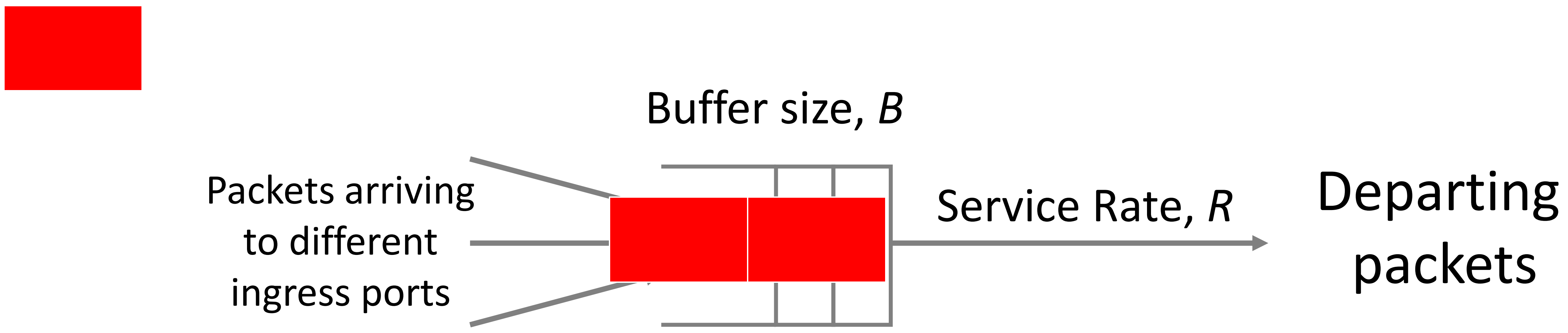
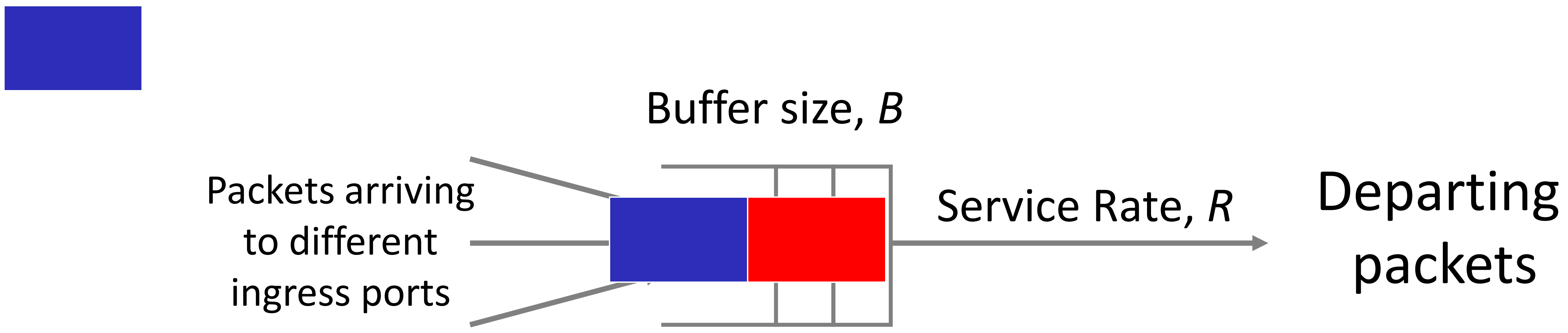
Packet Switching

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

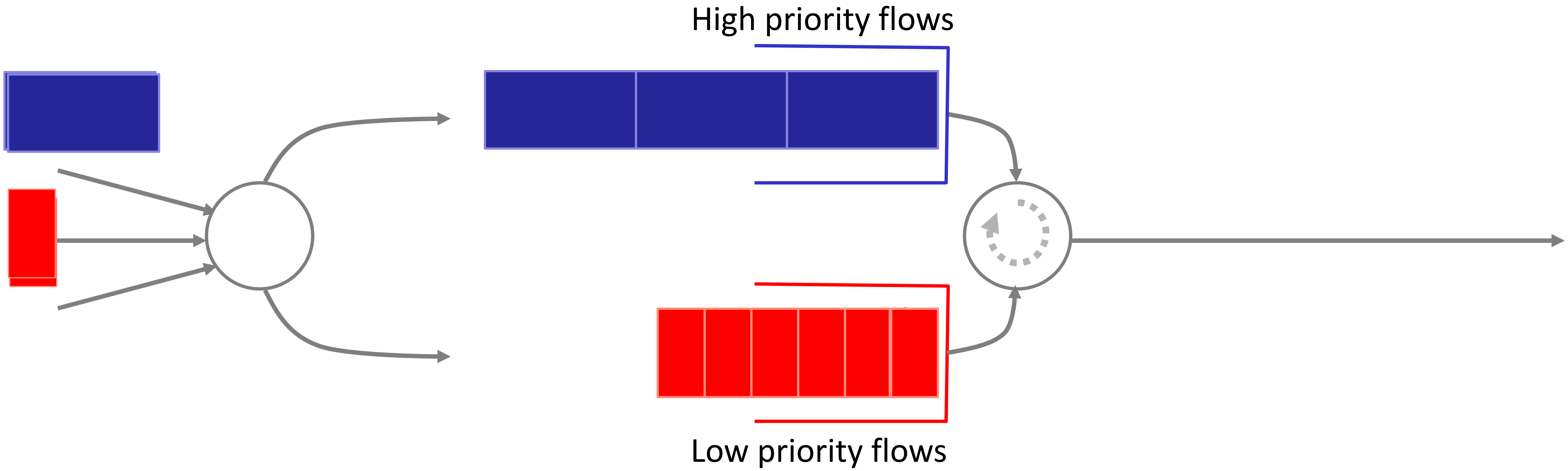
When talking about priorities, it's convenient to talk about a "flow" of packets that all share a common set of attributes. For example:

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

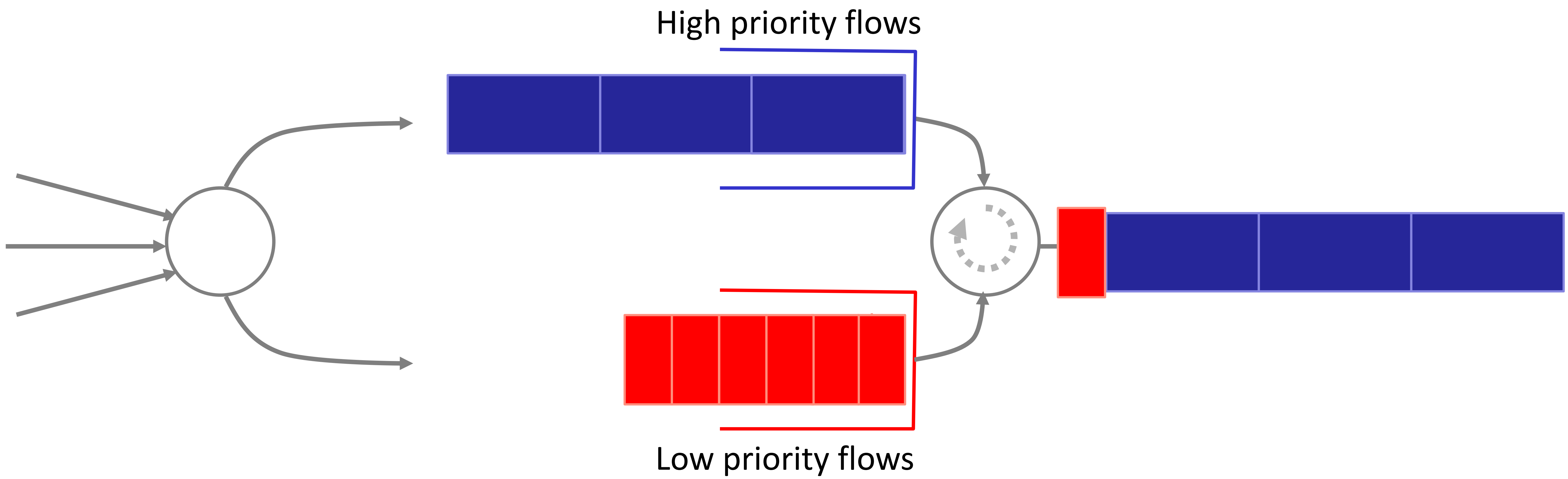
Outline of what's coming up next…

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

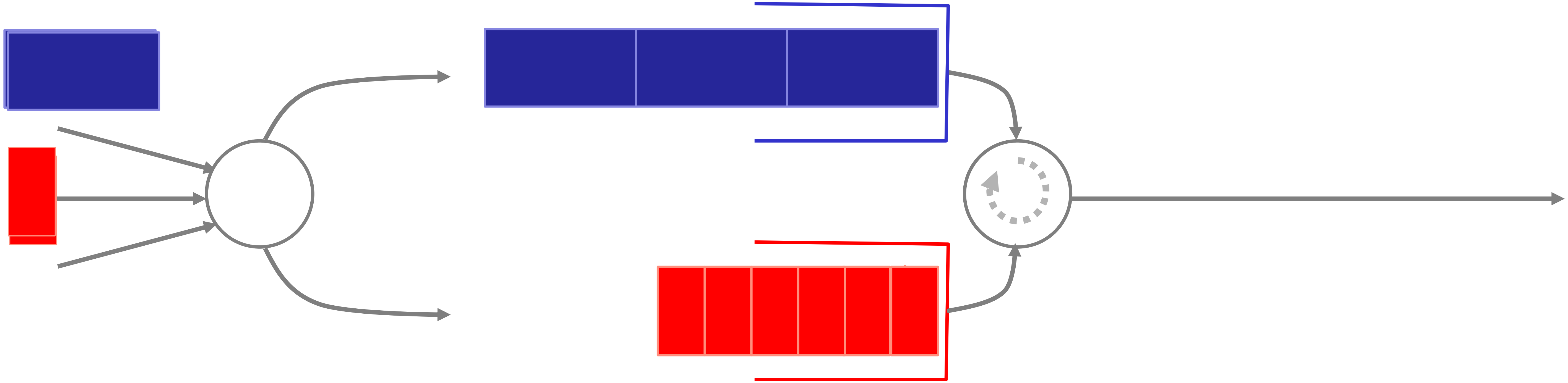


"Strict priorities" means a queue is only served when all the higher priority queues are empty

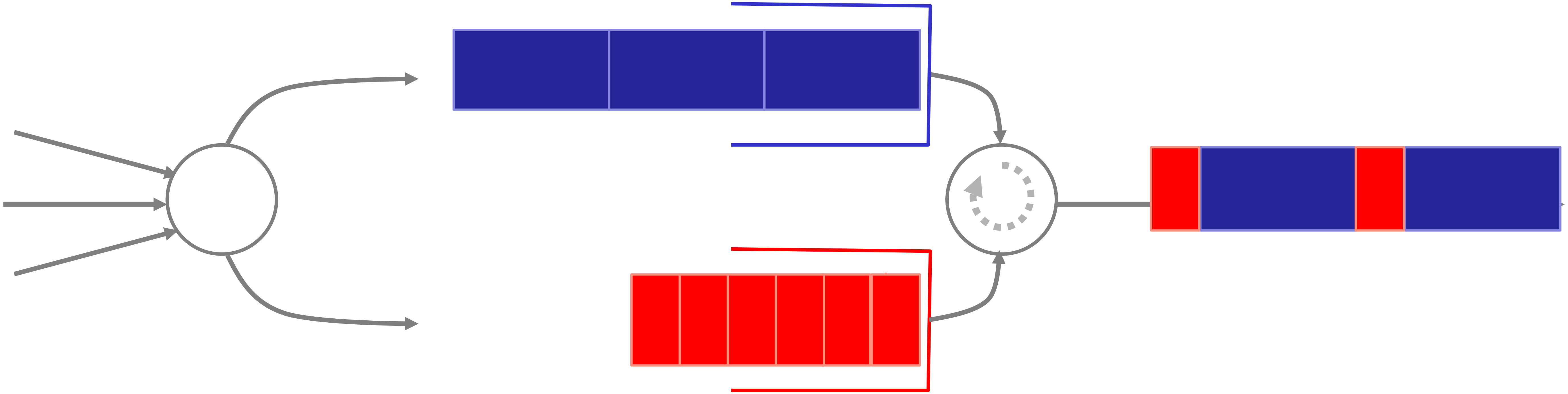
Strict Priorities: Things to bear in mind

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.

