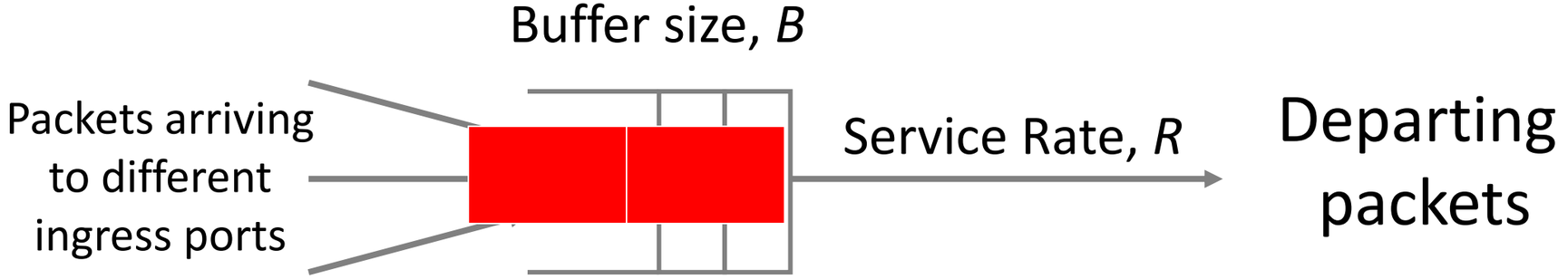
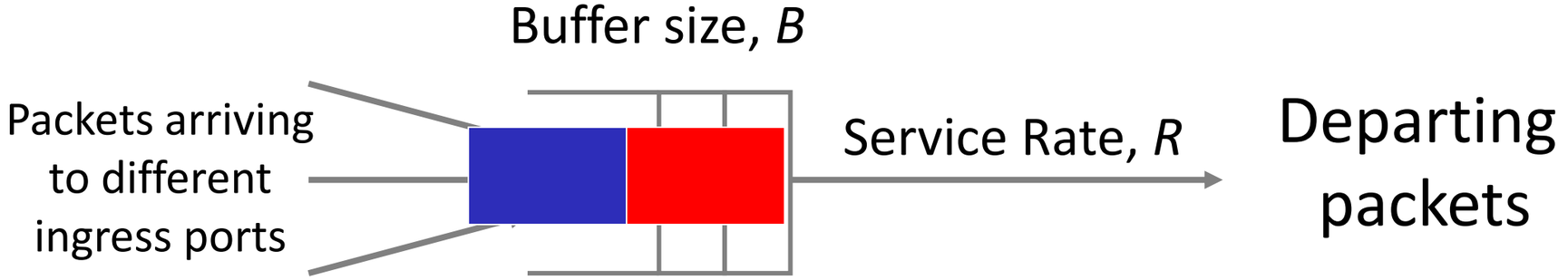
Packet Switching

What if some packets are more important than others?

By default, switches and routers use FIFO (*aka* FCFS) queues



By default, switches and routers use FIFO (*aka* FCFS) queues



Some packets are more important

For example:

1. Control packets that keeps the network working
   (e.g. packets carrying routing table updates)
2. Traffic from a particular user (e.g. a customer paying more)
3. Traffic belonging to an application (e.g. Zoom)
4. Traffic to/from specific IP addresses (e.g. emergency services)
5. Traffic that is time sensitive (e.g. clock updates)

Flows

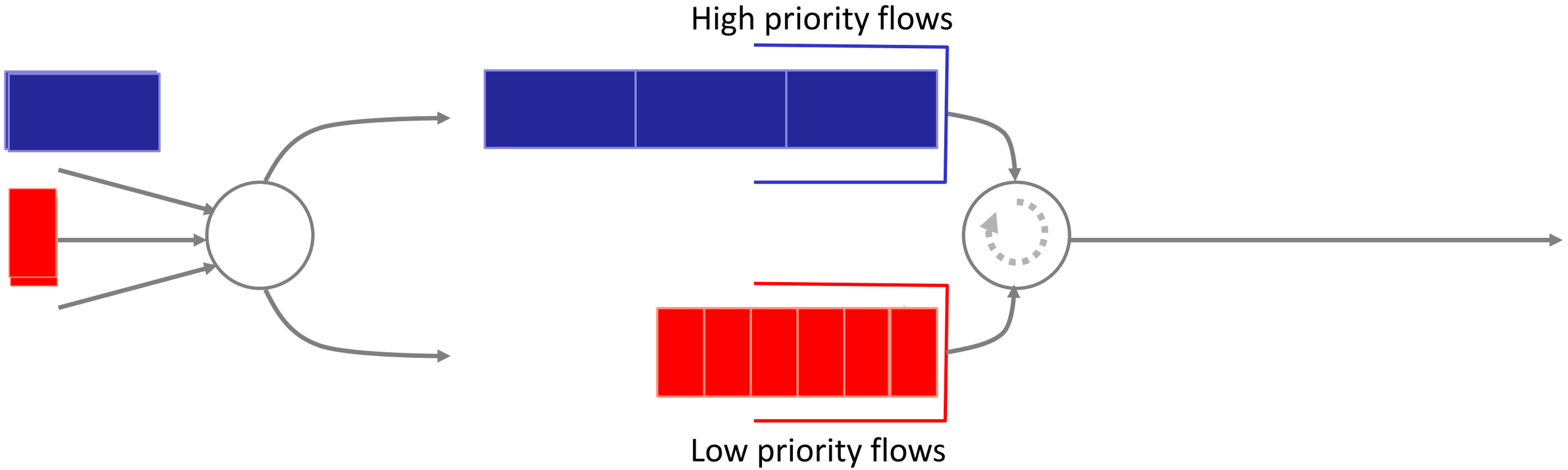
When talking about priorities, it's convenient to talk about a "flow" of packets that all share a common set of attributes. For example:

1. The flow of packets all belonging to the same TCP connection
   Identified by the tuple: TCP port numbers, IP addresses, TCP protocol
2. The flow of packets all destined to Stanford
   Identified by a destination IP address belonging to prefix 171.64/16
3. The flow of packets all coming from Google
   Identified by a source IP address belonging to the set of prefixes Google owns.
4. The flow of web packets using the http protocol
   Identified by packets with TCP port number = 80
5. The flow of packets belonging to gold-service customers
   Typically identified by marking the IP TOS (type of service) field

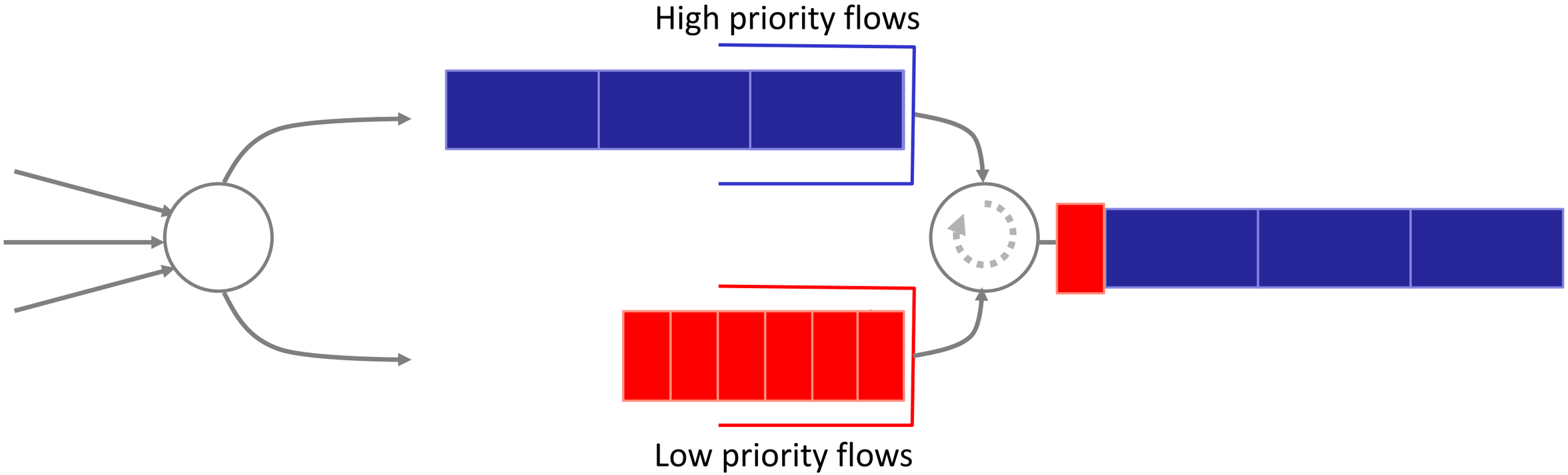
Outline of what's coming up next…

1. How to give "strict priority" to some flows
2. How to give "weighted priorities" to some flows
3. How to give "rate guarantees" to some flows
4. How to guarantee the end-to-end latency of a packet across a network

