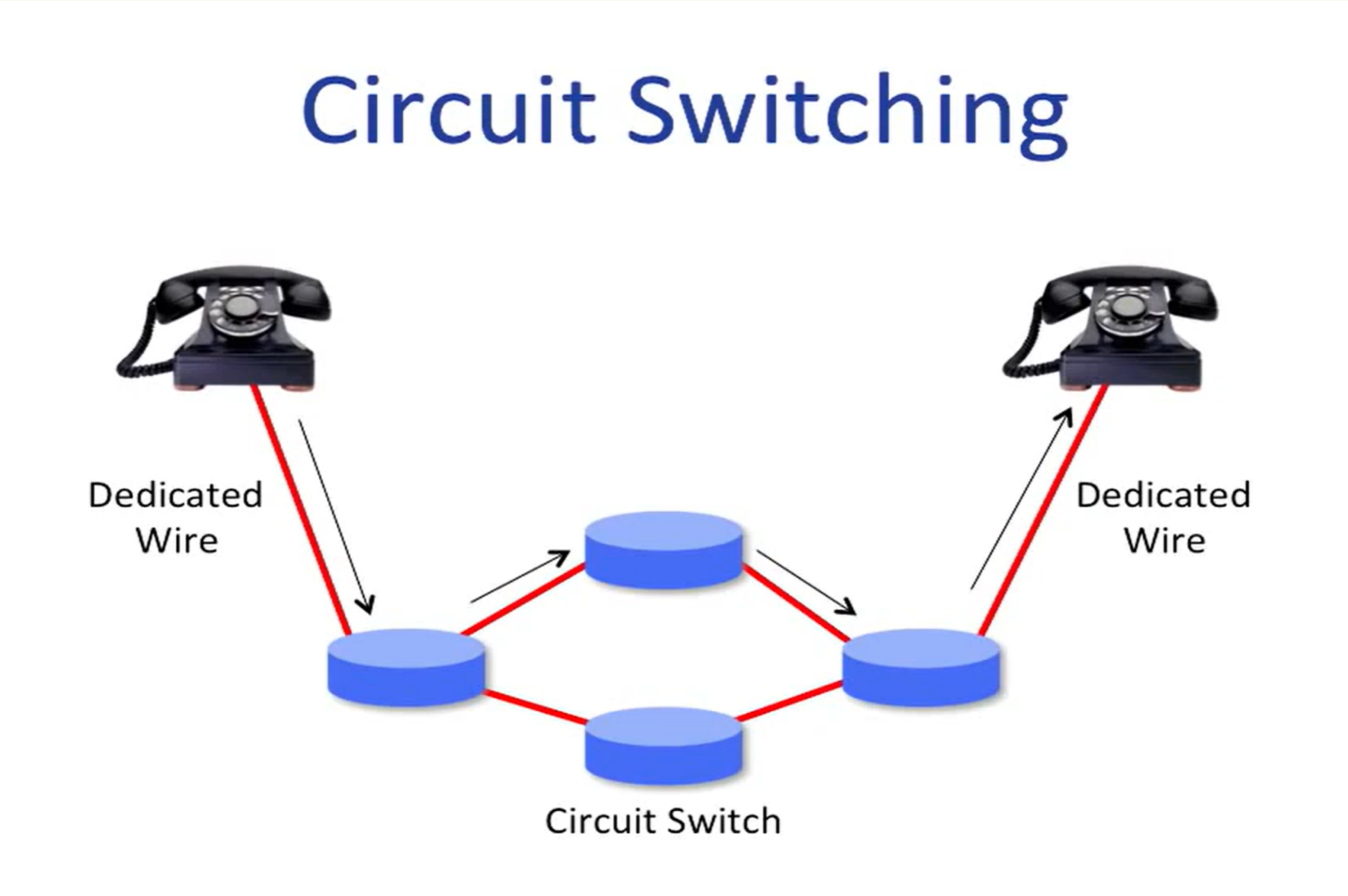
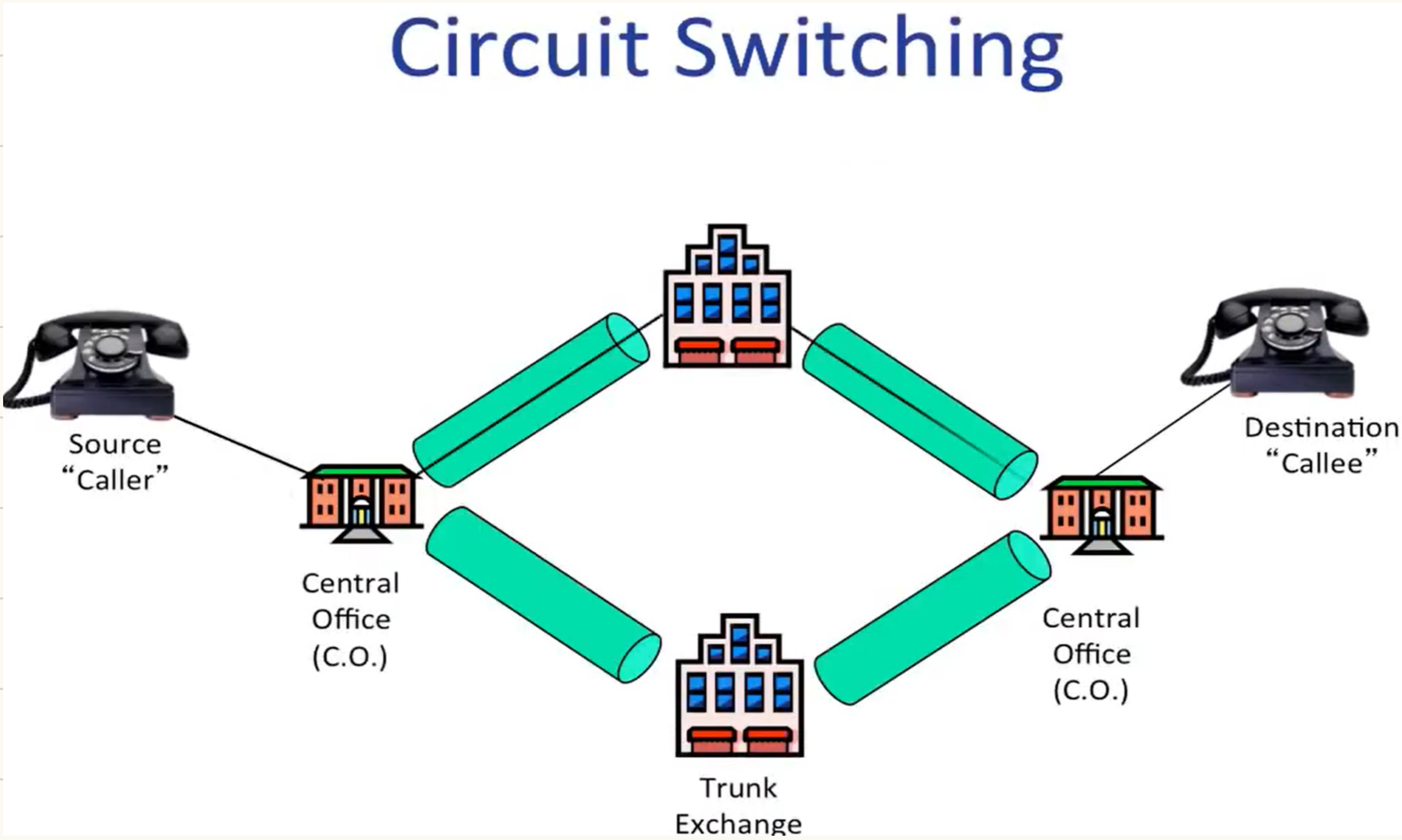
What is Packet Switching