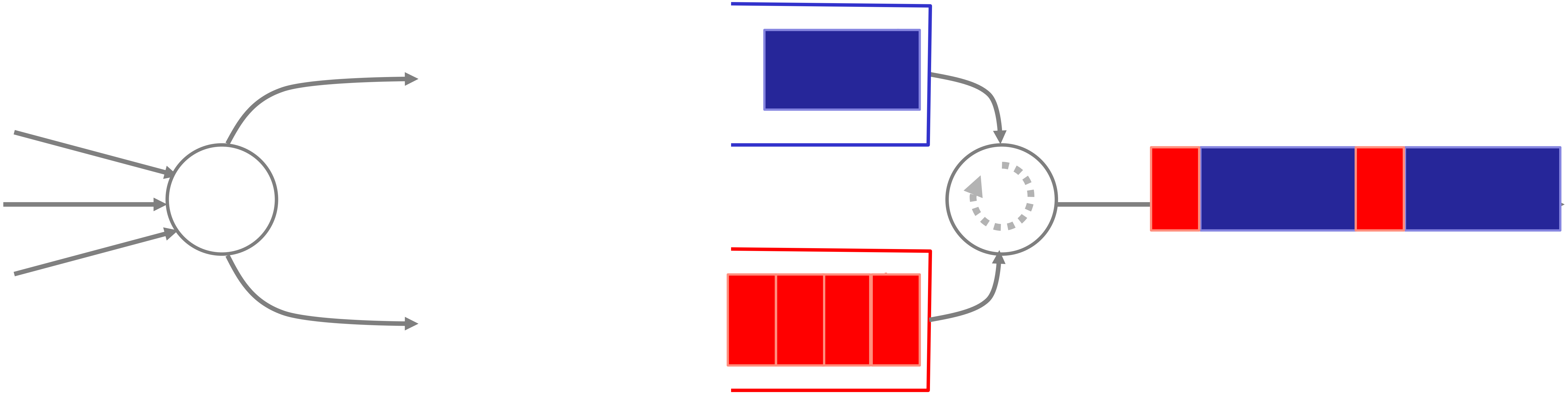
How do I give weighted (instead of strict) priority?



Trying to treat flows equally

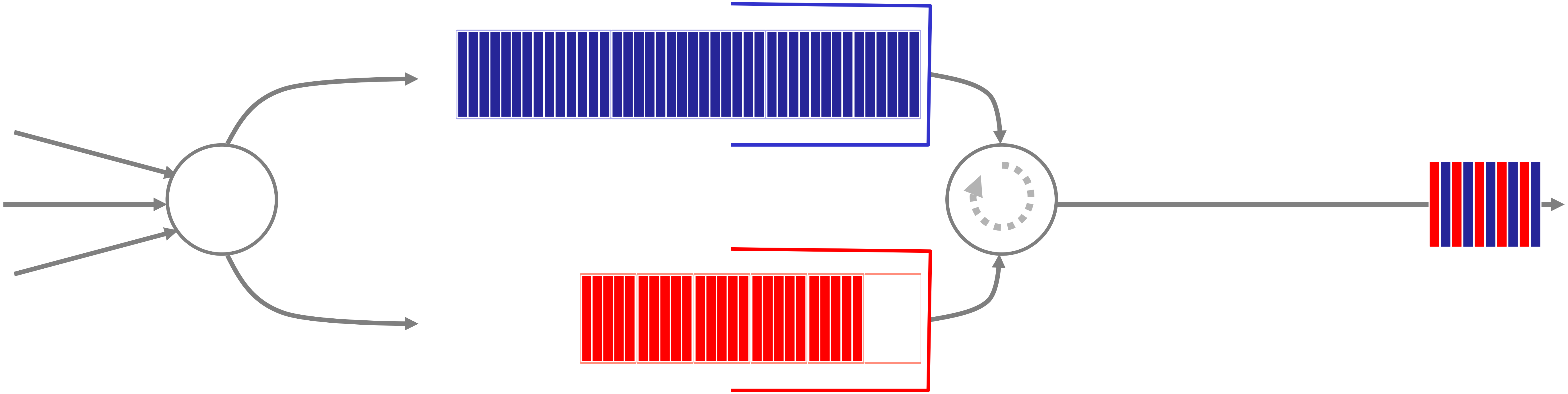


Trying to treat flows equally

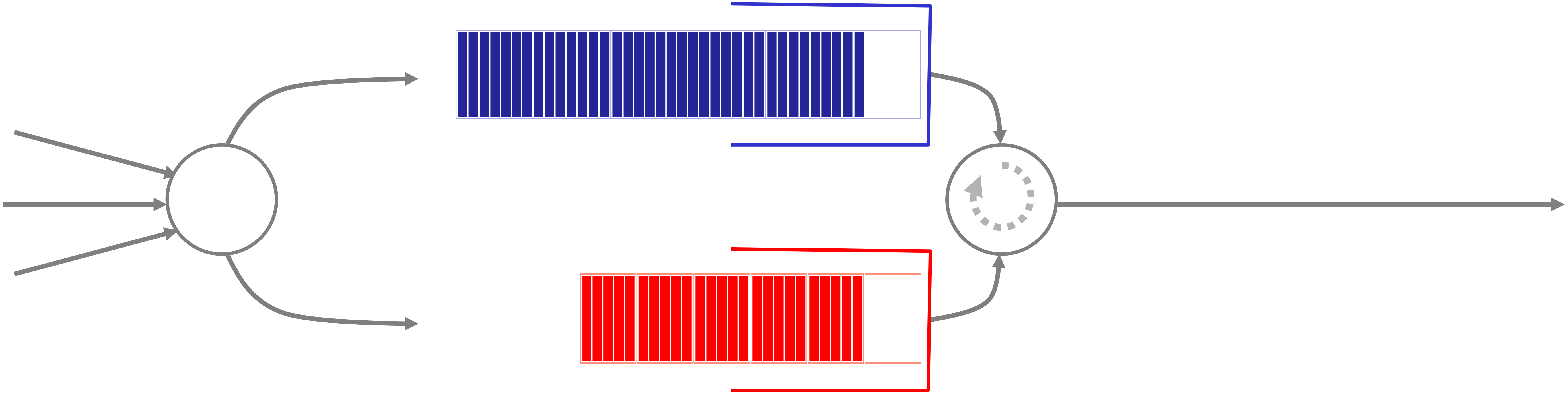


While each flow gets to send at the same packet rate,
the data rate is far from equal.

Scheduling flows bit-by-bit



Scheduling flows bit-by-bit

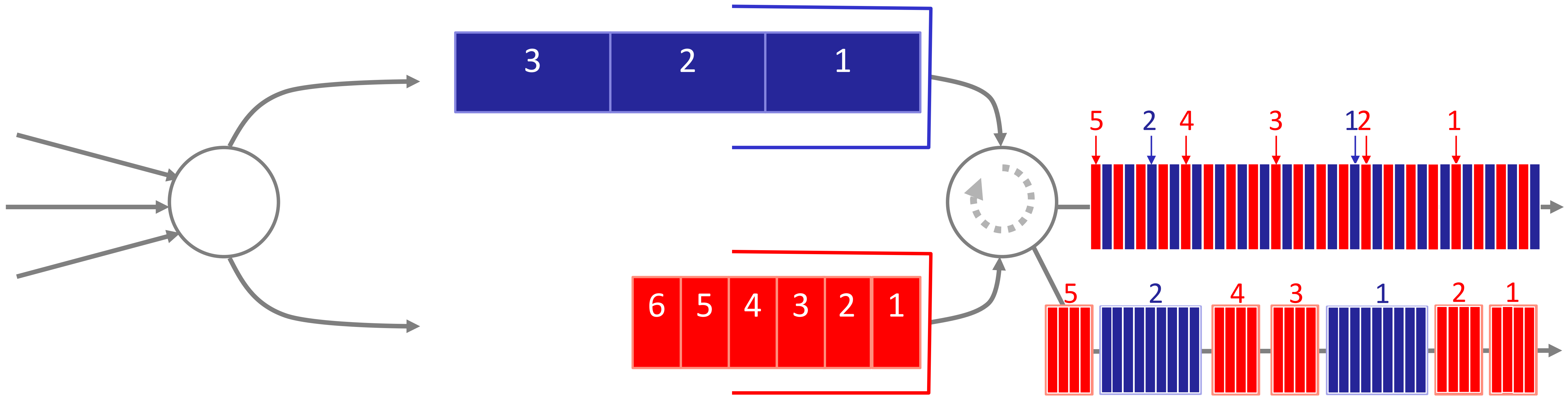


Now each flow gets to send at the same data rate,
but we no longer have “packet switching”.

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!

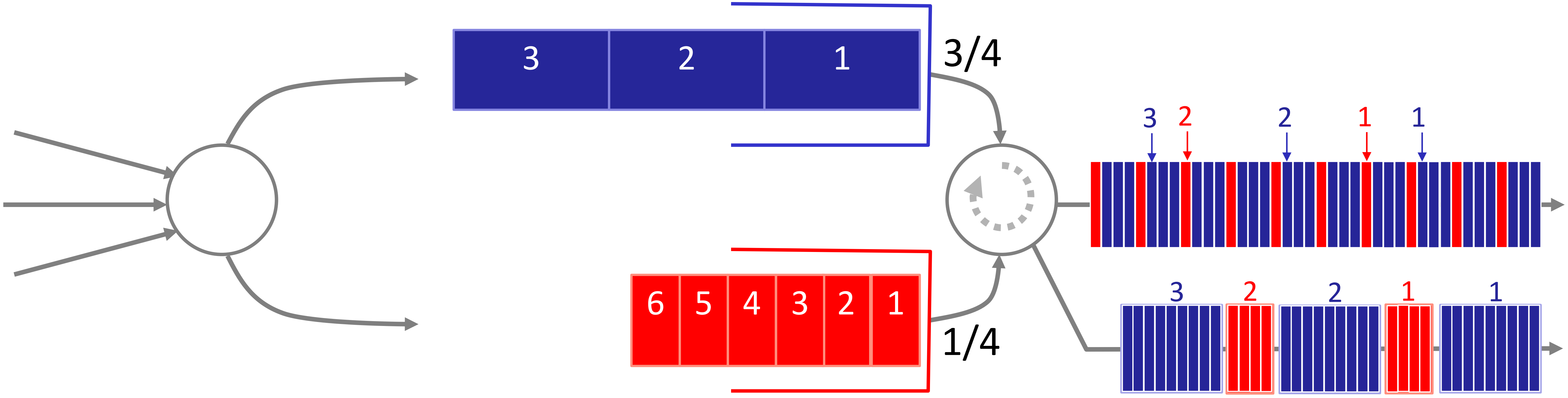
1. It can be proved that the departure time of a packet with Fair Queueing is no more than L_{max}/R seconds later than if it was scheduled bit-by-bit, where L_{max} is the maximum length packet and R is the data rate of the egress link.
2. In the limit, the two flows receive equal share of the data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

