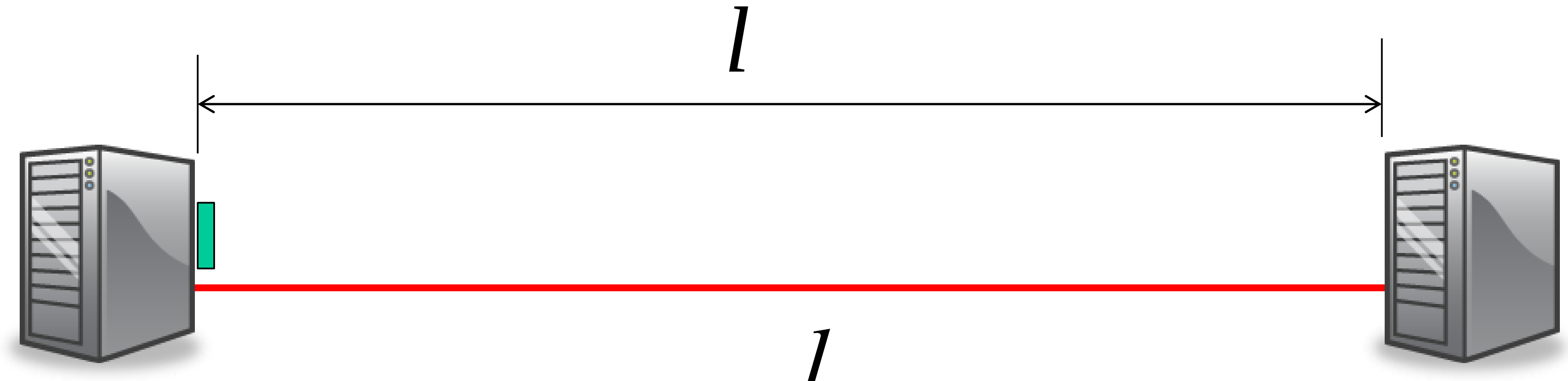
Packet Switching

Outline

1. End-to-end delay
2. Queueing delay
3. Simple deterministic queue model
4. Examples

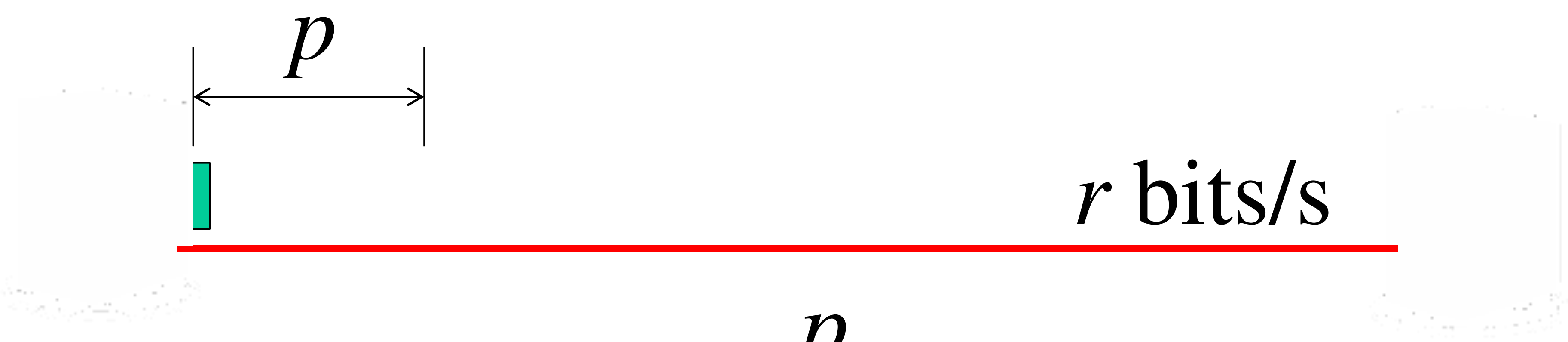
Propagation Delay, t_l: The time it takes a single bit to travel over a link at propagation speed c.



$$t_l = \frac{l}{c}$$

<u>Example</u>: A bit takes 5ms to travel 1,000km in an optical fiber with propagation speed 2 x 10^8 m/s.

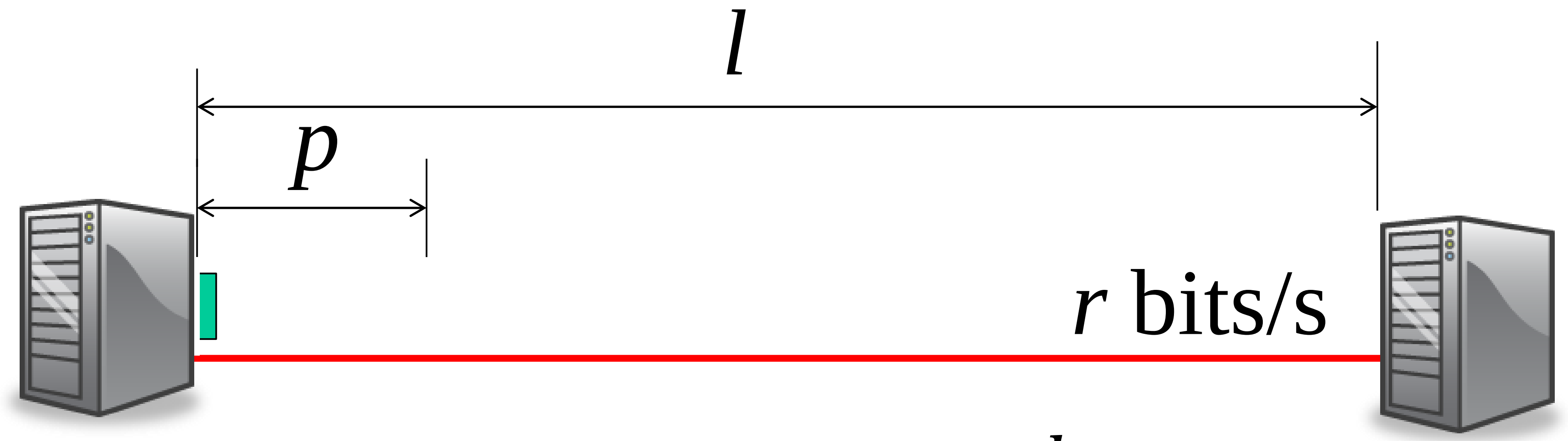
Serialization Delay, t_p: The time from when the first to the last bit of a packet is transmitted.



$$t_p = \frac{p}{r}$$

Example 1: A 64byte packet takes 5.12μs to be transmitted onto a 100Mb/s link.
Example 2: A 1kbit packet takes 1s to be transmitted onto a 1kb/s link.

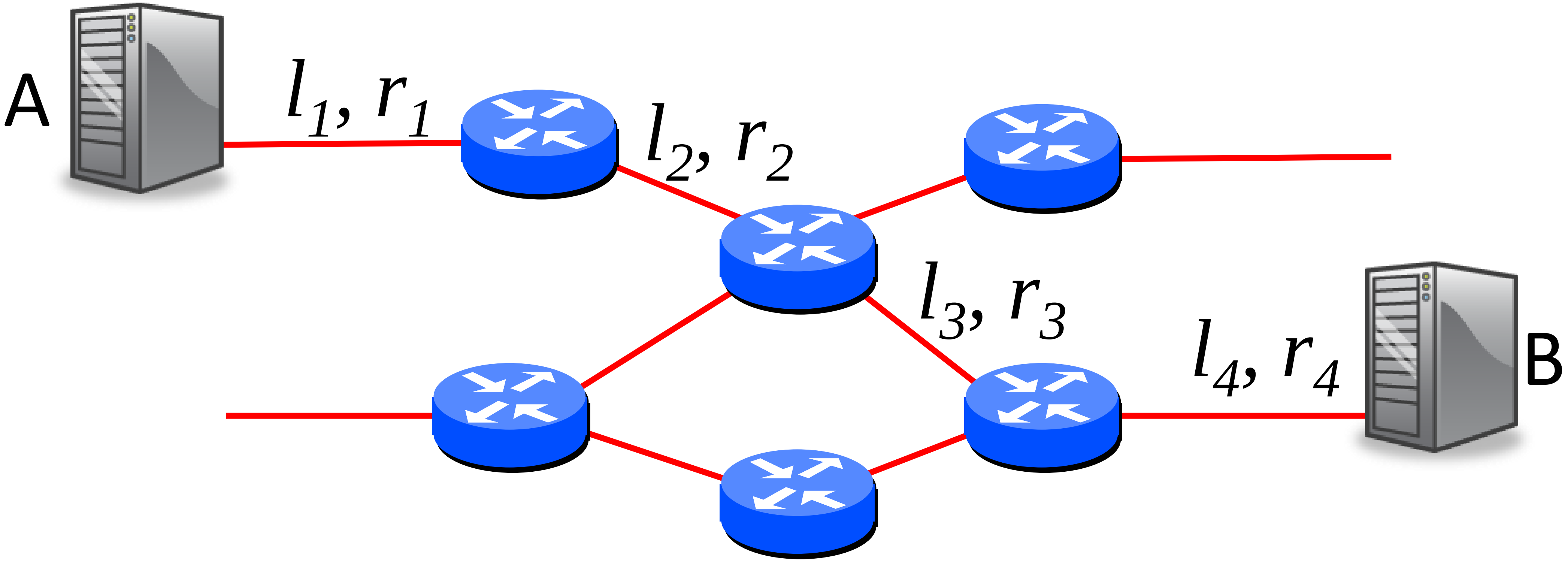
Total time to send a packet across a link: The time from when the first bit is transmitted until the last bit arrives.



$$t = t_p + t_l = \frac{p}{r} + \frac{l}{c}$$

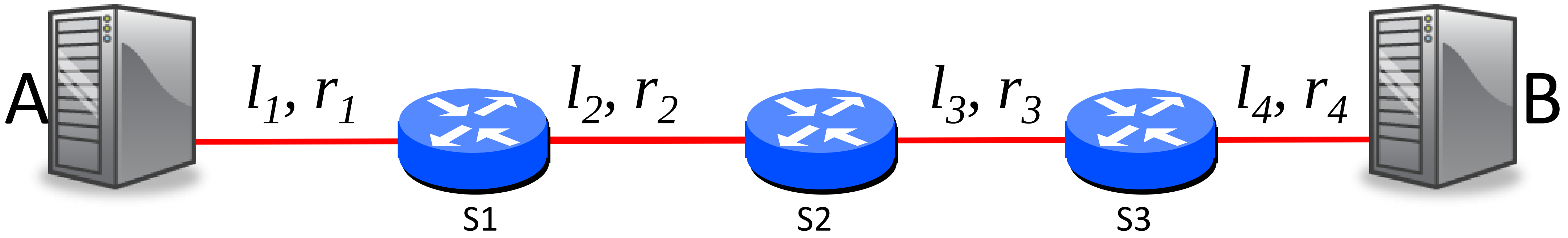
<u>Example</u>: A 100bit packet takes 10 + 5 = 15μs to be sent at 10Mb/s over a 1km link.

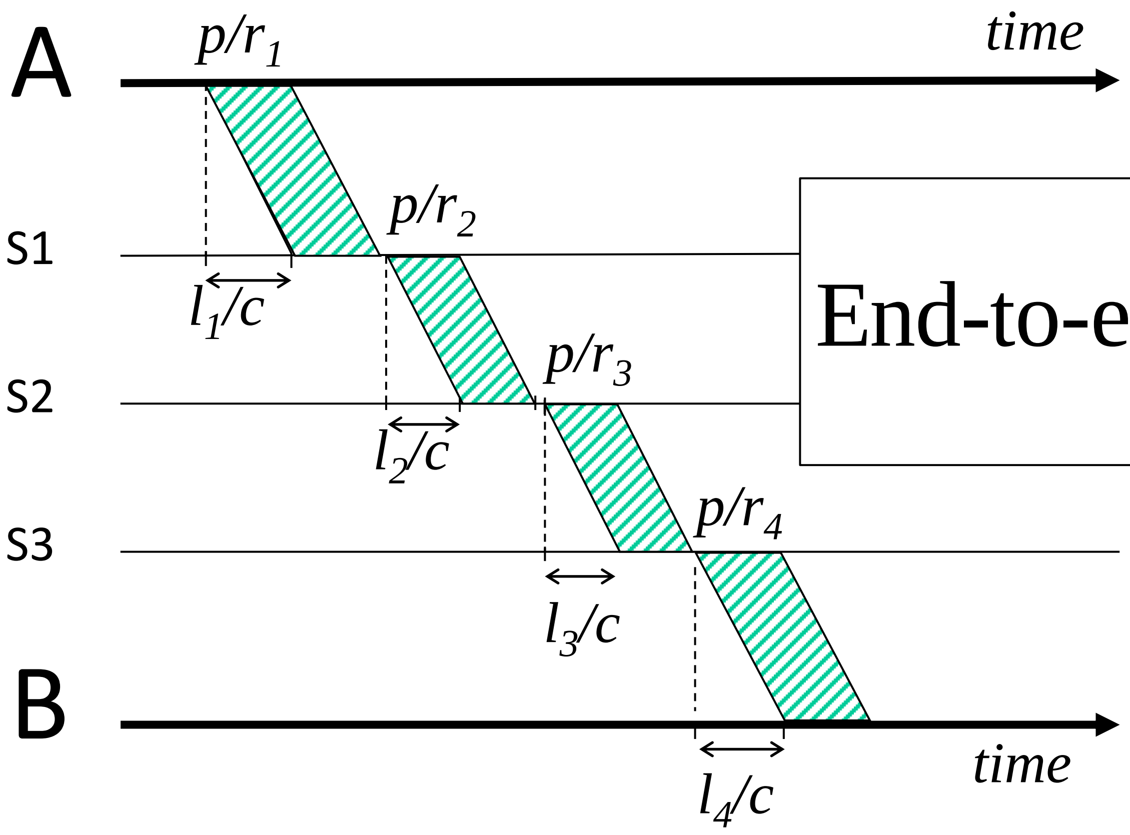
End-to-end delay



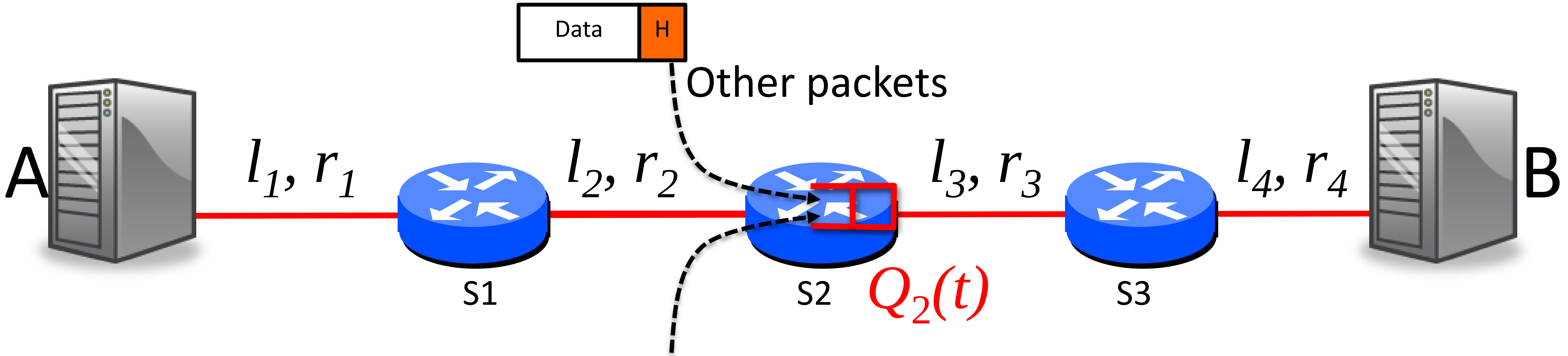
<u>Example</u>: How long will it take a packet of length p to travel from A to B, from when the 1st bit is sent, until the last bit arrives? Assume the switches *store-and-forward* packets along the path.