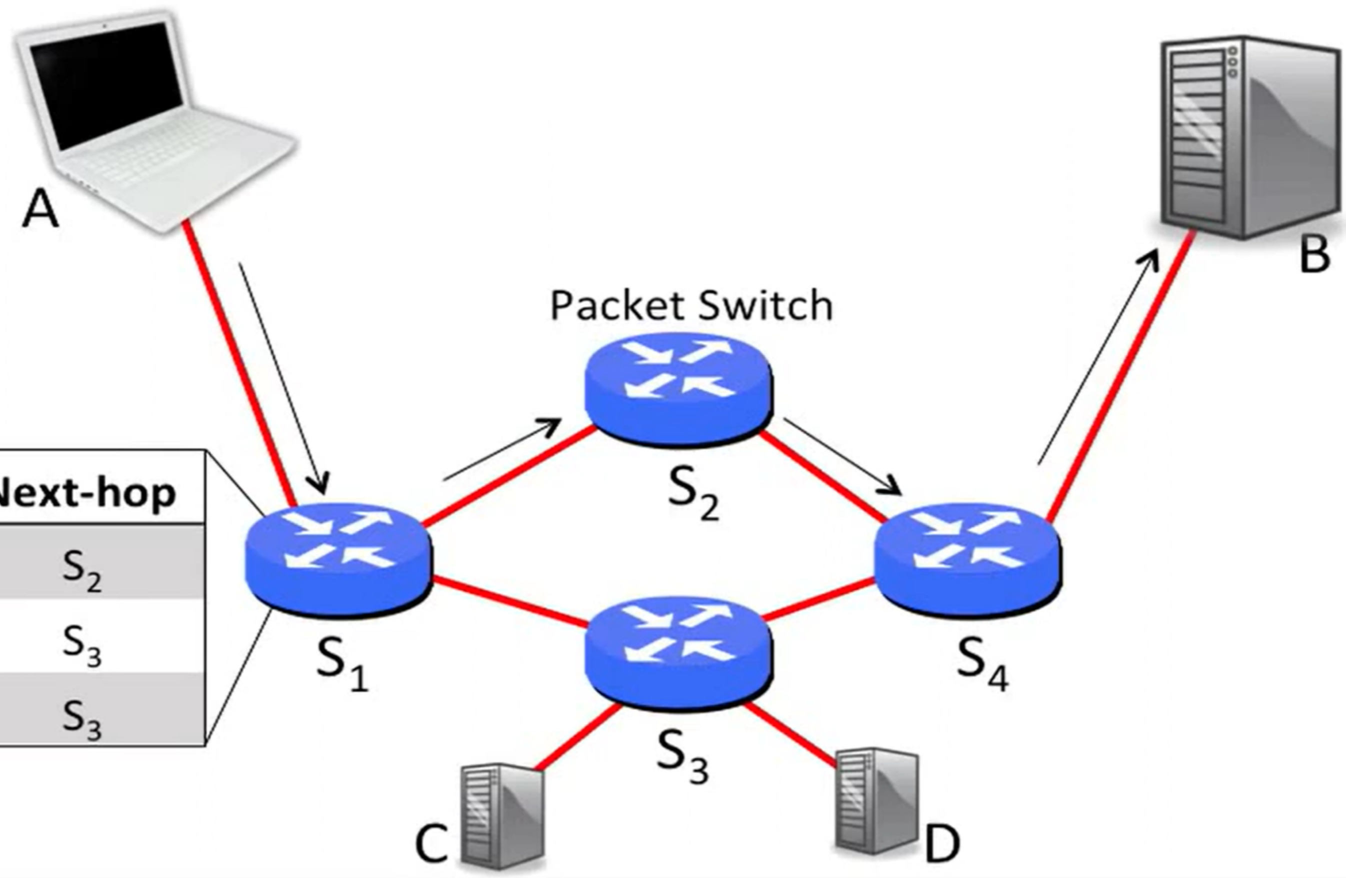
Circuit Switching

- Each call has its own private, guaranteed, isolated data rate from end-to-end.
- A call has three phases:
  1. Establish circuit from end-to-end ("dialing")
  2. Communicate
  3. Close circuit ("tear down")
- Originally, a circuit was an end-to-end physical wire.
- Nowadays, a circuit is like a virtual private wire.

Problems

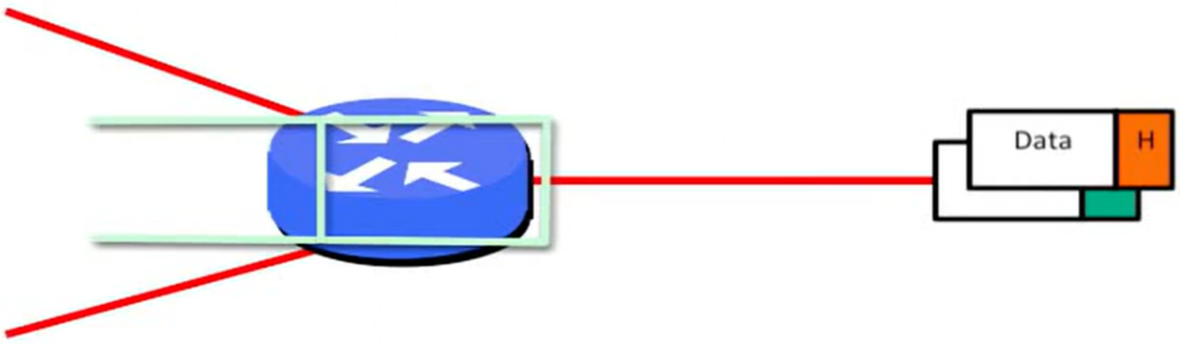
1. **Inefficient**. Computer communication tends to be very bursty. *e.g.* typing over an ssh connection, or viewing a sequence of web pages. If each communication has a dedicated circuit, it will be used very inefficiently.
2. **Diverse Rates**. Computers communicate at many different rates. *e.g.* a web server streaming video at 6Mb/s, or me typing at 1 character per second. A fixed rate circuit will not be much use.
3. **State Management**. Circuit switches maintain per-communication state, which must be managed.

Packet Switching



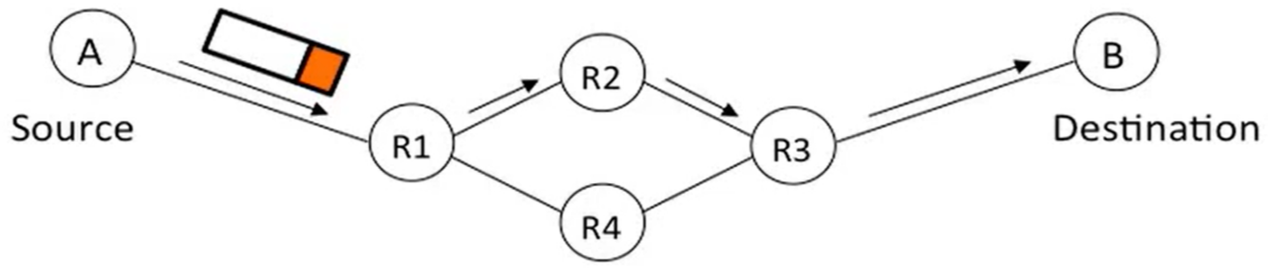
Address	Next-hop
B	S_2
C	S_3
D	S_3

Packet switches have buffers



Buffers hold packets:
- When two or more packets arrive at the same time
- During periods of congestion

Packet Switching



– Packets are routed individually, by looking up address in router's local table.
– All packets share the full capacity of a link.
– The routers maintain no per-communication state.

Why choose packet switch

Efficient use of expensive links

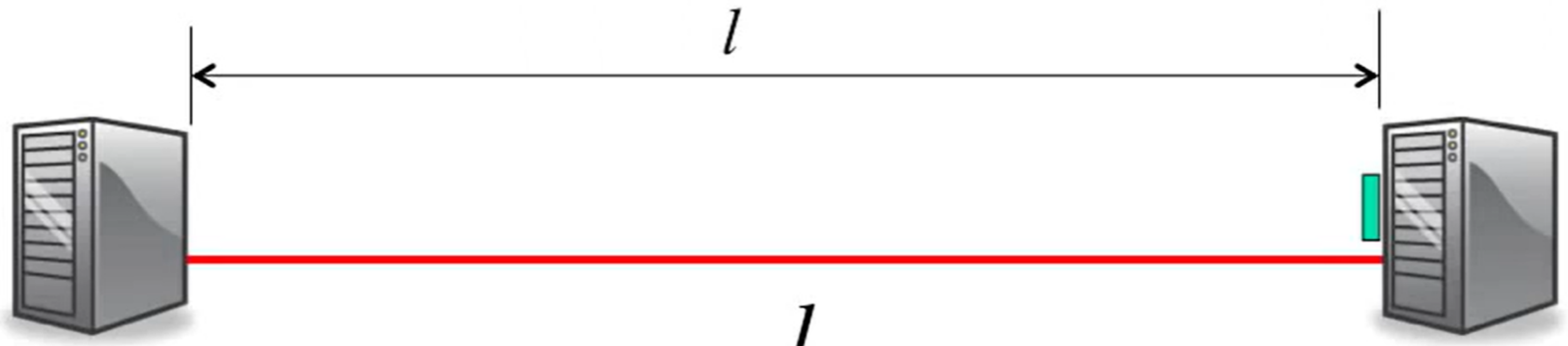
– Links were assumed to be expensive and scarce.
– Packet switching allows many, bursty flows to share the same link efficiently.
– "Circuit switching is rarely used for data networks, ... because of very inefficient use of the links"
– Bertsekas/*Gallager*

Resilience to failure of links & routers

– "For high reliability, ... [the Internet] was to be a datagram subnet, so if some lines and [routers] were destroyed, messages could be ... rerouted" - *Tanenbaum*

Some Definition

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



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

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

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

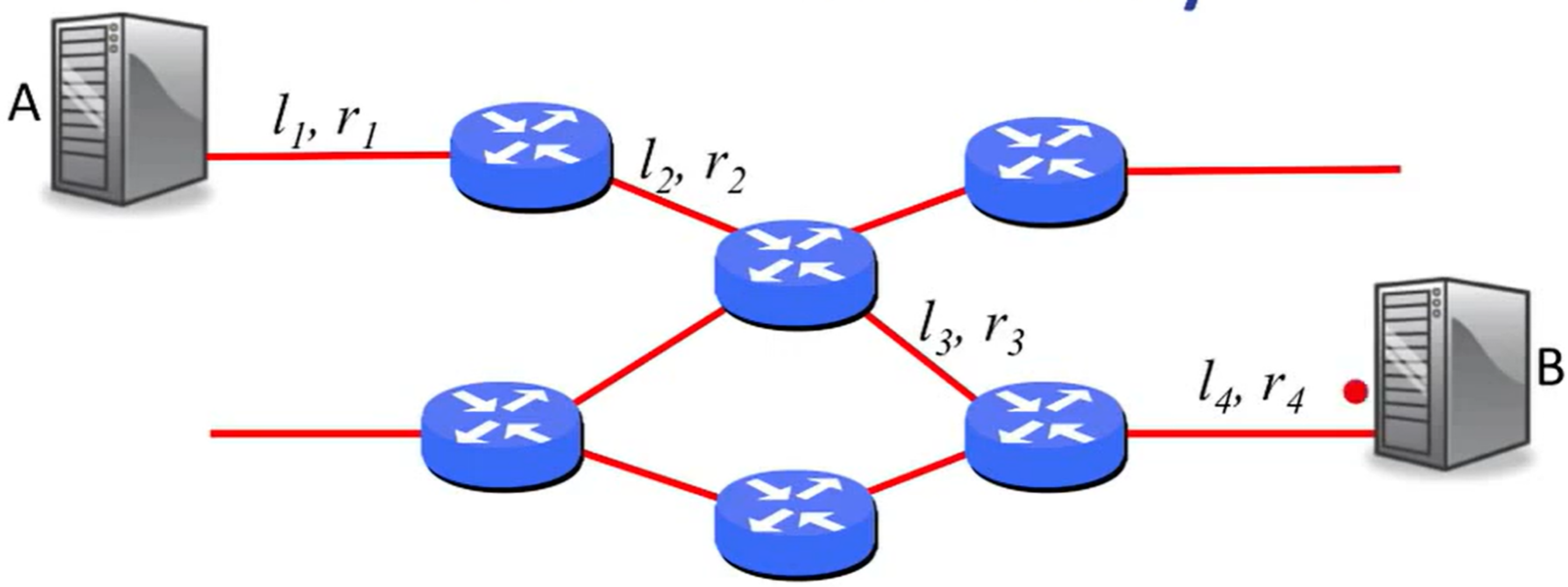
$$t_p = \frac{p}{r}$$

Example 1: A 64byte packet takes 5.12μs to be transmitted onto a 100Mb/s link.
Example 2: A 1kbit packet takes 1.024s to be transmitted onto a 1kb/s link.

