Nagle's - unfair; but O(1) enque / deque; too many queues

Fair - Queueing - fair; but O(log n) operations per packet

~~DRR~~ └ per-flow state

DRR - fair; O(1) computations per packet; too many queues.

↓

Too many queues problem can be fixed using SFQ.

DRR + SFQ:

Fix number of queues ⟶ hash buckets.

Hash flows to queues

As long as # buckets >> active flows ⟶ no collisions. (low prob.)

Fairness properties of DRR hold but in a stochastic sense.

DRR + SFQ used at n/w edges today.

Challenge: can we think of a techniques that combines the good aspects and overcomes all the bad aspects?

Bad
- per-flow queues
- per-flow state
- Expensive computation per packet
- Expensive buffer management.

CSFQ: A heuristic approach that answers these challenges.

- Per-flow state at edges
- No state or queueing any place else
- long term fairness guarantees.

~~Each route~~

Max min fair allocation example

2, 3, 6

$x < 2$

$3x$

$x \geq 2$

$2 + 2x$

if $A > C$

FSR is unique solution to

$$C = \sum \min(r_i, FSR)$$

Else

$$FSR = \max(r_i)$$

If $r_i < FSR$ → all packets let through

else only FSR let through. and $\left(1 - \frac{FSR}{r_i}\right)$ dropped.

Can implement fair bandwidth allocation if FSR and r_i are known (r_i → for each flow) at each router.

CSFQ contribution: estimation of these in a light weight-fashion.

Edge router:

(i) r_i → EWMA.

(ii) assuming FSR is known

~~$traffic_i = r_i\left(1 - \max\left(0, 1 - \frac{FSR}{r_i}\right)\right)$~~

if $r_i < FSR$

$traffic_i = r_i$

else

$traffic_i = FSR$.

→ Thus, we have r_i's for downstream router ~~and they don't need to be~~ re-estimated.

How to estimate FSR. $\widehat{FSR}$ — running estimate of α.

↓ drives dropping behavior.

Net traffic leaving a queue → depends on FSR

→ piece-wise linear f...

if link is congested then FSR is unique solution
to $\sum traffic_i = C. = \sum max(r_i, \hat{FSR})$

else FSR = max (all labels in packet).

If r_i's are known then easy to solve and obtain C.
└ edge router case is solved

How to do it without per-flow state. at core routers.

Track arrival and total traffic rates. using EWMA.

if congested → if α is correct then traffic rate should match C.

if not α is too aggressive → increase α.
α too lax → decrease α.
→ linear interpolations help
converge to optimal α

- Evaluation criticism. { complexity ⇒ mechanism can be gamed
- benefits are there but it is not clear what is leading to them
- choice of parameters is not clear ⇒ impacts stability and effectiveness

- Why is fair allocation important?

- to deal with unfriendly flows < allocation or isolation / detection or identification (much harder.)

allocation → incentives for drop intolerant flows to adopt congestion control.

drop-tolerant flows: punishment
↳ identification and punishment may be necessary.

but this is easier in the allocation model.
↳ just look for significant drop rate.