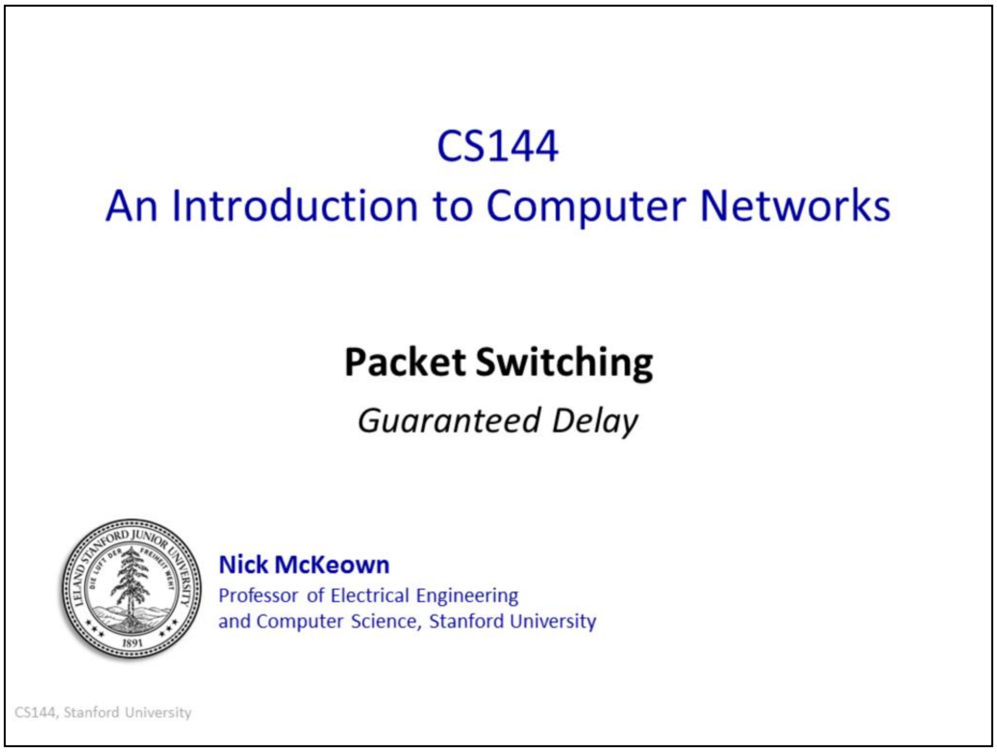
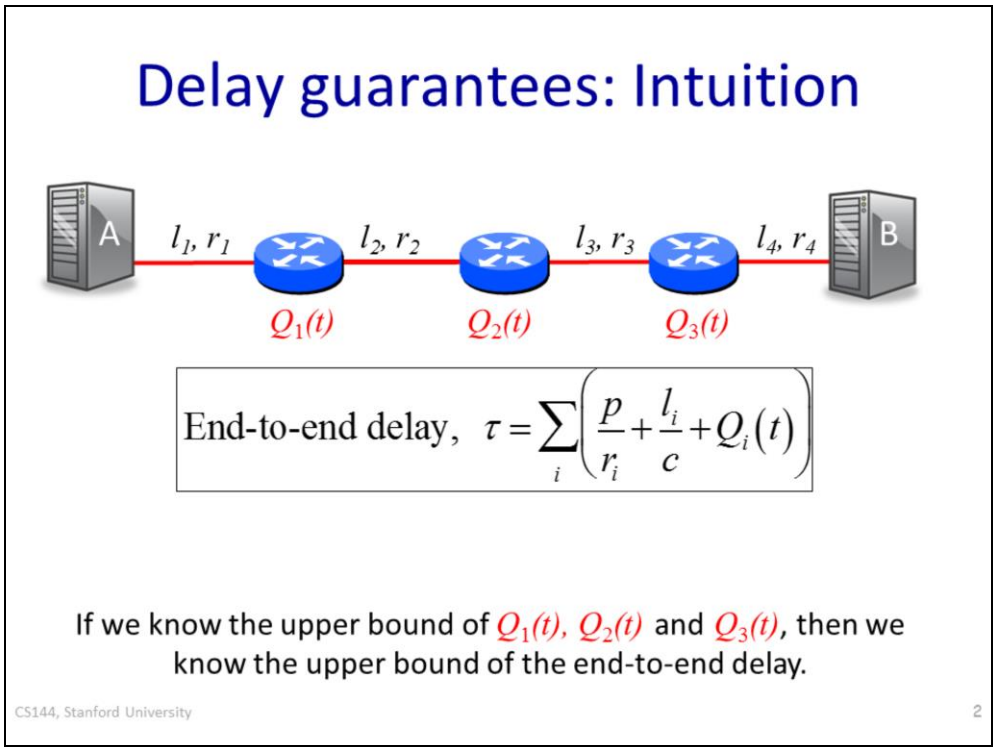