1024bits

1000bits/s

End-to-end delay

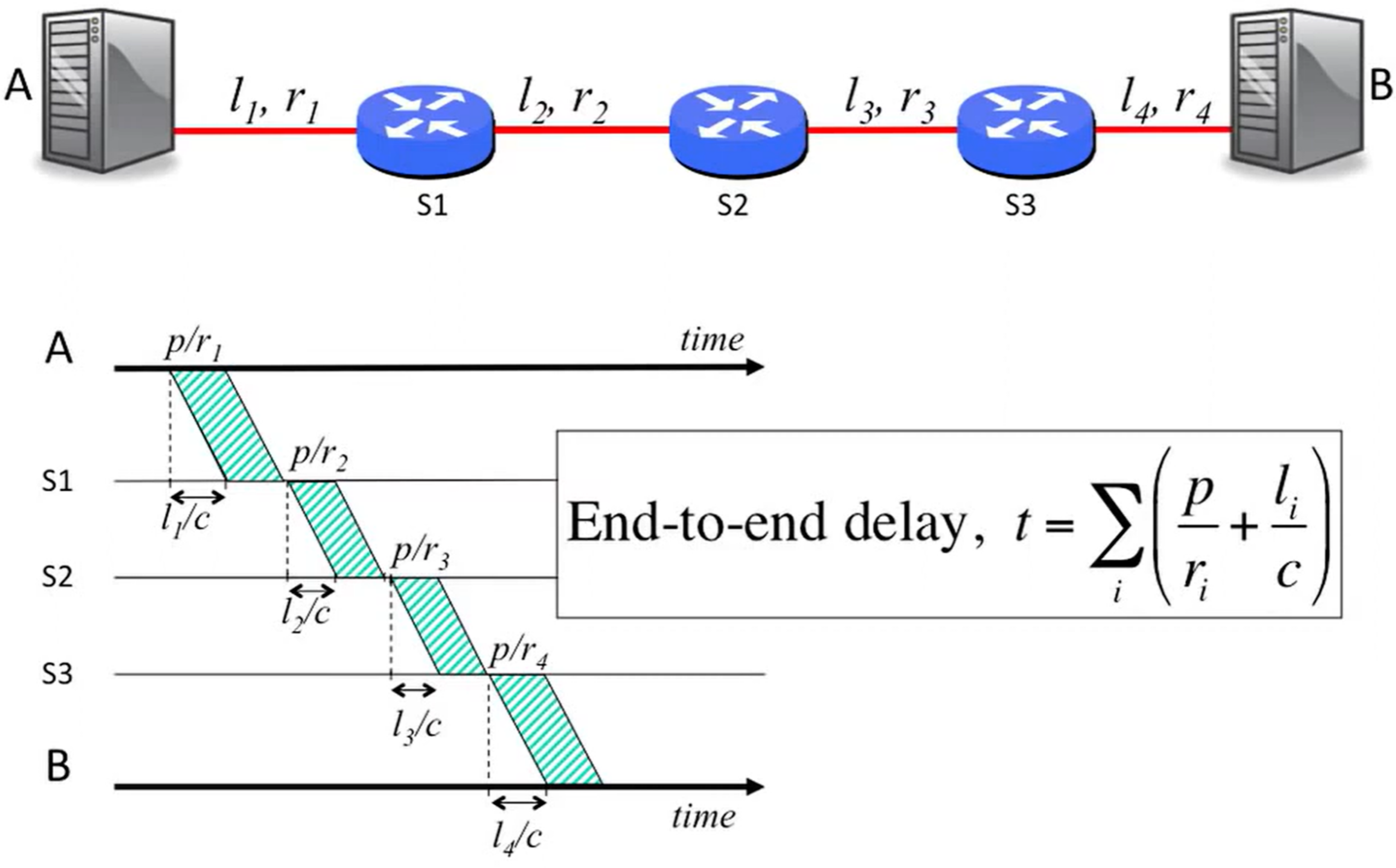


Example: How long will it take a packet of length p to travel from A to B, from when the 1st bit is sent, until the last bit arrives? Assume the switches *store-and-forward* packets along the path.

