pFabric: Minimal Near-Optimal Datacenter Transport

Problem

Provide little or no queuing for short, deadline-sensitive flows

Fully utilize the network for long flows

Goals:

(1) Provide little or no queuing for short, deadline-sensitive flows

(2) Fully utilize the network for long flows

Approaches

- Keep switch queues really short (1-2 packets)
  - E.g., DCTCP
  - Pro: little queuing for short flows
  - Con: hard to keep network well utilized (need pacing etc.)

- Assign explicit rates, pause large flows for small ones
  - E.g., D3 (lacked pause), PDQ
  - Pro: works well when configured correctly
  - Con: difficult to configure, requires lots of new switch func.

pFabric

Separate *rate control* from *flow scheduling*

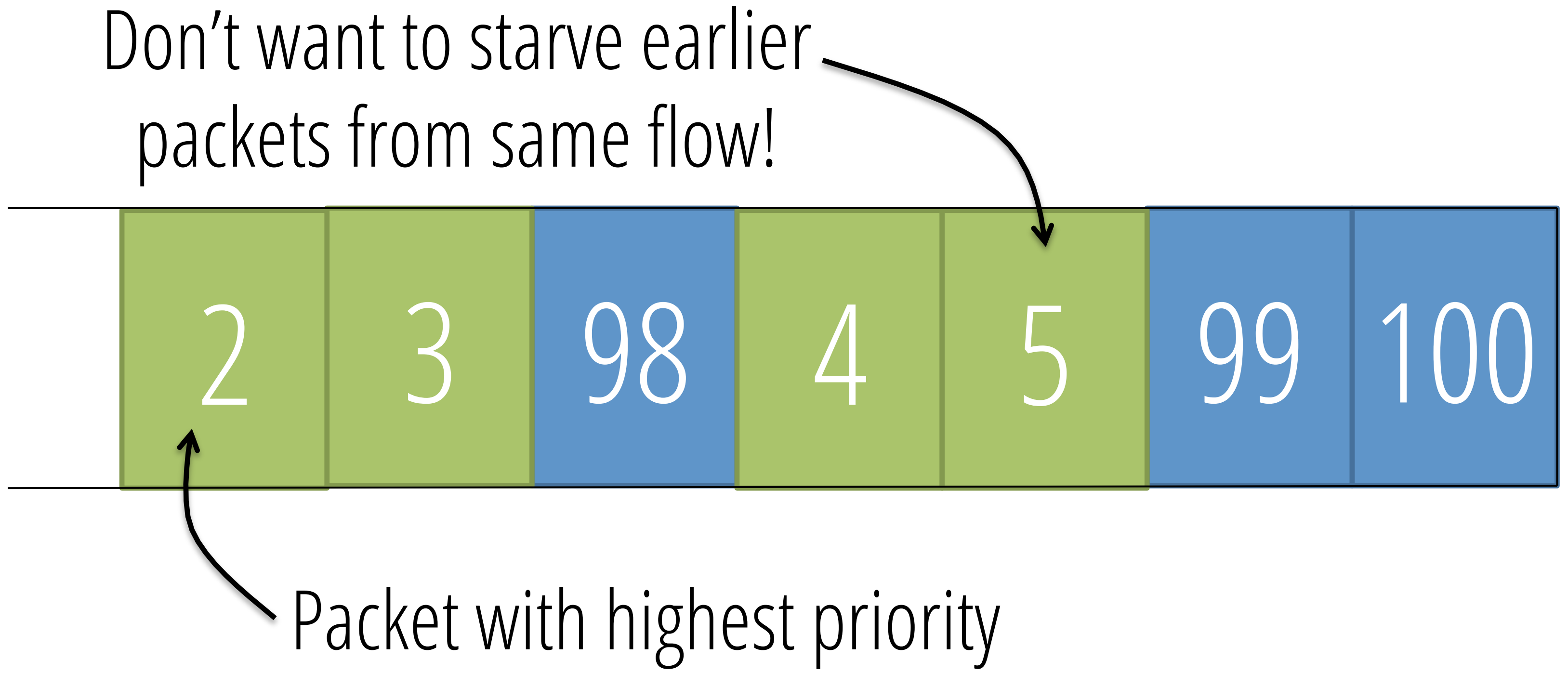
Really: leverage flow scheduling

To make short flows finish fast: prioritize them at switches

(end-host assigns priority: # remaining packets)

To fully utilize network (but not have congestion collapse):
flows start at line-rate, throttle if they experience loss

How do switches send packets?