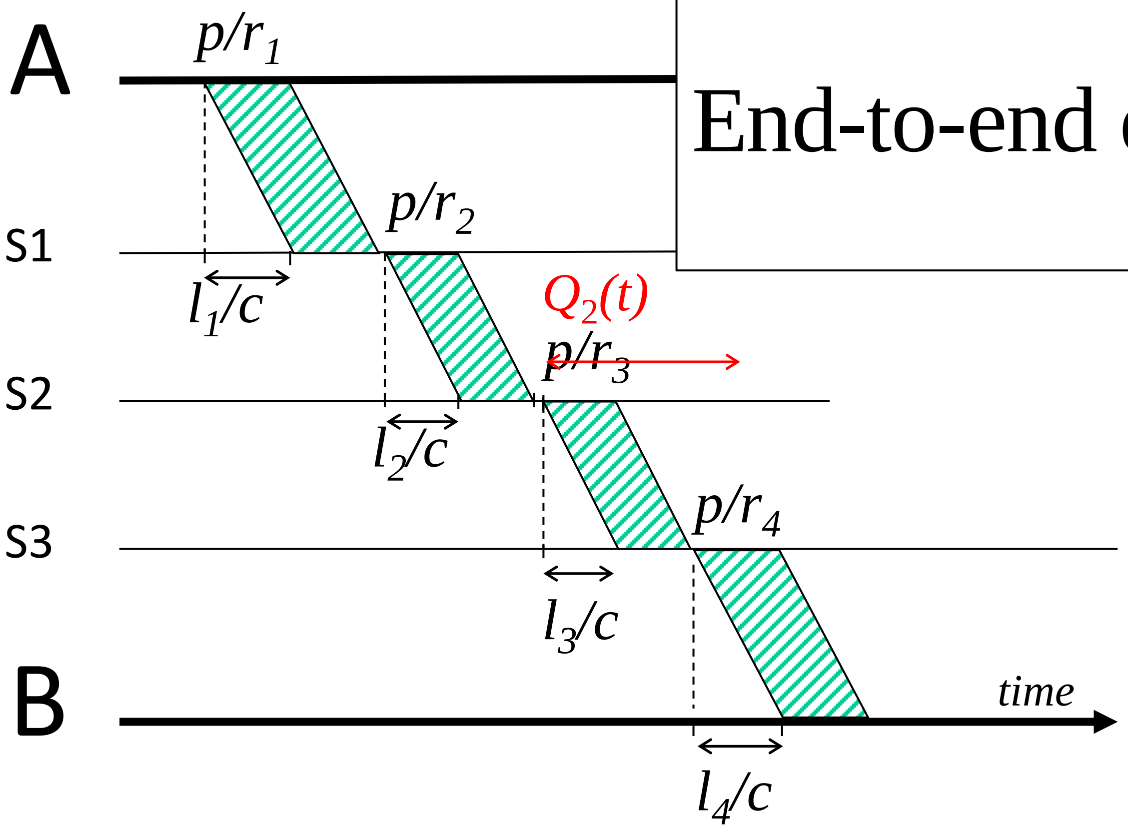
$$\text{End-to-end delay, } t = \sum_i \left(\frac{p}{r_i} + \frac{l_i}{c} \right)$$





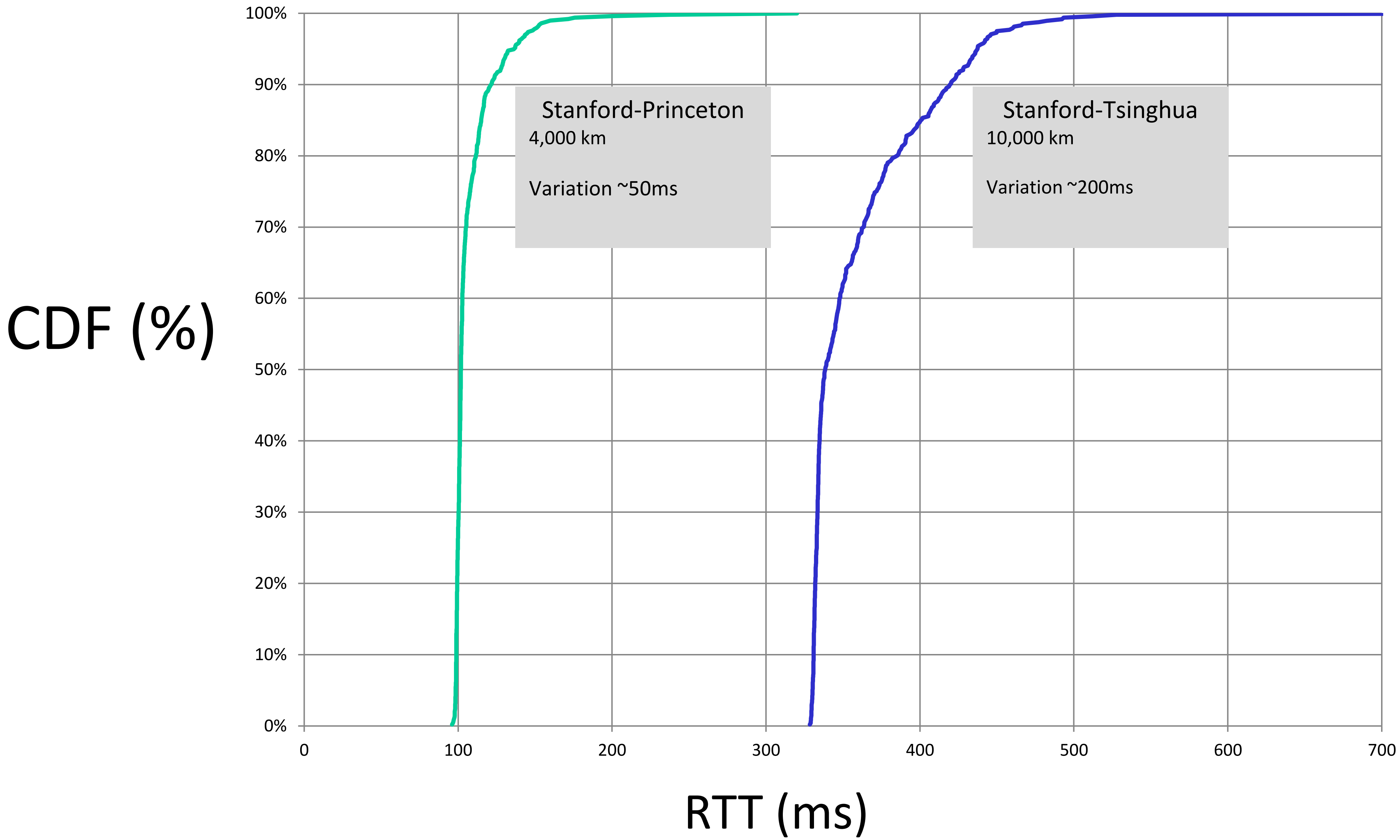
End-to-end delay, $t = \sum_i \left(\frac{p}{r_i} + \frac{l_i}{c} \right)$





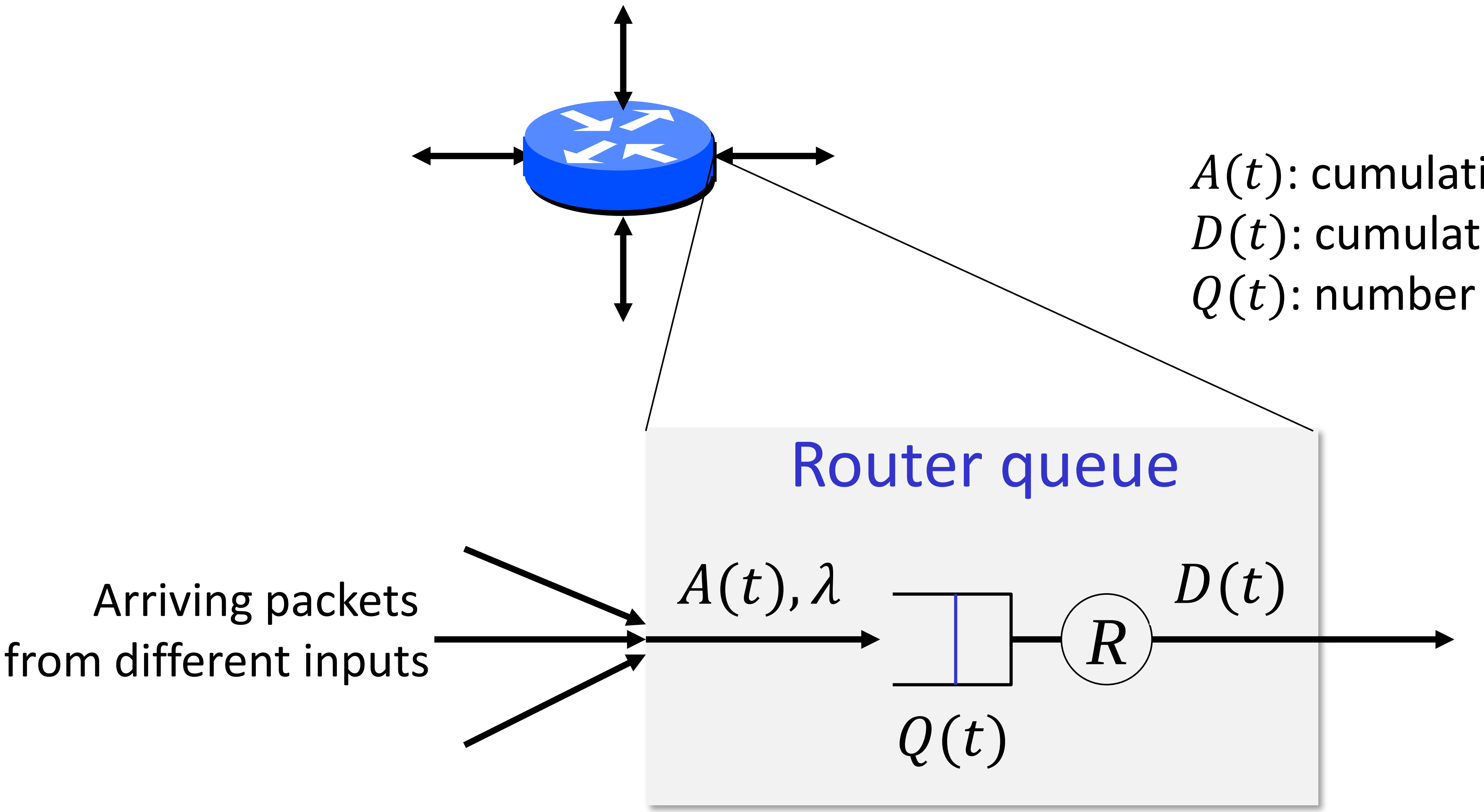
End-to-end delay, $\tau = \sum_i \left(\frac{p}{r_i} + \frac{l_i}{c} + Q_i(t) \right)$