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

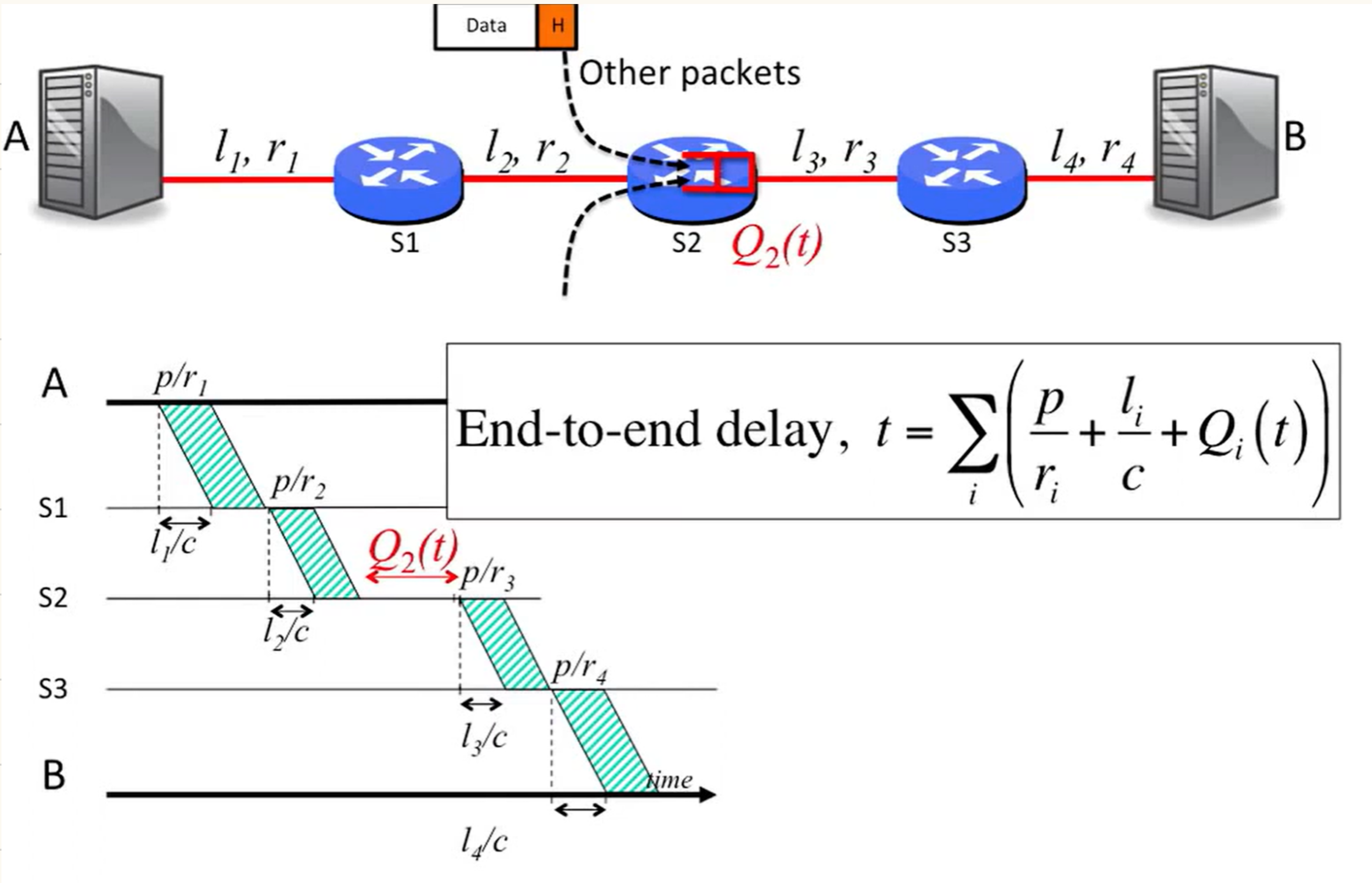
packetization

propagation



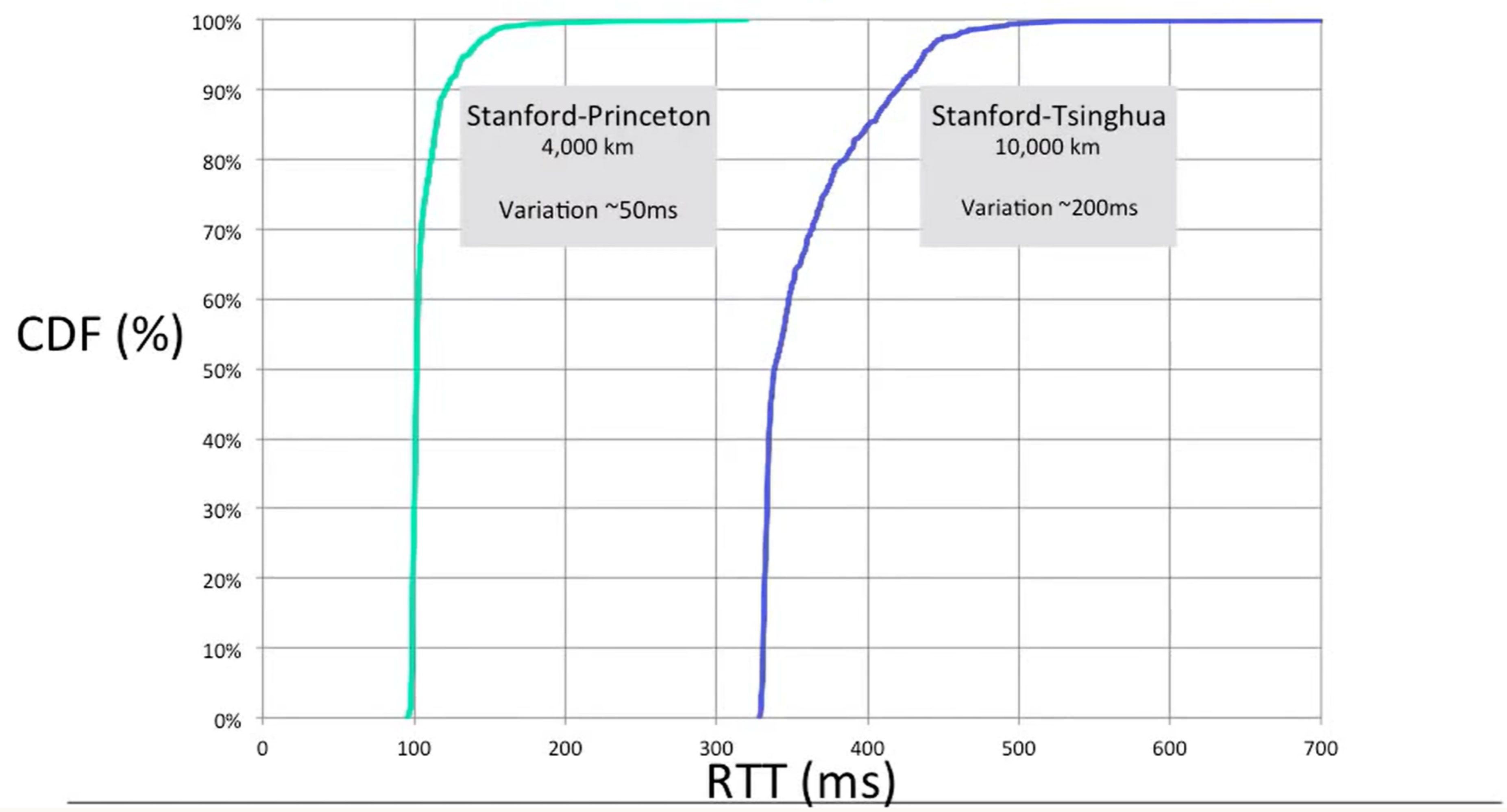
$$\text{End-to-end delay}, \quad t = \sum_i \left(\frac{p}{r_i} + \frac{l_i}{c} \right)$$

If packet queue is considered

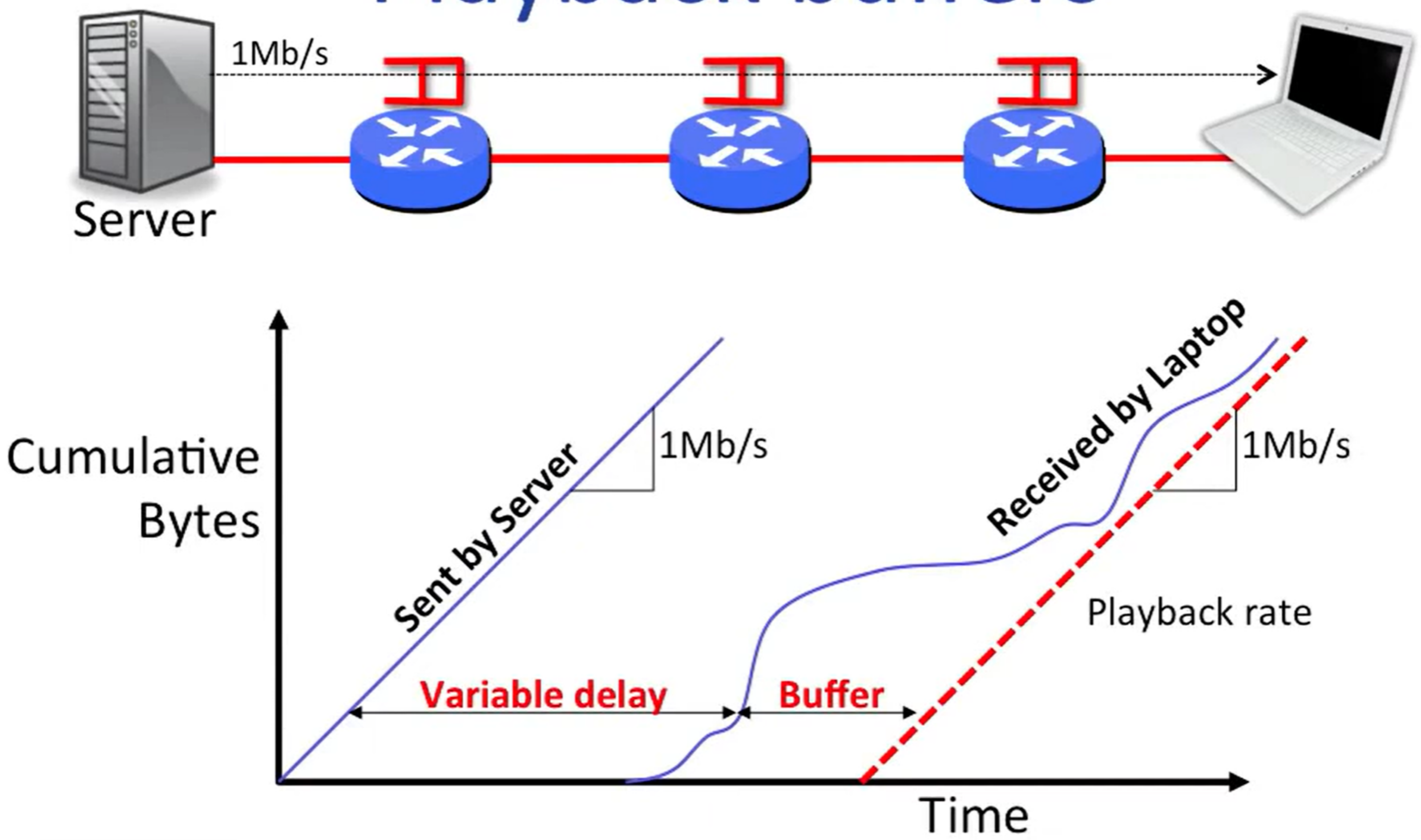


$$\text{End-to-end delay, } t = \sum_i \left(\frac{p}{r_i} + \frac{l_i}{c} + Q_i(t) \right)$$

