_ 5

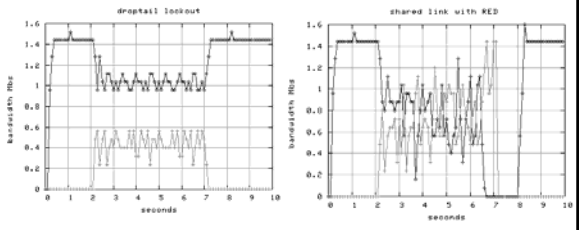
  Queue/RED set maxthresh_ 15

  these need to come before you define the links in your tcl

  ns defaults for these parameters are calculated dynamically

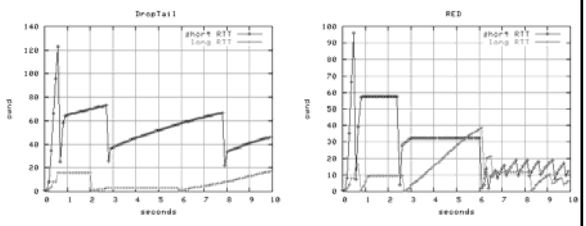
RED and ns

- Chap 11 shared link example with RED
  - No lockout
  - Both flows experience drops



RED and ns

- Chap 11 long vs short RTT (Fack)
  - RED improves RTT fairness



Monitoring queues in ns

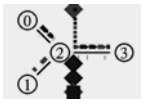
- Setting queue size on path between r1 r2 $ns queue-limit $r1 $r2 $qsize
- Tracing queue variables at specified interval

```
set qmon [$ns monitor-queue $n2 $n3 [open qm.out w] 0.1];
[$ns link $n2 $n3] queue-sample-timeout;
File:  1.3 2 3 1040.0 1.0 5 2 2 4120 1080 2000
    time src dst avrgB  avrgpkts arrivals depart drops barriv bdepart bdrops
```

- Monitoring a queue
  - Snapshot of queue activity with your record or finish procedure
  - Variables **pdrops_ pdepartures_ parrivals_ bdrops_ bdepartures_ barrivals_**

```
set qmon [$ns monitor-queue $n0 $n1 1 2]
set curr_qsize [$qmon set size_]
puts "drops [$qmon set pdrops_]  "
```

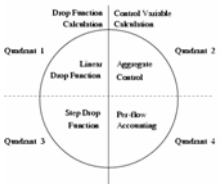
- Monitoring a flow
  - If you need to know which flows are experiencing drops
  - Need to set flow ids $tcp set fid_ 1




```
set fm [$ns makeflowmon Fid]
$ns attach-fmon [$ns link $r1 $r2] $fm 0
foreach f  [$fm flows] {
        puts " flow [$f set flowid_]  drops: [$f set pdrops_]"
}
```


RED variants

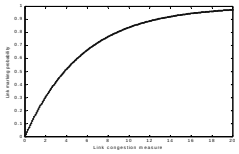
- Variants of RED based on
  - Selection of 4 parameters
  - Calculation of the control variable or drop function
- Try to improve fairness, throughput, and/or reduce delay and variance (jitter)
- Many variants: SRED, DSRED, BLUE, REM, …



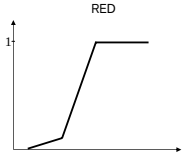
REM (Random Exponential Marking)

- RED variant, marking function exponential not linear
  - Uses aggregate input rate from multiple input links
  - Marking probability a function of link capacity and current buffer fill level

REM



RED



- Five parameters to ensure desired performance
  - ns Queue/REM
- Better throughput than RED, decrease in jitter
- See table 12.3 in text
- No silver bullet ☹

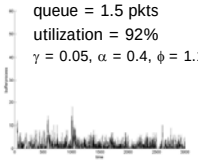
Comparison of AQM Performance

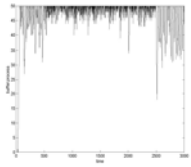
REM

queue = 1.5 pkts

utilization = 92%

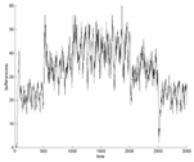
$\gamma = 0.05, \alpha = 0.4, \phi = 1.15$





DropTail

queue = 94%



RED

min_th = 10 pkts

max_th = 40 pkts

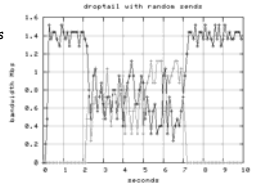
max_p = 0.1

Packet dropping schemes

- **Size-based Schemes**
  - drop decision based on the size of FIFO queue
  - e.g. RED
- **Content-based Schemes**
  - drop decision based on the current content of the FIFO queue
  - Fair queuing, e.g. CHOKe or CSFQ
- **History-based Schemes**
  - keep a history of packet arrivals/drops to guide drop decision
  - e.g. SRED, RED with penalty box