Strict Priorities



Strict Priorities

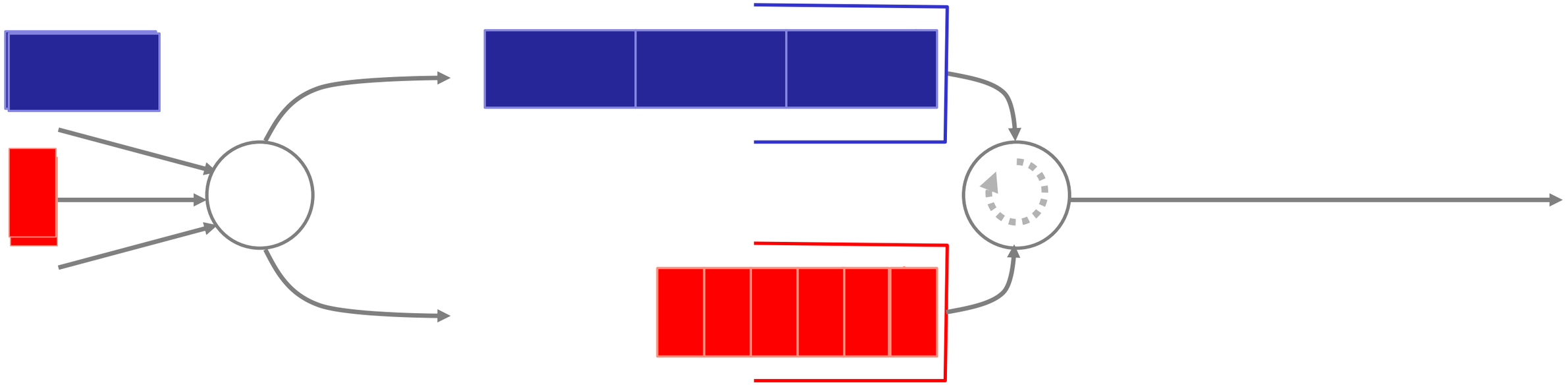


"Strict priorities" means a queue is only served when all the higher priority queues are empty

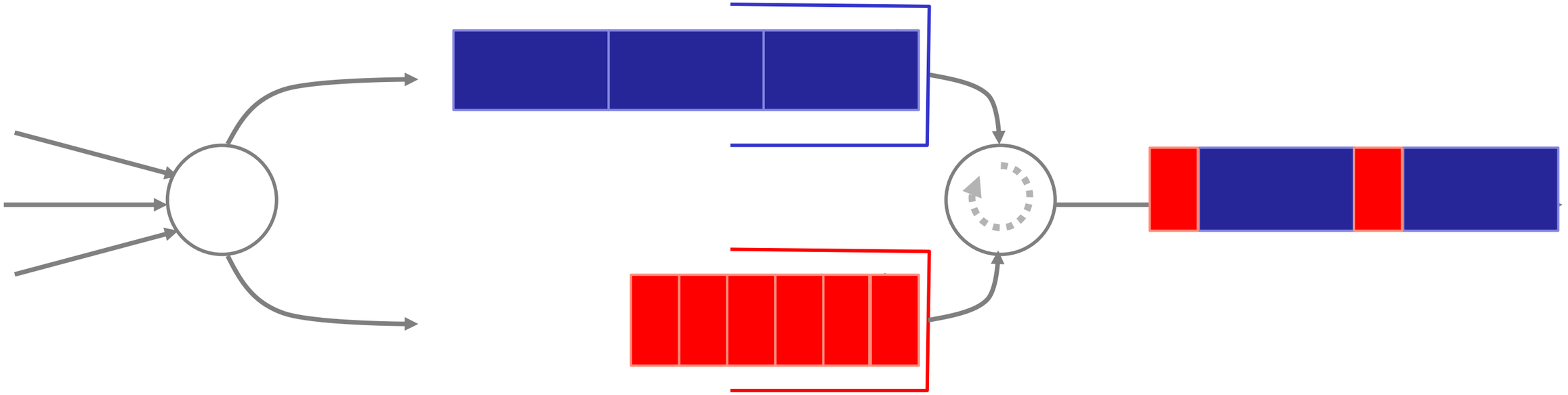
Strict Priorities: Things to bear in mind

1. Strict priorities can be used with any number of queues.
2. Strict priorities means a queue is only served when all the higher priority queues are empty.
3. Highest priority flows “see” a network with no lower priority traffic.
4. Higher priority flows can permanently block lower priority flows. Try to limit the amount of high priority traffic.
5. Not likely to work well if you can’t control the amount of high priority traffic.
6. Or if you really want *weighted* (instead of strict) priority.

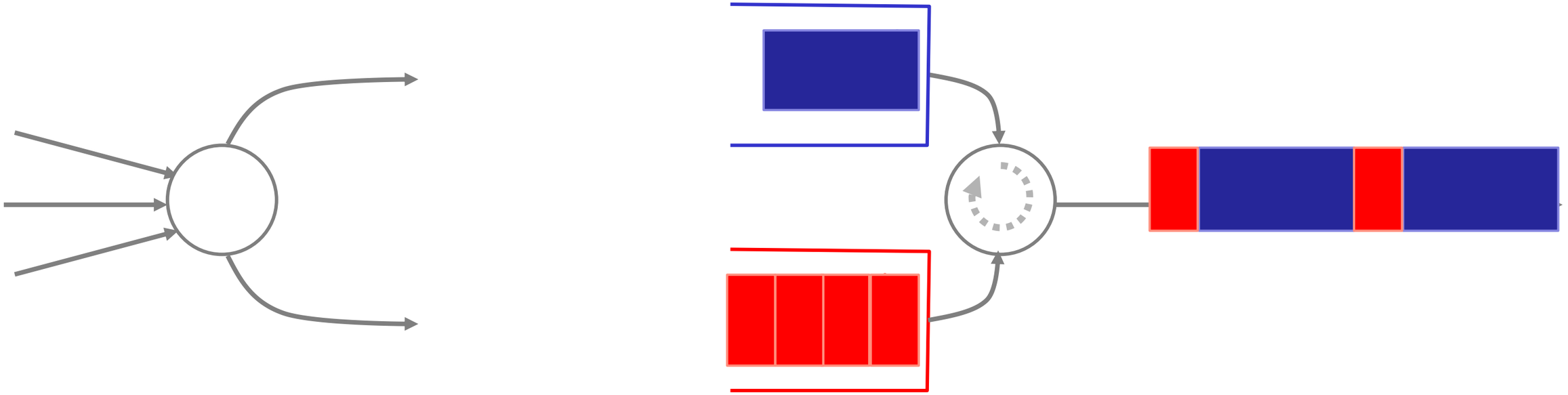
How do I give weighted (instead of strict) priority?



Trying to treat flows equally

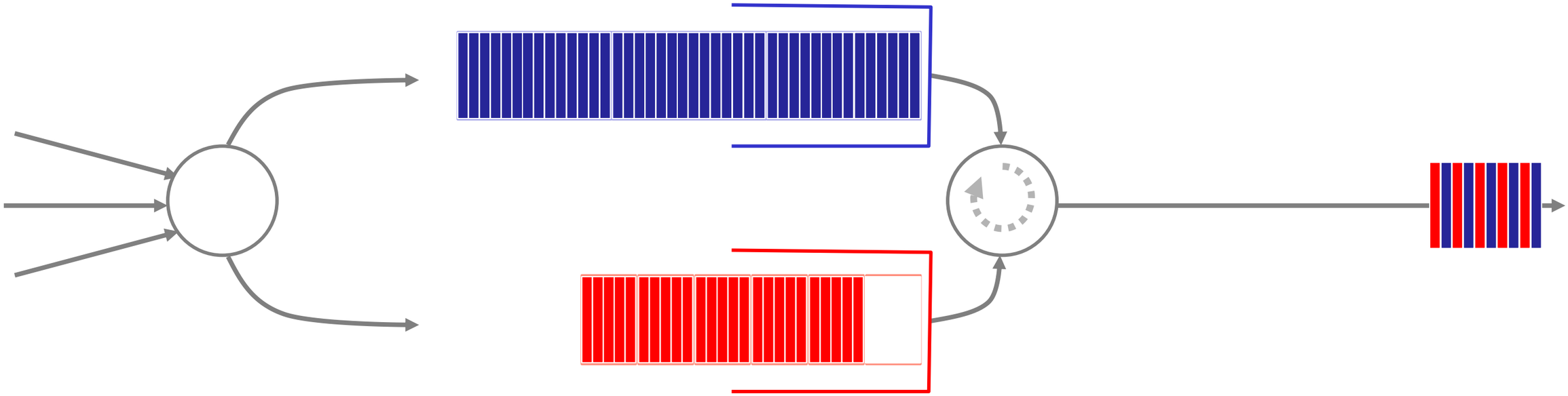


Trying to treat flows equally

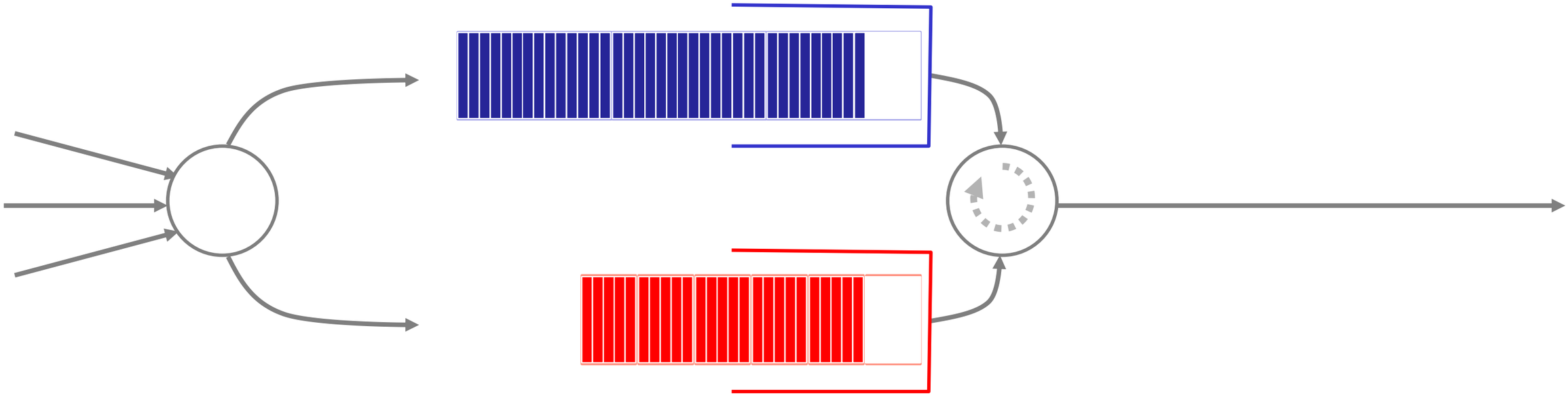


While each flow gets to send at the same packet rate,
the data rate is far from equal.