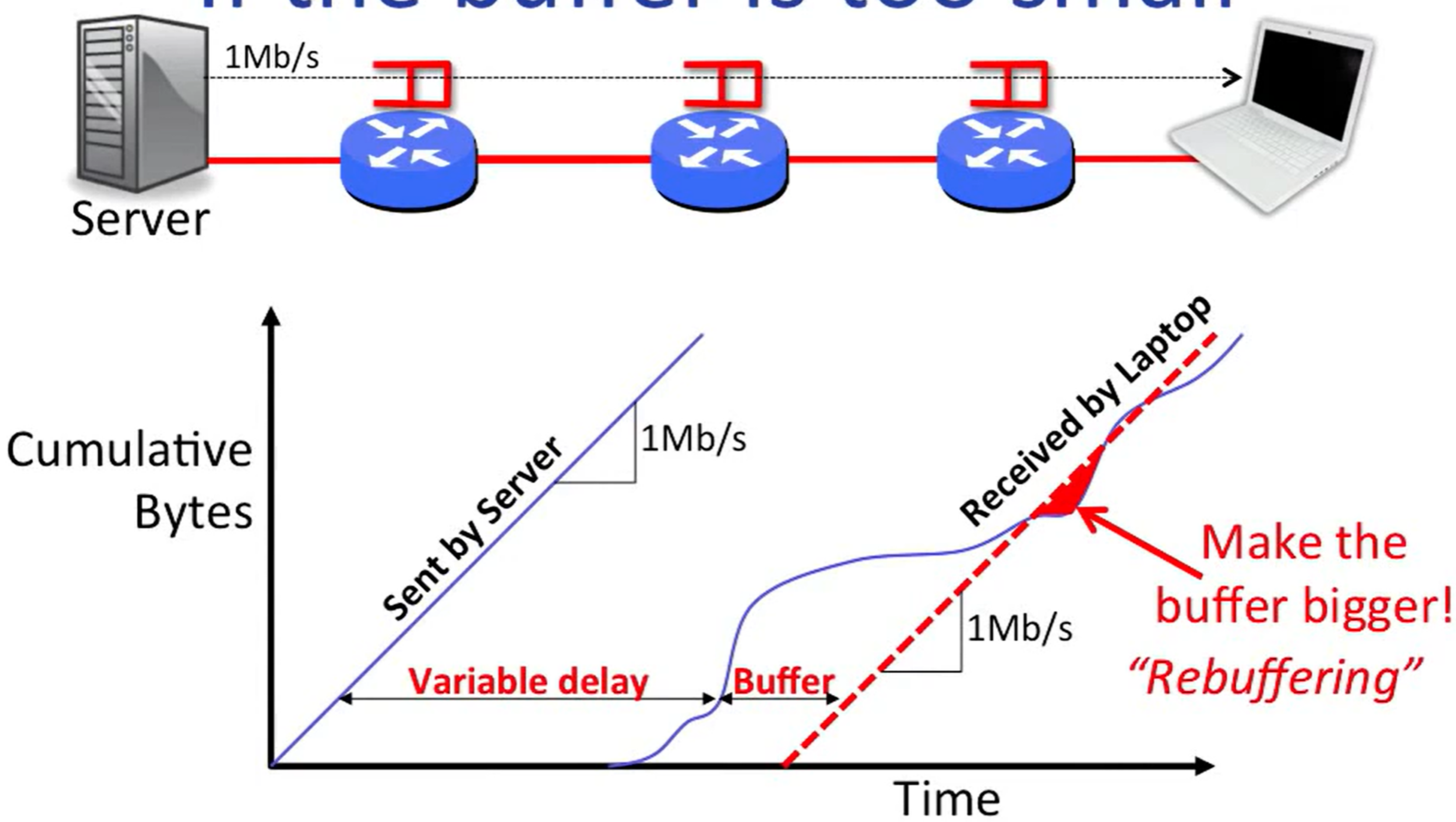
Packet delay variation



Playback buffers



If the buffer is too small

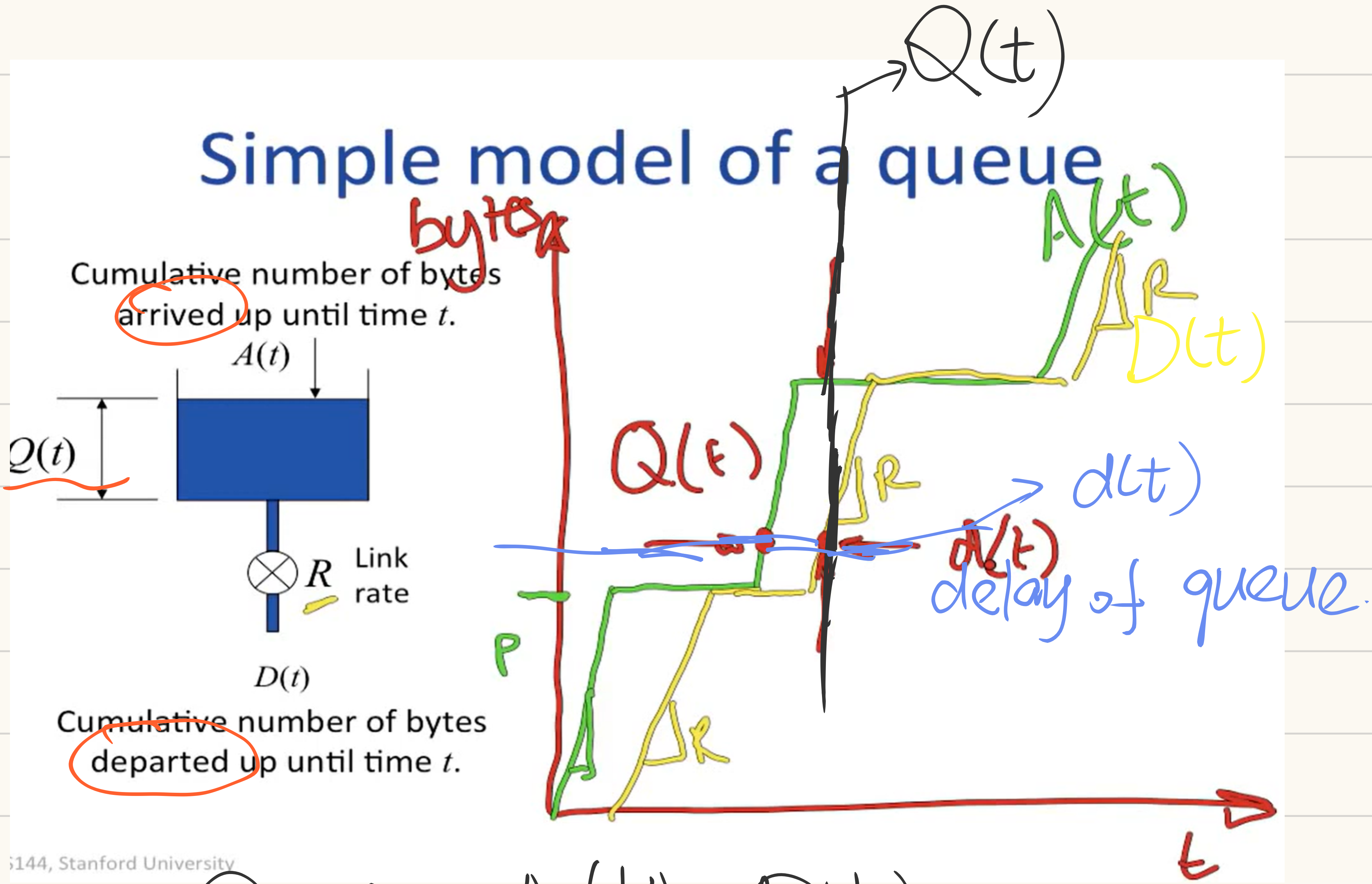


Playback buffer

- With packet switching, end-to-end delay is variable.
- We use a playback buffer to absorb the variation.

- We could just make the playback buffer very big, but then the video would be delayed at the start.
- Therefore, applications estimate the delay, set the playback buffer, and resize the buffer if the delay changes.

Queue Models

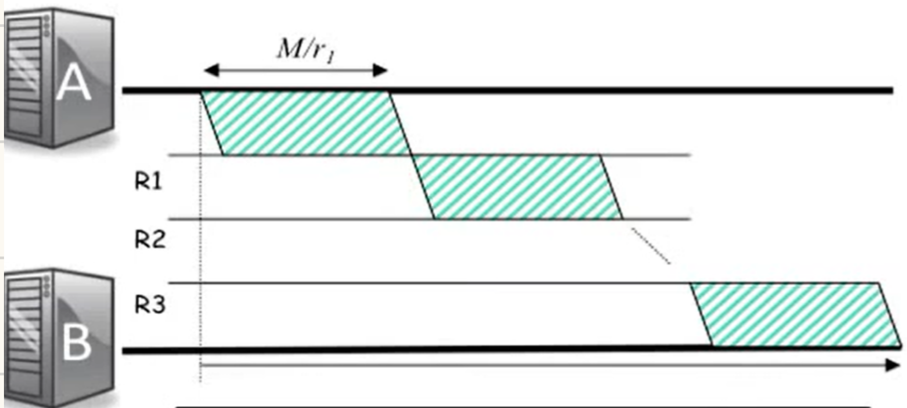


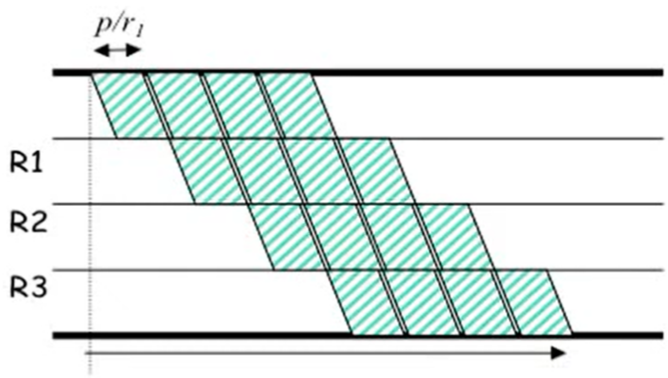
$$Q(t) = A(t) - D(t)$$

$d(t)$: time spent in the queue by a byte arriving at time t.

Packet Switching

Why not send the entire message in one packet?





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

$\frac{M}{r_3}$

End-to-end delay, $t = \sum_i \left(\frac{p}{r_i} + \frac{l_i}{c} \right) + \cancel{\frac{M}{p} r_3}$

Breaking message into packets allows parallel transmission across all links, reducing end to end latency.

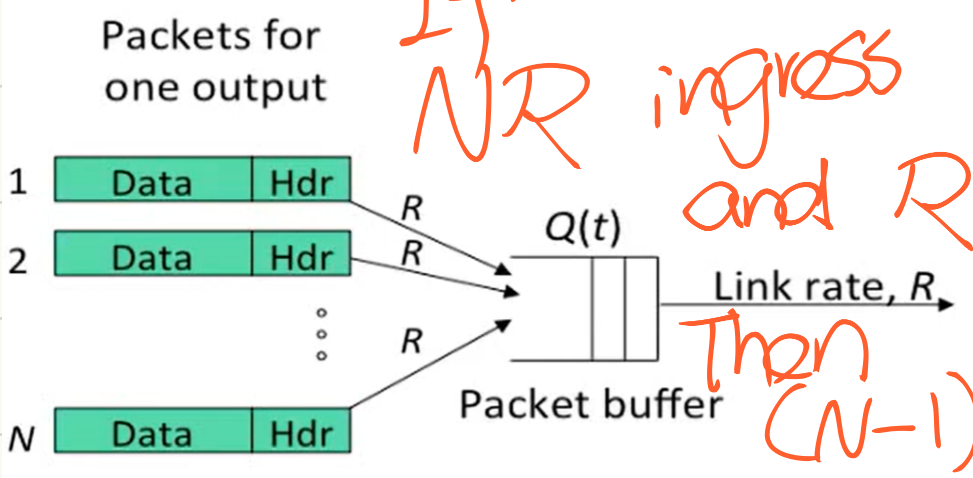
Correction: The second term should be $(M/p - 1) \cdot p/r_{min}$, where r_{min} is the rate of the slowest link.