Don’t want to starve earlier packets from same flow!
2
3
98
4
5
99
100
Packet with highest priority

Send earliest packet from flow with highest priority packet

pFabric Benefits

Near-ideal flow completion times in all cases

Pretty high utilization

- Really care about bursts, pFabric does well in this case

MAYBE implementable in real switches

- Amin: why this isn’t used at the Goog

pFabric Issues

Large flows can still starve

Need to determine remaining flow size

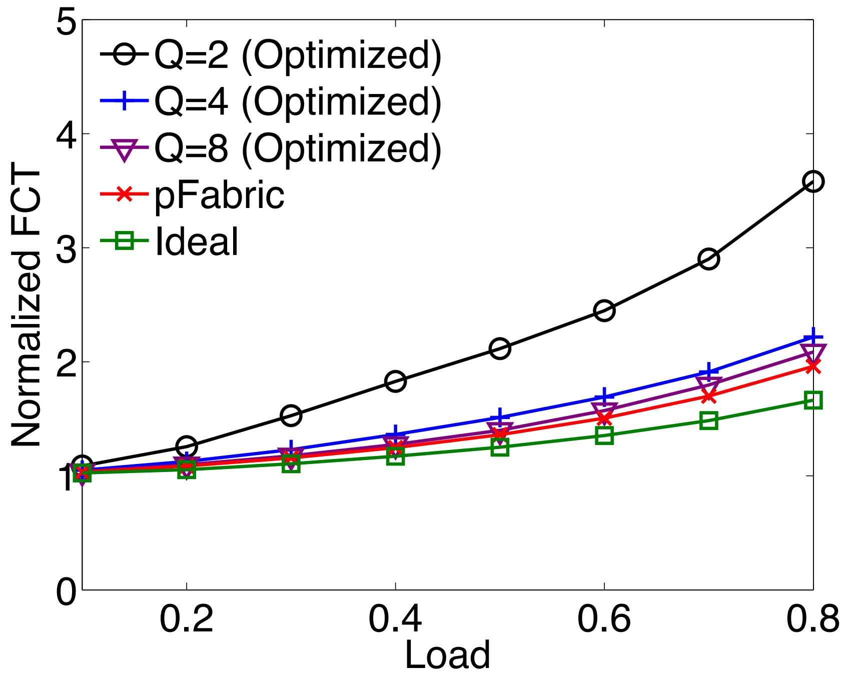
No isolation between tenants

Maybe FCT of background flows doesn’t matter

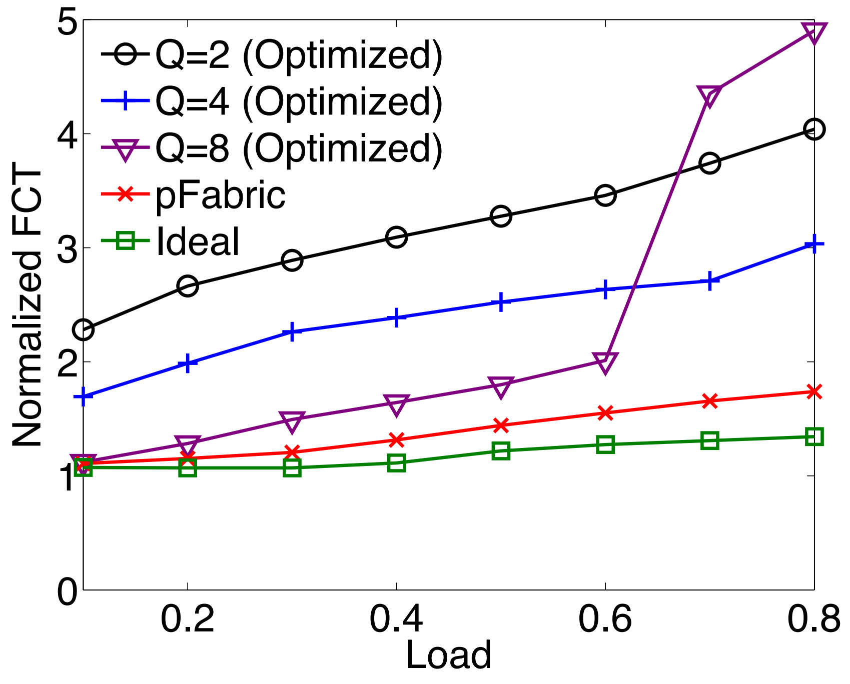
- Only use pFabric for short flows where remaining size known?

Could we make this simpler?

What about just two priority levels?



(a) Overall: Avg



(b) (0, 100KB]: 99th prctile