Scheduling flows bit-by-bit



Scheduling flows bit-by-bit

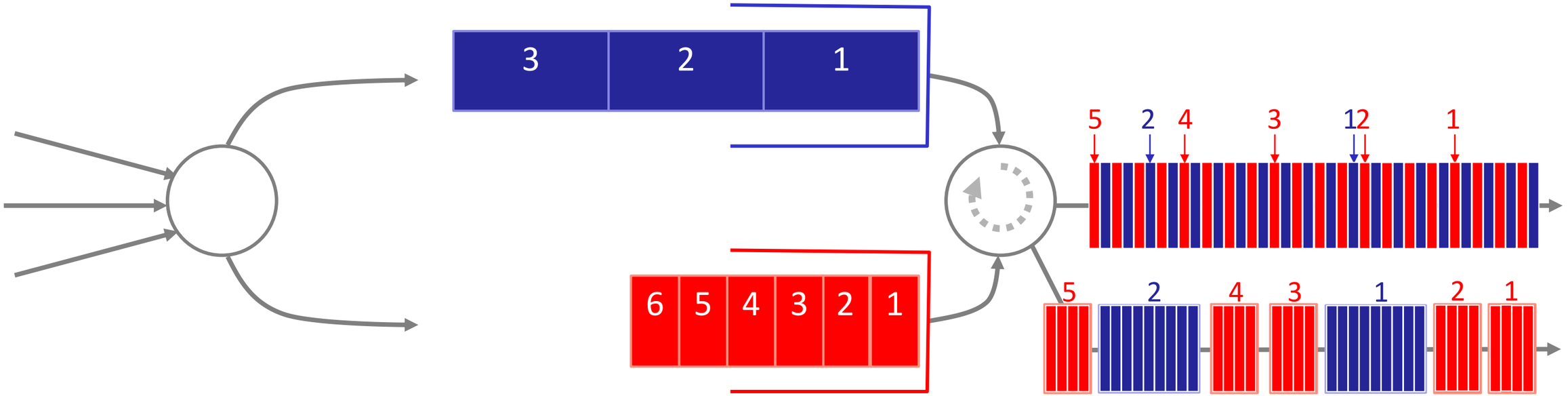


Now each flow gets to send at the same data rate,
but we no longer have “packet switching”.

Can we combine the best of both?

i.e. packet switching, but with bit-by-bit accounting?

Fair Queueing



Packets are sent in the order they would complete in the bit-by-bit scheme.

Does this give fair (i.e. equal) share of the data rate?

Yes!

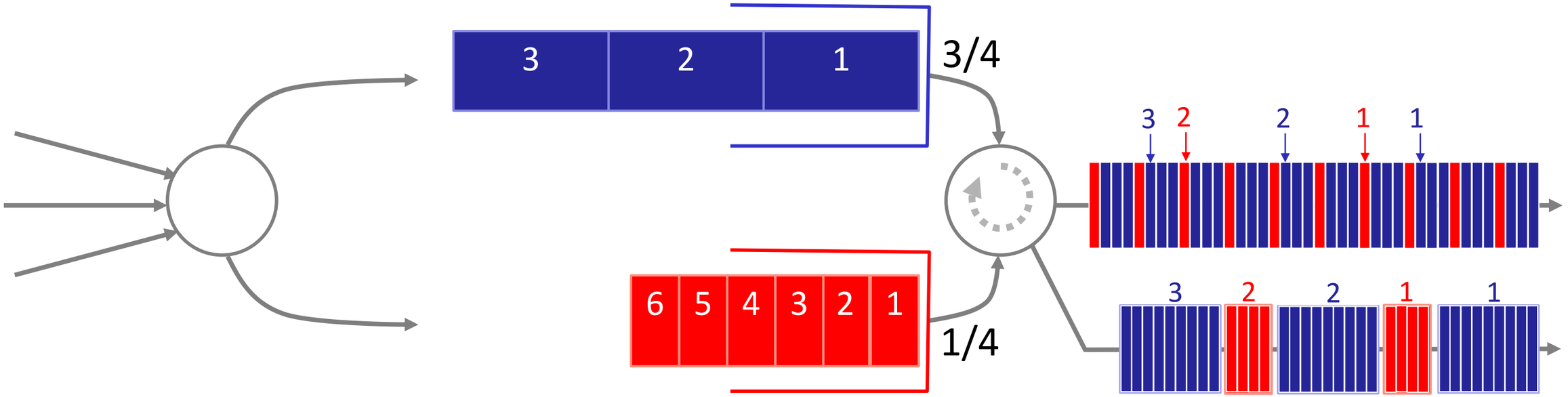
1. It can be proved that the departure time of a packet with Fair Queueing is no more than L_{max}/R seconds later than if it was scheduled bit-by-bit, where L_{max} is the maximum length packet and R is the data rate of the egress link.
2. In the limit, the two flows receive equal share of the data rate.
3. The result extends to any number of flows sharing a link.[1]

[1] "Analysis and Simulation of a Fair Queueing Algorithm" Demers, Keshav, Shenker. 1990.

What if we want to give a different share of the link to each flow?

i.e., a weighted fair share.

Weighted Fair Queueing

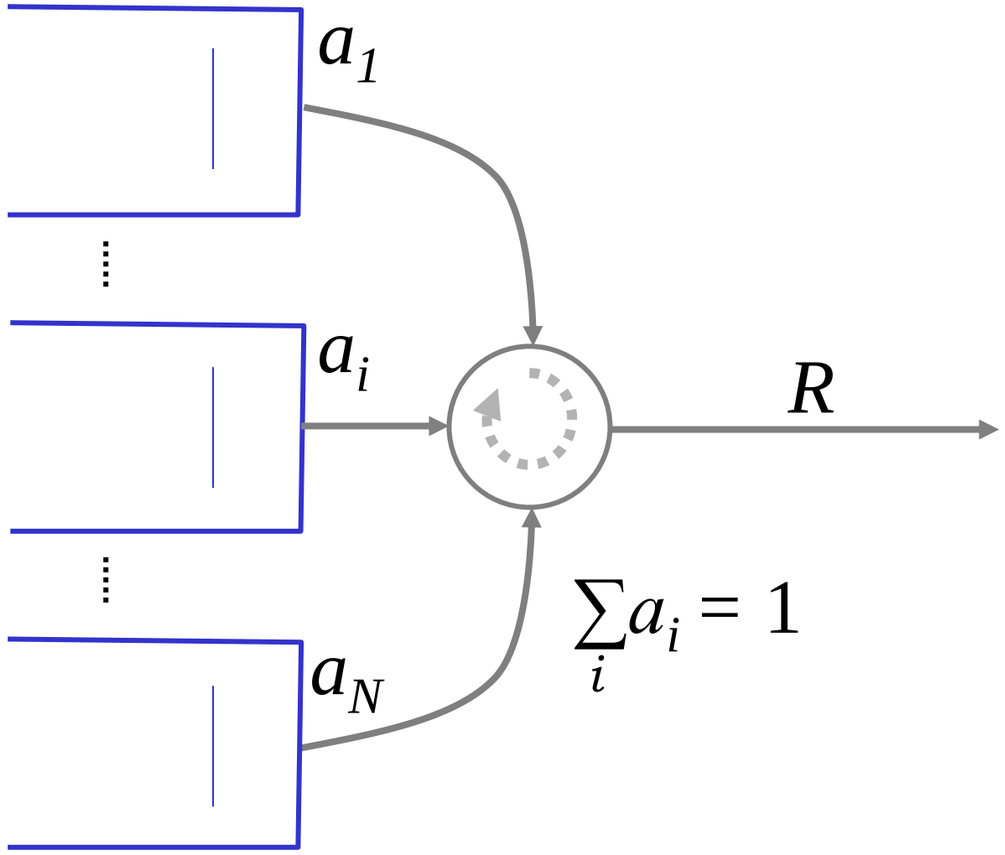


As before, packets are sent in the order they would complete in the bit-by-bit scheme.