the same as $\frac{M}{r_3}$

which means the packetization delay of that set of packets over the last

Packet Switching

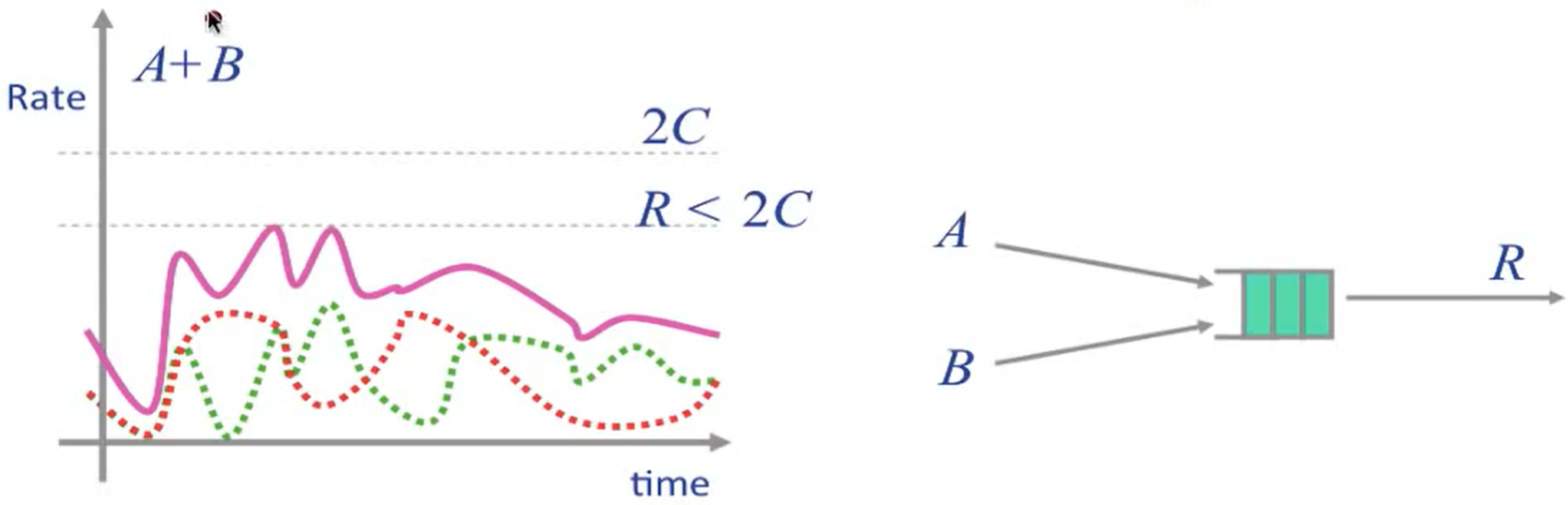
Statistical Multiplexing



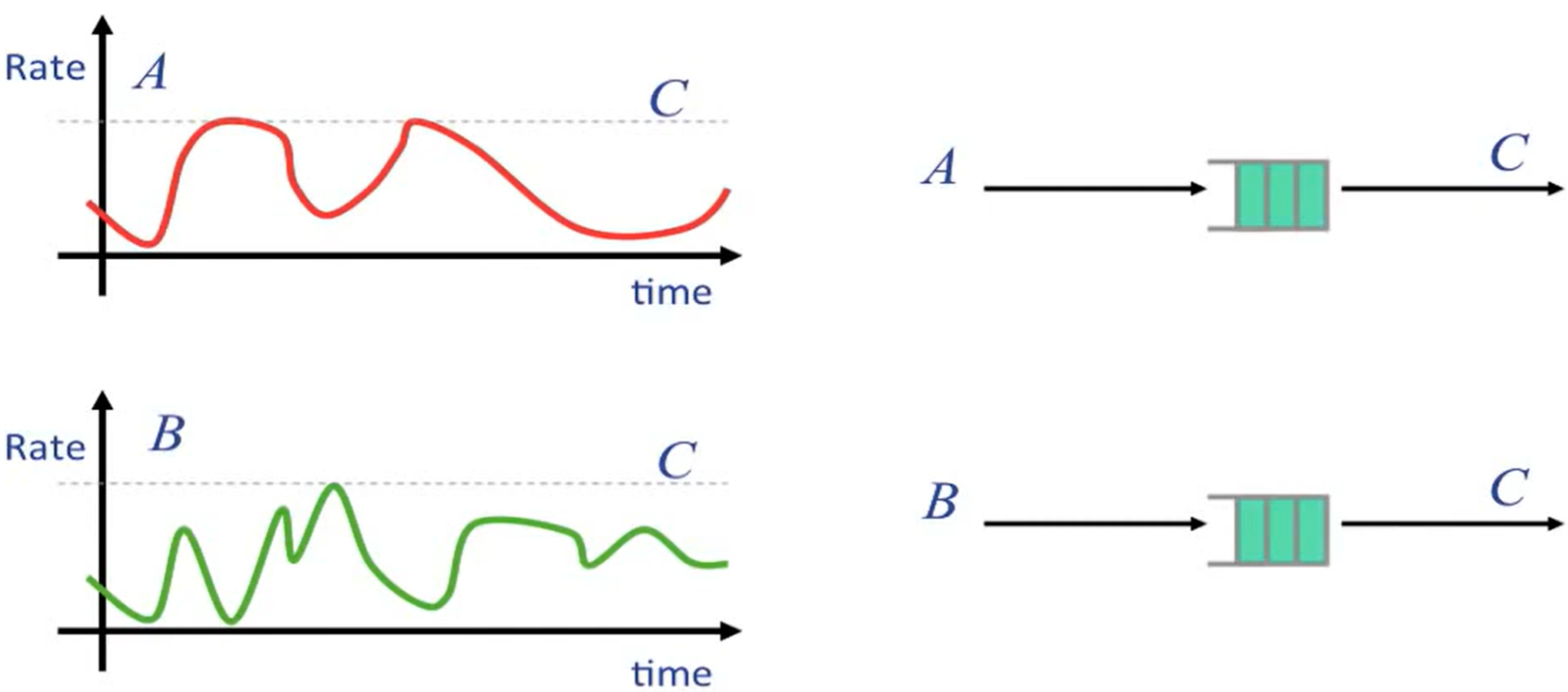
If NR ingress and R egress Then (N−1)R Loss

- Statistical multiplexing means the egress link need not run at rate NR.
- The buffer absorbs brief periods when the aggregate rate exceeds R.
- Because the buffer has finite size losses can occur.

Statistical Multiplexing Gain



Statistical multiplexing gain = $2C/R$



Useful Queue Properties

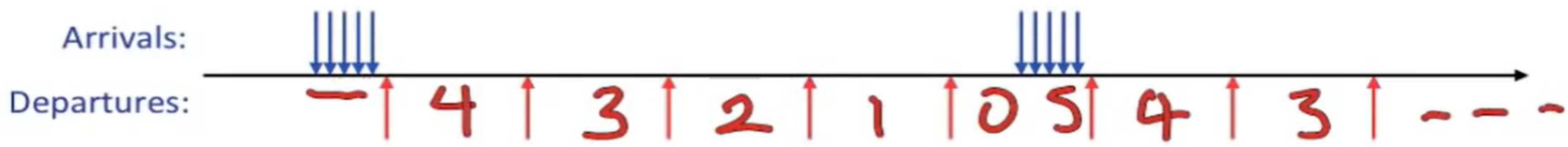
#1 "Burstiness increases delay"

Example 1: Periodic single arrivals 1 pkt/sec

1 0 1 0 1 0 1 0 - - -

$Q(t) \leq 1$

Example 2: Periodic burst arrivals N every N secs



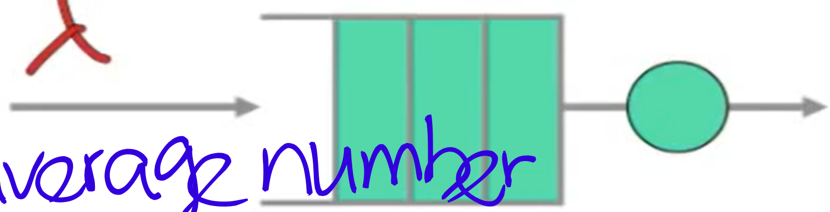
$Q(t) = 0, \ldots, 5$

#2 "Determinism minimizes delay"

counterbalance of the first
i.e. random arrives wait longer on average than simple periodic arrivals.

#3. "Little's Result"

λ → well-defined arrival rate



the average number in the queue ← L
(the average occupancy of a queue)

d = average delay

$$L = \lambda d$$

If none lost or dropped

4 "M/M/1 Queue"

The Poisson process

An arrival process is Poisson if:

1. Pr{k arrivals in an interval of t seconds} is

$$P_k(t) = \frac{(\lambda t)^k}{k!} e^{-\lambda t}$$

2. $E[\text{number of arrivals in interval } t] = \lambda t$ — arrival rate
3. Successive interarrival times are independent (i.e. not bursty).

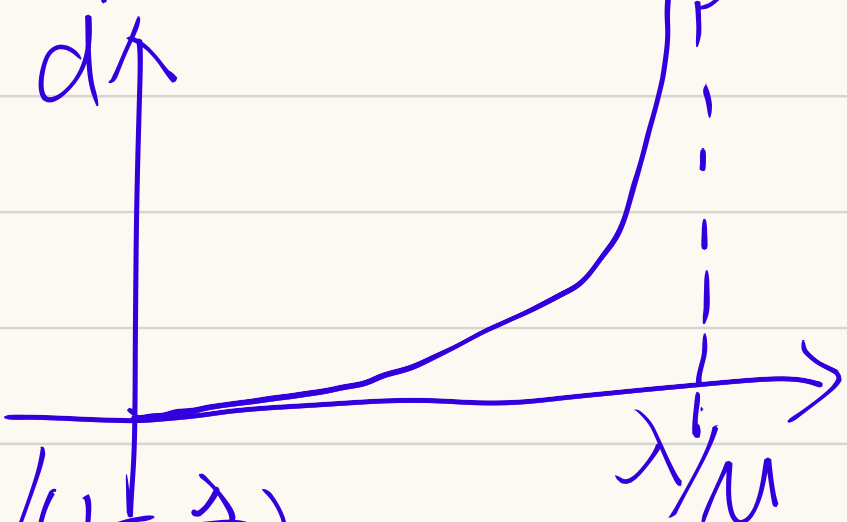
Be warned:

① Network traffic is very bursty!
② Packet arrivals are not Poisson.
③ But it models quite well the arrival of new flows.