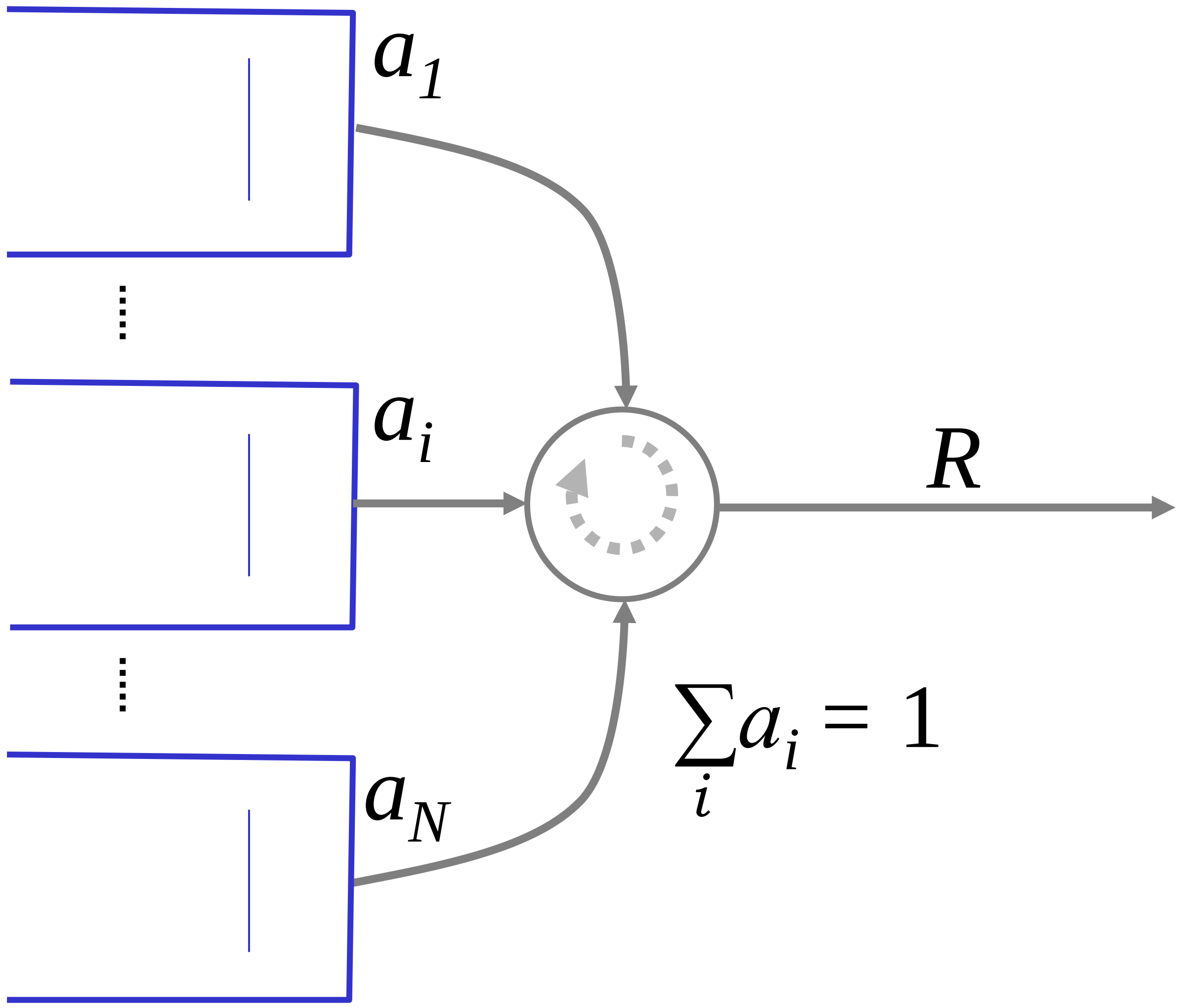


As before, packets are sent in the order they would complete in the bit-by-bit scheme.

Weighted Fair Queueing (WFQ)

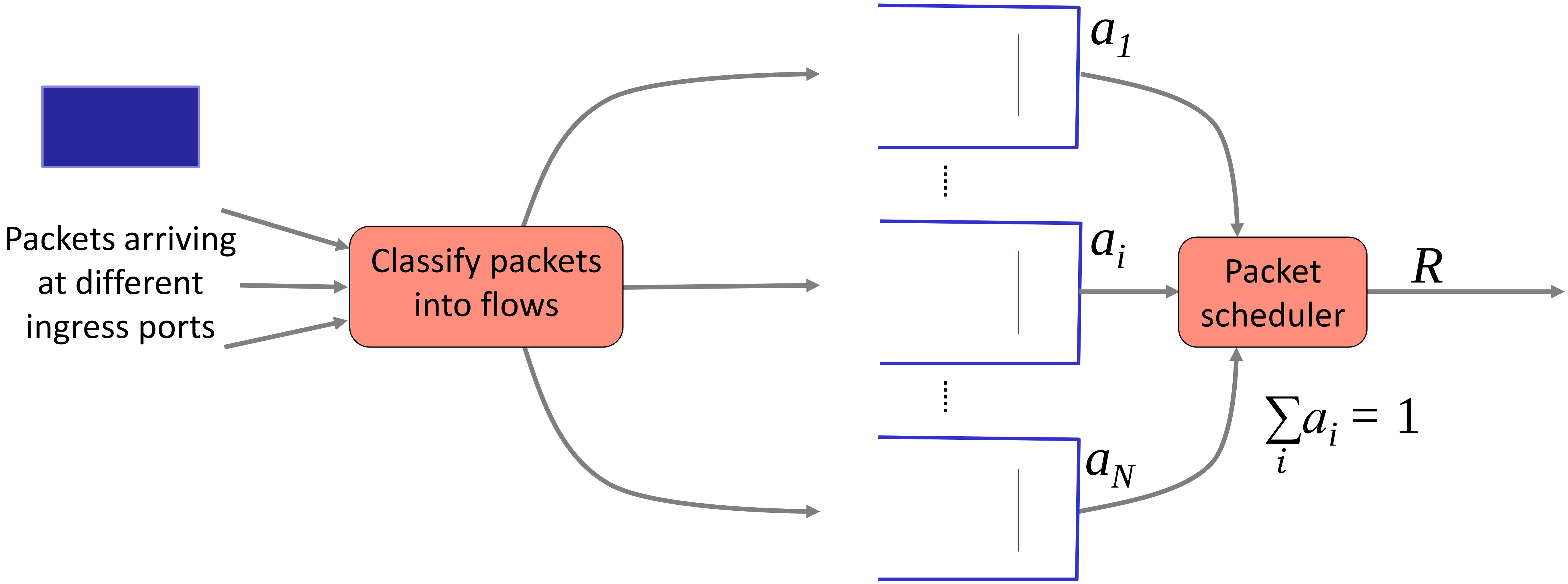
For any number of flows, and any mix of packet sizes:

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 $a_i R$

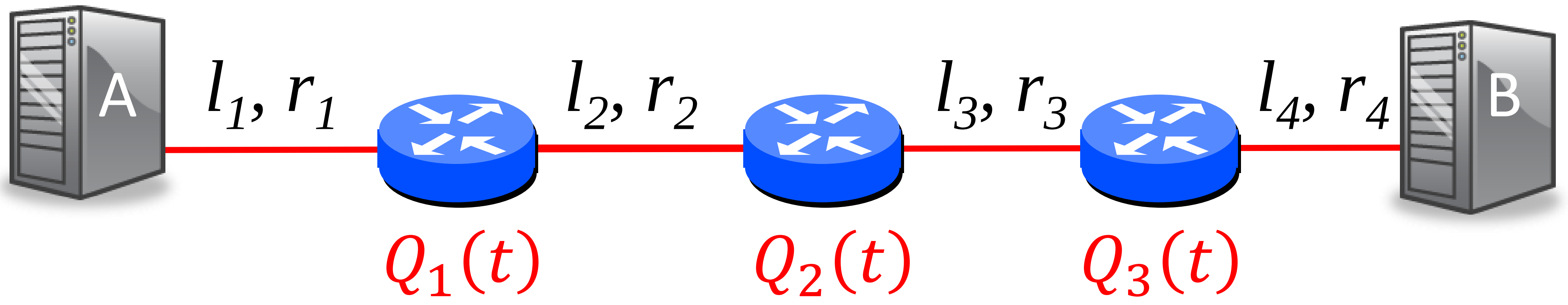
Summary

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

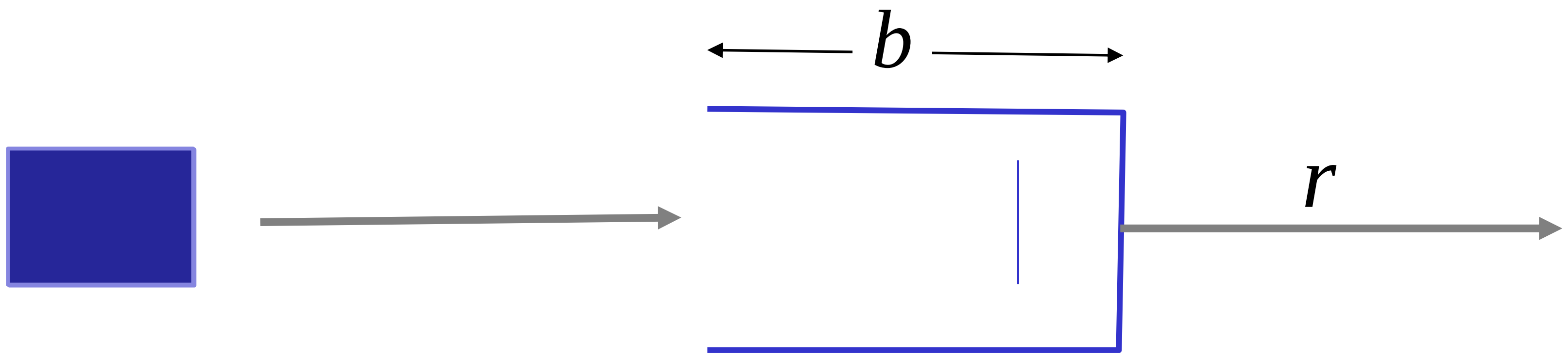
Delay guarantees: Intuition



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.