Packet delay variation



Simple model of a router queue

Simple model of a router queue

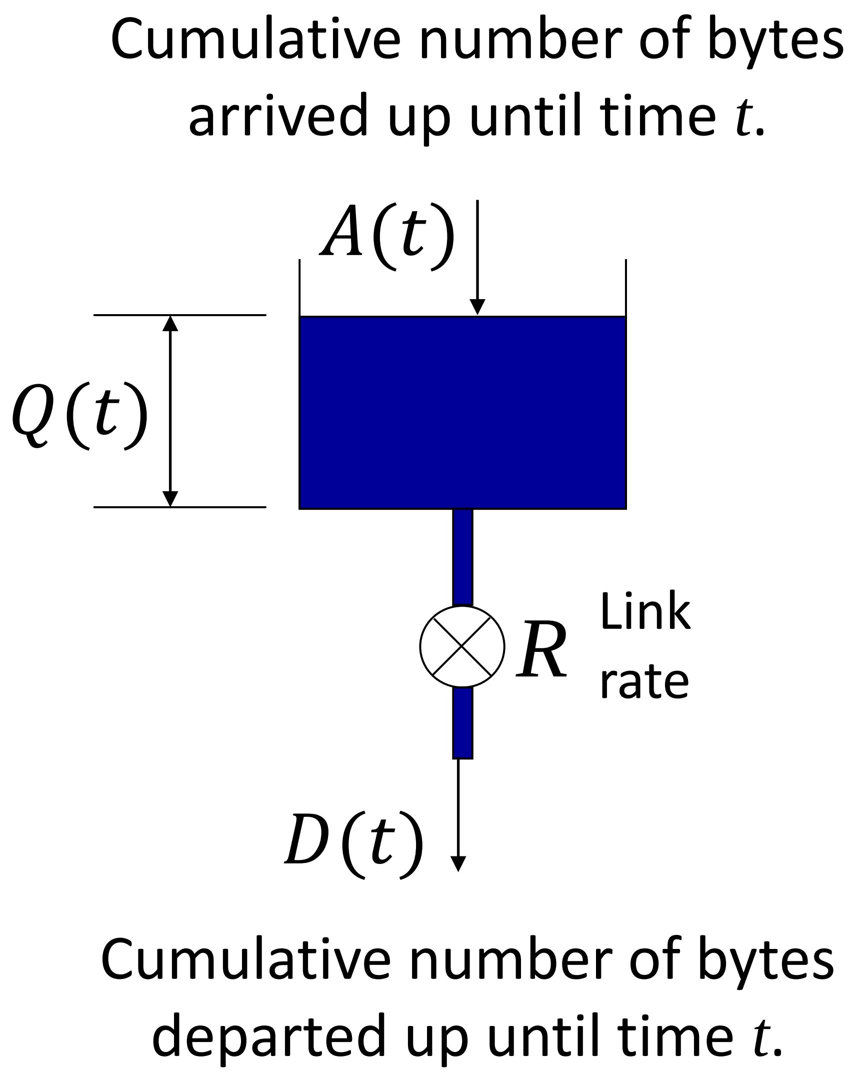


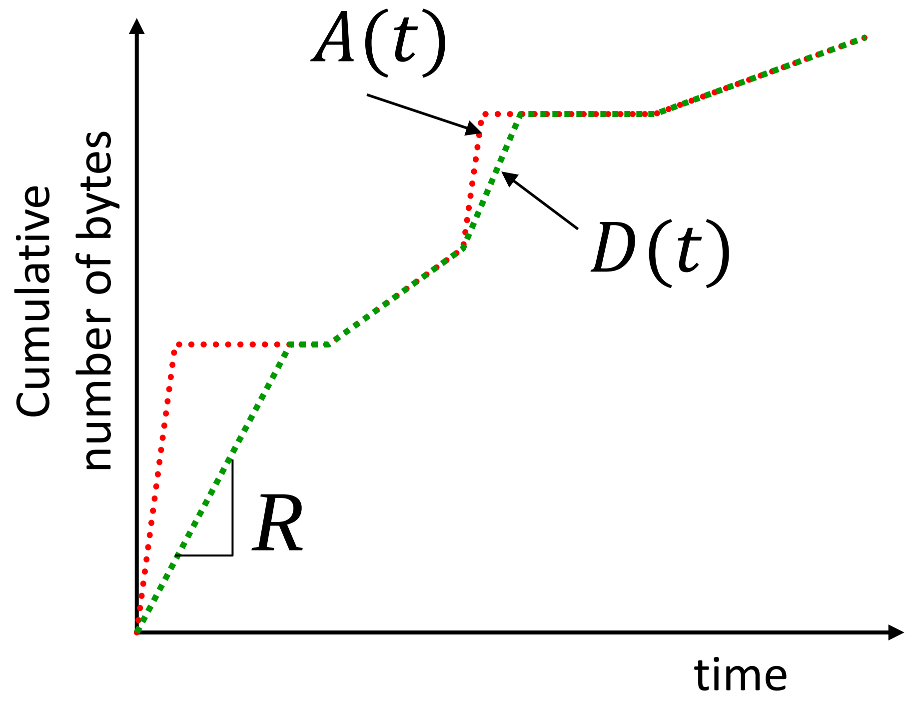
$A(t)$: cumulative arrivals up until time t
$D(t)$: cumulative departures up until time t
$Q(t)$: number of packets in queue at time t

Properties:

1. $A(t), D(t)$ *non-decreasing*
2. $A(t) \geq D(t)$

Simple model of a queue

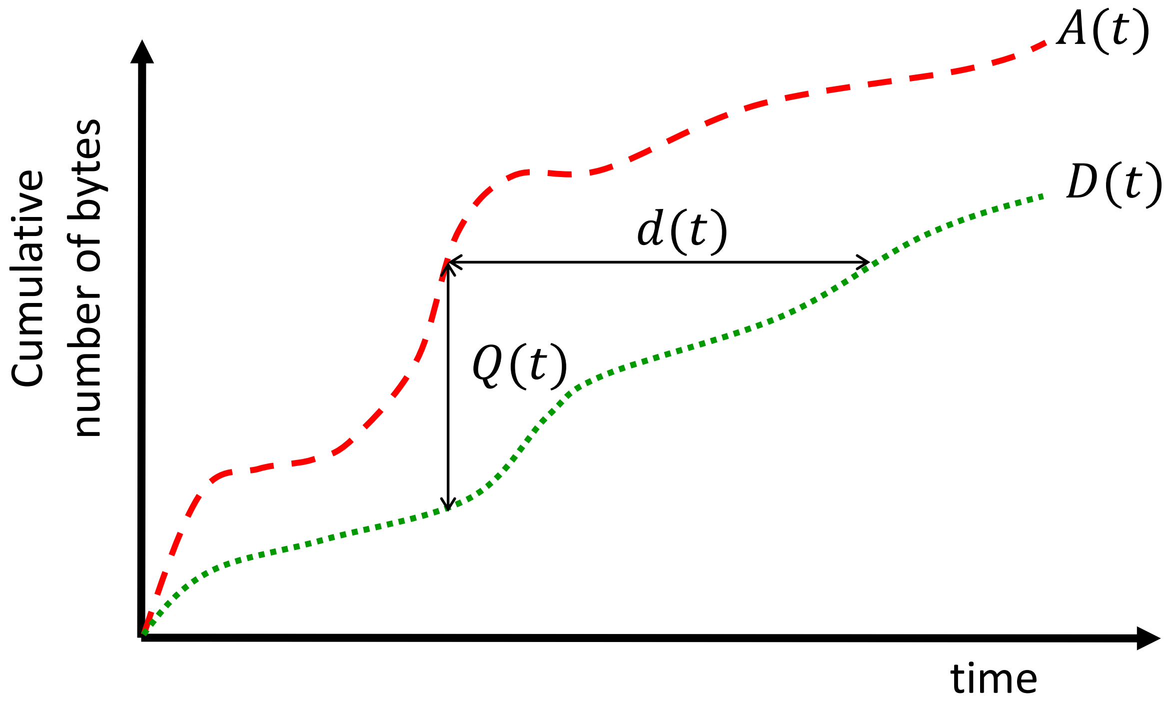




Properties:

1. $A(t), D(t)$ *non-decreasing*
2. $A(t) \geq D(t)$

Simple model of a queue

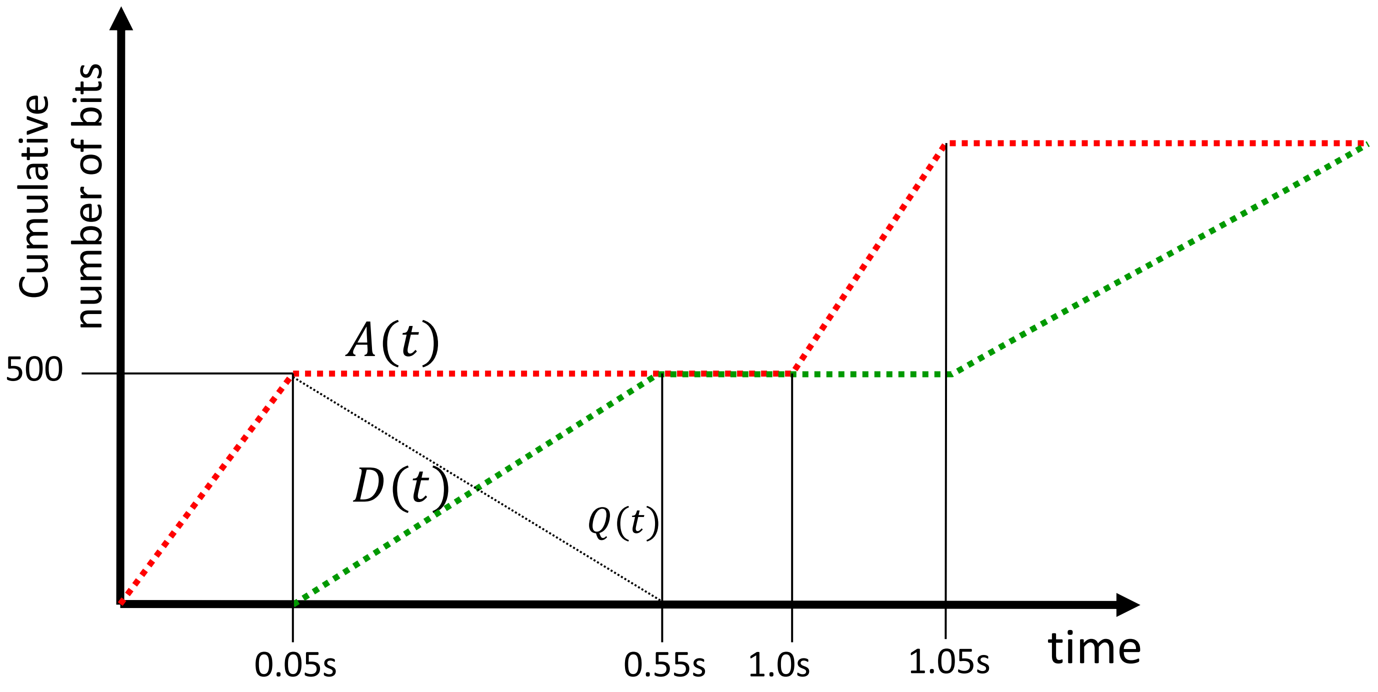


Queue occupancy: $Q(t) = A(t) - D(t)$.

Queueing delay, $d(t)$, is the time spent in the queue by a byte that arrived at time t, assuming the queue is served first-come-first-served (FCFS).

Example (store & forward)

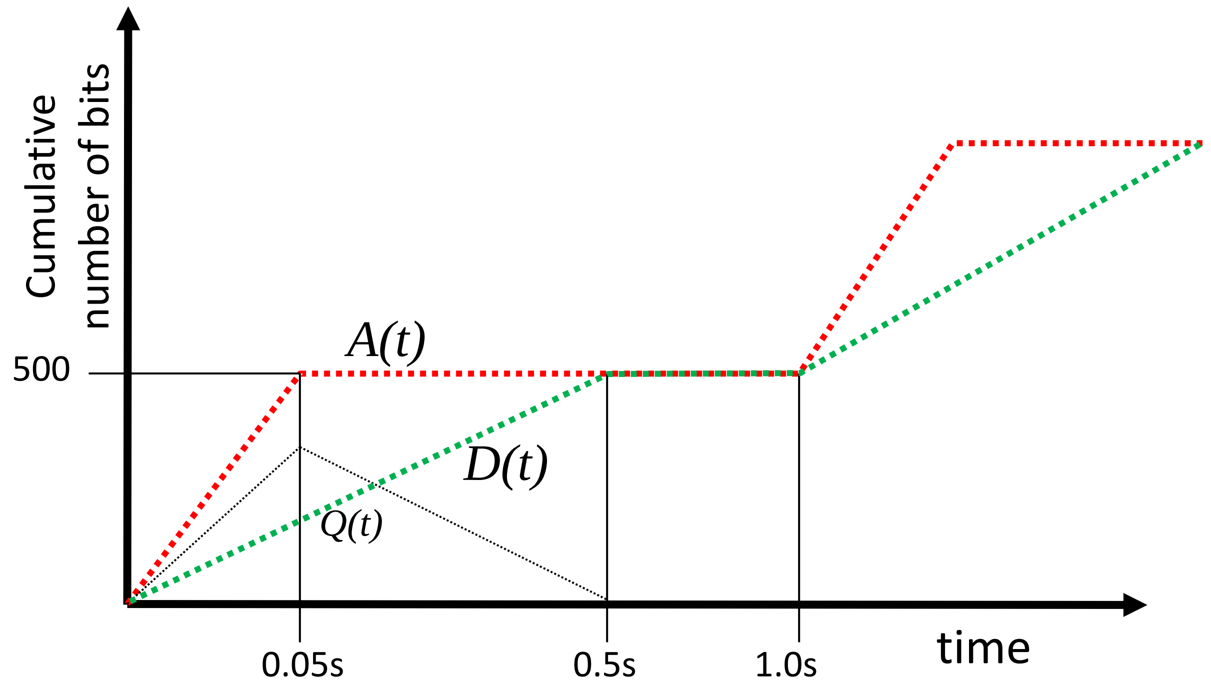
Every second, a 500 bit packet arrives to a queue at rate 10,000b/s. The maximum departure rate is 1,000b/s. What is the average occupancy of the queue?



Solution: During each repeating 1s cycle, the queue fills at rate 10,000b/s for 0.05s, then empties at rate 1,000b/s for 0.5s. Over the first 0.55s, the average queue occupancy is therefore 250 bits. The queue is empty for 0.45s every cycle, and so average queue occupancy is (0.55 * 250) + (0.45 * 0) = 137.5 bits.

Example ("cut through")

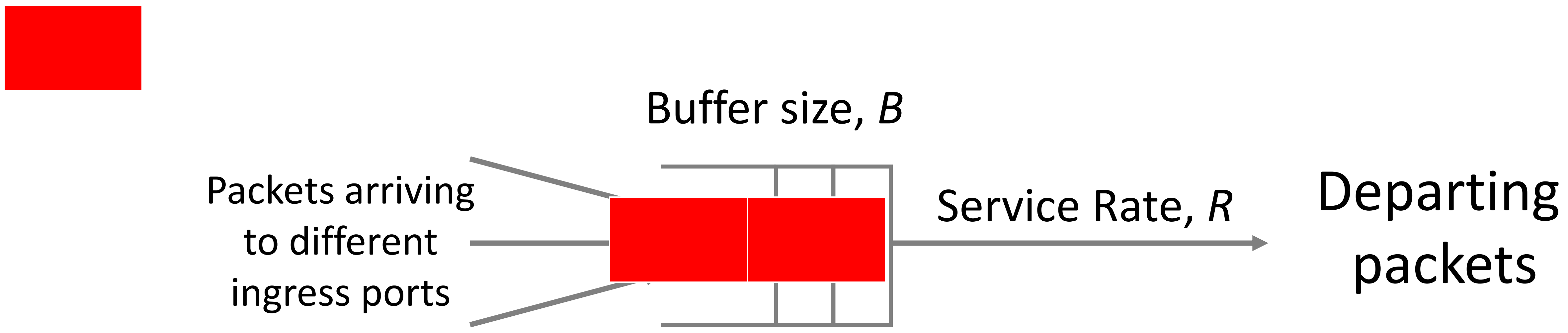
Every second, a 500 bit packet arrives to a queue at rate 10,000b/s. The maximum departure rate is 1,000b/s. What is the time average occupancy of the queue?