M/M/1

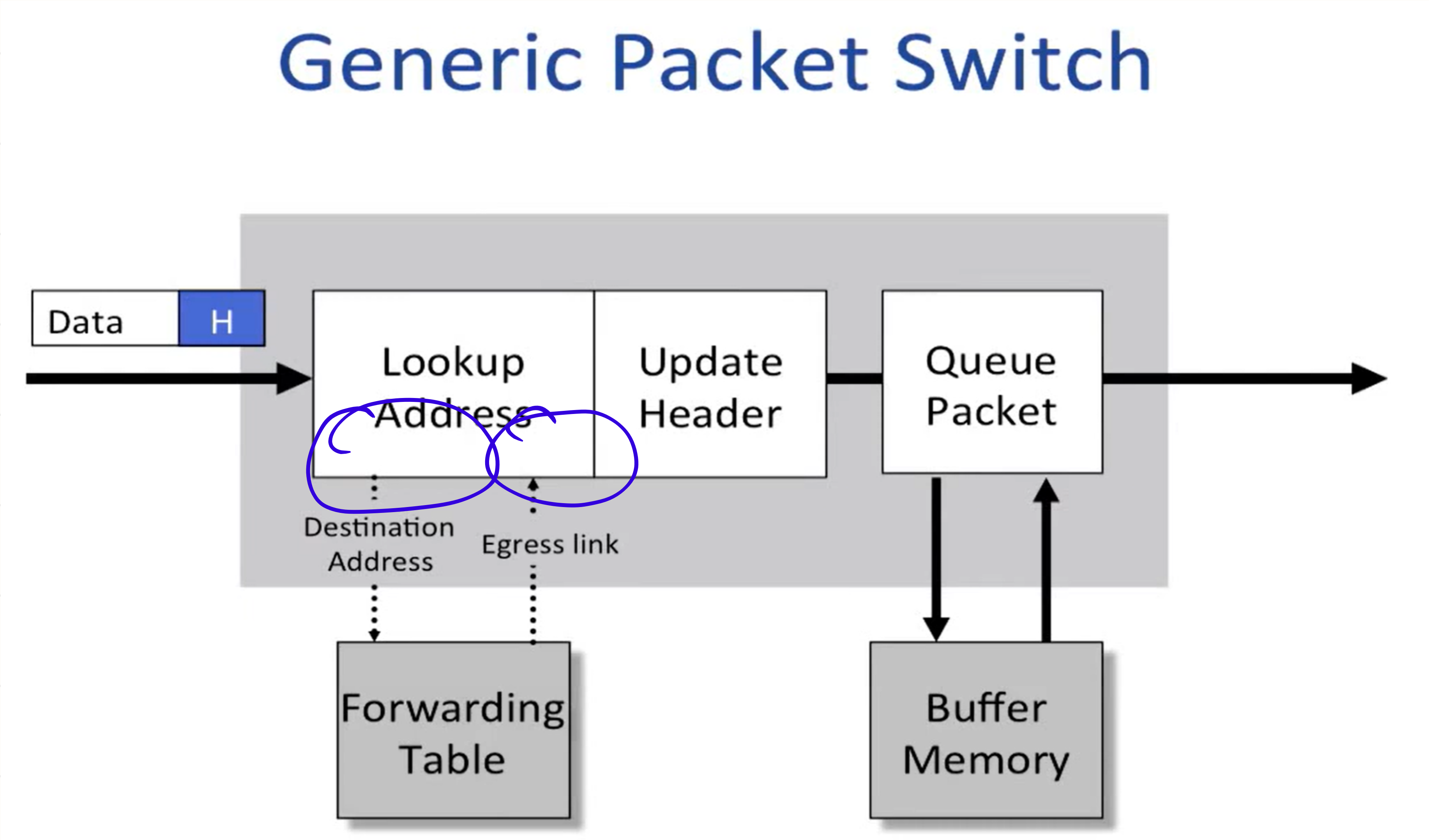
- M: Markovian arrival process (Possion)
- M: Markovian service process (Exponential)
- 1: server (One outgoing line serving this queue)

$$d = \frac{1}{\mu - \lambda}$$



$$L = \lambda d = \frac{\lambda}{\mu - \lambda} = \frac{\lambda}{\mu} \Big/ \left(1 - \frac{\lambda}{\mu}\right)$$

How a packet switch works



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.

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.

if it does, drop the packet

the checksum includes the TTL

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.

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 2-way hash)
- Look for exact match in hash table

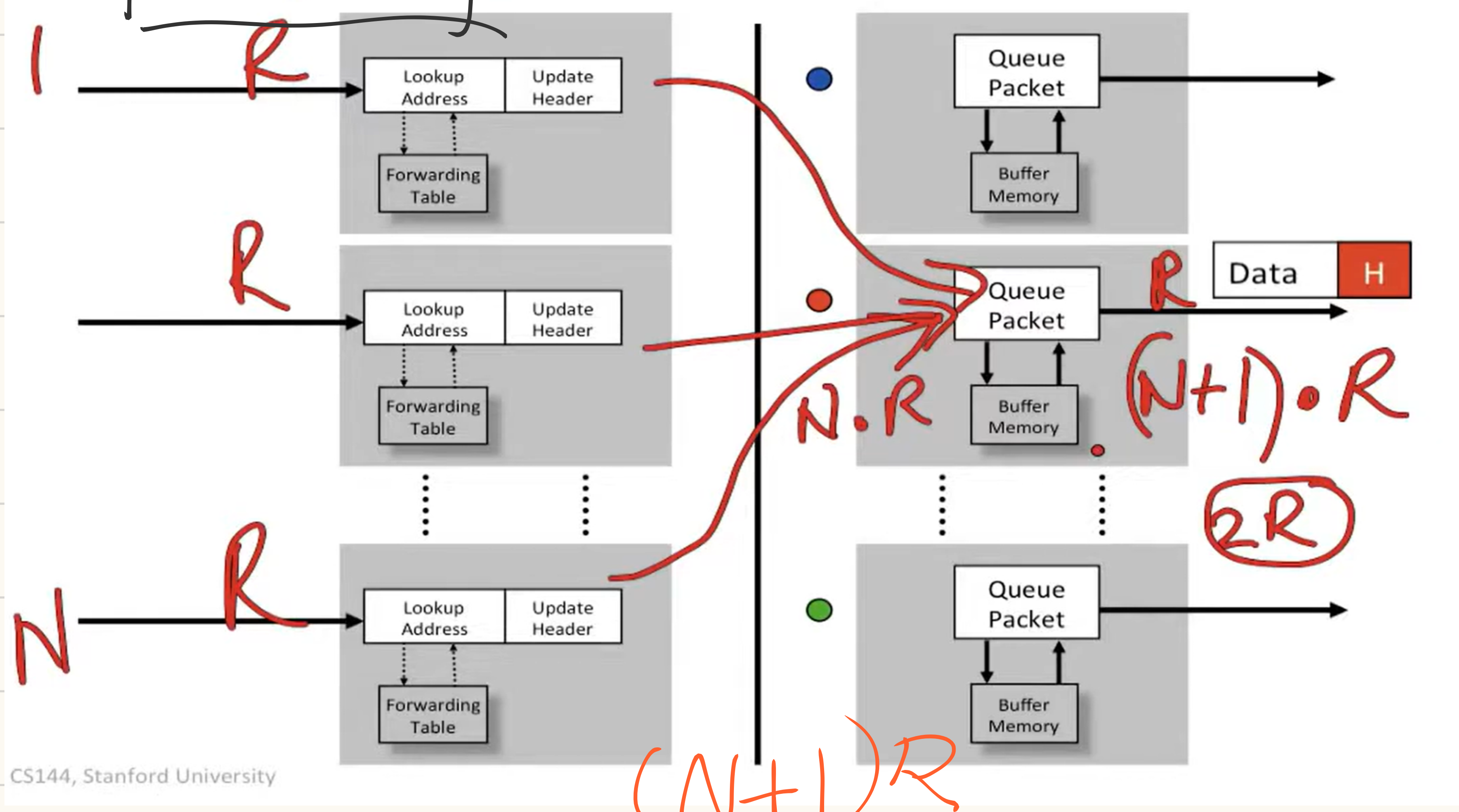
Lookup Address: IP

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 <u>longest prefix match</u>, not an exact match