Weighted Fair Queueing (WFQ)

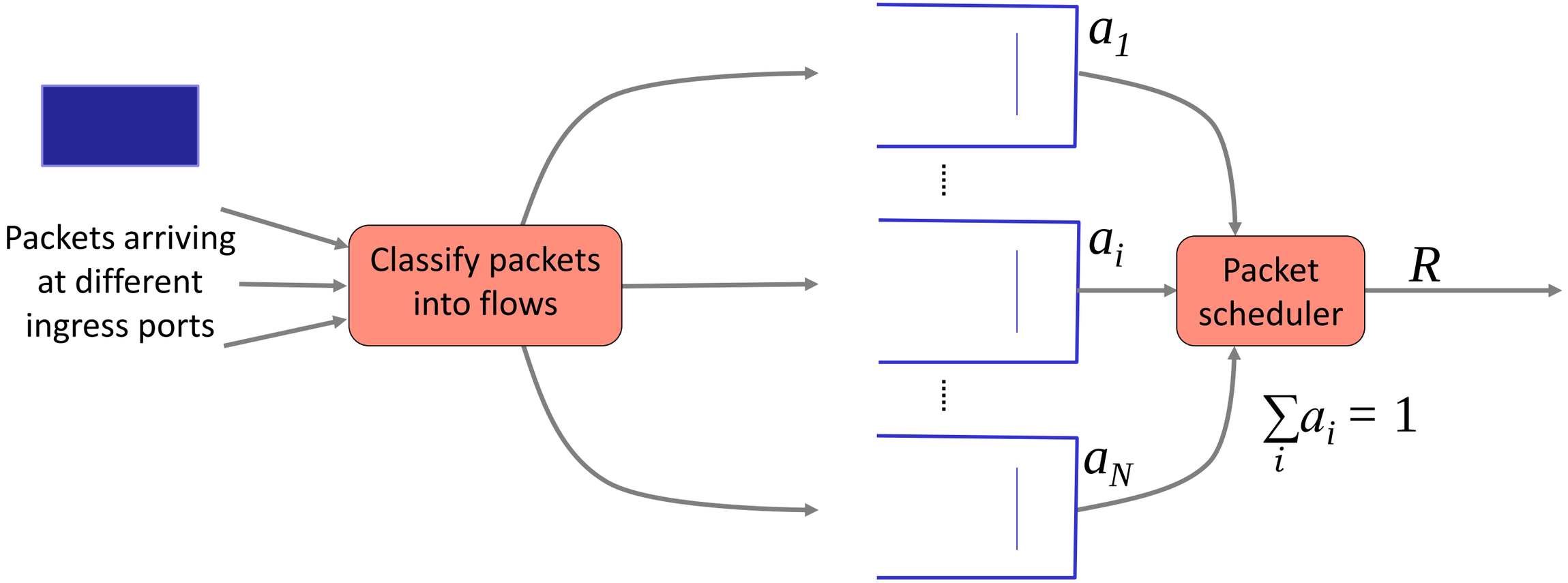
For any number of flows, and any mix of packet sizes:

1. Determine the departure time for each packet using the weighted bit-by-bit scheme.
2. Forward the packets in order of increasing departure time.



Flow i is guaranteed to receive at least rate $a_i R$

Weighted Fair Queueing (WFQ)



Flow i is guaranteed to receive at least rate $a_i R$

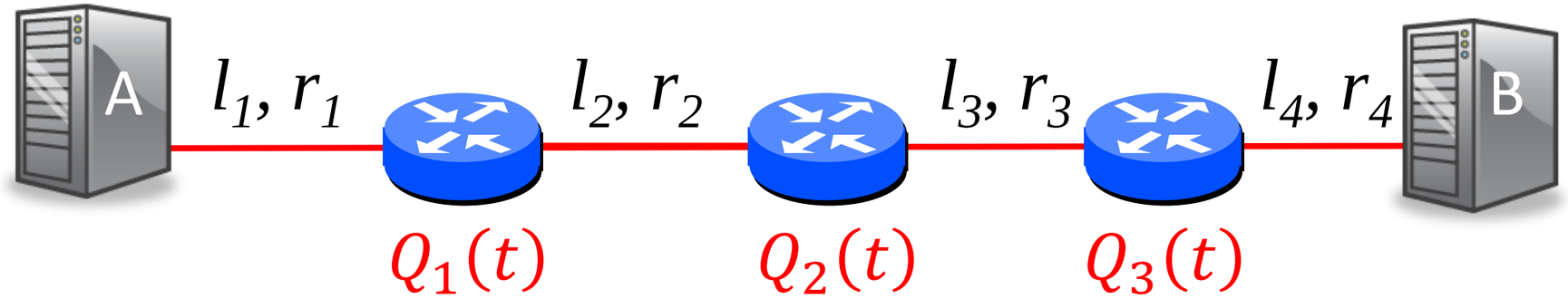
Summary

1. FIFO queues are a free for all: No priorities, no guaranteed rates.
2. Strict priorities: High priority traffic “sees” a network with no low priority traffic. Useful if we have limited amounts of high priority traffic.
3. Weighted Fair Queueing (WFQ) lets us give each flow a guaranteed service rate, by scheduling them in order of their bit-by-bit finishing times.

Outline of what's coming up next…

1. ~~How to give "strict priority" to some flows~~
2. ~~How to give "weighted priorities" to some flows~~
3. ~~How to give "rate guarantees" to some flows~~
4. How to guarantee the end-to-end latency of a packet across a network

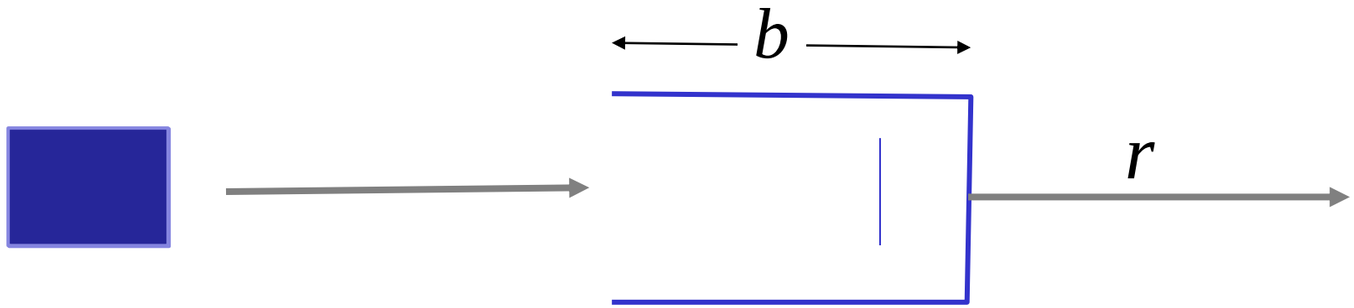
Delay guarantees: Intuition



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

The following values are fixed (or under our control): p, c, l_i and r_i. 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.

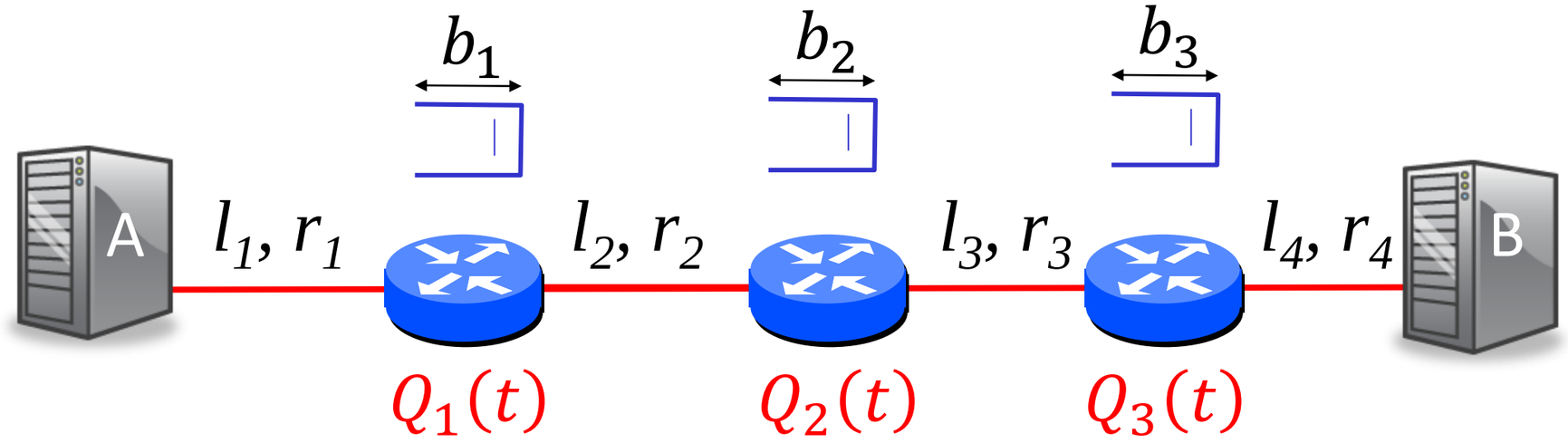
Upper bound on $Q(t)$



$$Q(t) \leq \frac{b}{r}$$

Example: If a packet arrives to a FIFO queue of size 1 million bits, and the queue is served at 1Gb/s, then the packet is guaranteed to depart within $^{10^6}/_{10^9} = 1$ms.