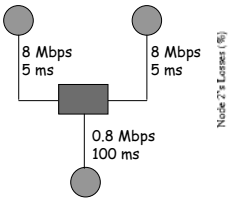
Randomized TCP to reduce phase effects (side bar)

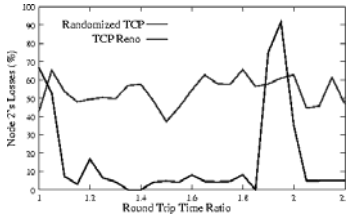
- Phase effects can be reduced by adding random delay at the TCP sender!
- Benefits
  - Breaks synchronization
  - Spreads losses over time
    - Independent losses
  - Removes Phase Effects
  - Removes Bias against large RTT flows
  - Reduces burst losses
- You can experiment with ns
  - Agent/TCP set overhead_ 0.01
  - Chap 11 lockout example →
    - No lockout with random sends
    - Both flows experience drops



- Routing updates can become synchronized leading to packet loss, adding a random offset to their periodic updates helps.

Randomized TCP

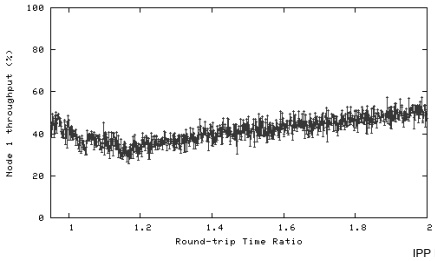




Randomized TCP removes phase effects

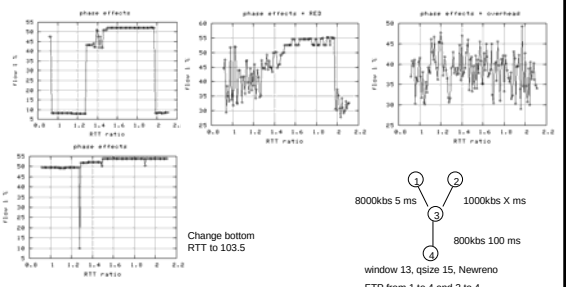
Phase effects

- Mixing in other traffic (telnet, reverse path) reduces phase effects
- Same two-flow example as before, but now with Telnet and reverse path traffic



Phase effects

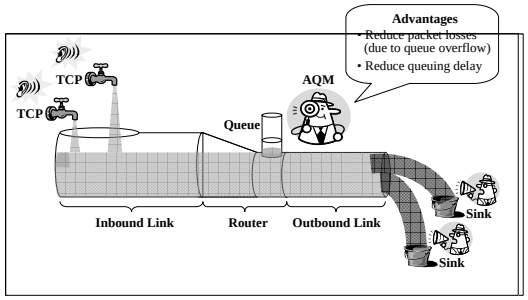
- Series of tests varying RTT of right link (ratio to left link)
- RED and randomization reduce phase effects of DropTail



Active Queue Management (marking)

- Using AQM, e.g., RED, instead of dropping packets in early phase, router "marks" a packet
  - Set a bit in the randomly selected IP packet in the queue indicating congestion is about to occur
  - Receiver copies the bit into the ACK packet so the "mark" gets back to the sender
  - When sender gets notice of "congestion", reduces sending rate
    - Linux treats such a notice (ECN) as if it were a packet loss
    - But potential is there now to distinguish between congestive loss and random loss – e.g. maybe use Westwood if non-congestive loss … alas, you still could get congestive loss
- Early notification may avoid loss, keep queue sizes small
- Mark is in IP header, so other transports (UDP) "could" respond too
- Trouble with marking
  - Packet or ACK could be lost, sender never notified
  - Sender ignores marking and doesn't adjust
  - Requires changes to routers and end-node transport (but incremental)

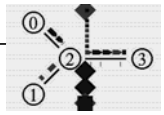
Active Queue Management (marking)



ECN (Explicit Congestion Notification)

- Use explicit notification of congestion (in old days, ICMP Source Quench) rather than implicit (packet loss)
- RFC 3168, uses low-order 2 bits of IP TOS byte in IP header
  - 00 no ECN support
  - 01 or 10 ECN capable
  - 11 ECN notification from router
- TCP uses bits 8 and 9 of the reserved (flags) field to negotiate (ECE) ECN capability and to set ECN mark (CWR)
  - ECE capability negotiated between end-nodes during SYN/SYN-ACK
  - Receiver sees IP ECN and sets CWR in TCP header of ACK
  - Sender reduces flow rate when CWR bit is set in ACK packet
- NOTE: Vegas/Fast try to infer what ECN is explicitly providing