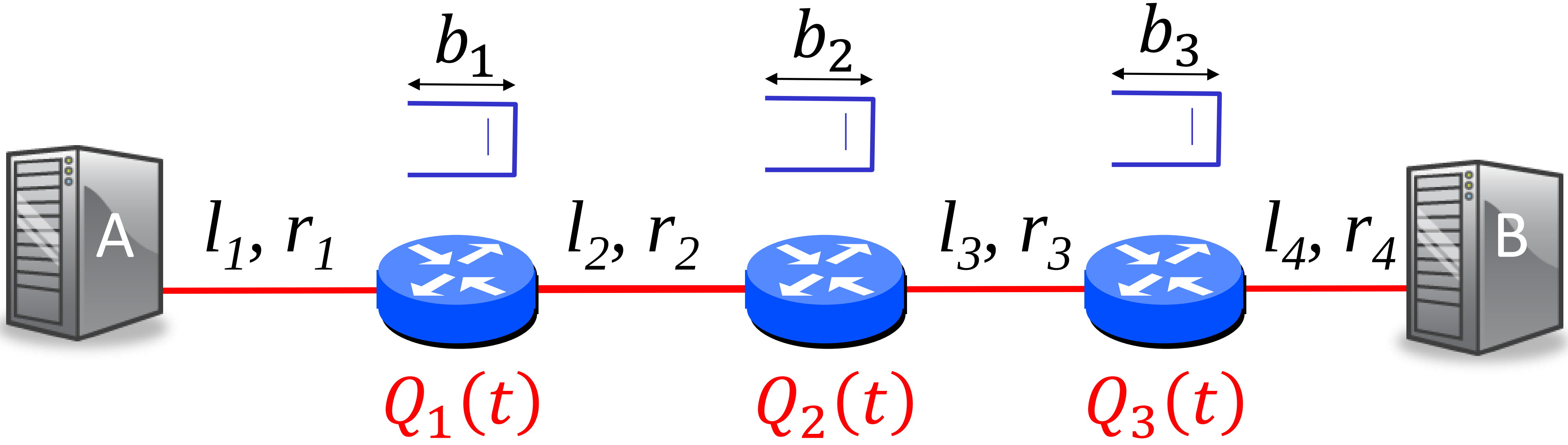
Upper bound on $Q(t)$



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

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

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

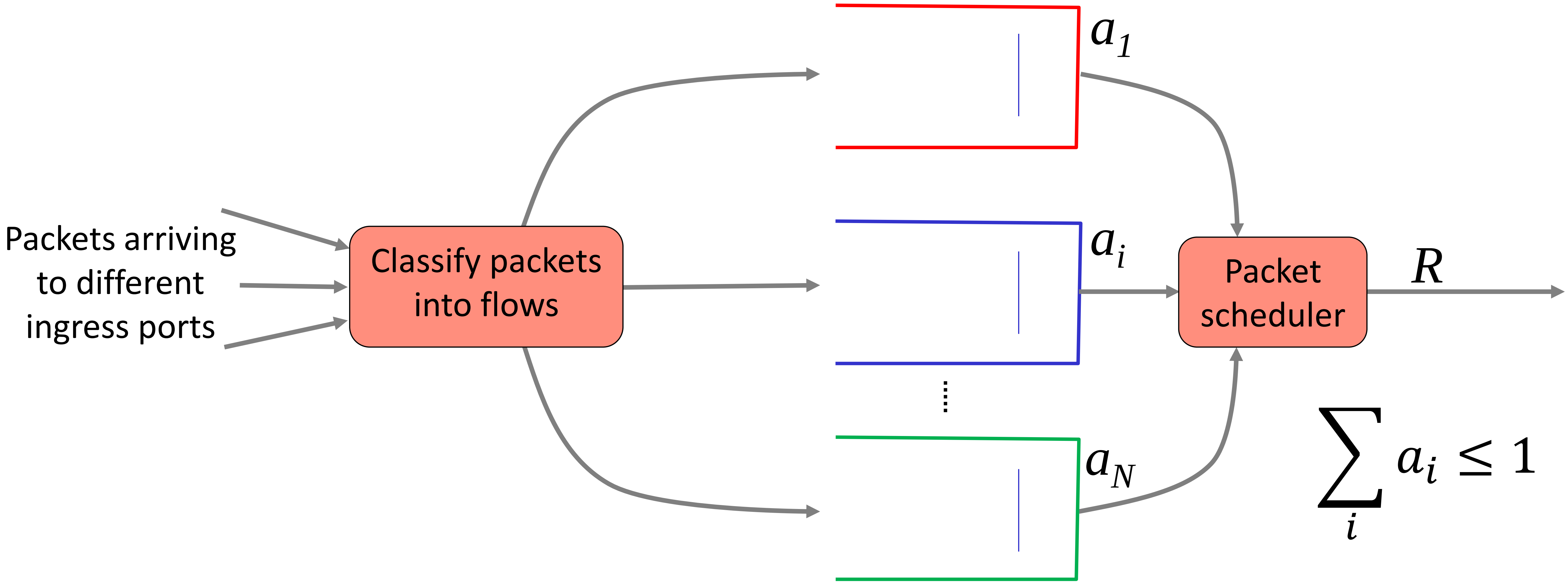
Why this is only an intuition…

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.

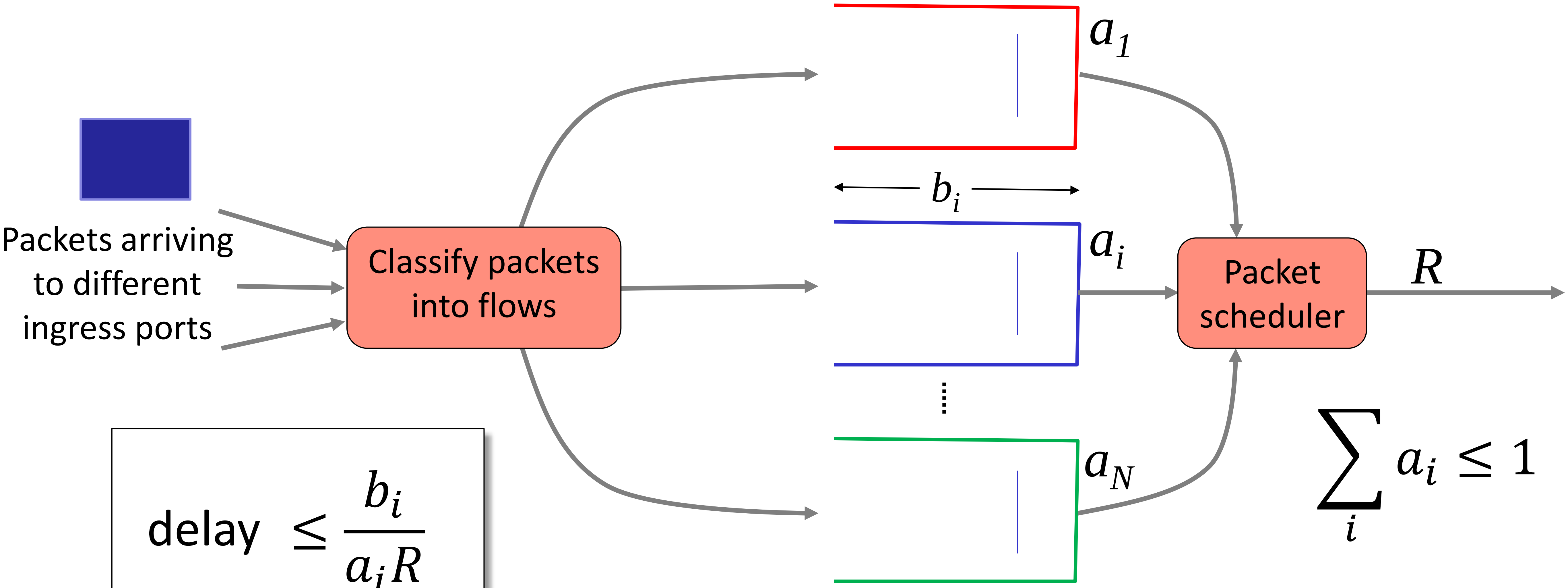
Q: How can we give an upper bound on delay for packets in each flow?

Weighted Fair Queueing (WFQ)



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

Weighted Fair Queueing (WFQ)

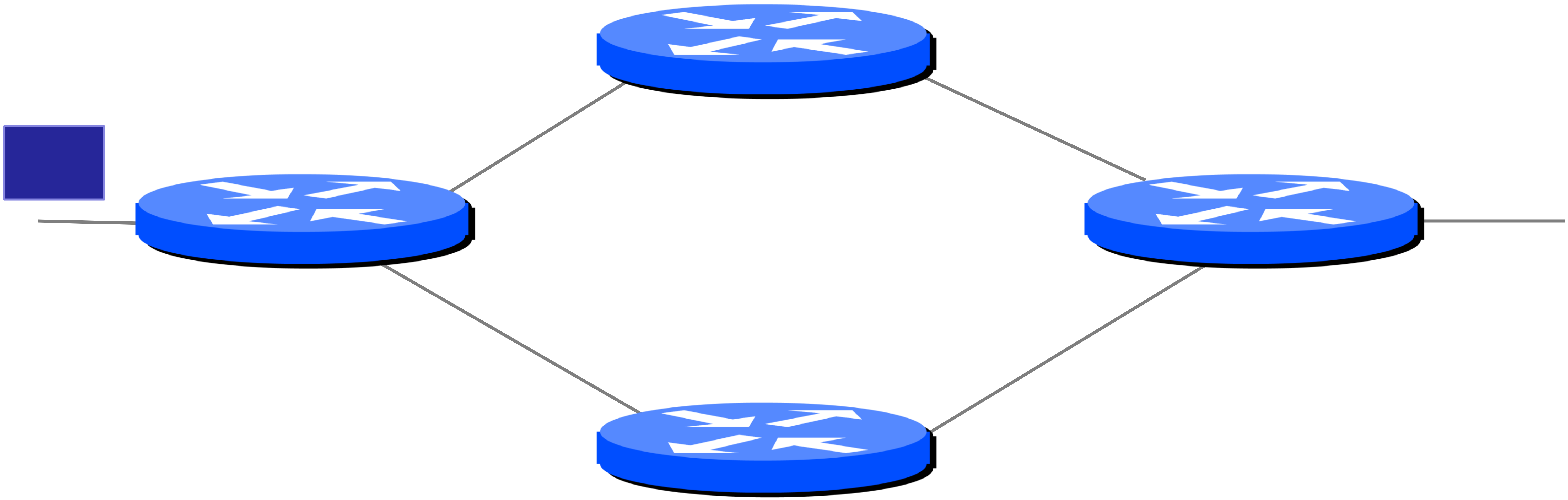


$$\sum_i a_i \leq 1$$

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

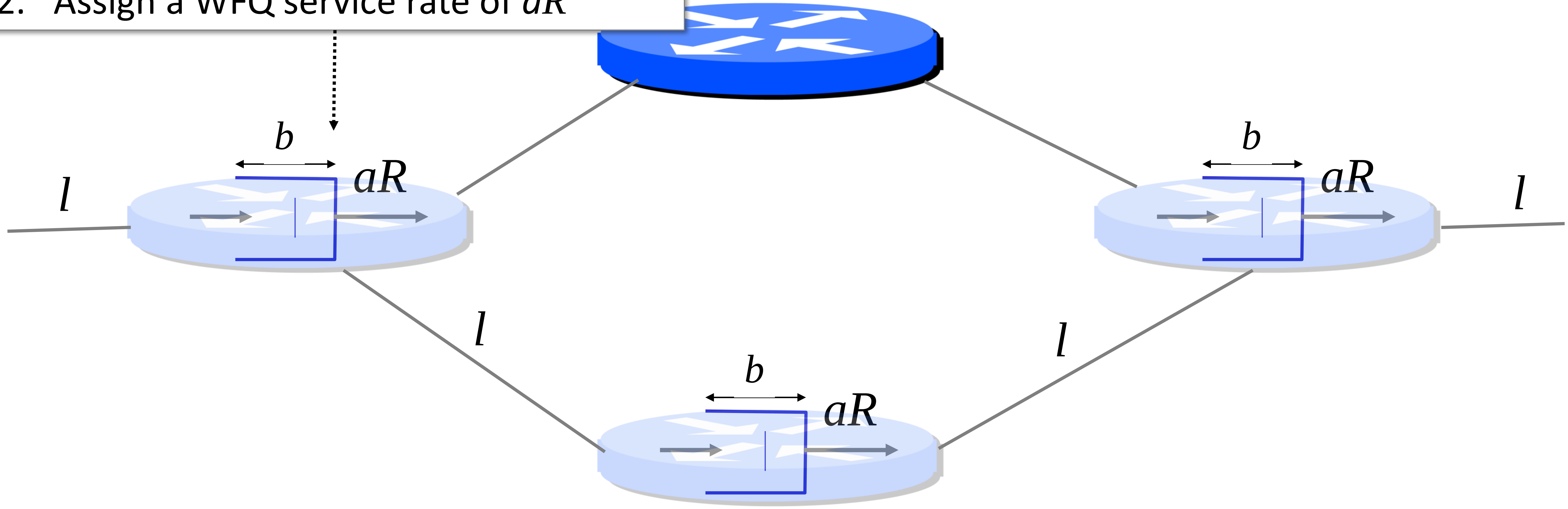
Flow i is guaranteed to receive at least rate $a_i R$

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")

