2/11/2010

Queueing : FQ

Ideally,

Role of routers in congestion control : many, but here are a few.

- Actively monitor onset of congestion
- Provide early indication of congestion
- Isolate misbehaving / or unresponsive flows so that end-host congestion control is effective.

"Active Queue Management (AQM)"

→ FQ → does third really well.

→ Others that do subsets of these → discuss in the next class.

How to isolate : ideal → max-min fairness ; efficient use of resour
→ also buffer and delay allocation.

[~~FQ Goal : allocate bandwidth, buffer space and delay in a fair fashion across multiple end to end flows.~~] ~~at~~

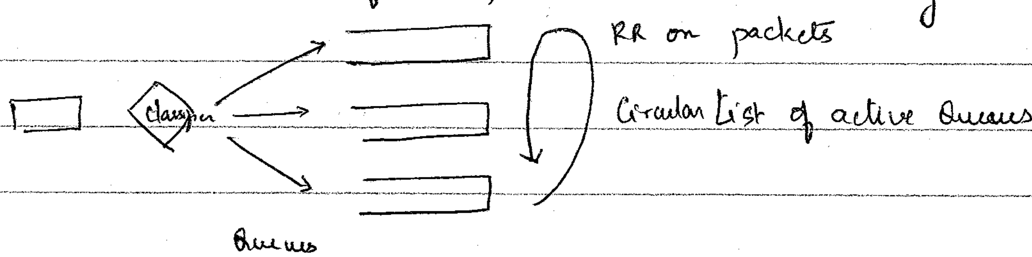
what we have today → ① - FCFS : ~~intertwines all three goals and messes up the outcome above~~ ineffective at all three

② - Router backplane switching does this using starvation-free fair scheduling. Simple, efficient, but...

- Too granular : focuses on input ports as opposed to flows. parking lot problem where flows that traverse multiple routers get exponentially lesser share.

Question : → How to build a fair mechanism that is easy to implement / use?

Nagle's idea : Instead of links, allocate bandwidth by isolating flows.



Bad — ① # Queues = # flows ever seen (provision many queues for worst case)

Bad — ② Unfairness = relative proportions of n/w shares received

$= \frac{\text{Max Pkt}}{\text{Min Pkt}} \rightarrow \infty$

Good — ③ Work complexity = O(1) ~~expected~~ O(1) enqueuing, O(1) dequeuing

FQ proposal of DKS:

- fixes the unfairness problems
- Buffer allocation and incentives / disincentives
- But somewhat complex to implement
- Other salient features.

- Simulate bit-by-bit RR.

Upon packet arrival, calculate finish time (→ ~~not just round number~~) and insert into a sorted Queue based on finish times.

- ~~per flow state~~

~~Finish time = size + max(current time, finish time of prev pack in flow)~~

Finish Round = size in bits + max(current round - δ, finish round of prev. pkt in flow)

$\frac{d\,R(t)}{dt}$ → can change over time as the number of active flows grows and shrinks.

also R(t) is fractional. → packets arriving almost at the same time will see different R(t)

① Insert O(log m) → m is the number of packets in the queue.

② For each flow, track finish time of last packet in flow.
O(num active flows)

FQ Buffer management: Buffer has a certain maximum length.

What to do when buffer is full.

~~Two options~~.
"Buffer stealing" ← remove packet from flow with maximum byte occupancy → O(logn) if done naively; can be done in O(1) time.

But don't update

- Email PC.

Delay allocation with a δ

Salient properties:
1. Fair
2. Low delay for telnet flows
3. Good congestion control still important to make good efficient use of n/w.
4. Incentives to adopt good congestion control
5. Punishment to misbehaving flows.

Can we achieve the same fairness as FQ but at $O(1)$ cost per packet, and lower space requirement, and low amount of state tracking.

RR of Nagle $\rightarrow$ $O(1)$ but unfairness.

idea: RR + deficit

Additional variables

Quantum: Q

Deficit per-Queue: D_i

~~At round of i, send $Q_i + DC_i$ bytes from queue i~~

Round Robin: when servicing Queue i

check if size of packet at head, $H_i \leq Q + D_i$

yes $\rightarrow$ send $B_i = |H_i|$ bytes

No $\rightarrow$ Send $B_i = 0$

$D_i \leftarrow B_i + D_i$

Thm: Difference between FQ and DR.

FQ $\rightarrow$ maximum fine-grained difference across flows MAX

DRR $\rightarrow$ " " = 3MAX

~~Another difference~~

Complexity: Enqueue: finding the queue, append to tail of Queue

Dequeue: $Q \geq Max \rightarrow$ guaranteed to send on packet every ~~round~~ time a queue is visited

Next active queue, update counters, Active list.

Unfairness : 3MAX (N-1) MAX delay.

Problem : queues are way too many

How to bound the number of Queues : SFQ.

if the Hash index is larger than the number of active flows at any given point then fairness guarantees become static.

From the last lecture :

- Buffer sizing
- TCP throughput
- TFRC

QOS