CS144
An Introduction to Computer Networks

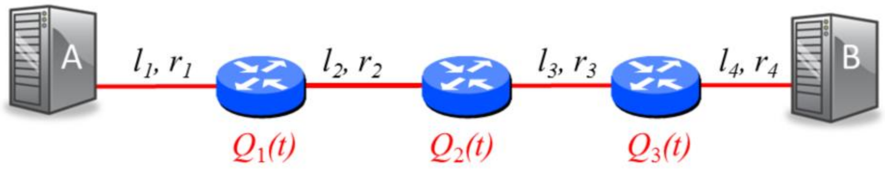
Packet Switching

Guaranteed Delay

Nick McKeown
Professor of Electrical Engineering
and Computer Science, Stanford University



Delay guarantees: Intuition

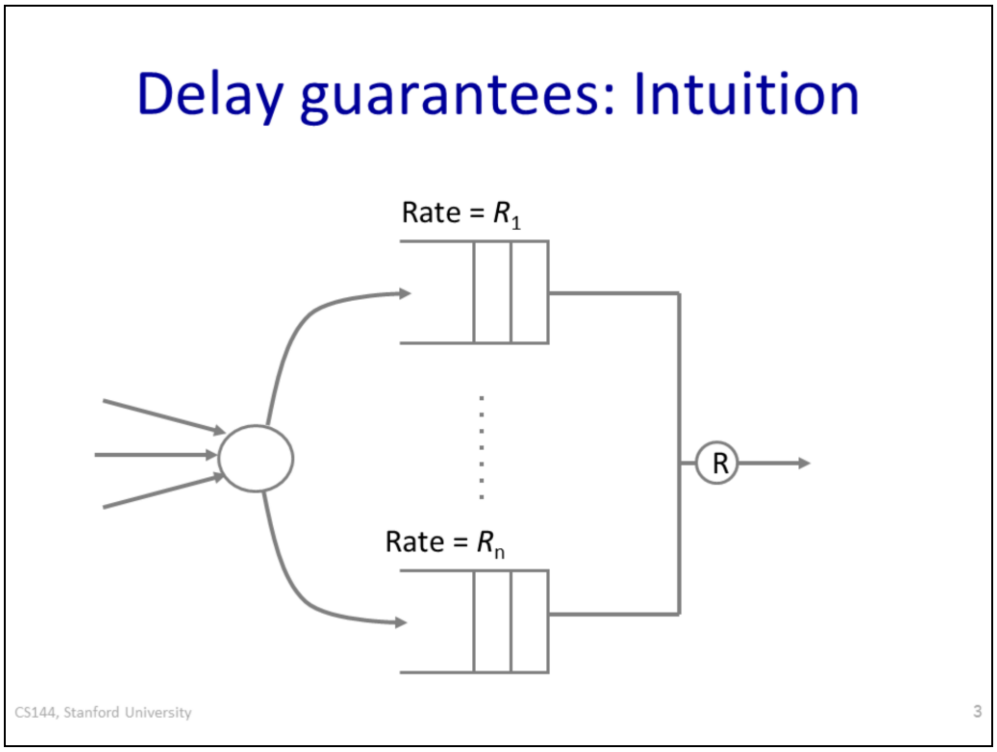


End-to-end delay, $\tau = \sum_i \left(\frac{p}{r_i} + \frac{l_i}{c} + Q_i(t) \right)$

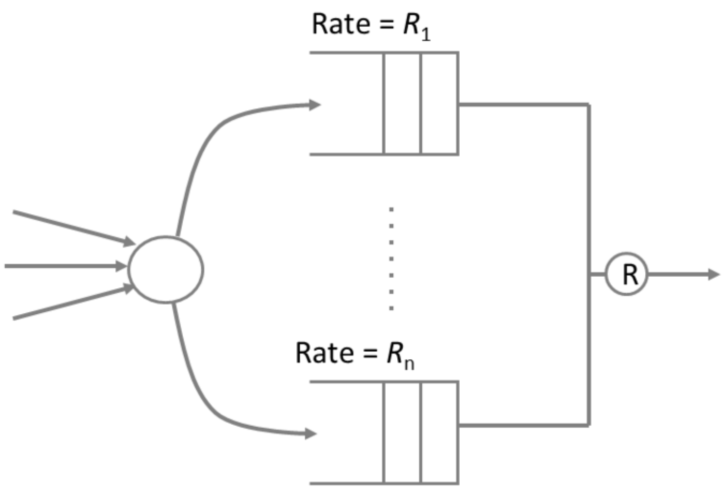
If we know the upper bound of $Q_1(t)$, $Q_2(t)$ and $Q_3(t)$, then we know the upper bound of the end-to-end delay.