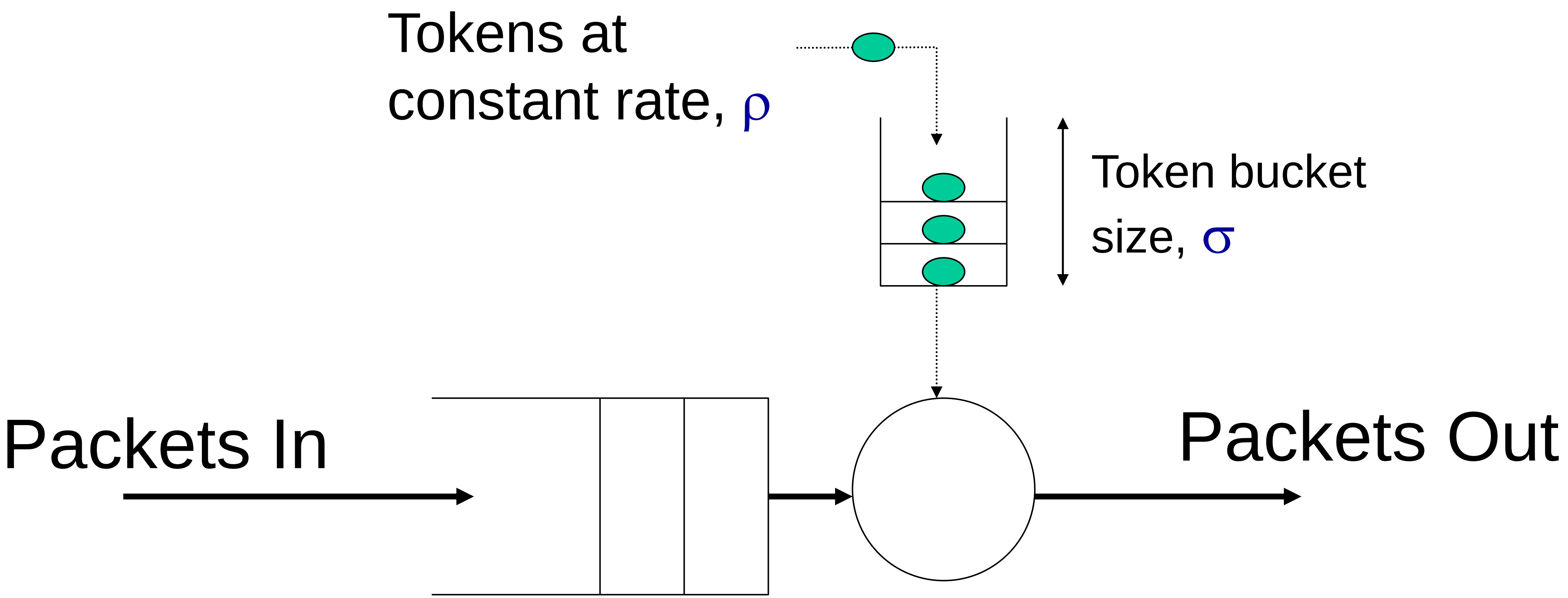
1. If the packets are far apart, then the queues drain the first packet before the second one arrives. All is good, and the delay equation holds.

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 <u>all</u> packets, then we need to deal with bursts. How?

The leaky bucket regulator

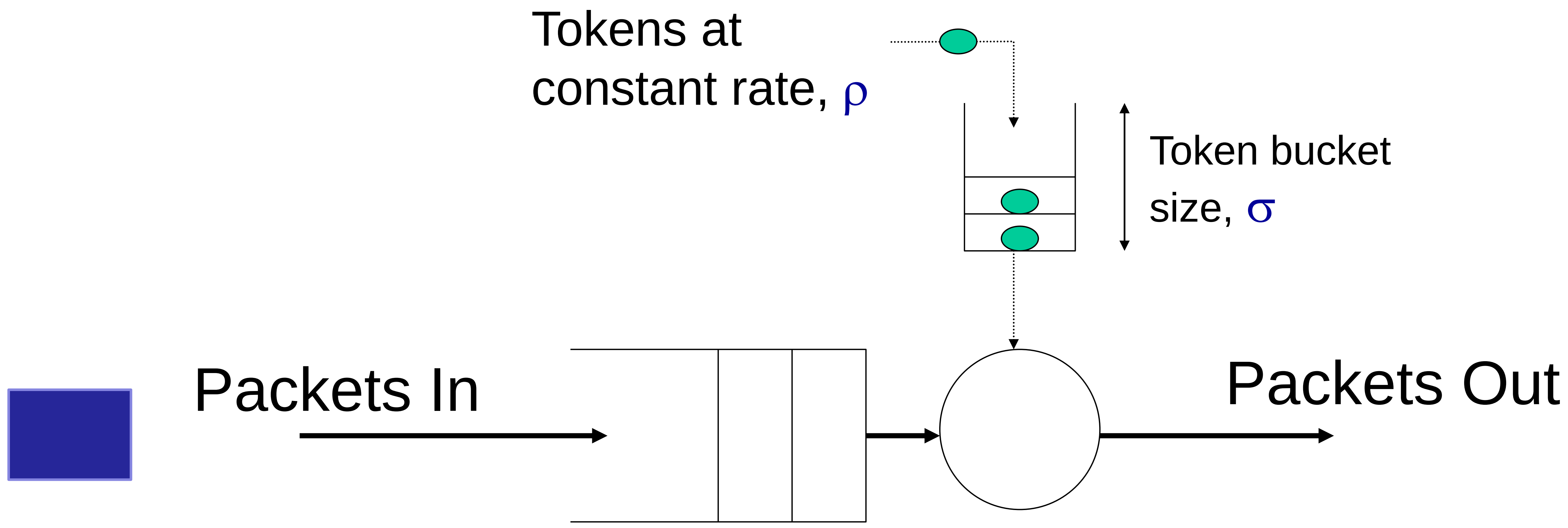
Limiting the "burstiness"



End-host sends packet if and only if there are tokens in the bucket

The leaky bucket regulator

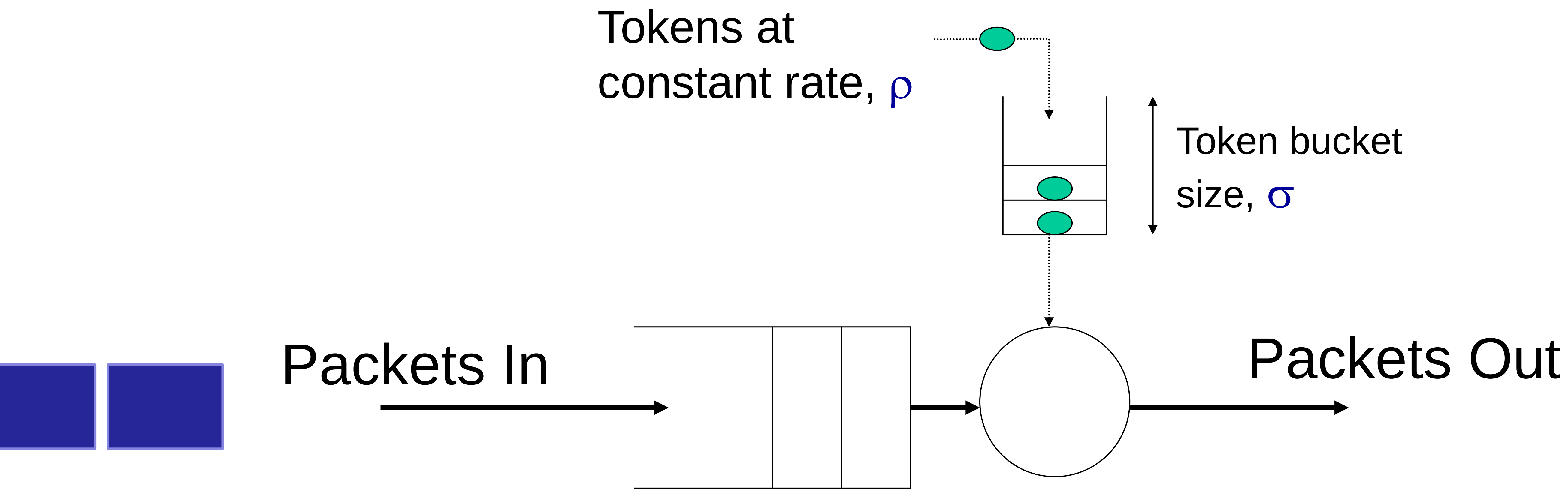
Limiting the “burstiness”