Delay guarantees: Intuition



End-to-end delay for a single packet, $\tau = \sum_{i=1}^{4}\left(\frac{p}{r_i}+\frac{l_i}{c}\right)+\sum_{i=1}^{3}Q_i(t)$

$$\leq \sum_{i=1}^{4}\left(\frac{p}{r_i}+\frac{l_i}{c}\right)+\sum_{i=1}^{3}\frac{b_i}{r_i}$$

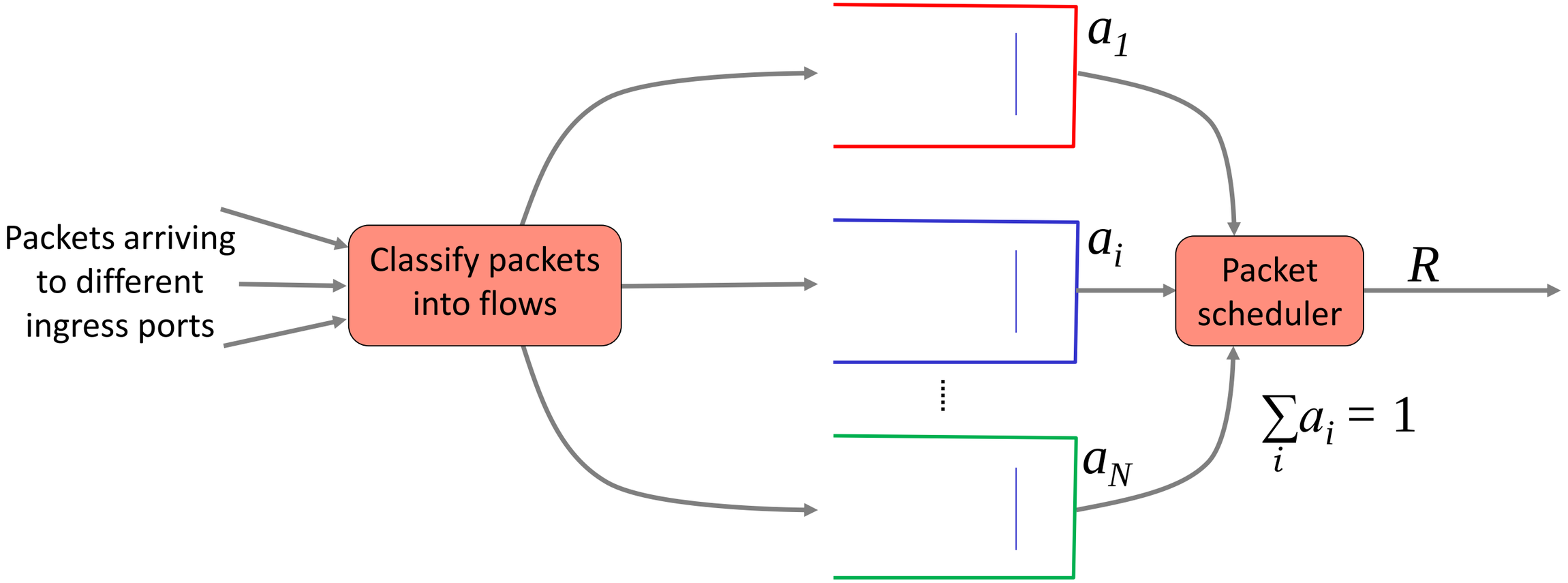
Why this is only an intuition…

1. Doesn't tell us what happens when $r_2 < r_1$. Will packets be dropped?

2. Treats all packets sharing a queue as one big flow; it doesn't give a different end-to-end delay to each flow.

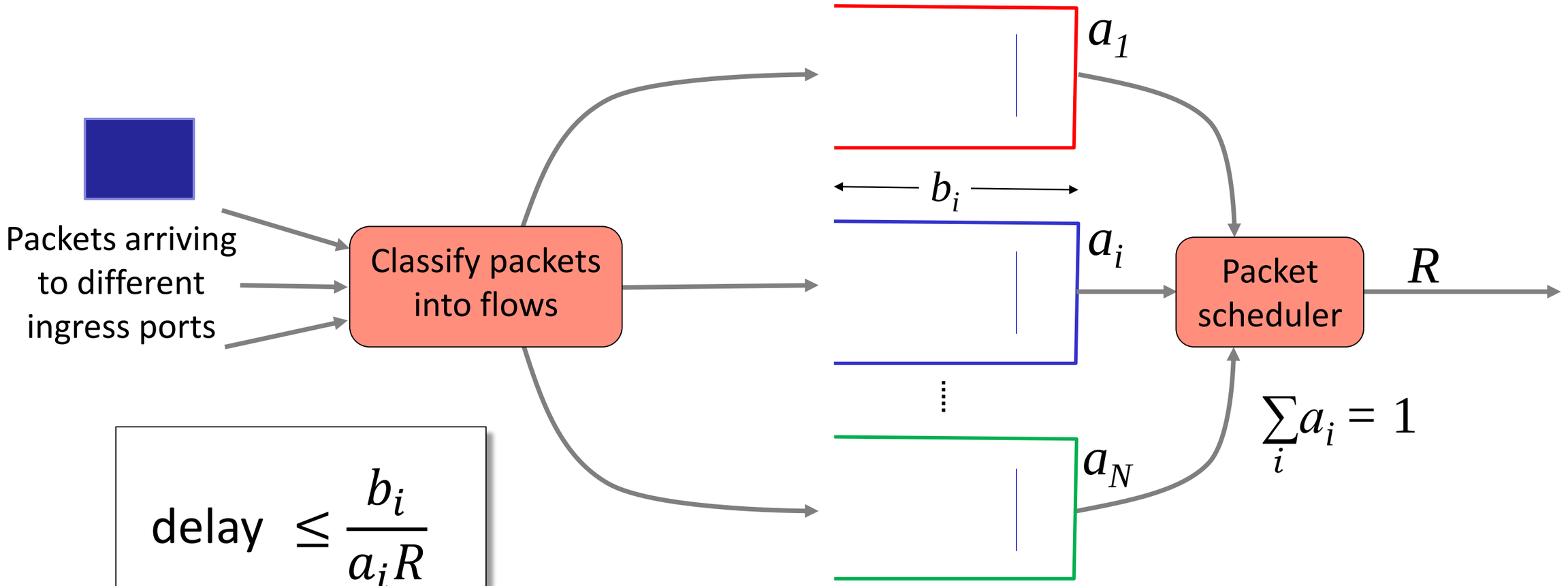
Q: How can we give an upper bound on delay for packets in each flow?

Weighted Fair Queueing (WFQ)



Flow i is guaranteed to receive at least rate $a_i R$

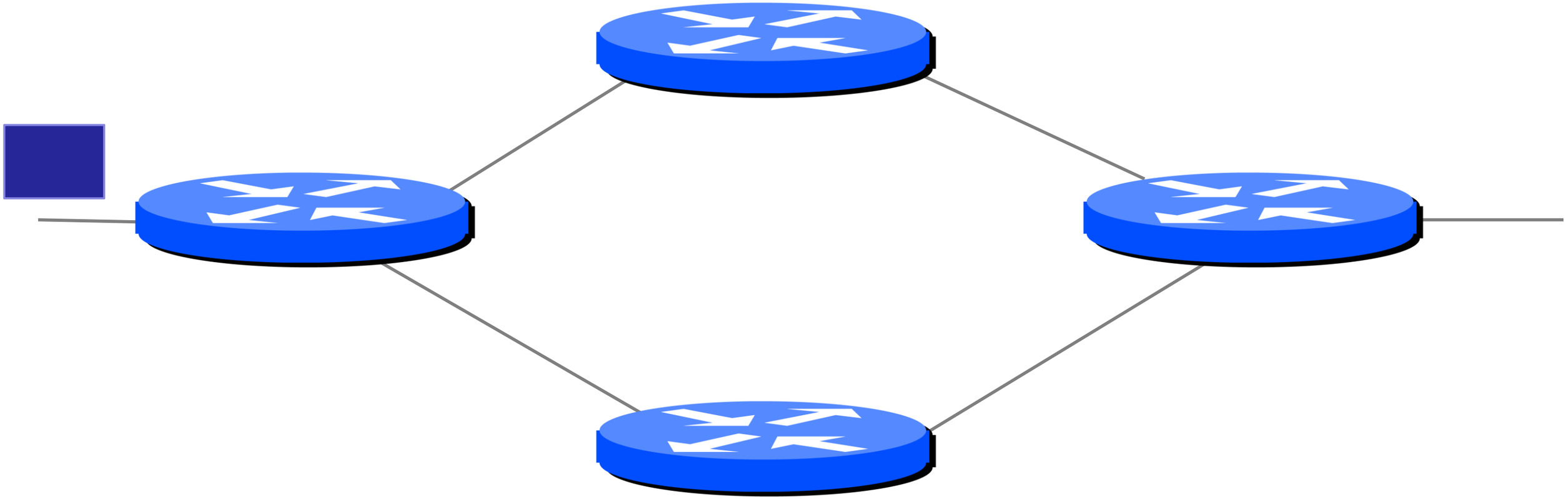
Weighted Fair Queueing (WFQ)