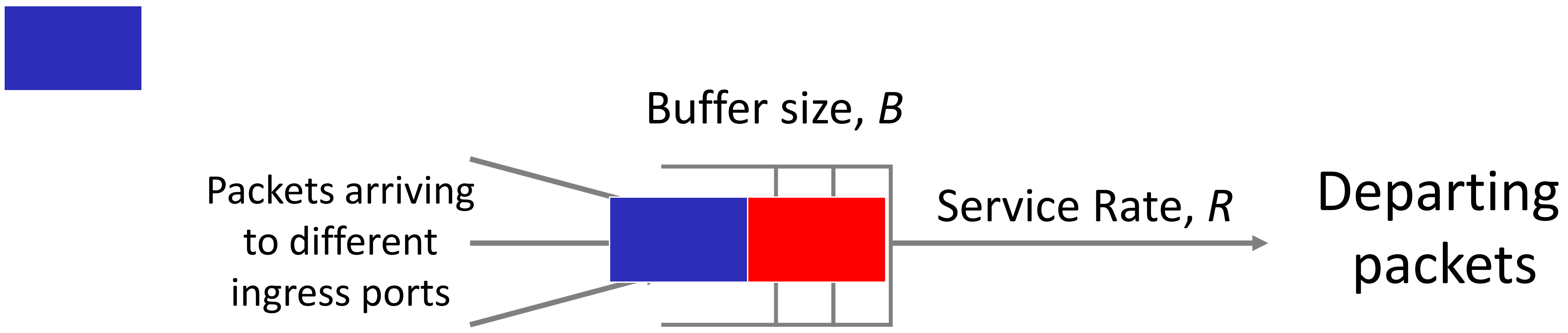
Solution: During each repeating 1s cycle, the queue fills at rate 10,000b/s to 500-50=450 bits over the first 0.05s, then drains at rate 1,000b/s for 0.45s. Over the first 0.5s, the average queue occupancy is therefore 225 bits.
The queue is empty for 0.5s every cycle, and so average queue occupancy: $\bar{Q}(t) = (0.5 \times 225) + (0.5 \times 0) = 112.5$

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. The control traffic 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 a particular application (e.g. videoconference)
4. Traffic to/from particular IP addresses (e.g. emergency services)
5. Traffic that is time sensitive (e.g. clock updates)

Flows

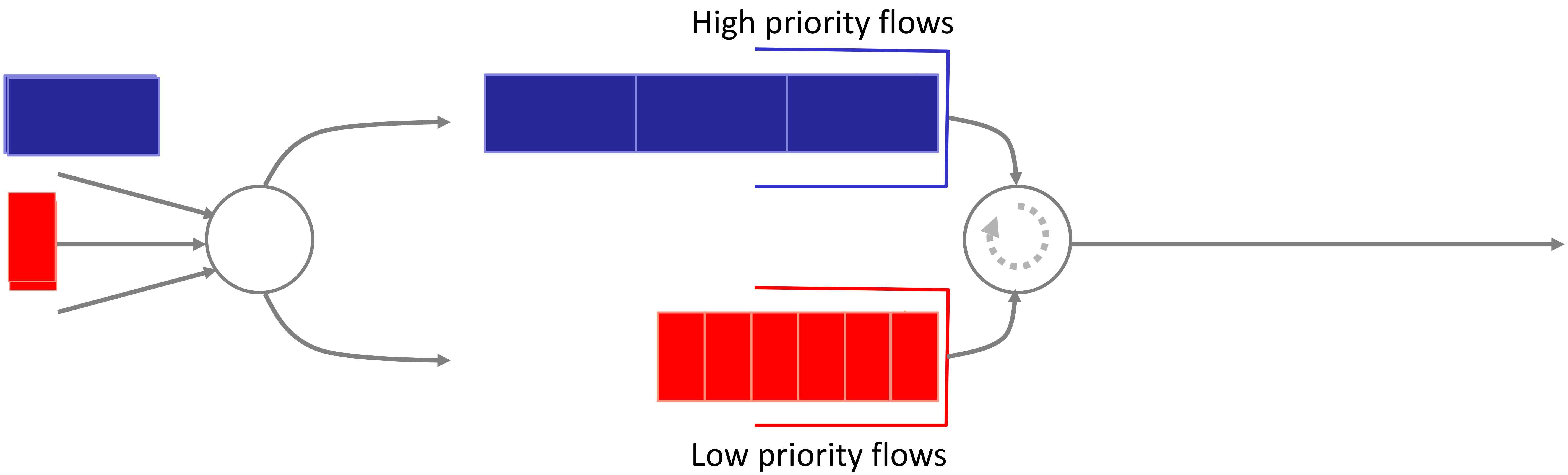
When talking about priorities, it's convenient to talk about a "flow" of packets that all share a common set of attributes. For example:

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

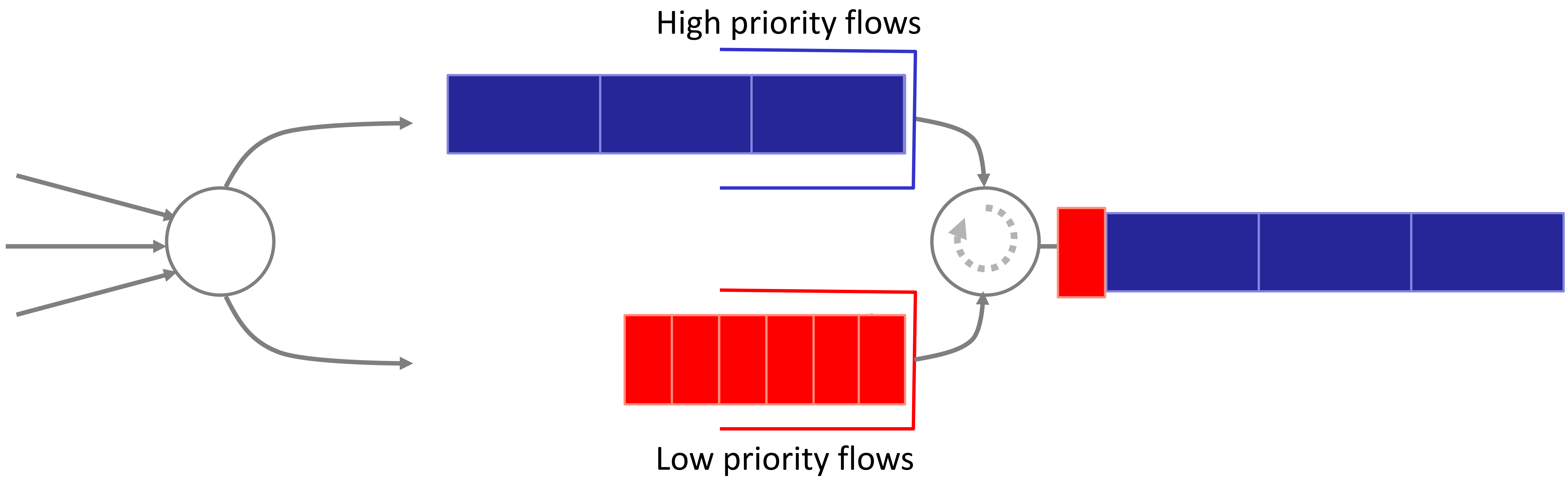
Outline

1. Strict Priorities
2. Weighted Priorities and Rate Guarantees

Strict Priorities



Strict Priorities

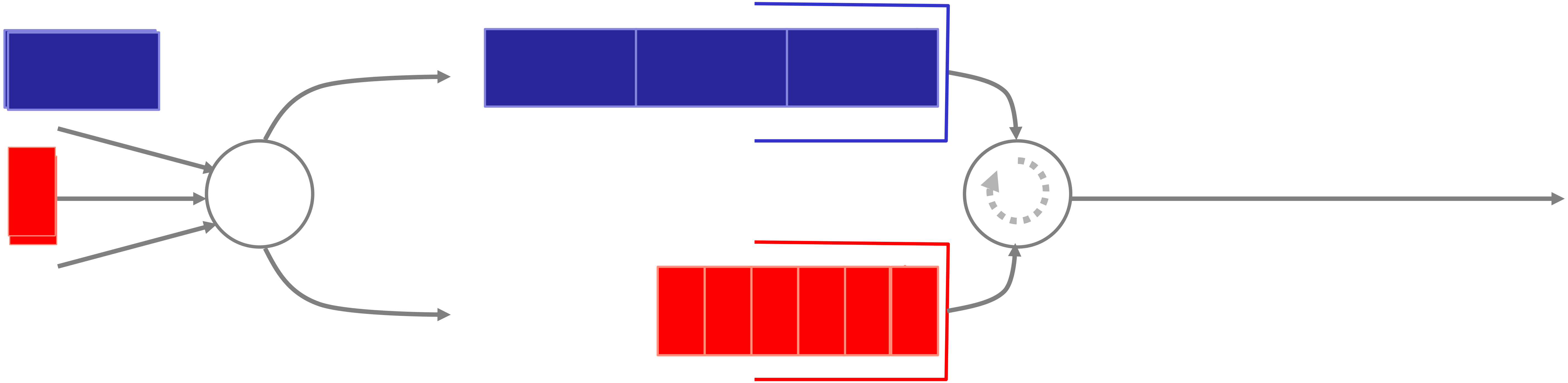


"Strict priorities" means a queue is only served when all the higher priority queues are empty

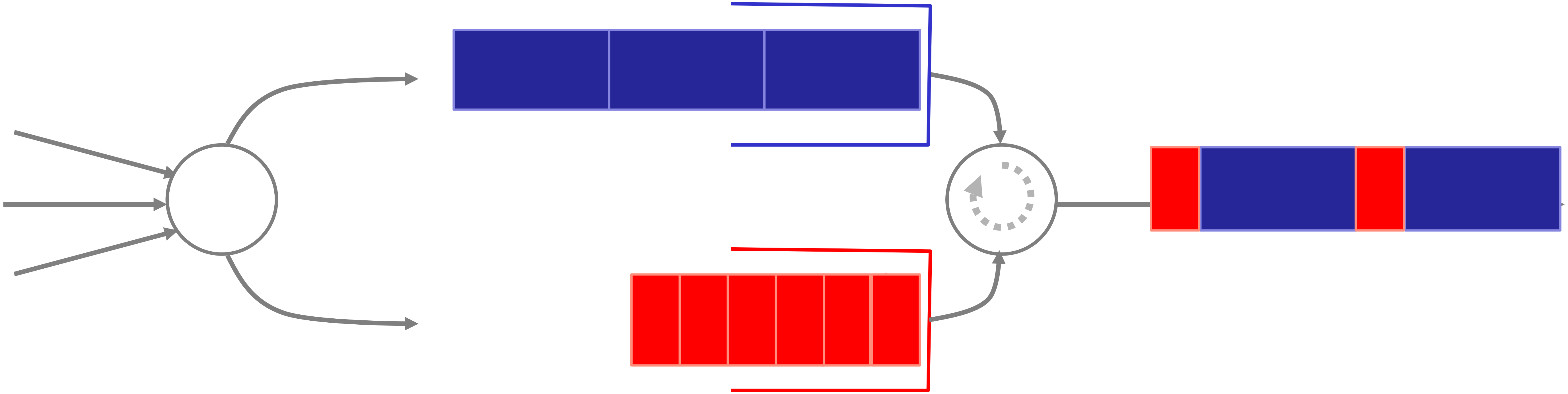
Strict Priorities: Things to bear in mind

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.

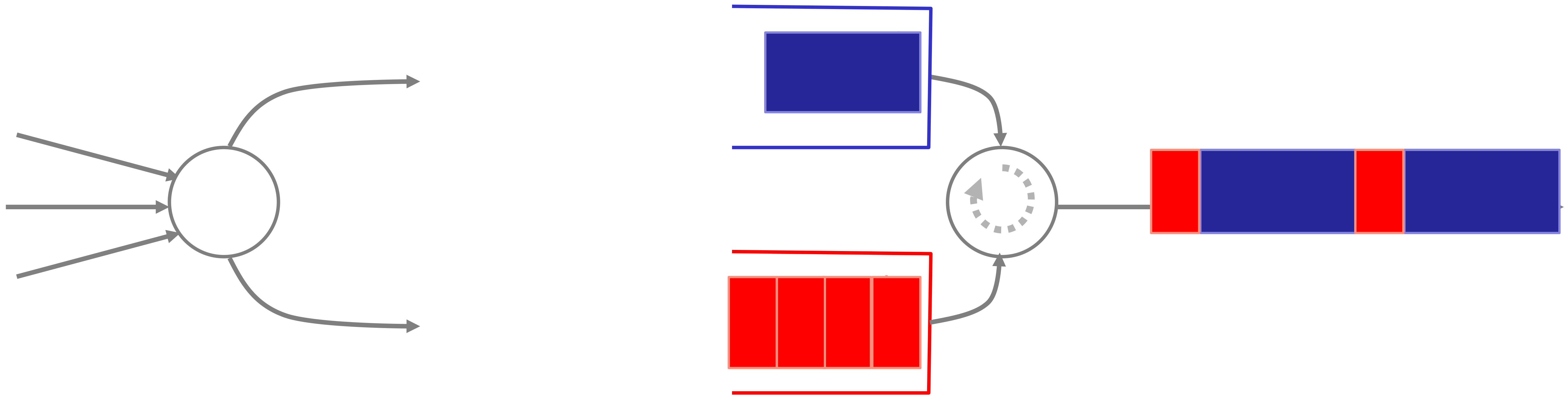
How do I give weighted (instead of strict) priority?