CS144, Stanford University 2

If we kn

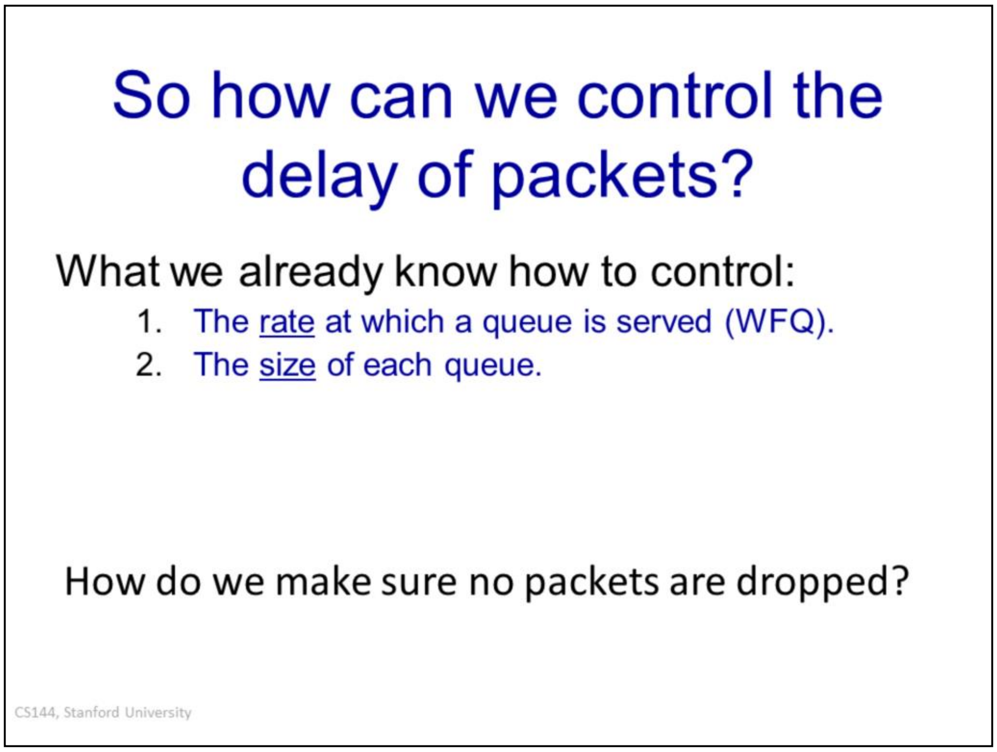


Delay guarantees: Intuition



If – in a router – I know which queue packets pass through, the size of the buffer and the rate at which it is served, then I know the maximum delay a packet can encounter.

WFQ gives me a lower bound on delay. DRAW expression again: r_i = w_i/Sum(w_i) * R. DRAW size of buffer, B.



So how can we control the delay of packets?

What we already know how to control:

1. The rate at which a queue is served (WFQ).
2. The size of each queue.

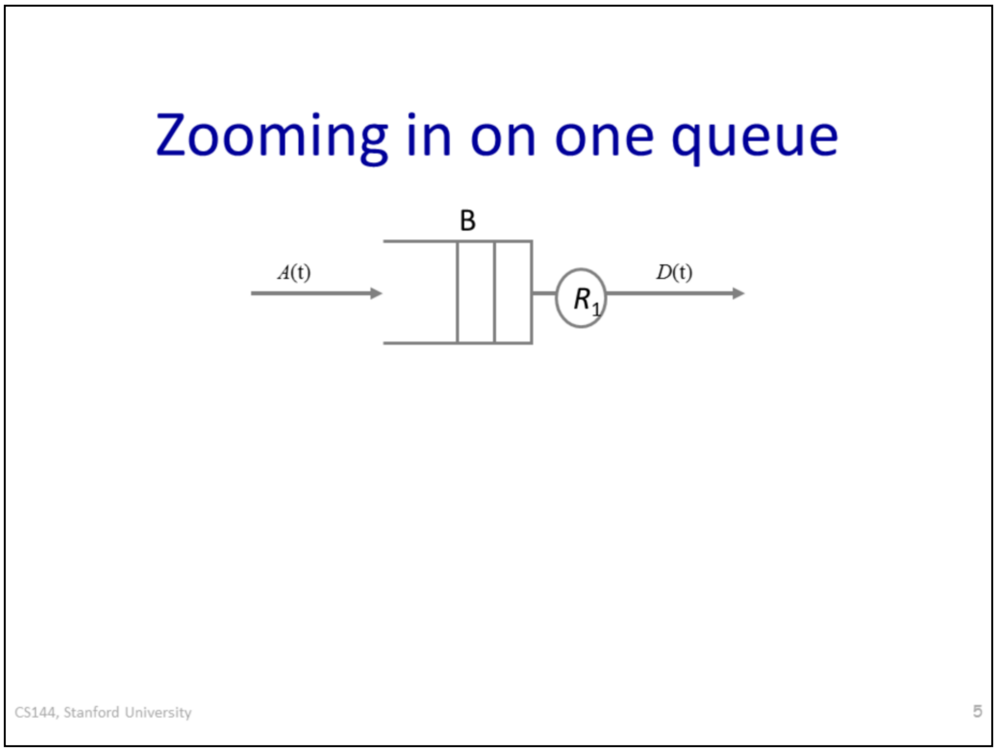
How do we make sure no packets are dropped?