$$\text{delay} \leq \frac{b_i}{a_i R}$$

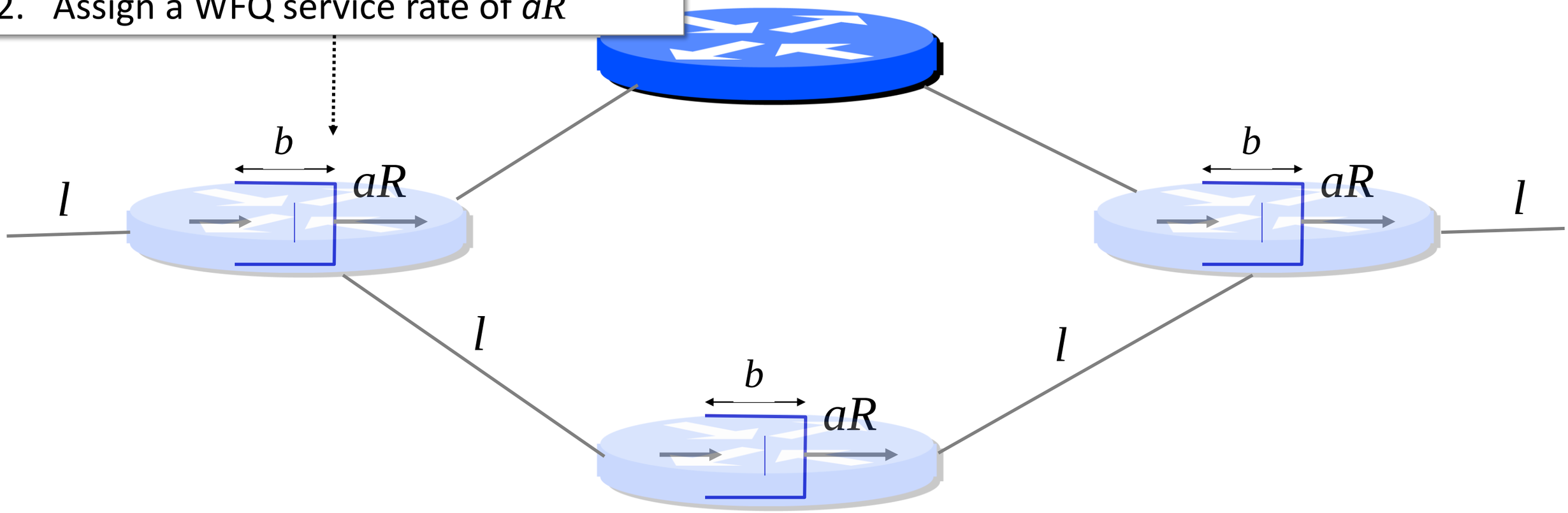
Flow i is guaranteed to receive at least rate $a_i R$

Bounding end-to-end delay



Bounding end-to-end delay

1. Allocate a queue of size b for this flow
2. Assign a WFQ service rate of aR



The end-to-end delay of a single packet of length p $\leq 4\left(\frac{l}{c}+\frac{p}{R}\right)+3\frac{b}{aR}$

What if two of the flow's enter the network back-to-back? (A "burst")

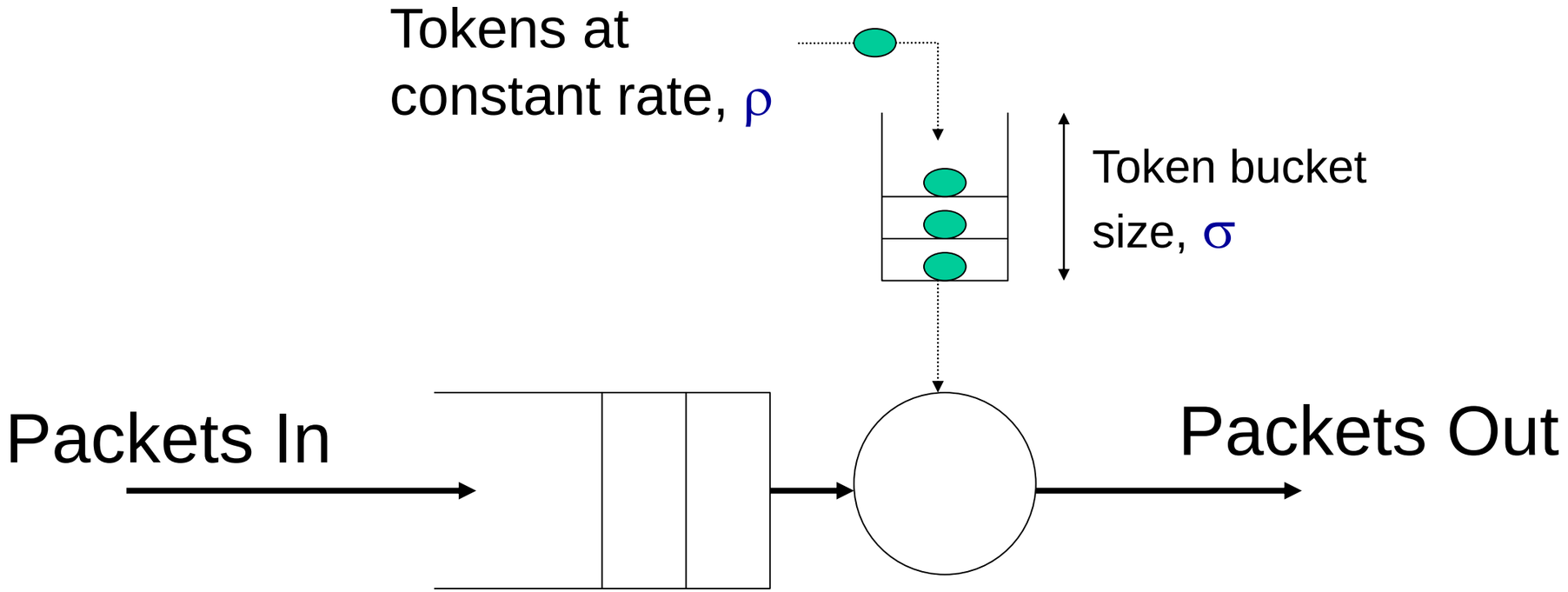
1. If the packets are far apart, then the queues drain the first packet before the second one arrives. All is good, and the delay equation holds.

2. If the packets are close together in a "burst", then they can arrive faster than aR and the queue might overflow, dropping packets.

3. This might be OK in some cases. But if we want to bound the end-to-end delay of all packets, then we need to deal with bursts. How?

The leaky bucket regulator

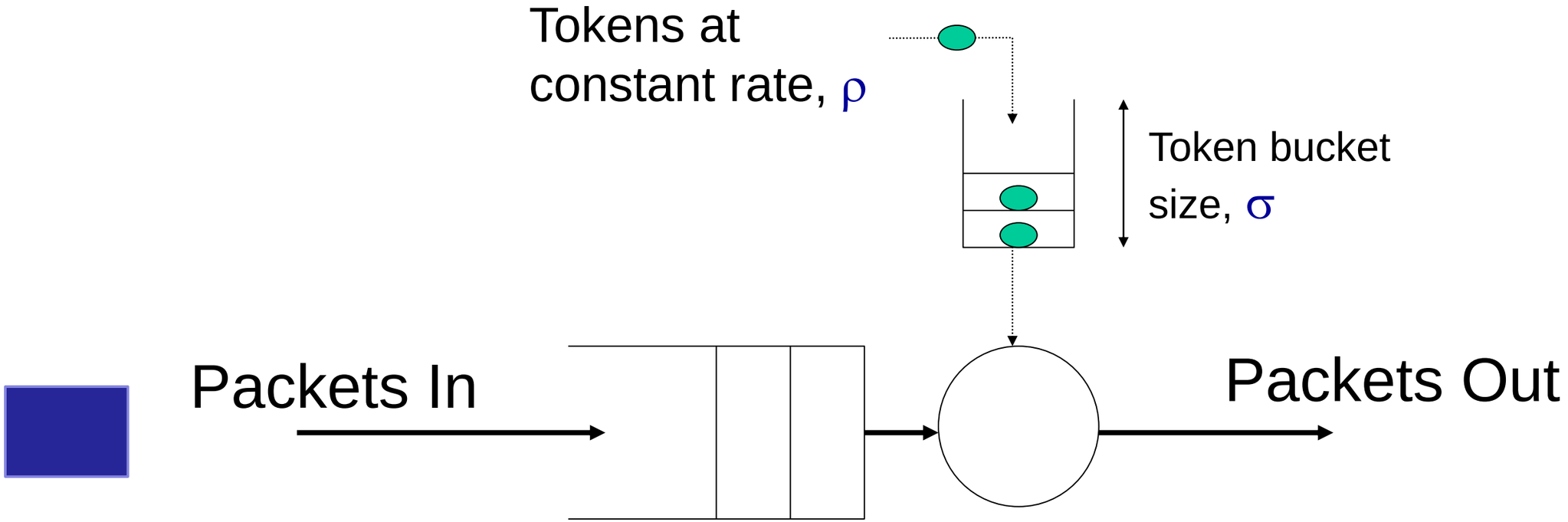
Limiting the “burstiness”