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

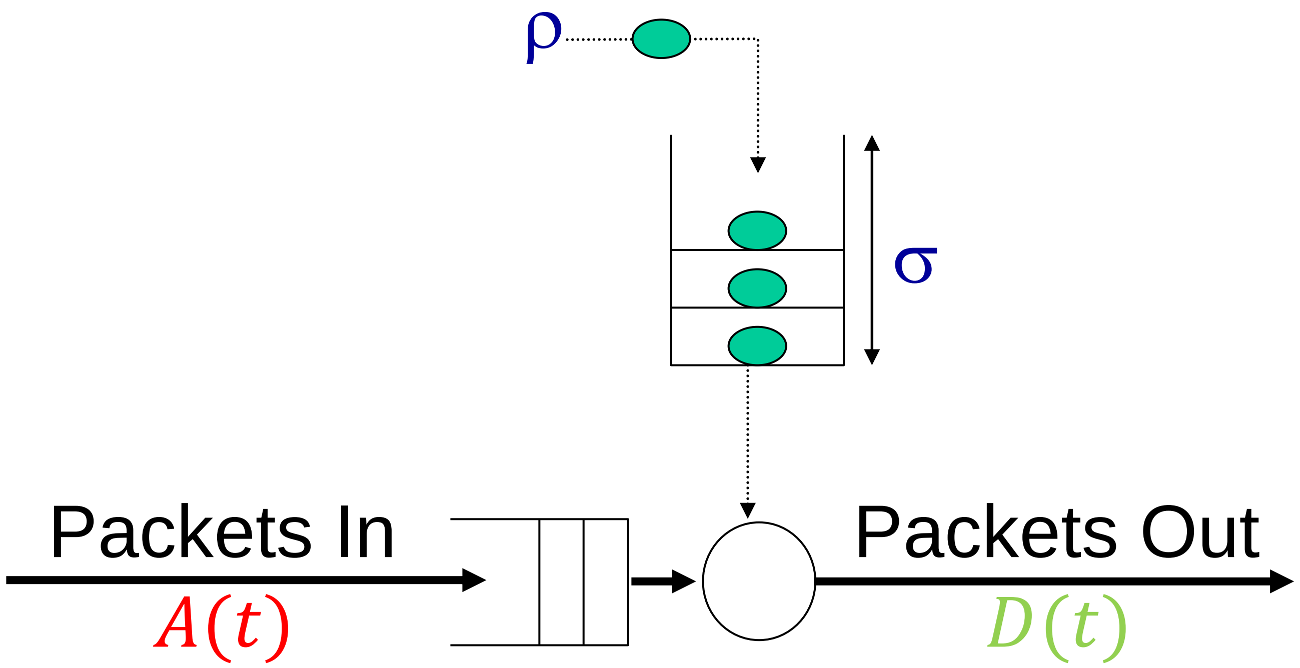
The leaky bucket regulator

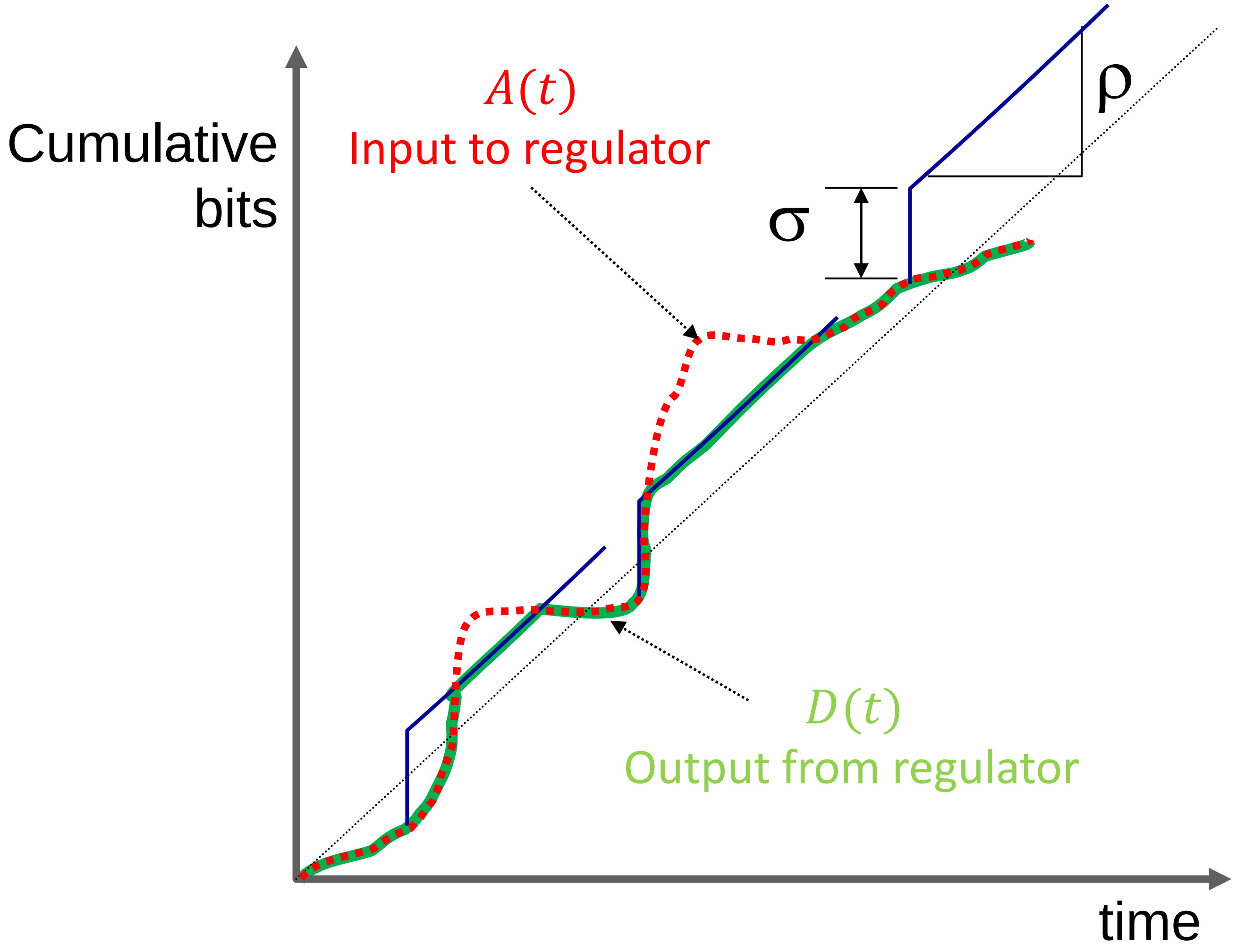
Limiting the "burstiness"



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

The leaky bucket regulator



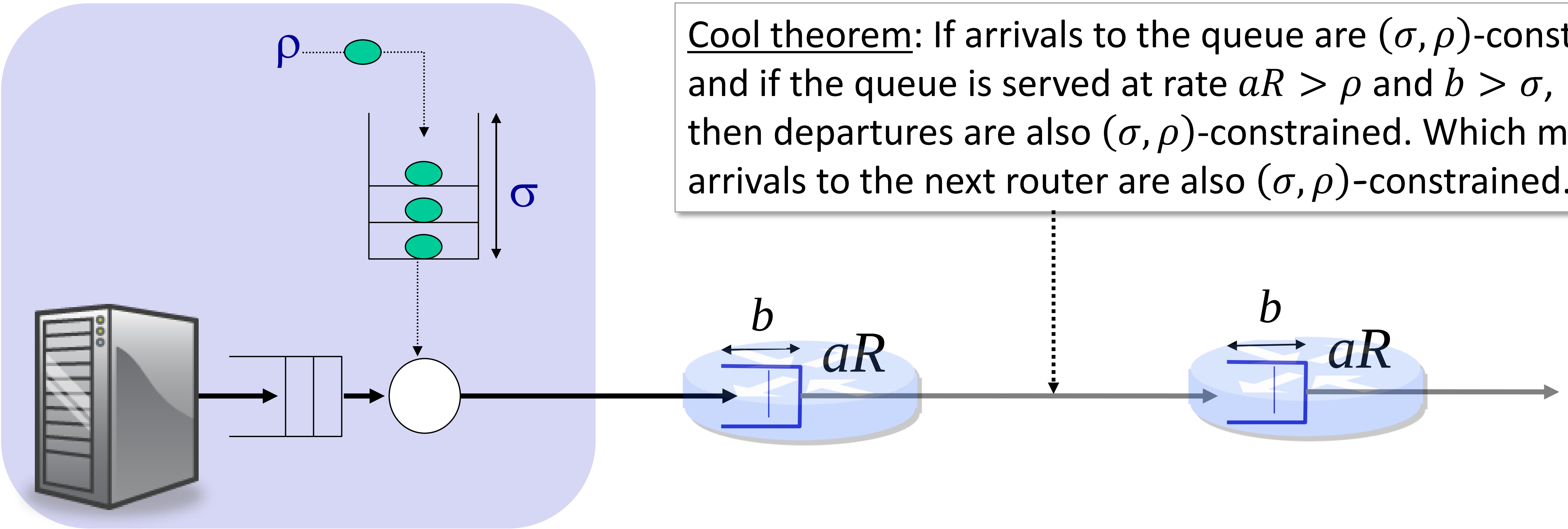


Number of bits that can be sent in <u>any</u> period of length t is bounded by: $\sigma + \rho t$

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

The leaky bucket regulator

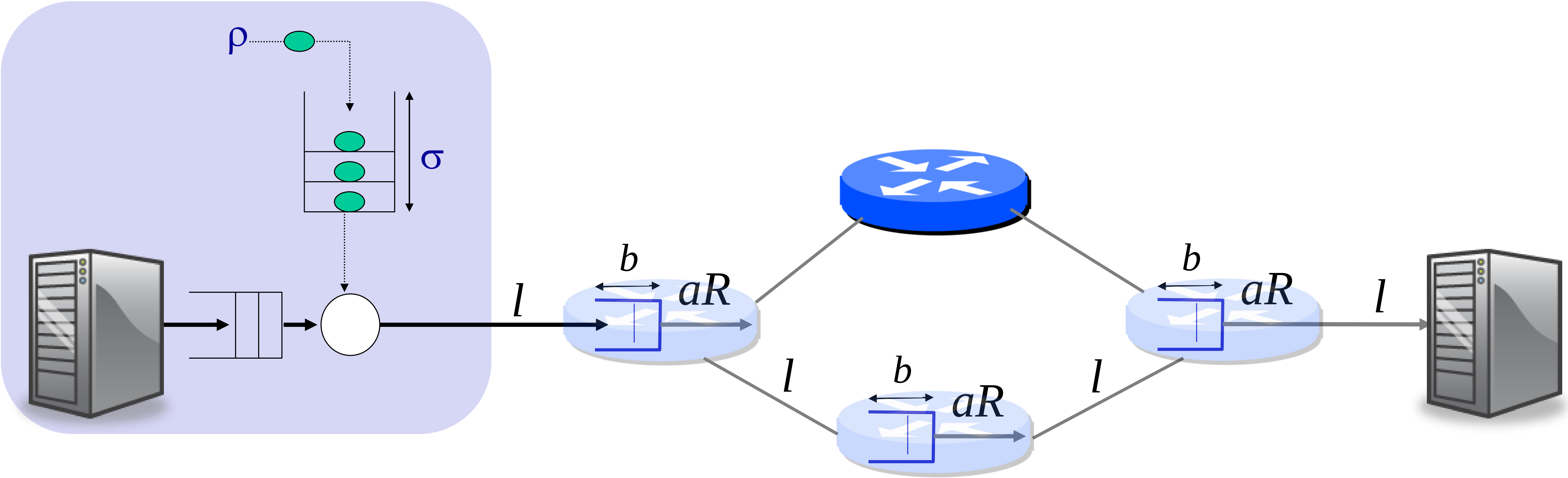
Limiting the "burstiness"



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.

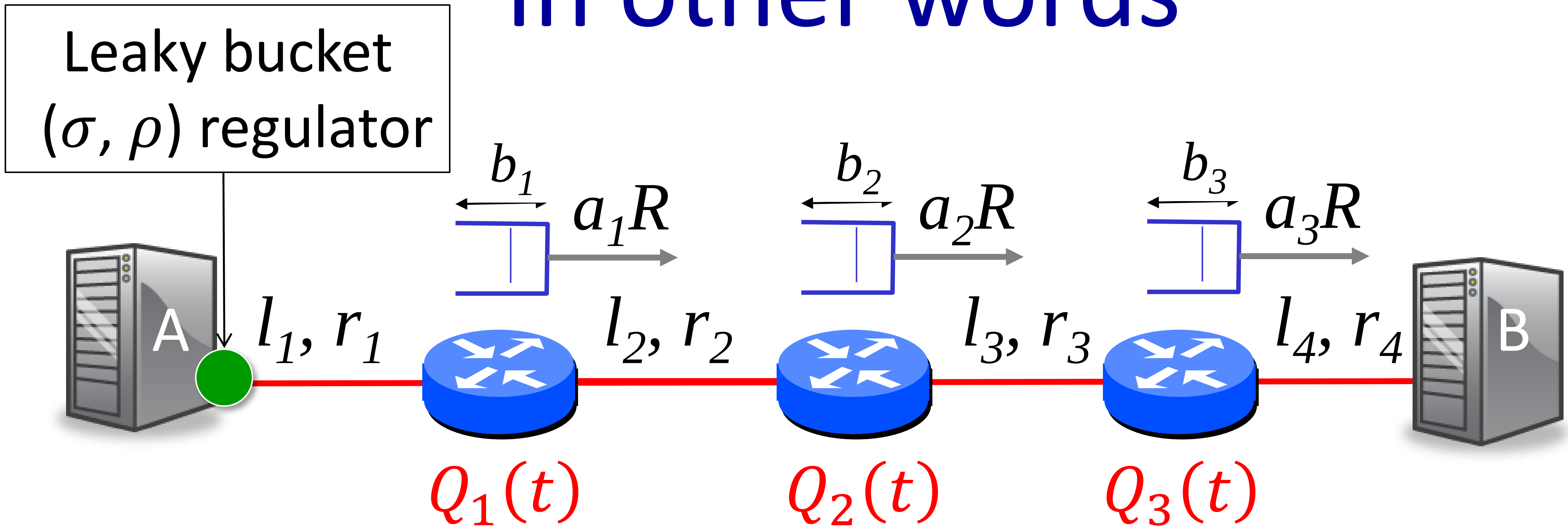
If $aR > \rho$ and $b > \sigma$ then delay through the first router for all packets in the flow $\leq \frac{b}{aR}$

Putting it all together



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

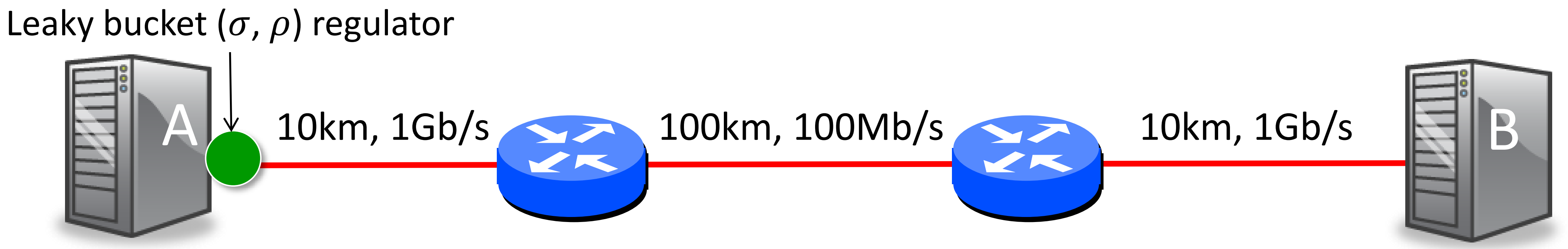
In other words



If we set $b_i > \sigma$, and $a_i R > \rho$ then
$$\tau = \sum_{i=1}^{4}\left(\frac{p}{r_i} + \frac{l_i}{c}\right) + \sum_{i=1}^{3} Q_i(t)$$
$$\leq \sum_{i=1}^{4}\left(\frac{p}{r_i} + \frac{l_i}{c}\right) + \frac{3\sigma}{\rho}$$

A Worked Example

Q: In the network below, we want to give an application flow a rate of at least 10Mb/s and an end to end delay of at most 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 $(120km/c) + 8{,}000bits(\frac{1}{10^9} + \frac{1}{100 \times 10^6} + \frac{1}{10^9})$ = 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 $a_i R \geq 10Mb/s$ and the flow's queue should have a capacity of at least 2500bytes.