Trying to treat flows equally

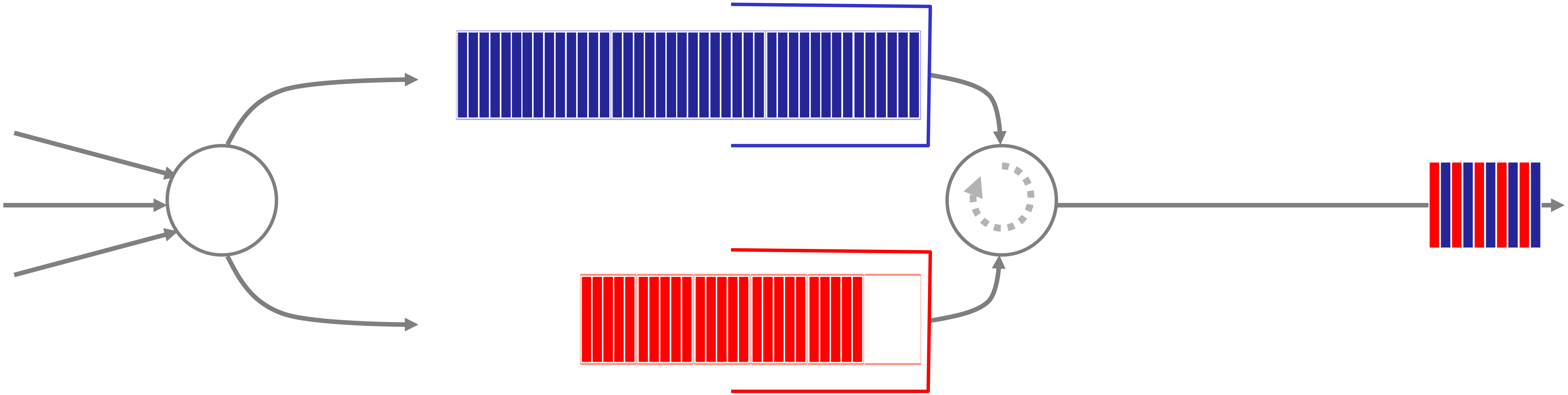


Trying to treat flows equally

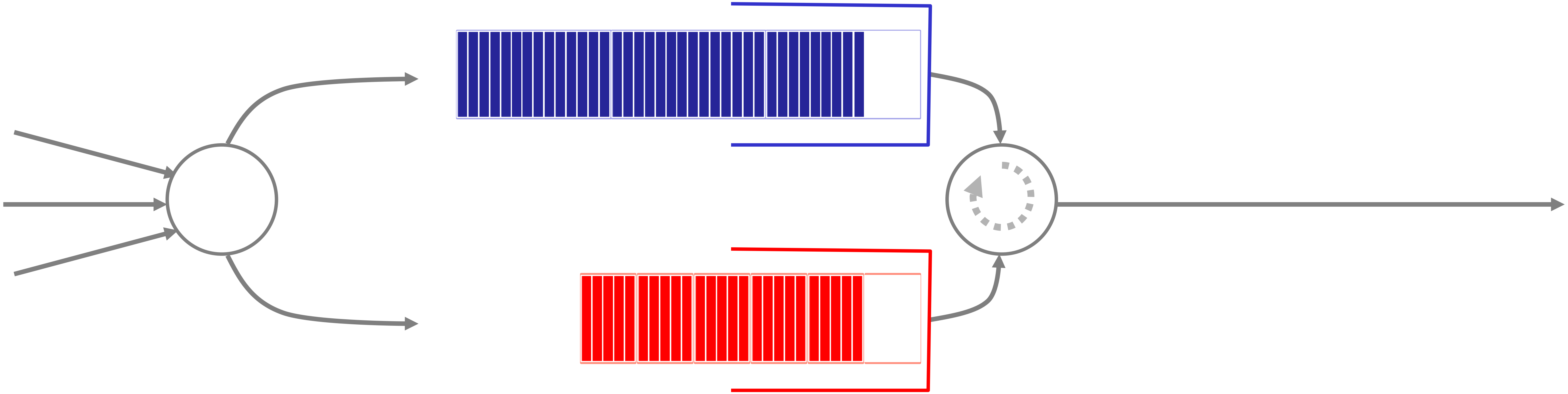


While each flow gets to send at the same packet rate,
the data rate is far from equal.

Scheduling flows bit-by-bit



Scheduling flows bit-by-bit

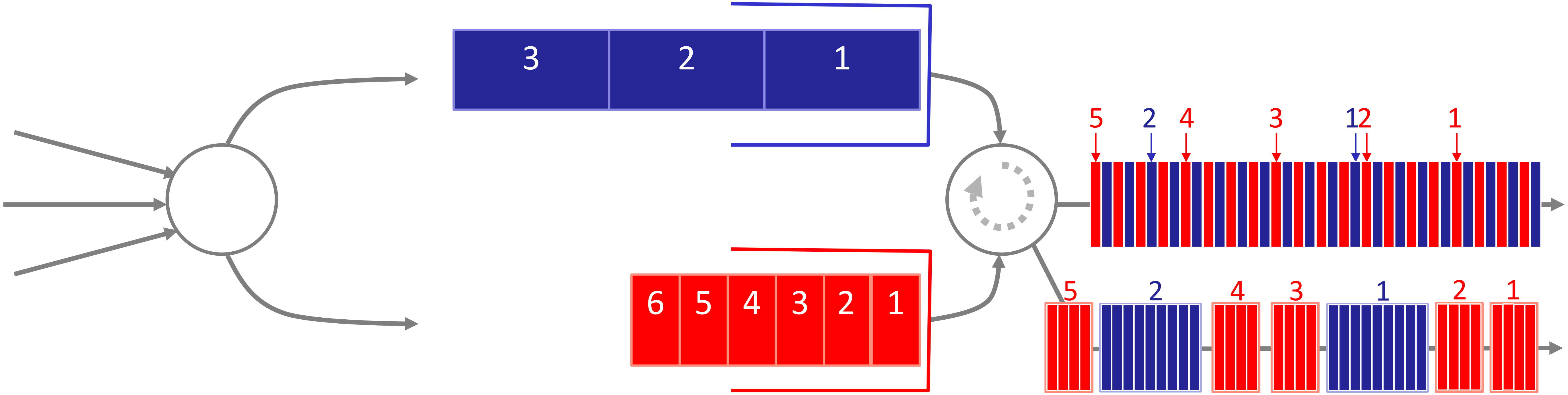


Now each flow gets to send at the same data rate,
but we no longer have “packet switching”.

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!

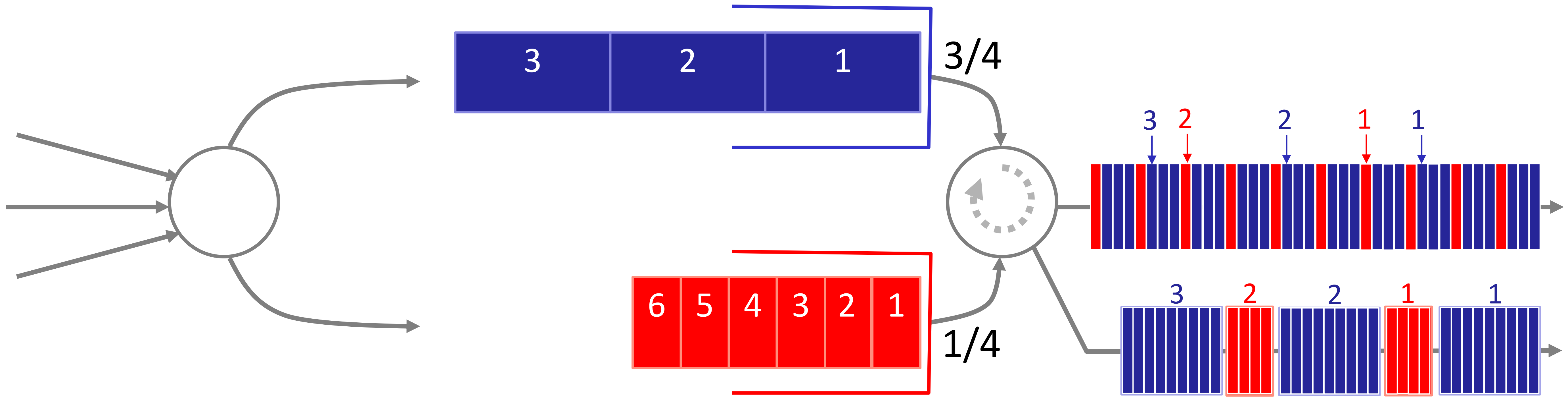
1. It can be proved that the departure time of a packet with Fair Queueing is no more than L_{max}/R seconds later than if it was scheduled bit-by-bit. Where L_{max} is the maximum length packet and R is the data rate of the outgoing link.
2. In the limit, the two flows receive equal share of the data 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 <u>weighted</u> fair share.

Weighted Fair Queueing

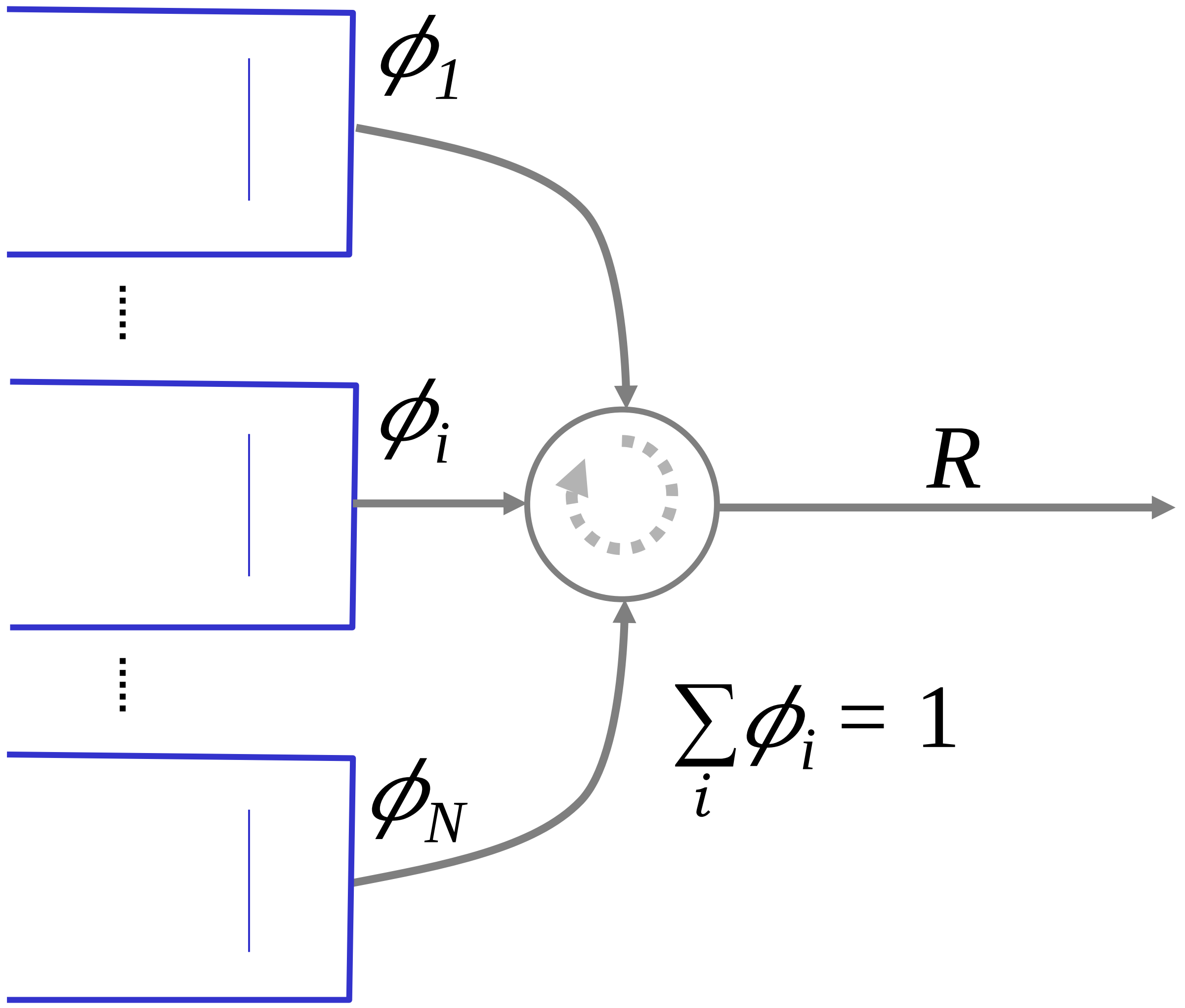


As before, packets are sent in the order they would complete in the bit-by-bit scheme.