CS144, Stanford University

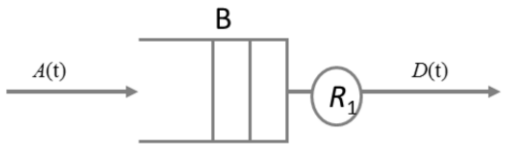
DRAW: This suggests a model for a router where we CLASSIFY incoming packets to put them in the right queue, set the queue size, and then serve the queue at the correct rate.

Any packet arriving to the router will have a bounded delay. If we add up all the components of delay correctly, then we can make it work end to end according to our equation.

This works for packets that make it all the way through. What if packets arrive too fast and they are dropped?



Zooming in on one queue



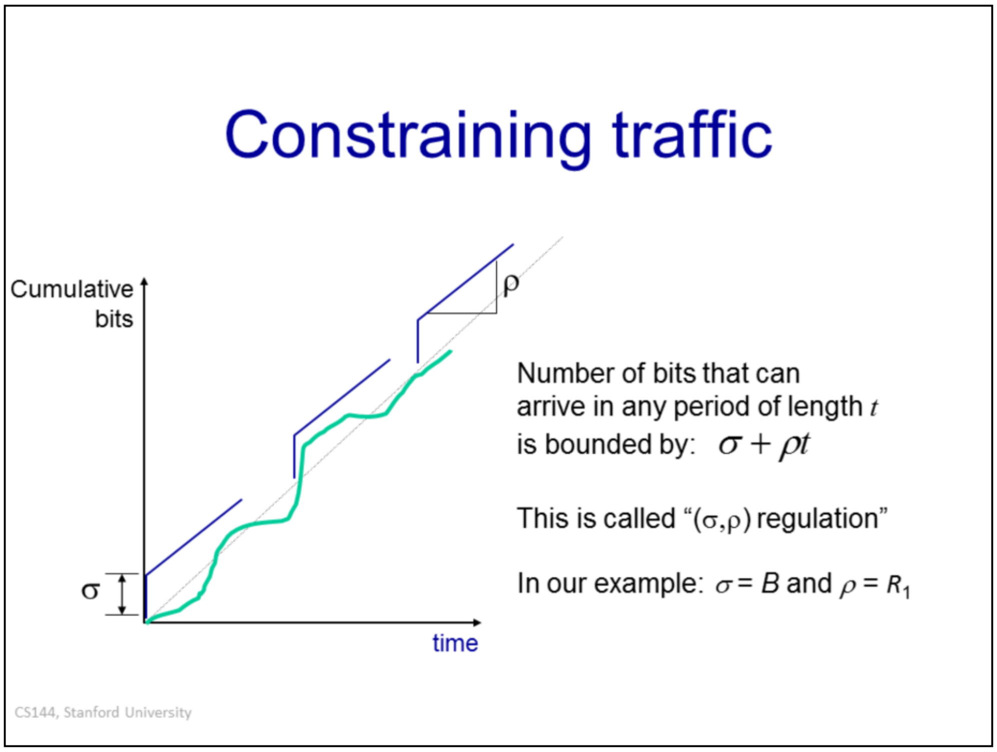
SKETCH: A(t) and D(t) from next slide. Remind us of the d(t) – the horizontal distance between D(t) and A(t) (it's a FIFO). D(t) is nicely bounded, but if the arrival process is too large, Q(t) will grow and the packets will be DROPPED. Not very useful.