Send packet if and only if we have tokens in the bucket

The leaky bucket regulator

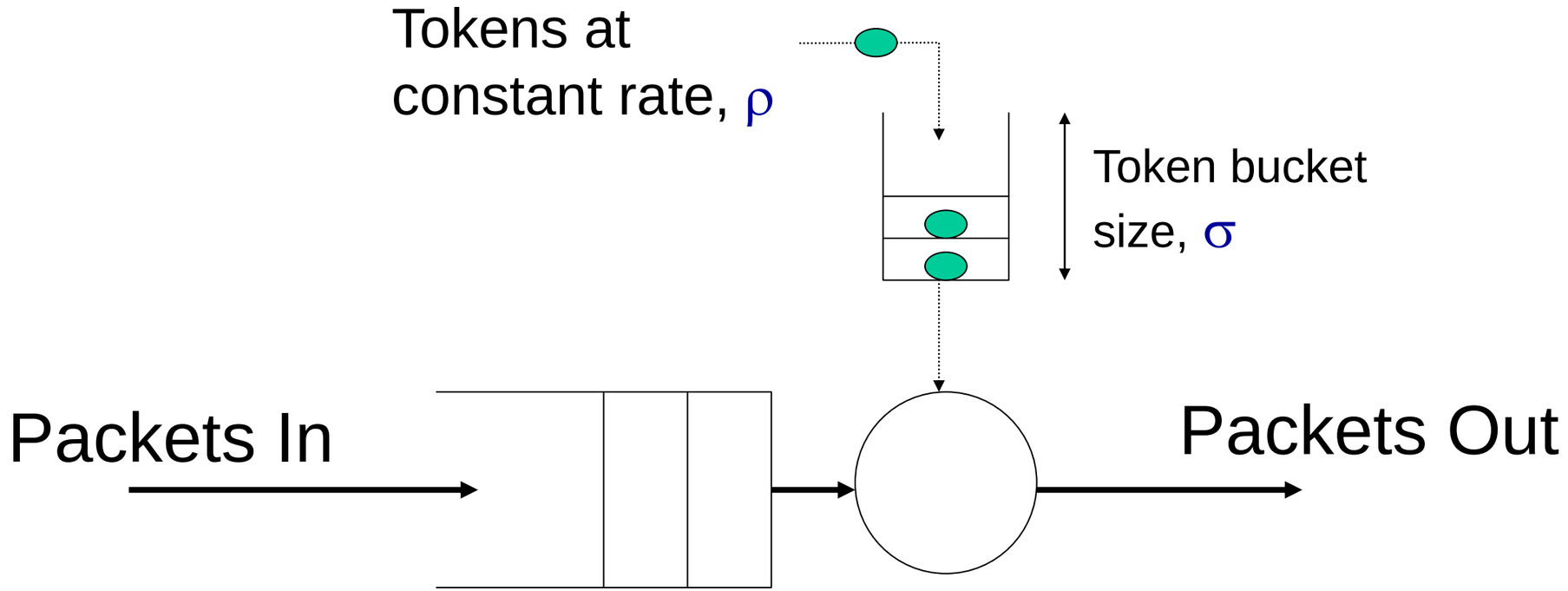
Limiting the “burstiness”



Send packet if and only if we have tokens in the bucket

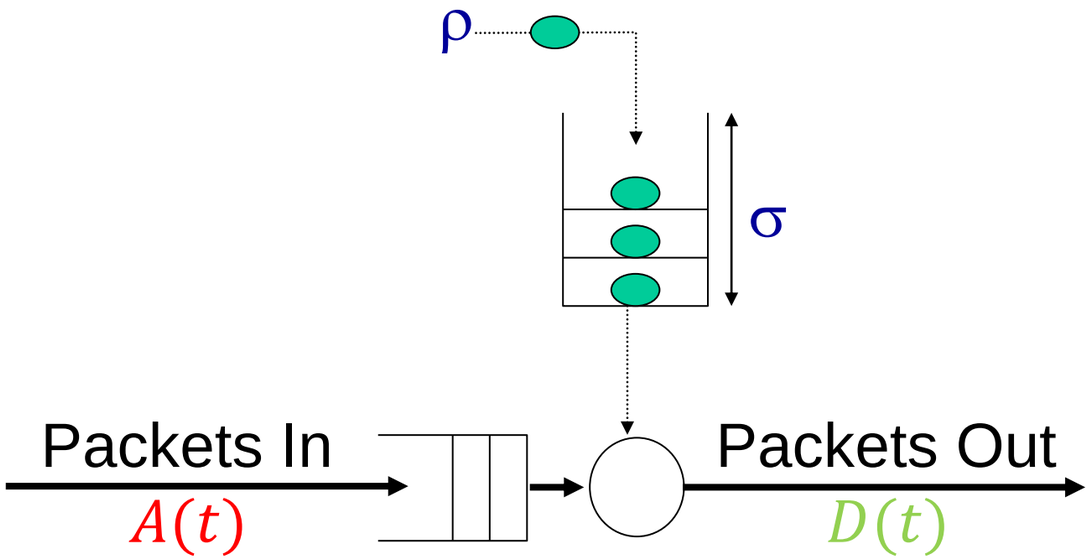
The leaky bucket regulator

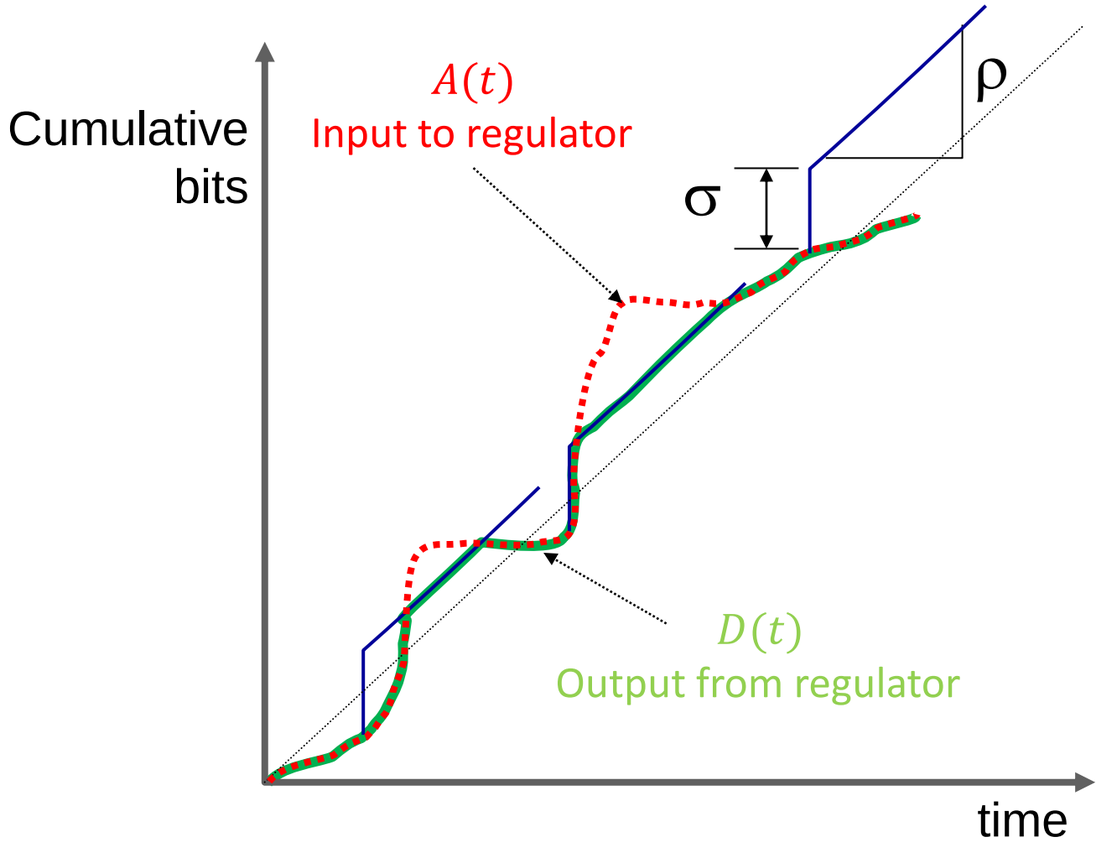
Limiting the “burstiness”



Send packet if and only if we have tokens in the bucket

The leaky bucket regulator



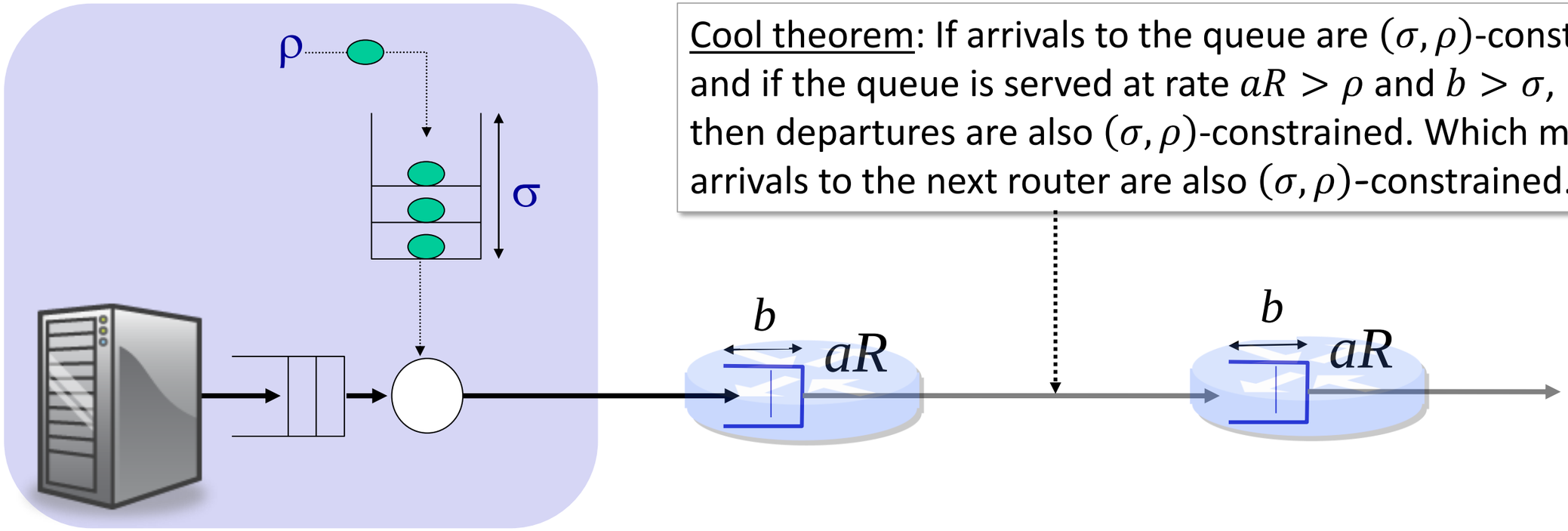


Number of bits that can be sent in <u>any</u> period of length t is bounded by: $\sigma + \rho t$