Weighted Fair Queueing (WFQ)

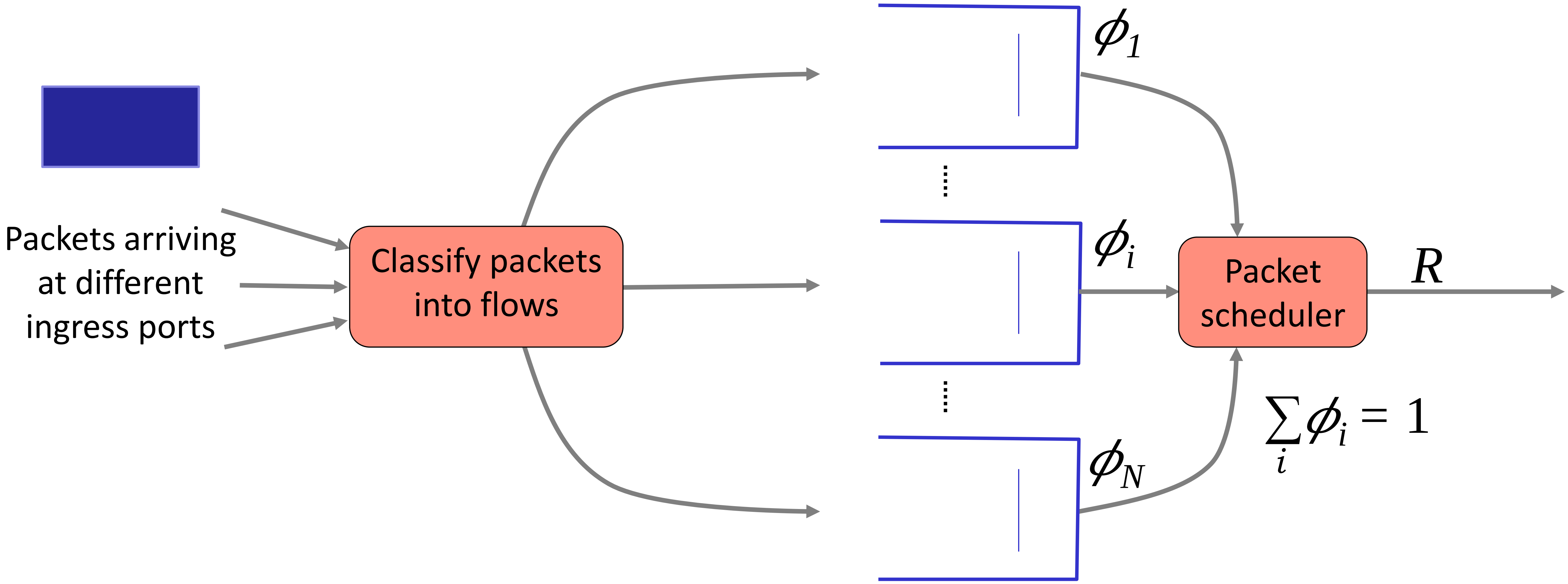
For any number of flows, and any mix of packet sizes:

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 $\phi_i R$

Weighted Fair Queueing (WFQ)



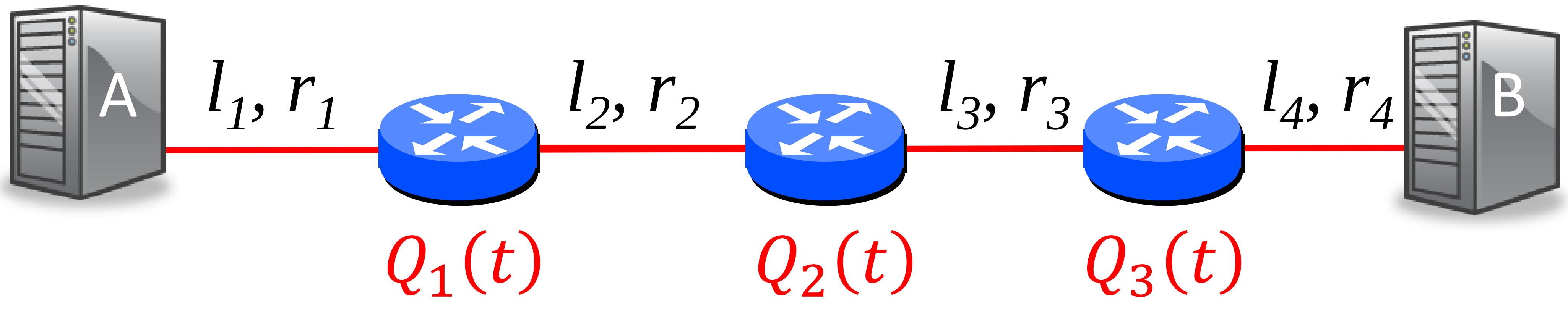
Flow i is guaranteed to receive at least rate $\phi_i R$

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.

Can we guarantee the delay of a packet across a network of packet switches?

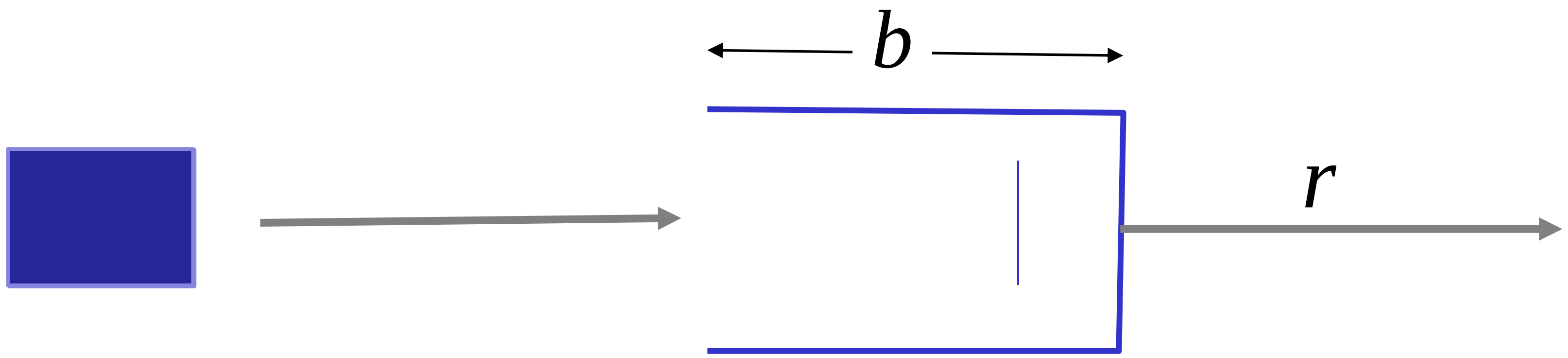
Delay guarantees: Intuition



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.

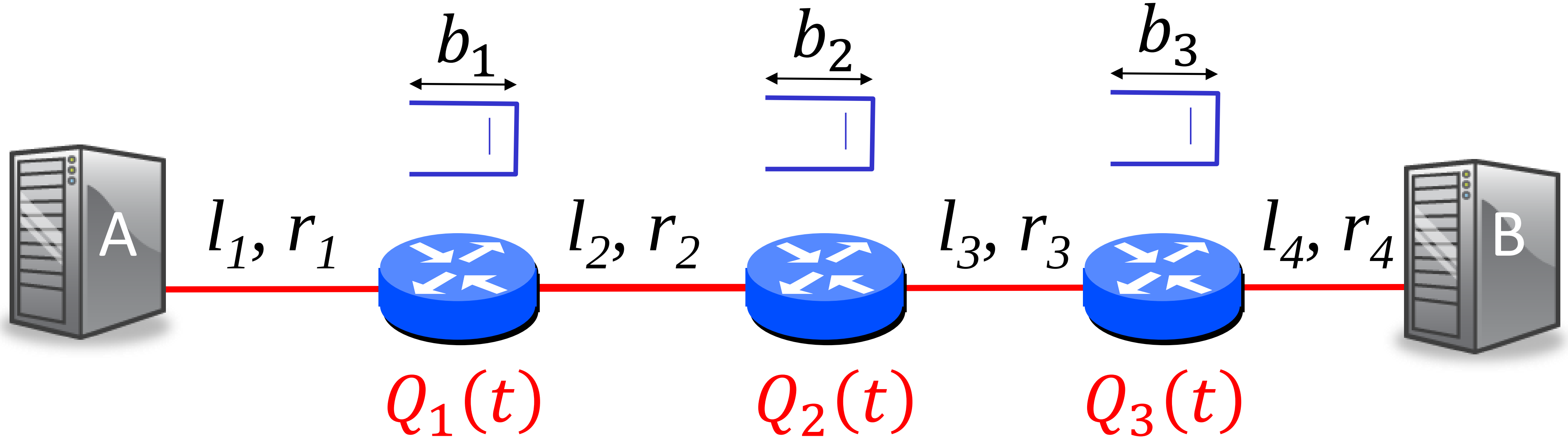
Upper bound on $Q(t)$



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

<u>Example</u>: 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 ${}^{10^6}/_{10^9} = 1\text{ms}$.

Delay guarantees: Intuition



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

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

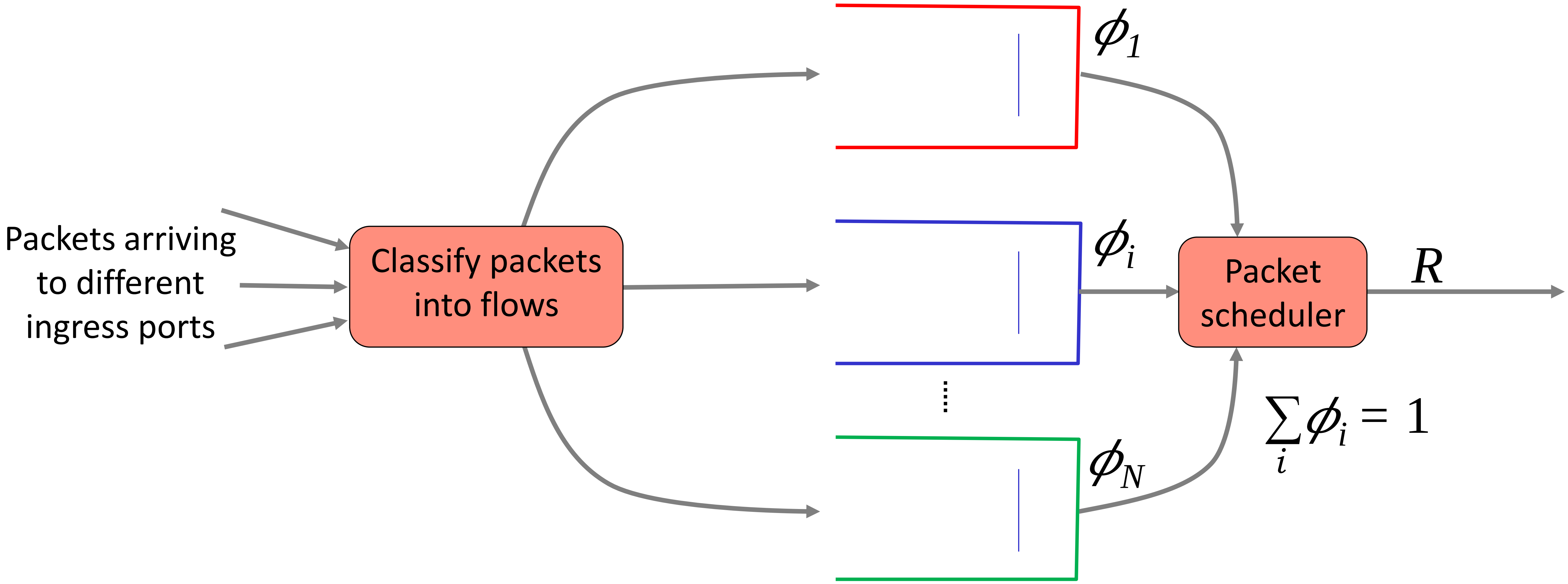
Why this is only an intuition…

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.

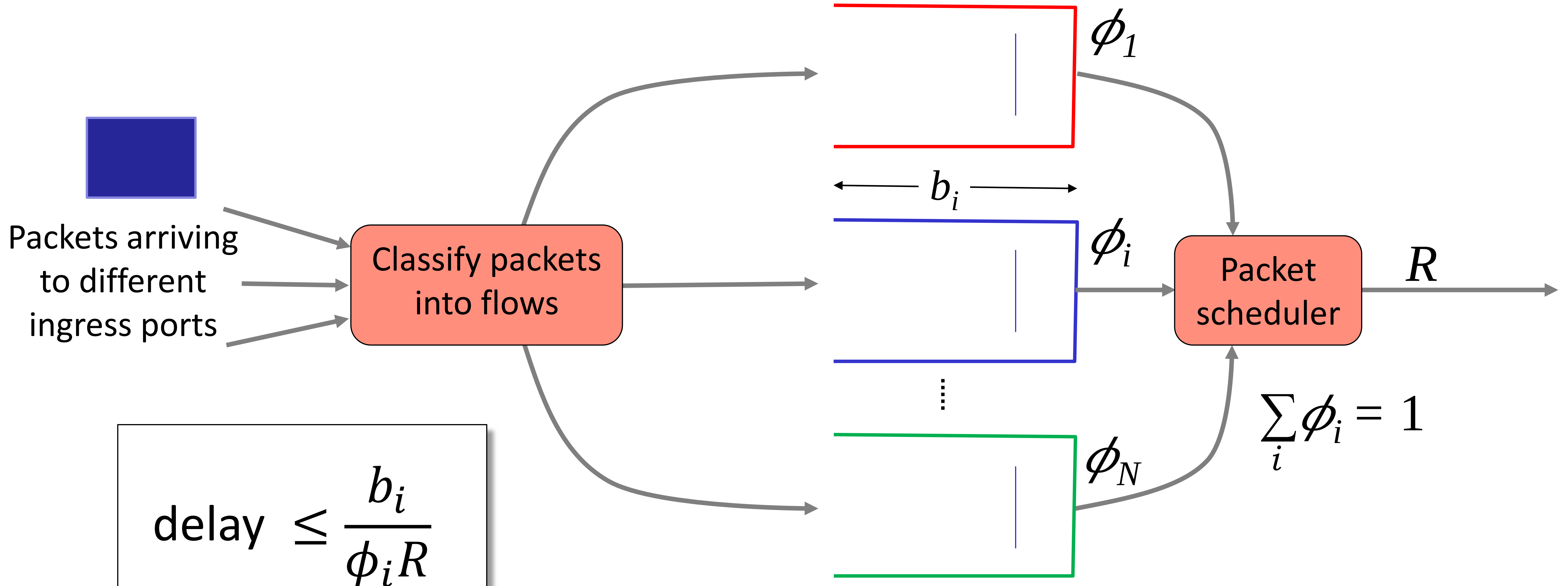
Q: How can we give an upper bound on delay to each individual flow?