In practice

While lots of network equipment implements WFQ (even your WiFi router at home might!), public networks don't offer a service to their customers to guarantee end-to-end delay.

Why?

- It requires coordination of all the routers from end to end. IETF RFC 4495 "RSVP" was designed to help coordinate.
- In most networks, a combination of over-provisioning and priorities work well enough.
- However, rate guarantees are commonly used by network owners to control how flows share network capacity.

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.

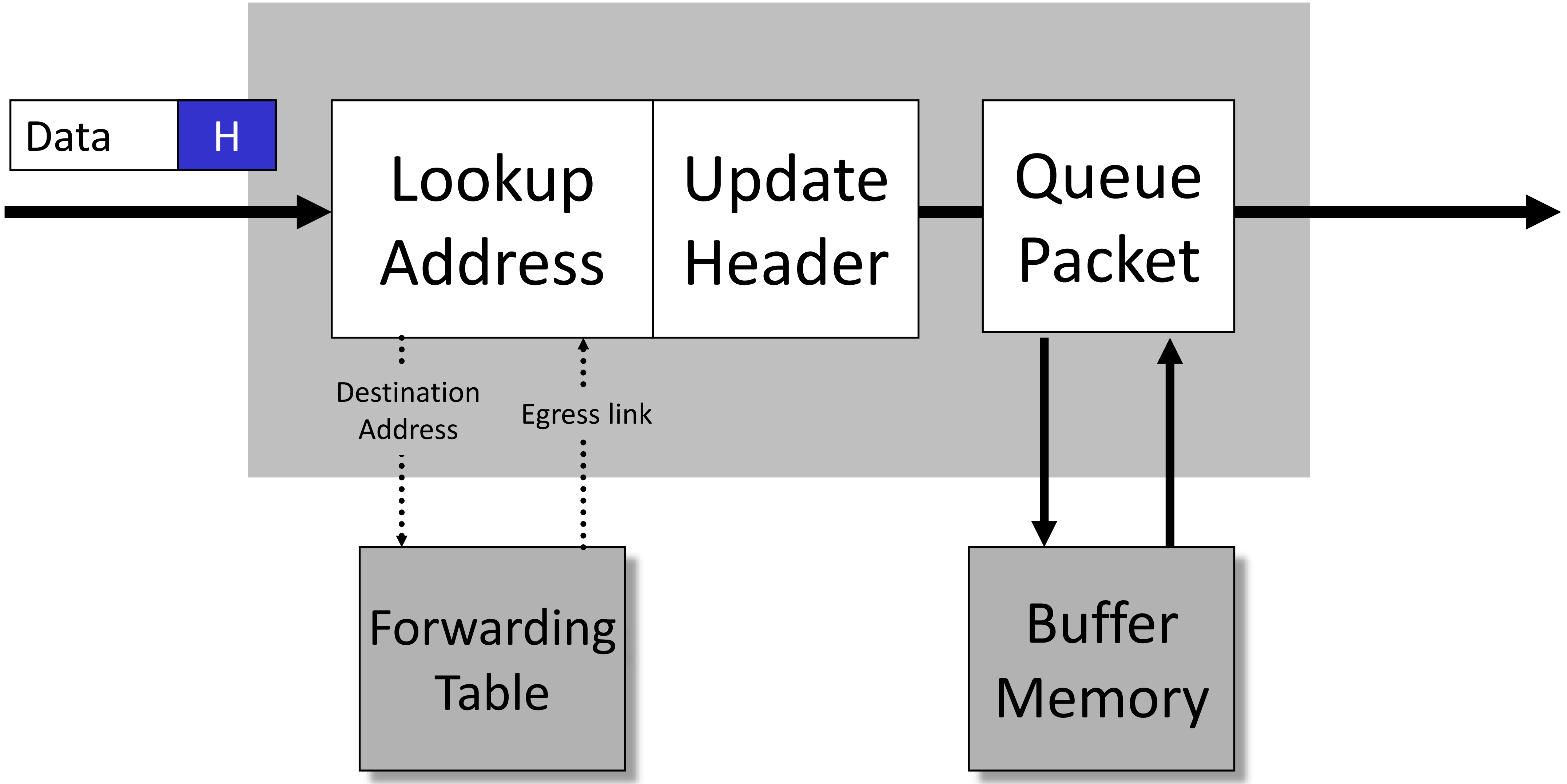
Packet Switching

How a packet switch works

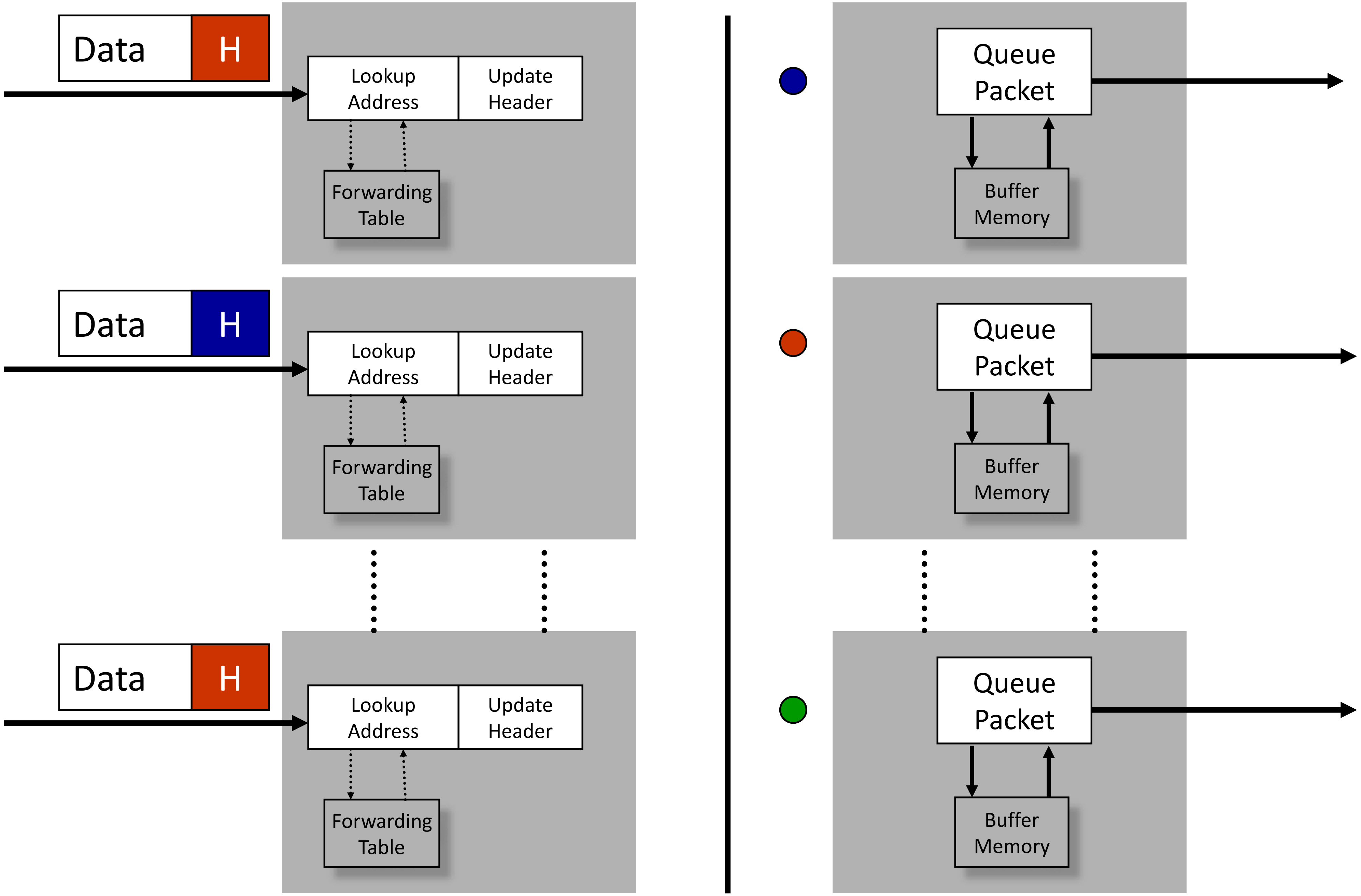
Outline

1. What does a packet switch look like?
2. What does a packet switch do?
   - Ethernet switch
   - Internet router
3. How address lookup works
   - Ethernet switch
   - Internet router

Generic Packet Switch



Generic Packet Switch



Packet processing by an Ethernet Switch

1. Examine the header of each arriving frame.
2. If the Ethernet DA is in the forwarding table, forward the frame to the correct output port(s).
3. If the Ethernet DA is not in the table, broadcast the frame to all ports (except the one through which the frame arrived).
4. Entries in the table are learned by examining the Ethernet SA of arriving packets.

Packet processing by an Internet Router

1. If the Ethernet DA of the arriving frame belongs to the router, accept the frame. Else drop it.
2. Examine the IP version number and length of the datagram.
3. Decrement the TTL, update the IP header checksum.
4. Check to see if TTL == 0.
5. If the IP DA is in the forwarding table, forward to the correct egress port(s) for the next hop.
6. Find the Ethernet DA for the next hop router.
7. Create a new Ethernet frame and send it.

Basic Operations

1. **Lookup Address**: How is the address looked up in the forwarding table?
2. **Switching**: How is the packet sent to the correct output port?