It is also called a "(σ, ρ) regulator"

The leaky bucket regulator

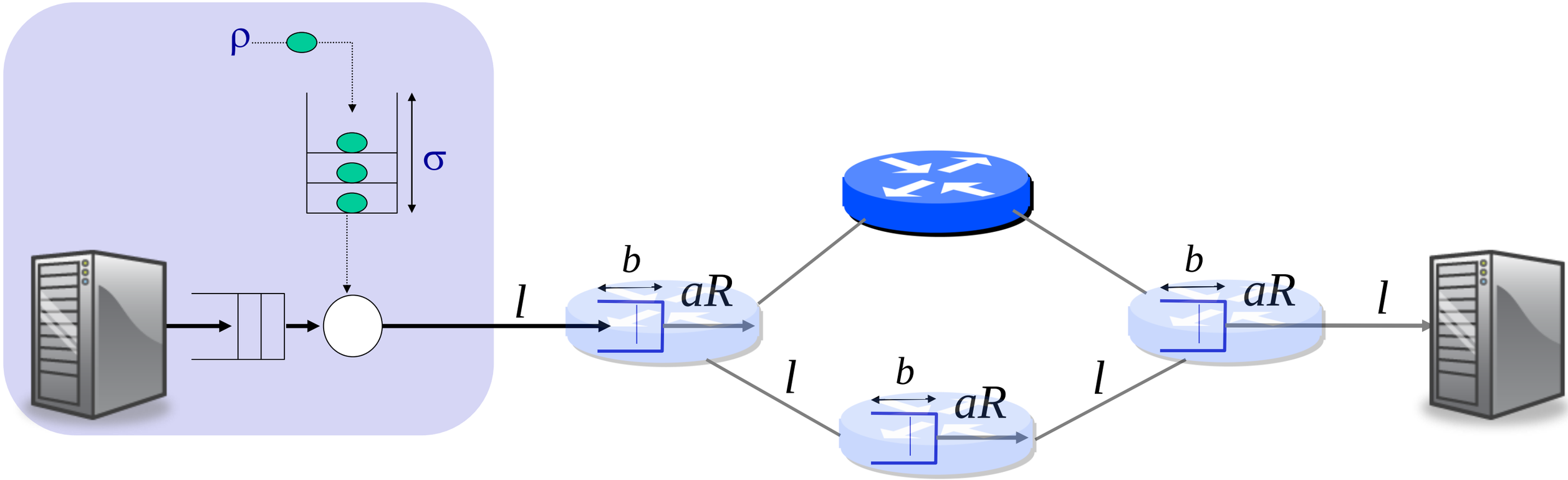
Limiting the “burstiness”



Cool theorem: If arrivals to the queue are (σ, ρ)-constrained, and if the queue is served at rate $aR > \rho$ and $b > \sigma$, then departures are also (σ, ρ)-constrained. Which means arrivals to the next router are also (σ, ρ)-constrained.

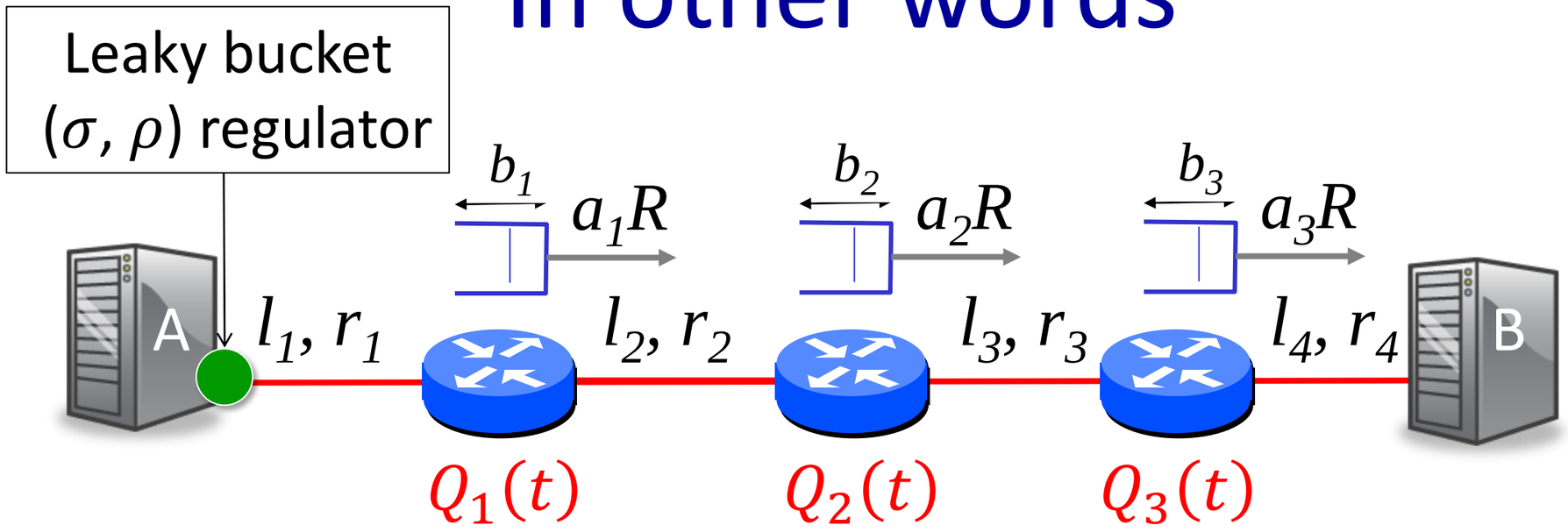
If $aR > \rho$ and $b > \sigma$ then delay through the first router for all packets in the flow $\leq \frac{b}{aR}$

Putting it all together



If $aR > \rho$ and $b > \sigma$ then the end-to-end delay of <u>every</u> packet of length $p \leq 4\left(\frac{l}{c} + \frac{p}{R}\right) + 3\frac{b}{aR}$

In other words



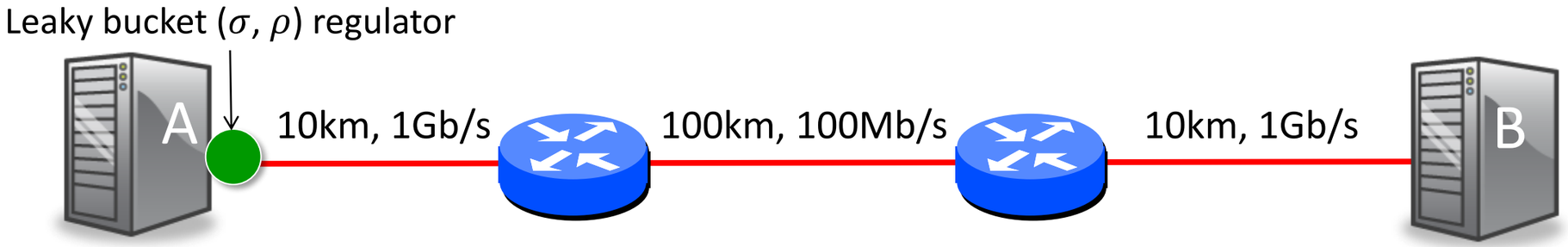
If we set $b_i > \sigma$, and $a_i R > \rho$ then

$$\tau = \sum_{i=1}^{4} \left(\frac{p}{r_i} + \frac{l_i}{c} \right) + \sum_{i=1}^{3} Q_i(t)$$

$$\leq \sum_{i=1}^{4} \left(\frac{p}{r_i} + \frac{l_i}{c} \right) + \frac{3\sigma}{\rho}$$

An Example

Q: In the network below, we want to give an application flow a rate of 10Mb/s and an end to end delay of less than 4.7ms for 1,000 byte packets. What values of σ and ρ should we use for the leaky bucket regulator? And what service rate and buffer size do we need in the routers? (Assume speed of propagation, $c = 2 \times 10^8$m/s).



A: The fixed component of delay is $(120km/c) + 8{,}000bits(\frac{1}{10^9} + \frac{1}{100\times10^6} + \frac{1}{10^9})$ = 0.7ms, leaving 4ms delay for the queues in the routers. Let's apportion 2ms delay to each router, which means the queue in each router need be no larger than $2ms \times 10$Mb/s $= 20{,}000$bits (or 2500bytes). Therefore, the leaky bucket regulator in Host A should have $\rho = 10Mb/s$ and $\sigma \leq 20{,}000bits$. WFQ should be set at each router so that $a_i R \geq 10Mb/s$ and the flow's queue should have a capacity of at least 2500bytes.

In practice

While almost all network equipment implements WFQ (even your WiFi router at home might!), public networks don’t provide a service to control end-to-end delay.

Why?

- It requires coordination of all the routers from end to end.
- In most networks, a combination of over-provisioning and priorities work well enough.

Summary

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

2. WFQ allows us to pick the rate at which a queue is served.

3. With the two observations above, if no packets are dropped, we can control end-to-end delay.

4. To prevent drops, we can use a leaky bucket regulator to control the “burstiness” of flows entering the network.