What if we could somehow control A(t) to make sure this doesn't happen? This is what we're going to look at next. If we can solve this, then we can provide the delay guarantee through the router.

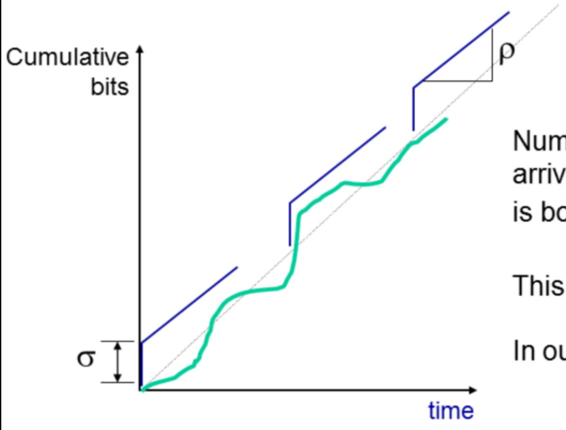
SKETCH: What if we could say that in any interval T, no more than B + r_1*T bits could arrive to the queue? In other words, A(t + T) – A(t) <= B + r_1*T.

Advantages of this approach:

* Queue never overflows, so we know the delay is guaranteed.
* We've given quite a lot of leeway to the arrival process A(t).



Constraining traffic

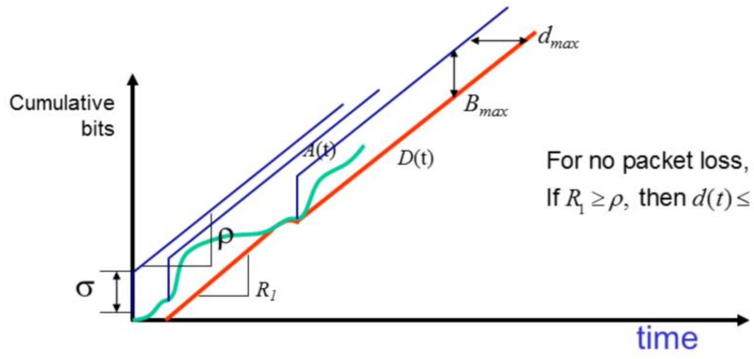


Number of bits that can arrive in any period of length t is bounded by: $\sigma + \rho t$

This is called "(σ,ρ) regulation"

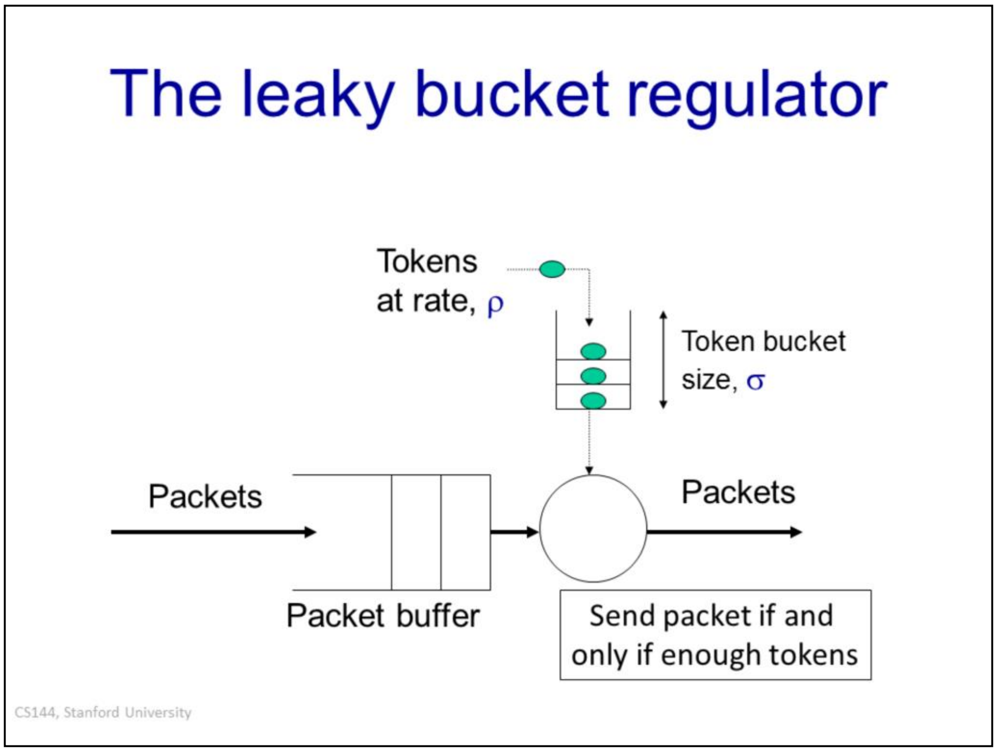
In our example: $\sigma = B$ and $\rho = R_1$