Weighted Fair Queueing (WFQ)



Flow i is guaranteed to receive at least rate $\phi_i R$

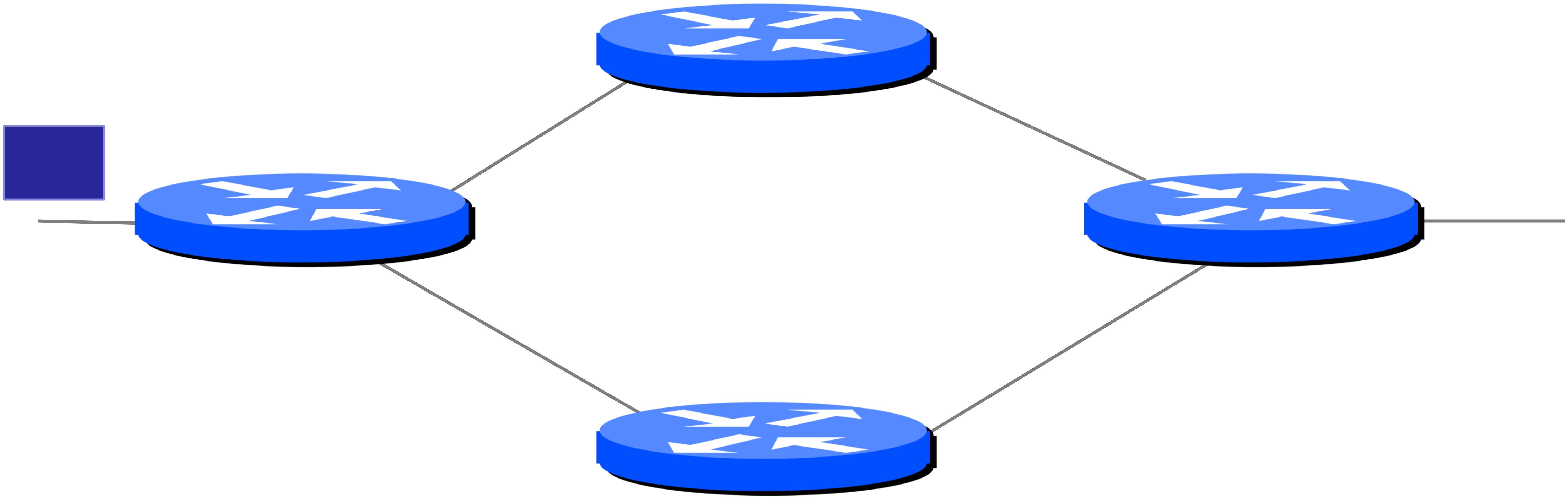
Weighted Fair Queueing (WFQ)



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

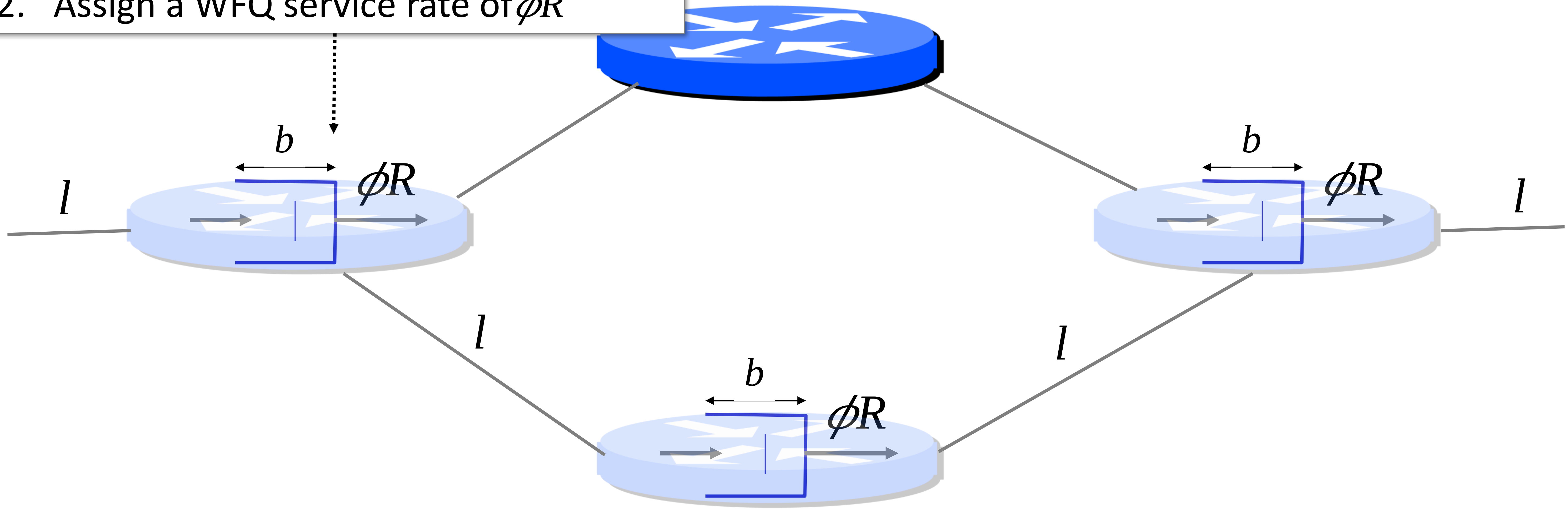
Flow i is guaranteed to receive at least rate $\phi_i R$

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 ϕR



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

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

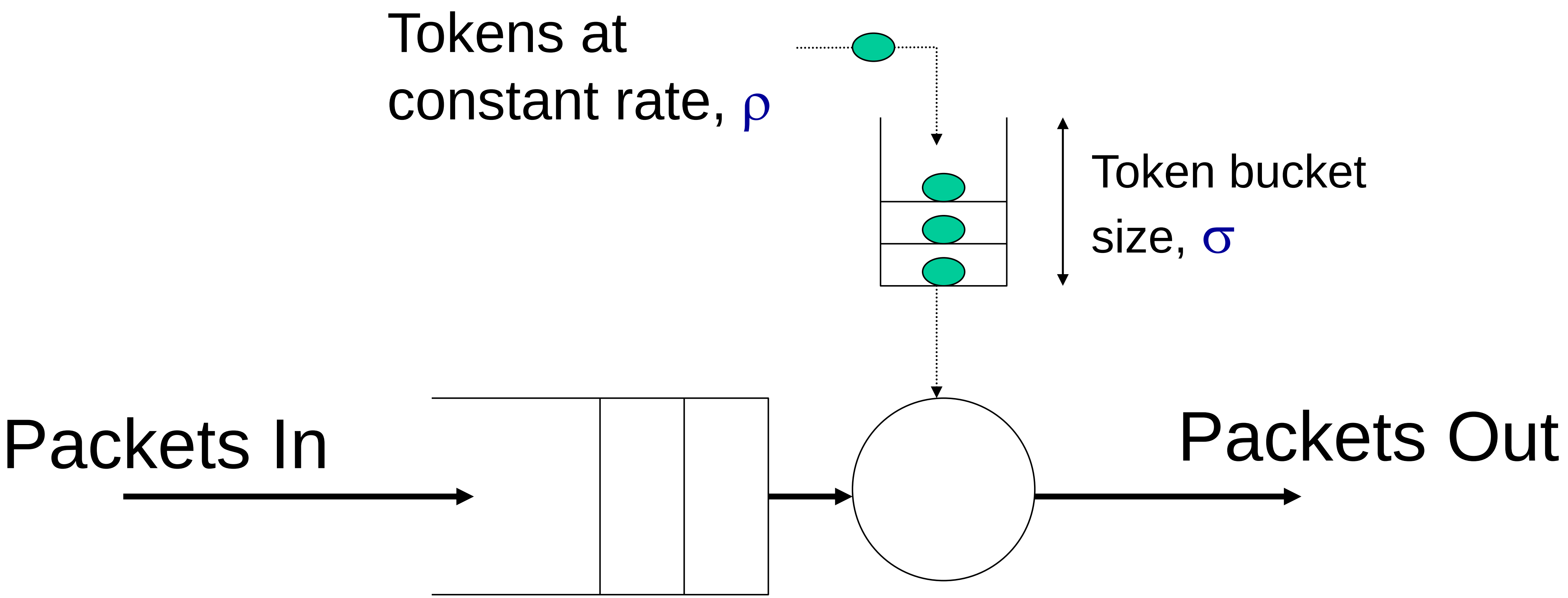
1. If the packets are far apart, then the queues drain the first packet before the second one arrives. All is good, and the delay equation holds.

2. If the packets are close together in a "burst", then they can arrive faster than ϕR 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 <u>all</u> packets, then we need to deal with bursts. How?

The leaky bucket regulator

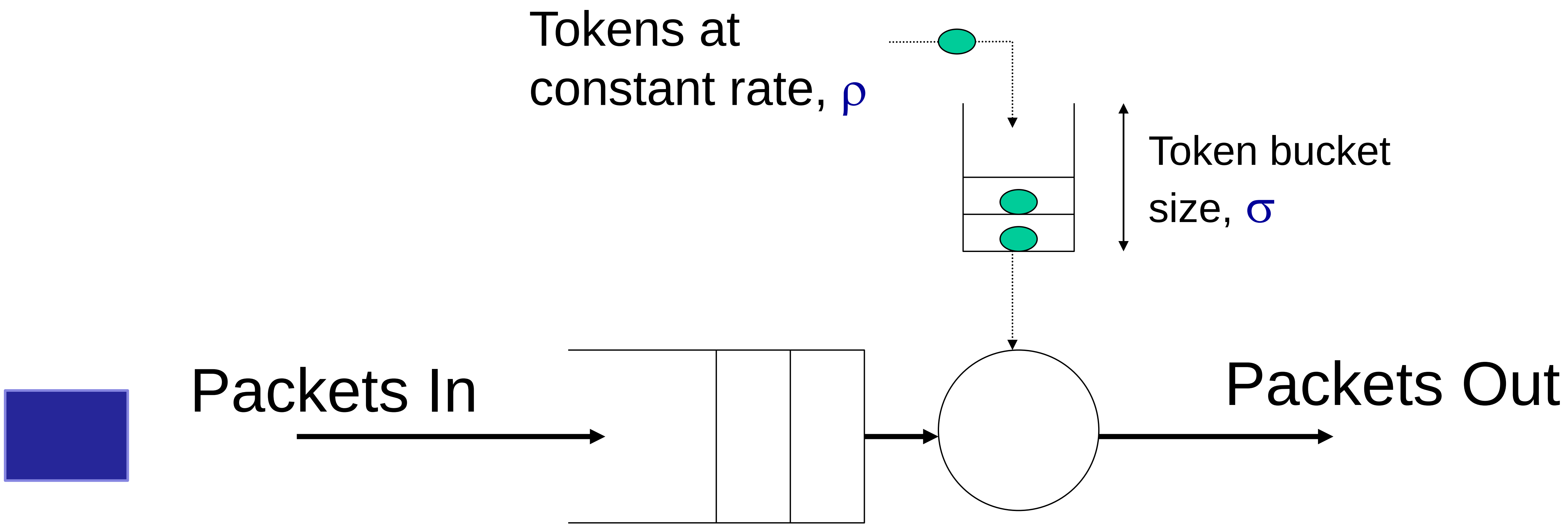
Limiting the "burstiness"



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

The leaky bucket regulator

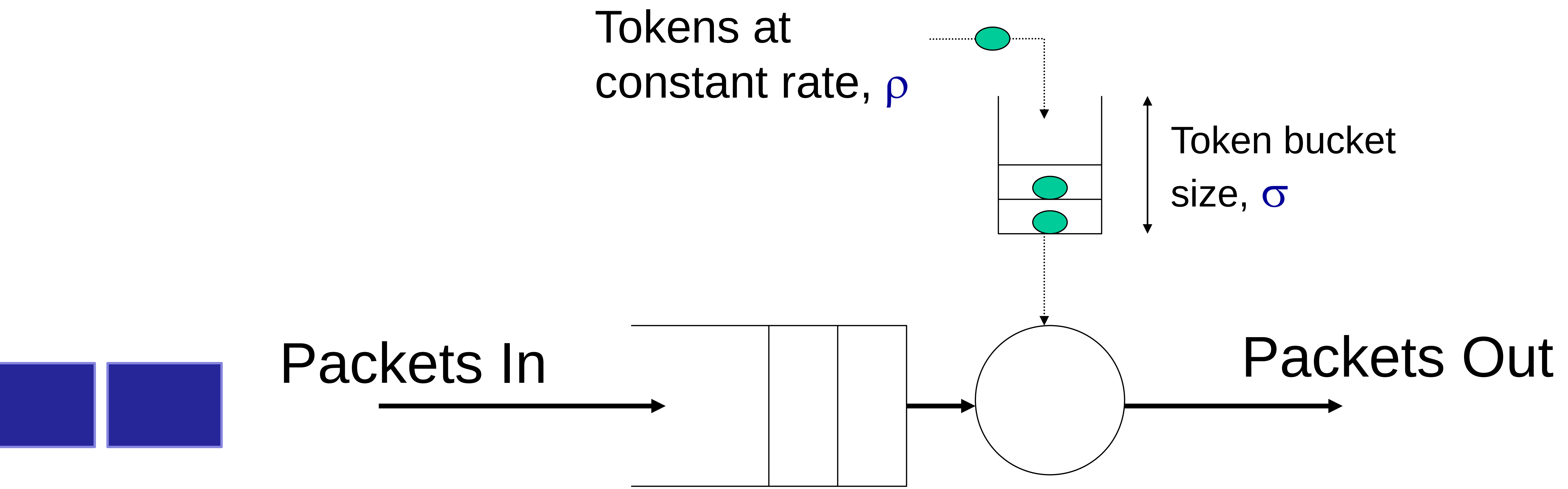
Limiting the “burstiness”