Lookup Address: Ethernet

Ethernet addresses (in a switch)

Match	Action
Ethernet DA = 0xA8B72340E678	Forward to port 7
Ethernet DA = 0xB3D22571053B	Forward to port 3
…	…

Methods

- Store addresses in hash table (maybe a multi-way hash)
- Lookup exact match in hash table

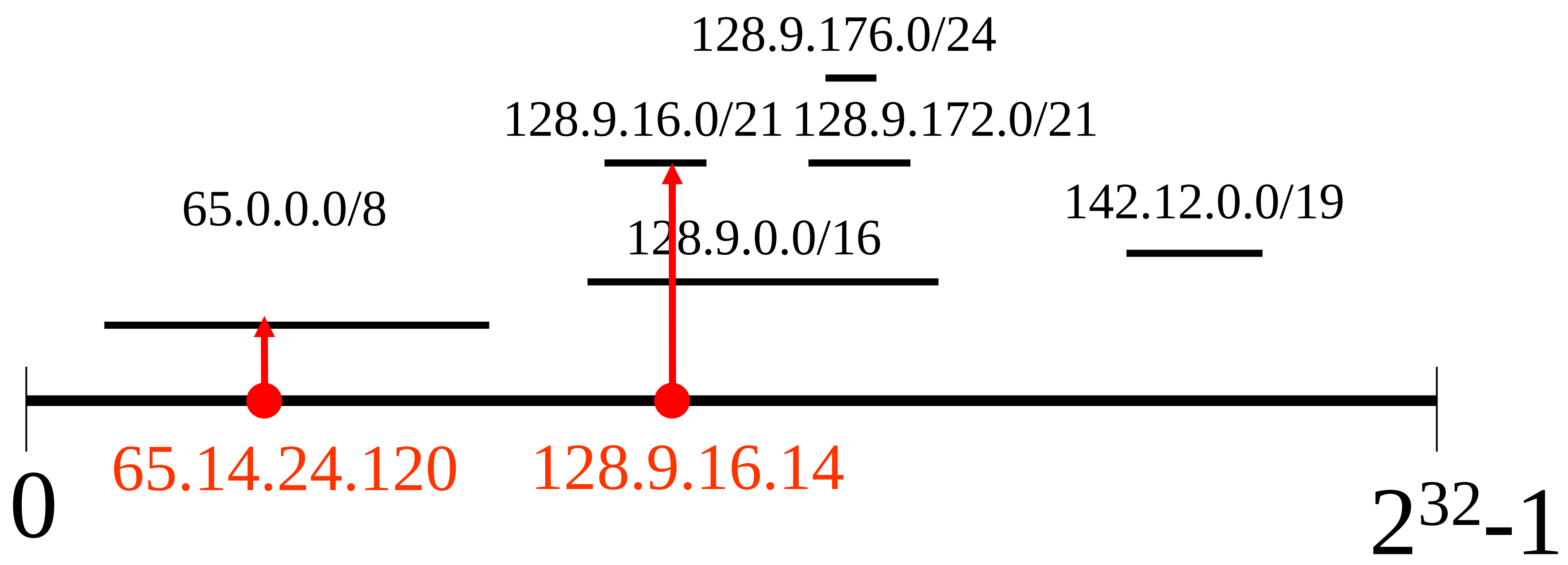
Lookup Address: IPv4

IP addresses (in a router)

Match	Action
IP DA = 127.43.57.99	Forward to 56.99.32.16
IP DA = 123.66.44.X	Forward to 22.45.21.126
IP DA = 76.9.X.X	Forward to 56.99.32.16
…	…

Lookup is a longest prefix match, not an exact match

Longest prefix match

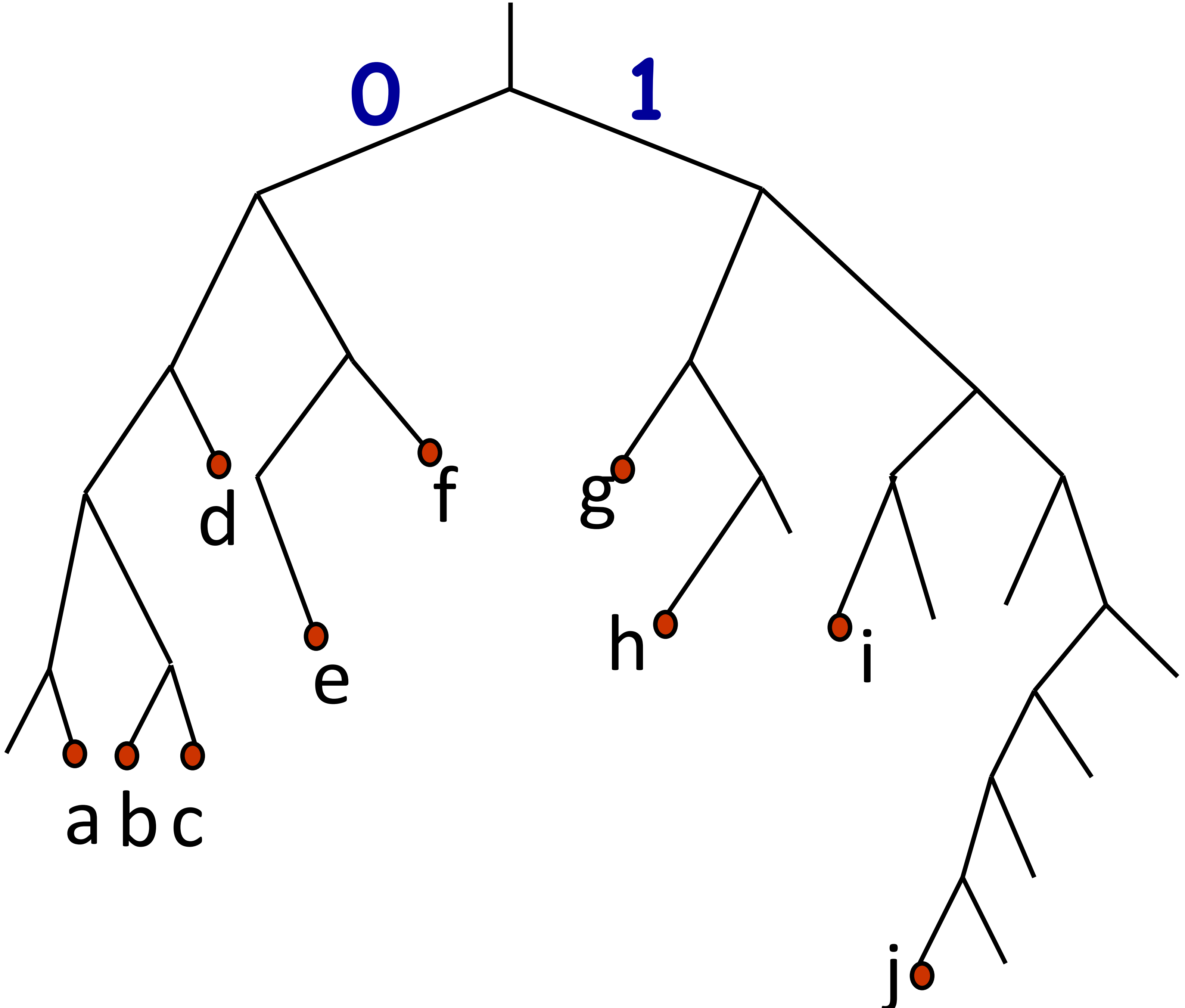


Routing lookup: Find the longest matching prefix (aka the most specific route) among all prefixes that match the destination address.

Longest prefix match lookup

Binary tries ("retrieval")

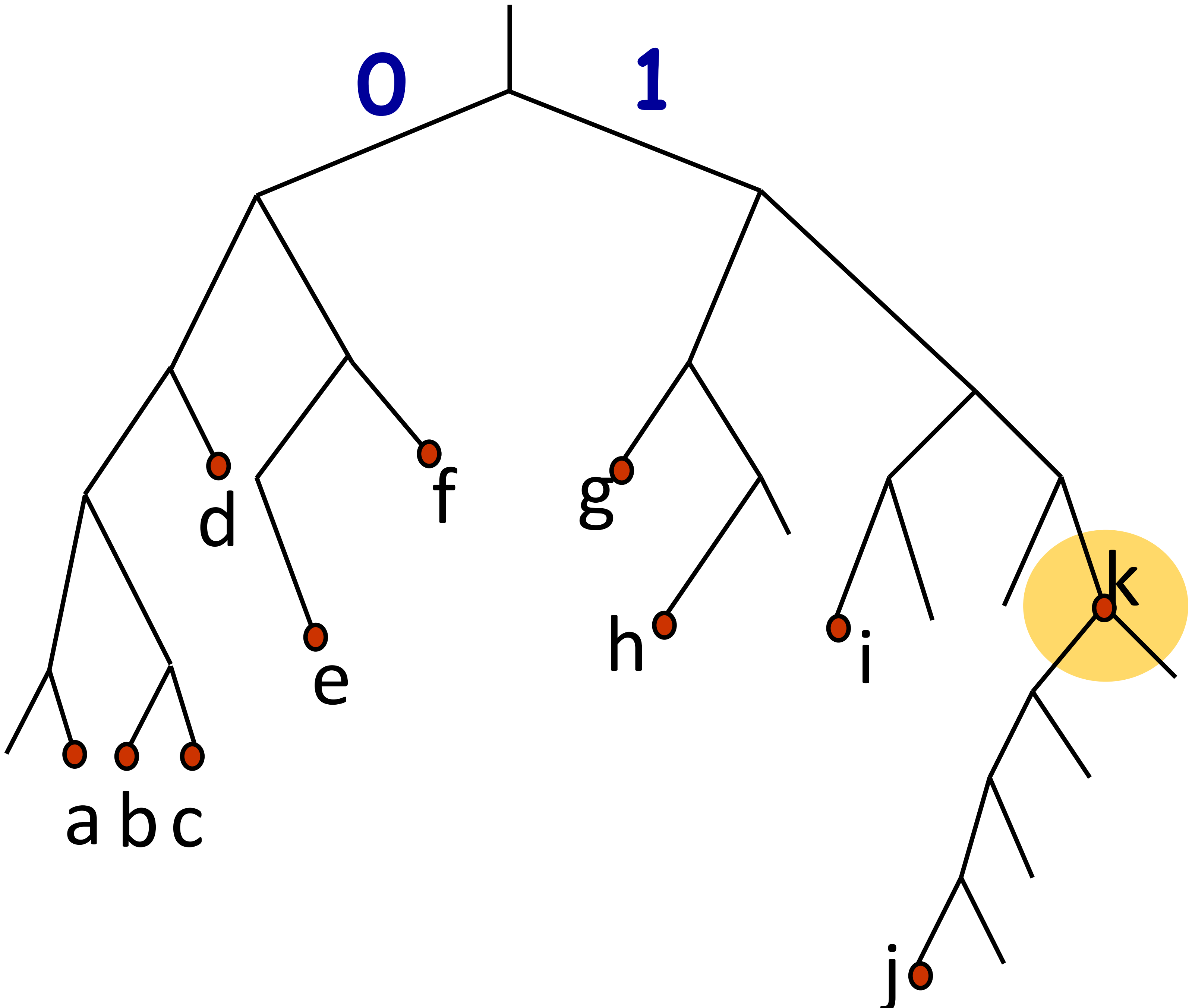
Entry	Prefix
a	00001
b	00010
c	00011
d	001
e	0101
f	011
g	100
h	1010
i	1100
j	11110000



Longest prefix match lookup

Entry	Prefix
a	00001
b	00010
c	00011
d	001
e	0101
f	011
g	100
h	1010
i	1100
j	11110000
k	1111

Binary tries ("retrieval")



Longest prefix match lookup

Ternary Content Addressable Memory (TCAM)

Entry	Prefix
a	00001
b	00010
c	00011
d	001
e	0101
f	011
g	100
h	1010
i	1100
j	11110000

Binary value + Mask

Entry	Prefix
a	00001XXX 11111000
b	00010XXX 11111000
c	00011XXX 11111000
d	001XXXXX 11100000
e	0101XXXX 11110000
…	…
j	11110000 11111111

Associative lookup: Compare address against every entry at the same time.

Lookup Address: Generic

Generic or abstract lookups: <Match, Action>

Match	Action
IP DA = X	Forward to port 7
Eth DA = Y AND IP DA = Z	Drop packet

- Generalization of lookups and forwarding action in switches, routers, firewalls, etc.
- This led to an abstraction for controlling switches using <match,action>rules, called OpenFlow

Summary

Packet switches perform two basic operations:

- Lookup addresses in a forwarding table
- Switching to the correct egress port

At a high level, Ethernet switches and Internet routers perform similar operations

Address lookup is different in Ethernet switches and IP routers.