(σ,ρ)-constrained Arrivals and Minimum Service Rate

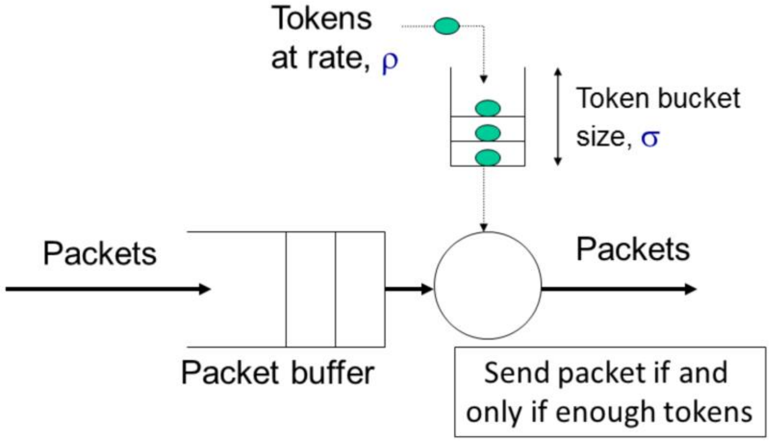


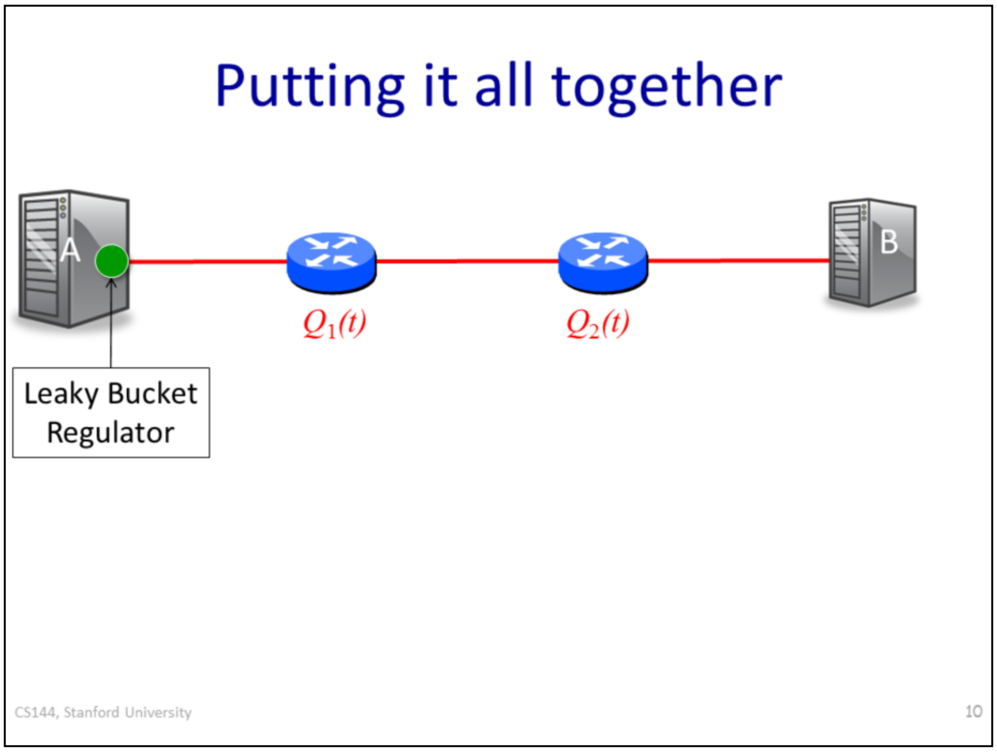
For no packet loss, $B \geq \sigma$.
If $R_1 \geq \rho$, then $d(t) \leq B / R_1$.

If flows are leaky-bucket constrained, and routers use WFQ, then end-to-end delay guarantees are possible.