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

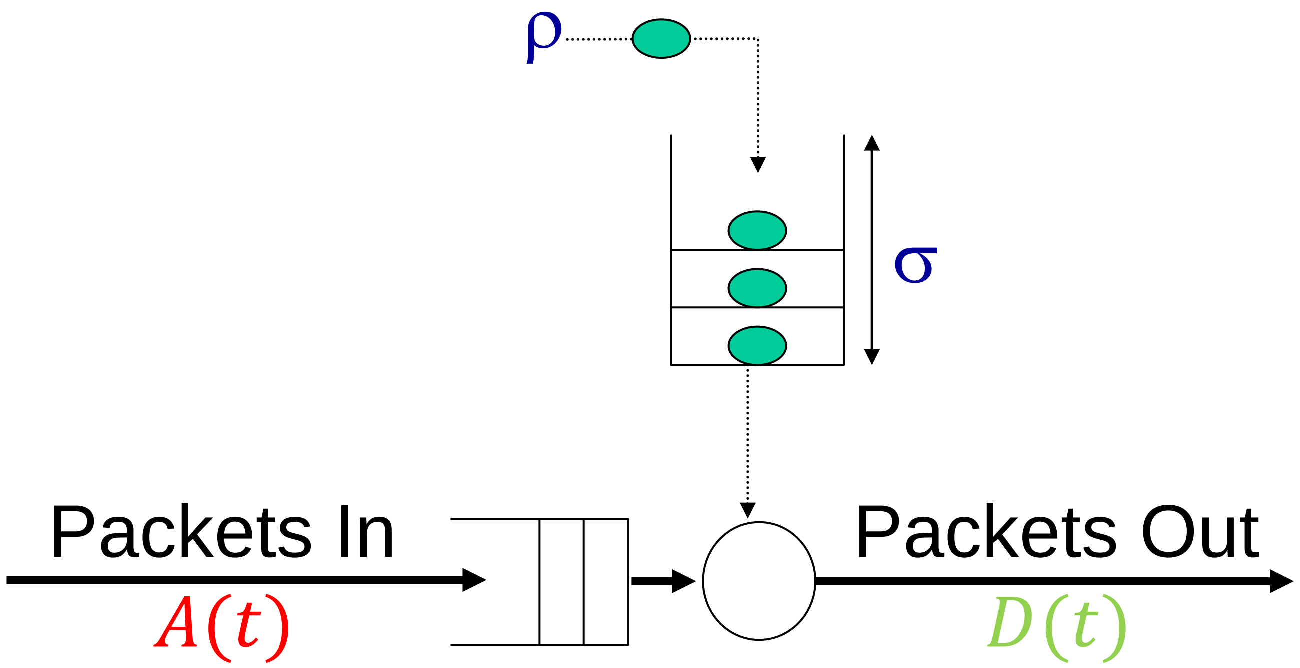
The leaky bucket regulator

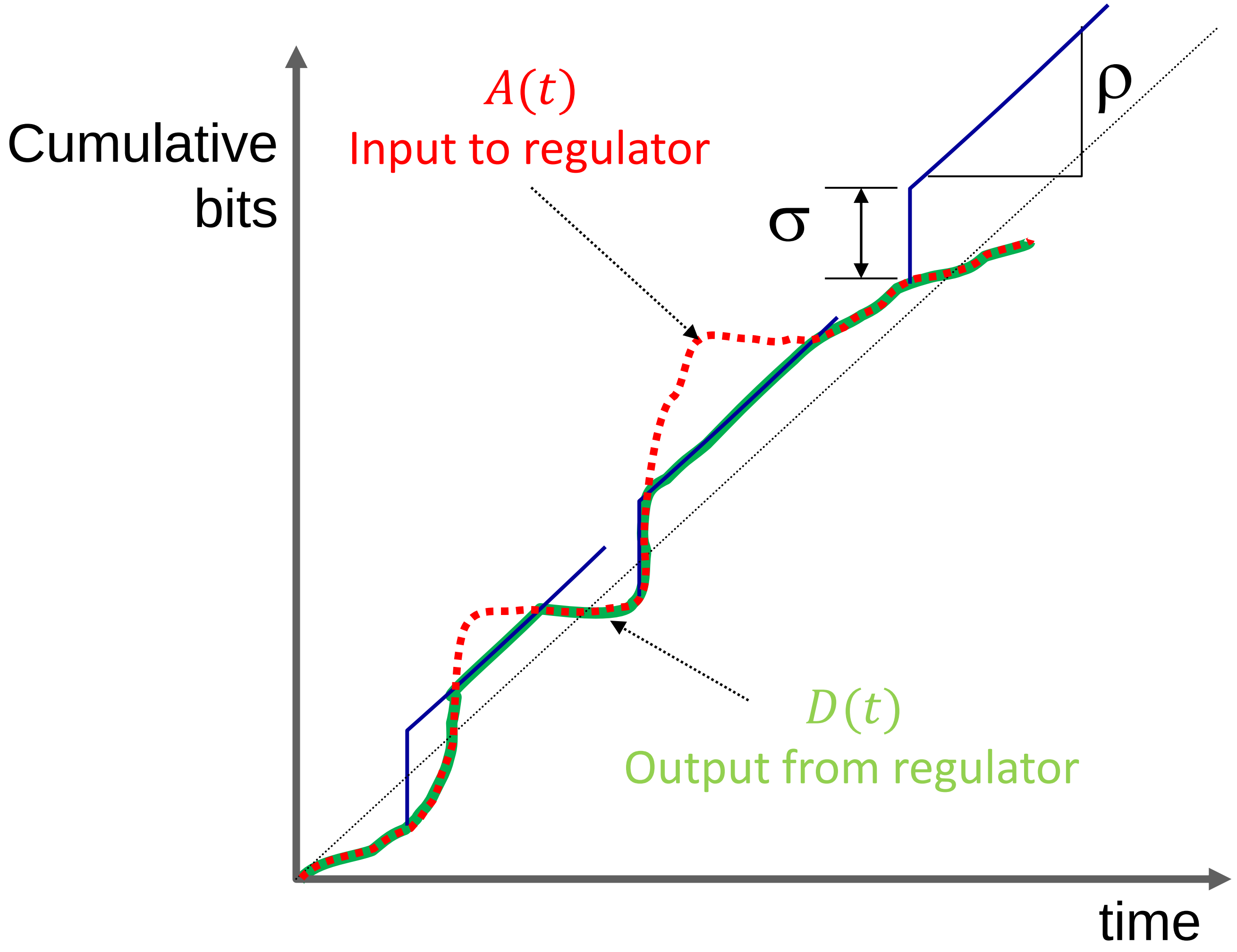
Limiting the "burstiness"



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

The leaky bucket regulator



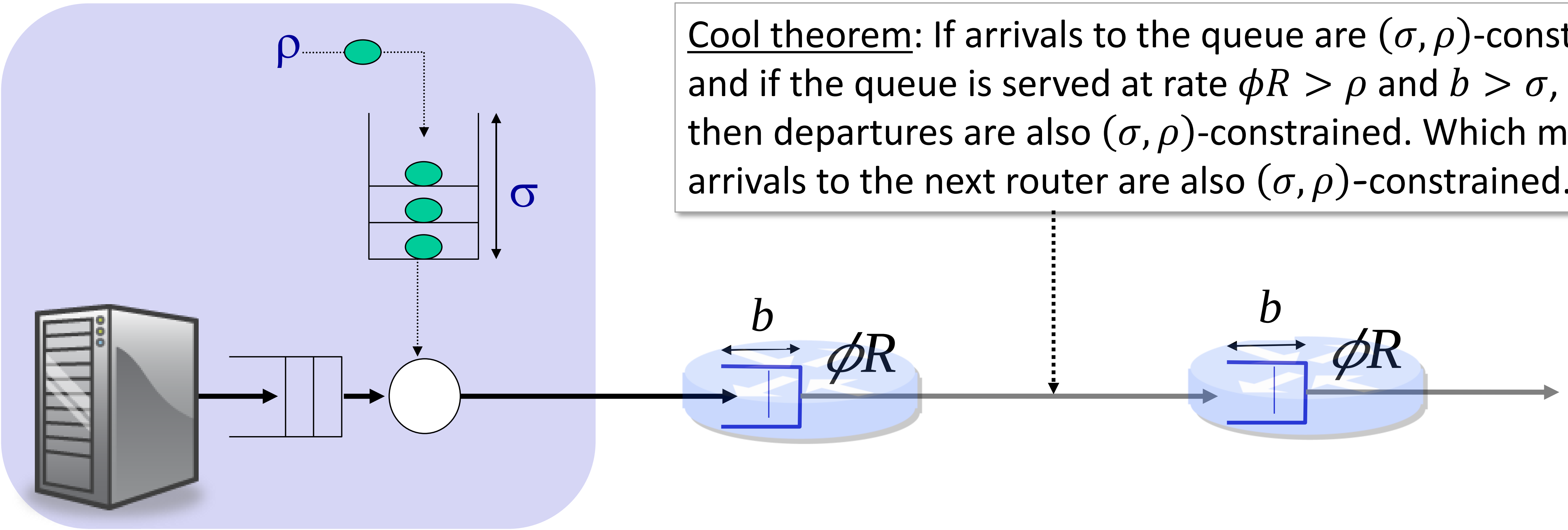


Number of bits that can be sent in <u>any</u> period of length t is bounded by: $\sigma + \rho t$

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

The leaky bucket regulator

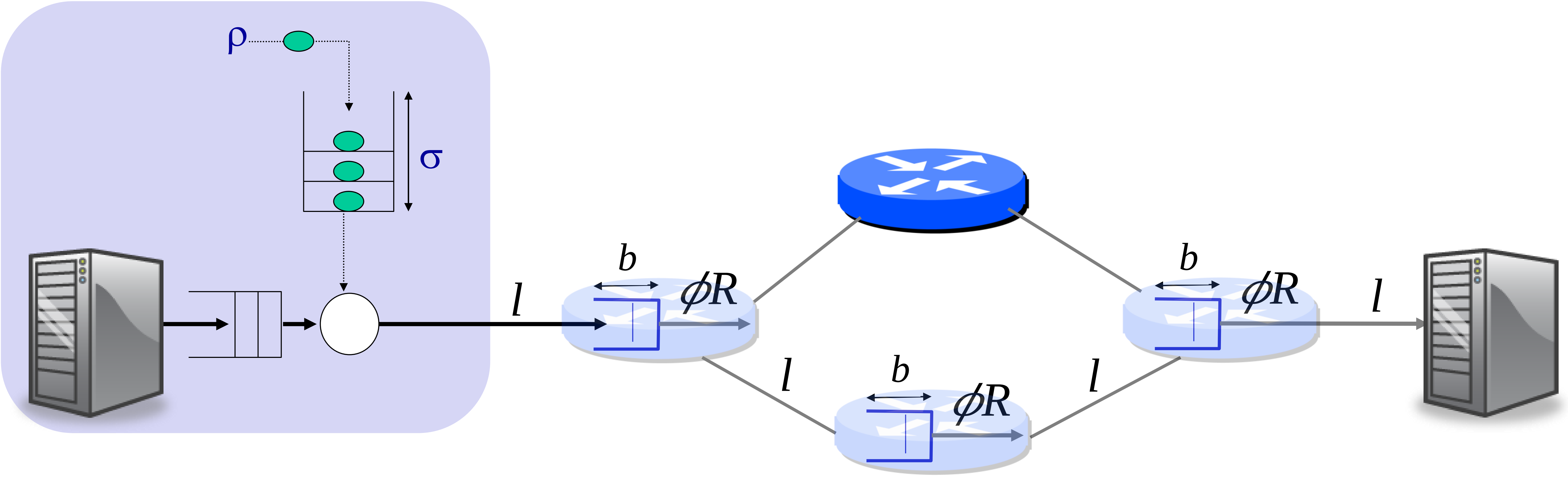
Limiting the "burstiness"



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

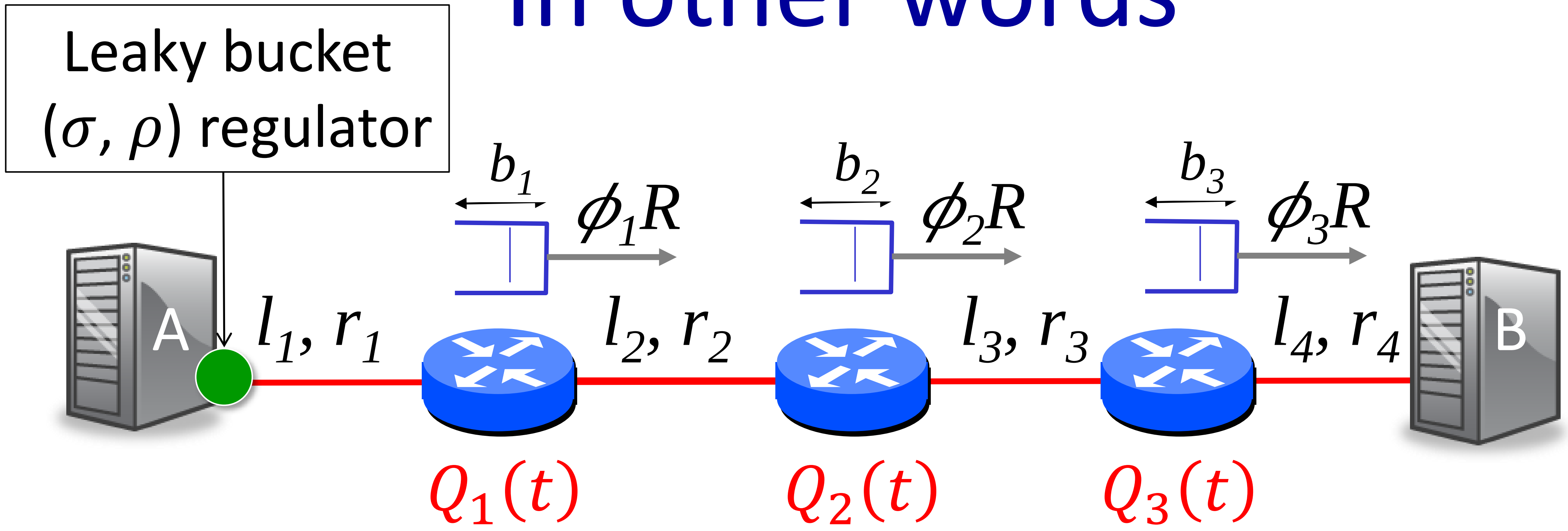
If $\phi R > \rho$ and $b > \sigma$ then delay through the first router for all packets in the flow $\leq \frac{b}{\phi R}$

Putting it all together



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

In other words



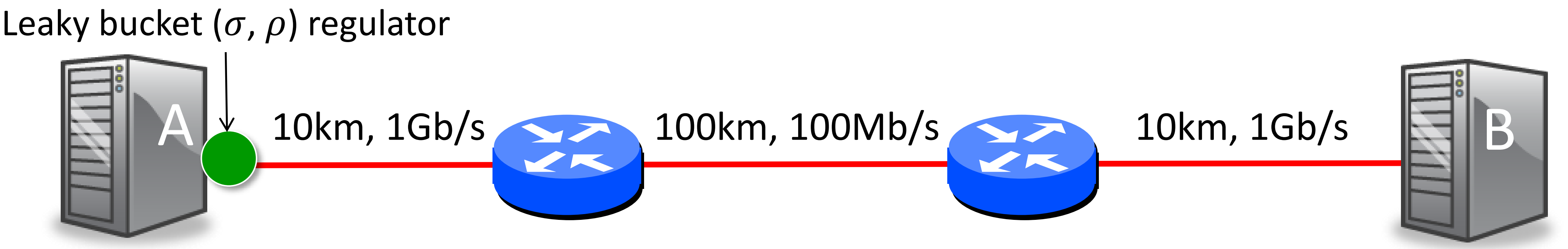
If we set $b_i > \sigma$, and $\phi_i R > \rho$ then

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

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

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 $2ms\ \times$ 10Mb/s $=$ 20,000bits (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 $\phi_i R \geq 10Mb/s$ 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.