ECN and ns



- In ns with RED queues

  Agent/TCP set old_ecn_ 1

  Agent/TCP set ecn_ 1
- Tables 11.5 and 11.6 in text
  - Pair-wise flows (1 KB MSS)
  - RED improves aggregate throughput, fairness, and reduces drops compared to DropTail
  - RED + ECN reduces drops even more, further improves fairness
  - Notice no drops for Vegas, so RED/ECN doesn't really help Vegas since Vegas is already trying to eliminate drops

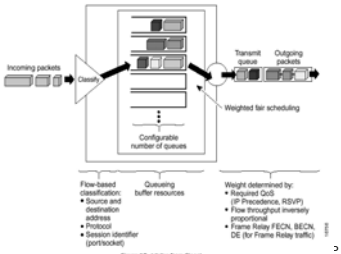
	DropTail	RED		RED+ECN	
Flavor	Goodput Kbs	Goodput	drops	Goodput	drops
Reno/Reno	642/570	640/571	48/38	653/648	42/32
Tahoe/Reno	1051/379	809/608	49/38	803/632	42/32
Vegas/Reno	380/1059	453/987	0/68	452/988	0/63
Newreno/Reno	931/394	798/524	50/38	684/640	42/32
Sack1/Reno	1122/329	902/548	50/38	858/592	42/32
Fack/Reno	1273/191	956/507	50/39	851/612	42/32

AQM and ECN

- Some router support for AQM/RED/ECN, but often not enabled
  - Would be nice if all routers in path were RED/ECN capable
  - But still can "work" with just some routers and some end-nodes
- Some TCP support for ECN (linux), disabled
  - sysctl net.ipv4.tcp_ecn = 0
  - Treats ECN notification as packet loss
- ORNL external internet traffic
  - 0. 02 % IP packets with ECN enabled (161 per million capable, 2 per million marked)
- Future internet routers and hosts may do more AQM
  - But proper settings, e.g, for RED, still a mystery
  - Potential for distinguishing congestive vs non-congestive loss and using different recovery functions (e.g. TCP Westwood)
- Problems with non-responsive transports (UDP)
- More complex considerations include
  - QoS, quality of service (charging ($$) for better service?)
  - Differentiated services

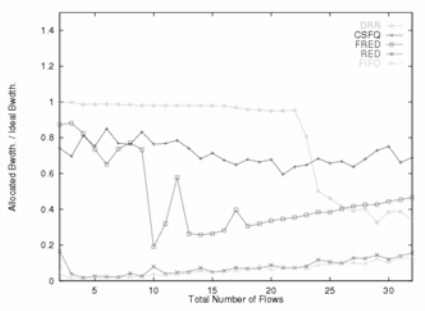
AQM for non-responsive flows

- Routers need to distinguish flow types (classifier) and then schedule based on some priority (IP precedence/qos field,UDP vs TCP, …?)
  - Trick is to do this efficiently and fairly
  - BRED, CBT-RED, FRED, DRR, CSFQ, WFQ
- Cisco uses WFQ for line speeds < 2 mbs, otherwise FIFO/DropTail



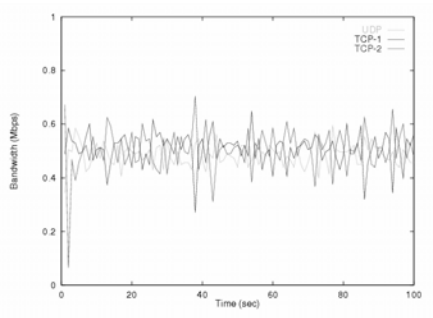
CSFQ/FRED/RED/DRR/DropTail

- One TCP flow vs N aggressive UDP flows (each sending at twice its "fair share")



CSFQ

- Two TCP (SACK) flows competing with 1 mbs UDP flow on 1.5 mbs link



XCP

- Provide more than a bit of congestion information
  - Addition of a congestion header to IP

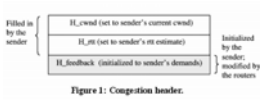


Figure 1: Congestion header.

- Changes routers and end-nodes
  - Simple arithmetic on routers, no per-flow accounting
    - Decouple congestion control from fairness
  - Modification to transport protocol to you use rate feedback (+ or -)
    cwnd ← cwnd + H_feedback
- Simulation results very promising

Slides from Dina Katabi MIT

Active queue management summary

- Routers cause some of the problems with TCP
  - FIFO/DropTail cause phase effects, lockout, unfair
- Routers should be part of solution – active queue management
  - Simple strategies: RED can help
  - For non-responsive flows, classifying flows or per-flow accounting is required
    - Added complexity
    - May not scale
  - Performance metrics:
    - Throughput
    - Delay
    - Fairness
- End-to-end modifications offer better performance
  - Changes to network layer (routers) and transport layer (TCP)
  - marking
  - ECN
  - XCP

Next time ...

- Parallel streams
- Rate-based UDP