The leaky bucket regulator





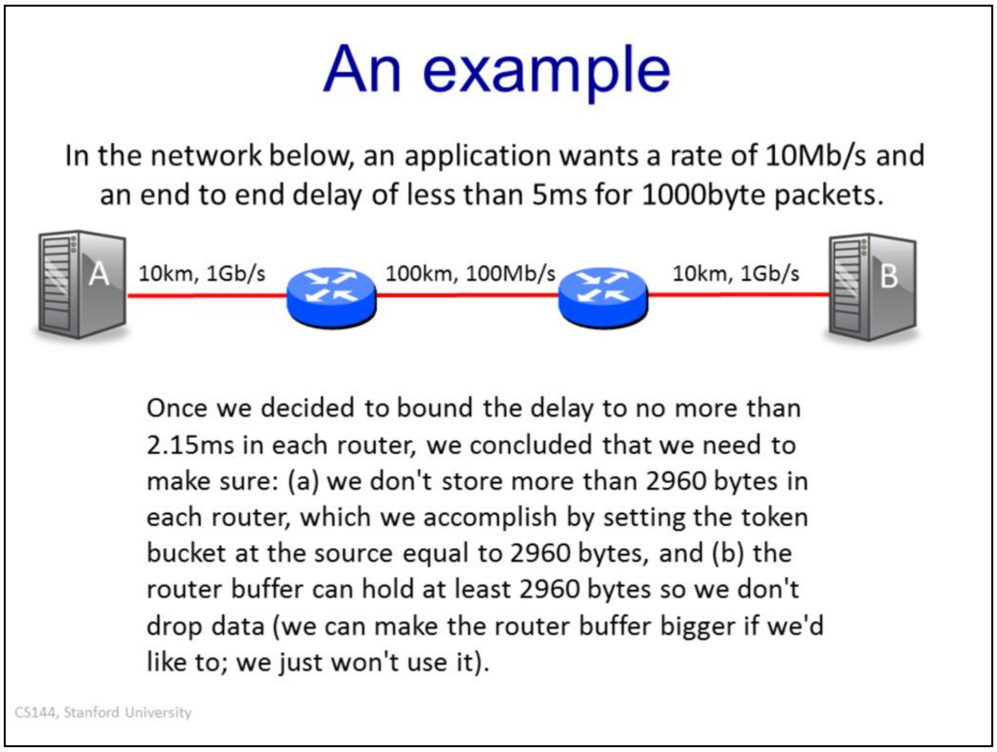
Putting it all together

SKETCH: sigma, rho constrained traffic from A.

Each router runs WFQ with pre-determined rates and buffer sizes. SHOW classification into right queue and its service rate.

Packets goe into router Q_1 and is served at R_1. We just require sigma <= B and rho <= R_1. Similarly for second router… and so on. Then using our equation for end-to-end delay we can calculate the entire delay along the path.

You may be wondering how the values for sigma, rho, ,R_1, B and so on get told to the router and who sets up the end to end delay in the first place. There is an IETF protocol called RSVP (IETF RFC 2205) for doing this. Good descripton in textbooks and on Wikipedia.



An example

In the network below, an application wants a rate of 10Mb/s and an end to end delay of less than 5ms for 1000byte packets.

A
10km, 1Gb/s
100km, 100Mb/s
10km, 1Gb/s
B

Once we decided to bound the delay to no more than 2.15ms in each router, we concluded that we need to make sure: (a) we don't store more than 2960 bytes in each router, which we accomplish by setting the token bucket at the source equal to 2960 bytes, and (b) the router buffer can hold at least 2960 bytes so we don't drop data (we can make the router buffer bigger if we'd like to; we just won't use it).

CS144, Stanford University

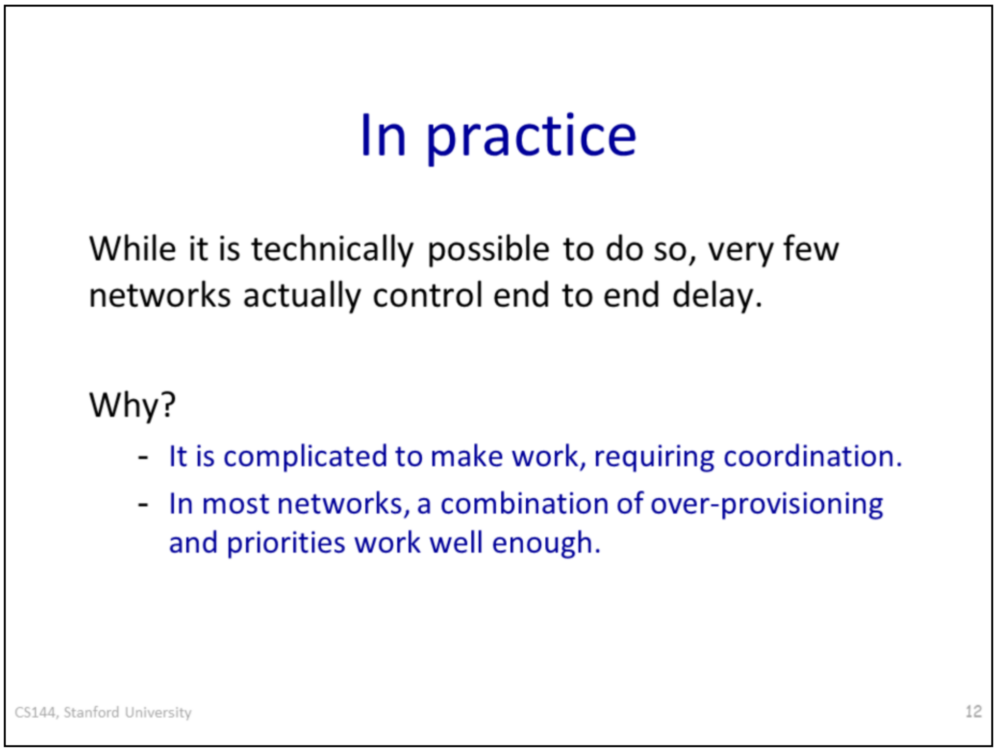
SOLLUTION:

Fixed delay is: Packetization delay + propagaton delay = (1000*8 / 10^9 + 1000*8 / 10^8 + 1000*8 / 10^9) + (120 * 10^3 / 2x10^8) = 0.696ms

Therefore queueing delay needs to be less than 5-.696 = 4.304ms.

Let's CHOOSE to split the delay equally among the two routers: i.e. 2.152ms at each. Therefore, B > 10Mb/s * 2.152ms = 21520 bits = 2690 bytes.

A needs to send with leaky bucket parameters 21520 bits and 10Mb/s.

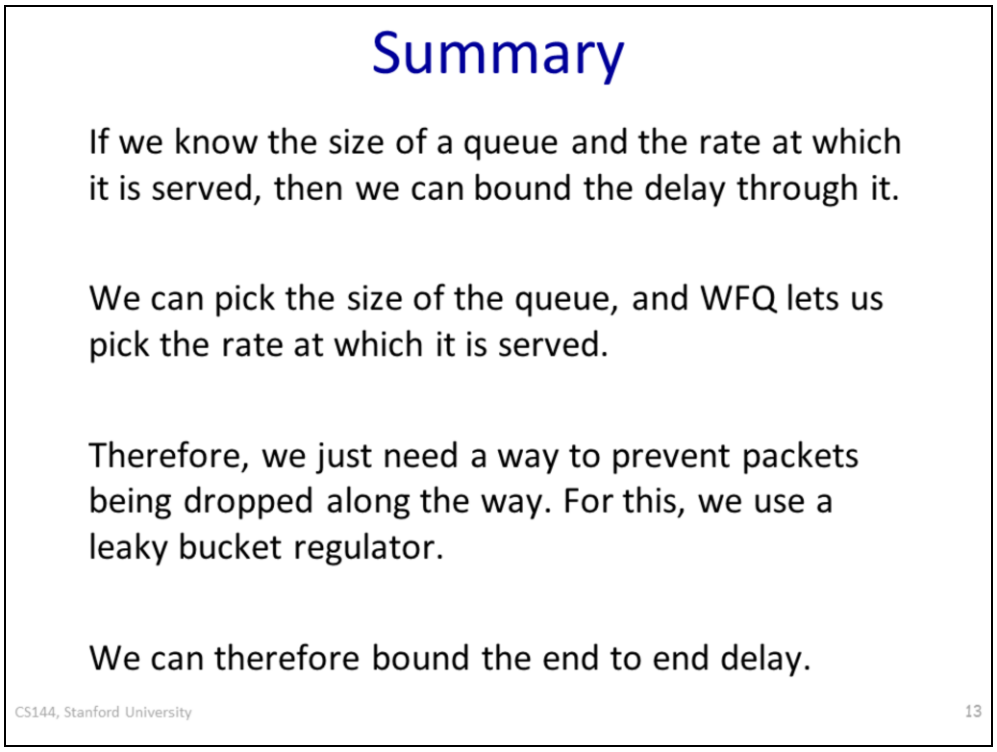


In practice

While it is technically possible to do so, very few networks actually control end to end delay.

Why?

- It is complicated to make work, requiring coordination.
- In most networks, a combination of over-provisioning and priorities work well enough.



Summary

If we know the size of a queue and the rate at which it is served, then we can bound the delay through it.

We can pick the size of the queue, and WFQ lets us pick the rate at which it is served.

Therefore, we just need a way to prevent packets being dropped along the way. For this, we use a leaky bucket regulator.

We can therefore bound the end to end delay.