Output Queued Packet Switch

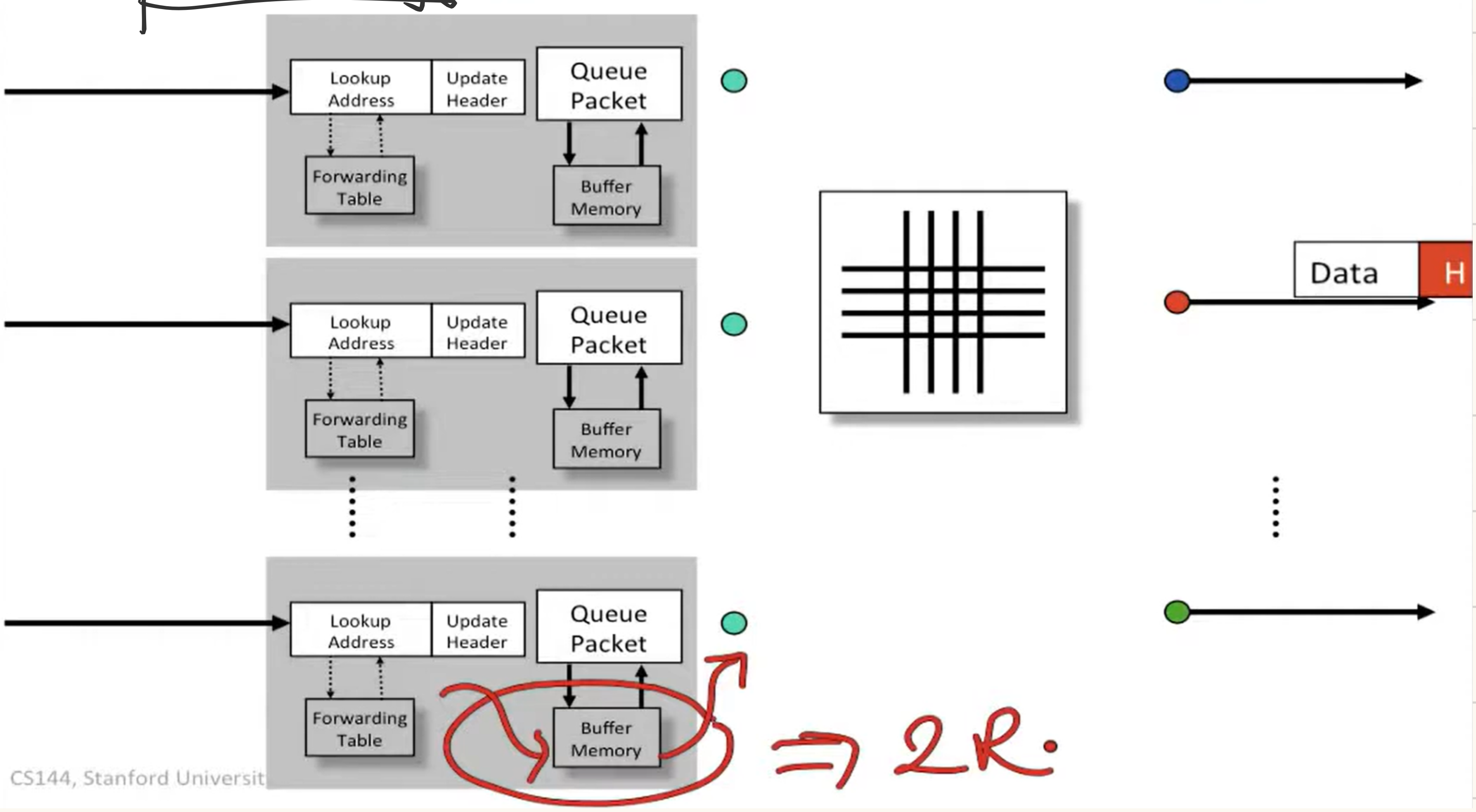


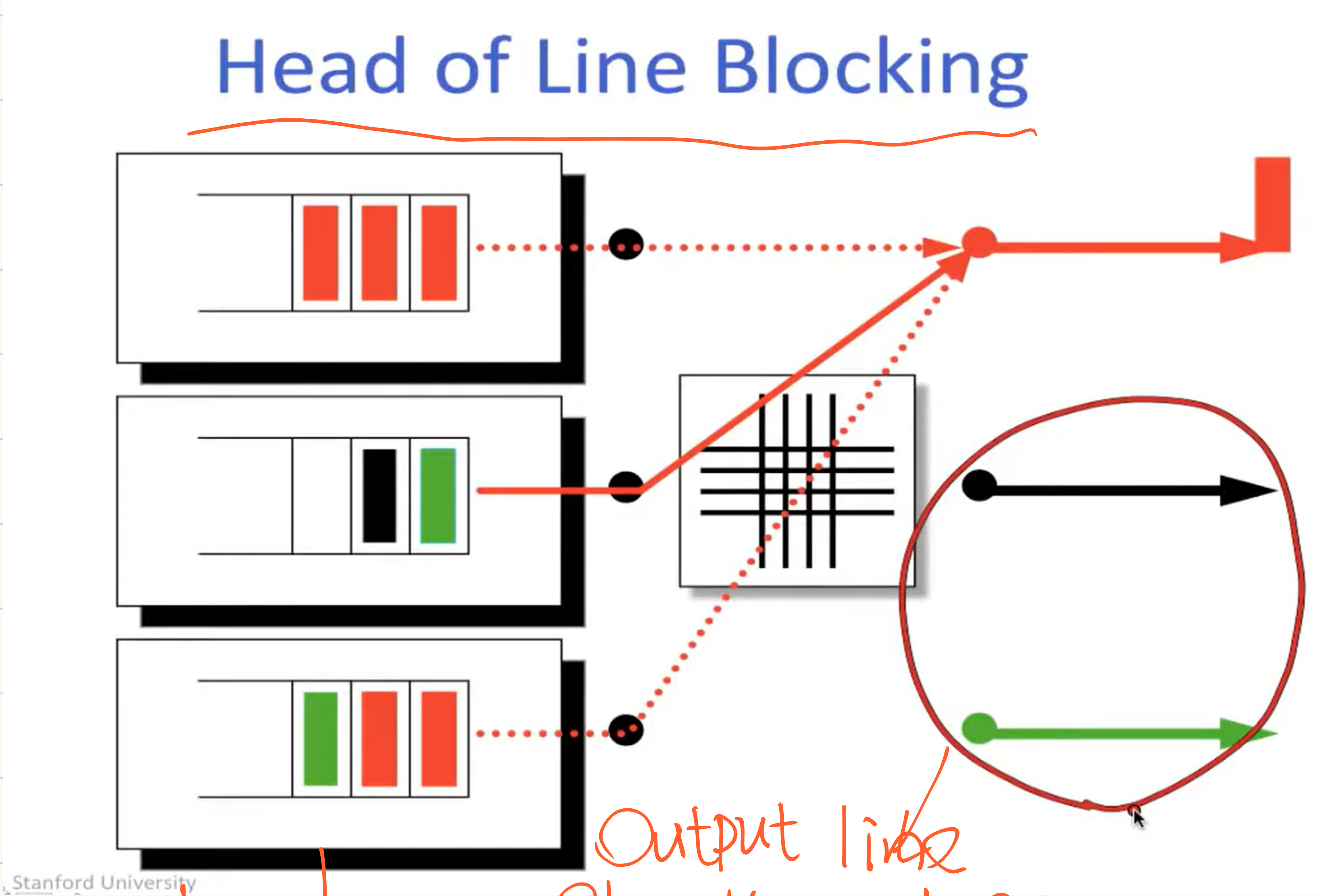
(N+1)R

↓

2R

Input Queued Packet Switch

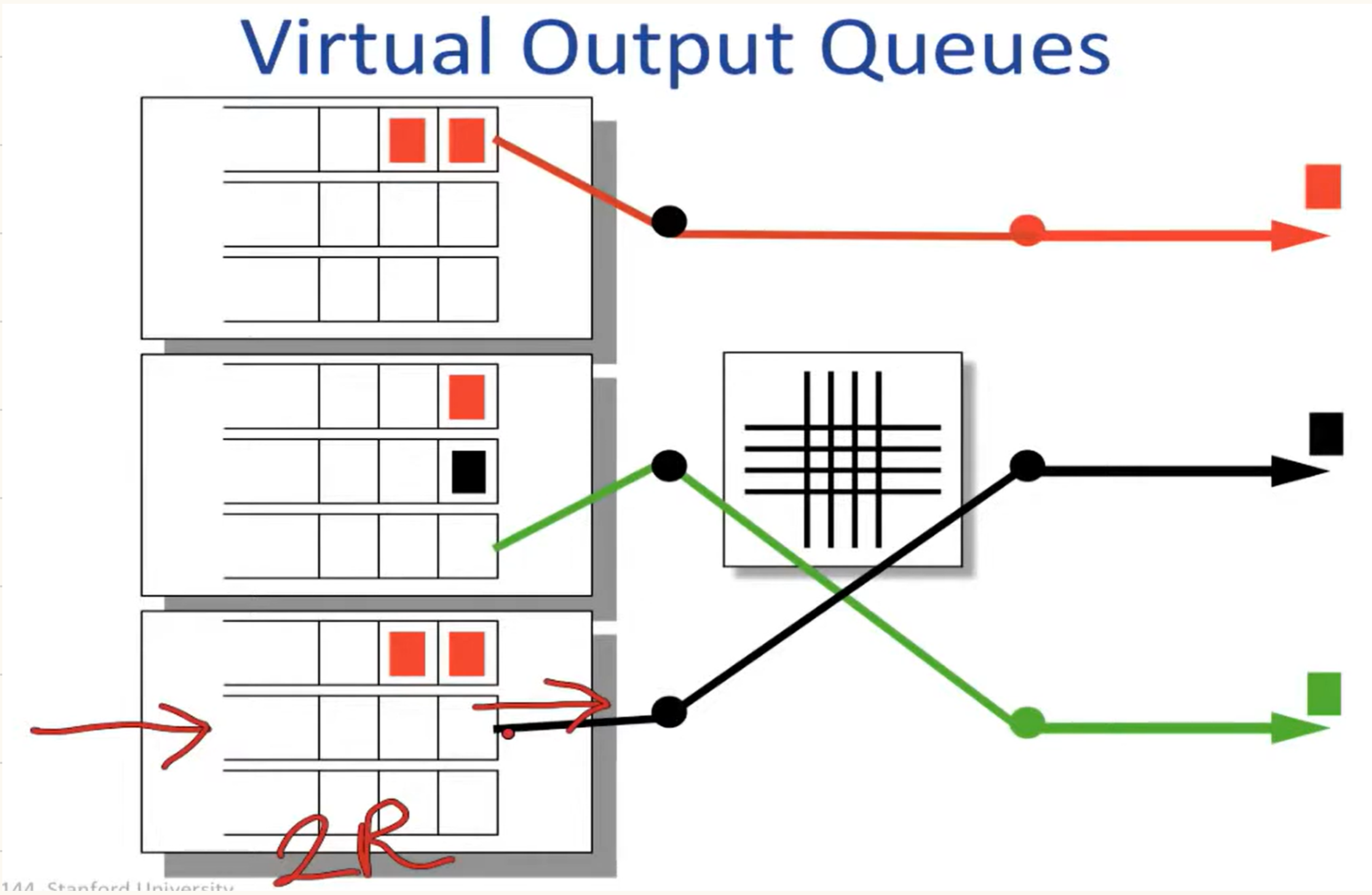




Output line

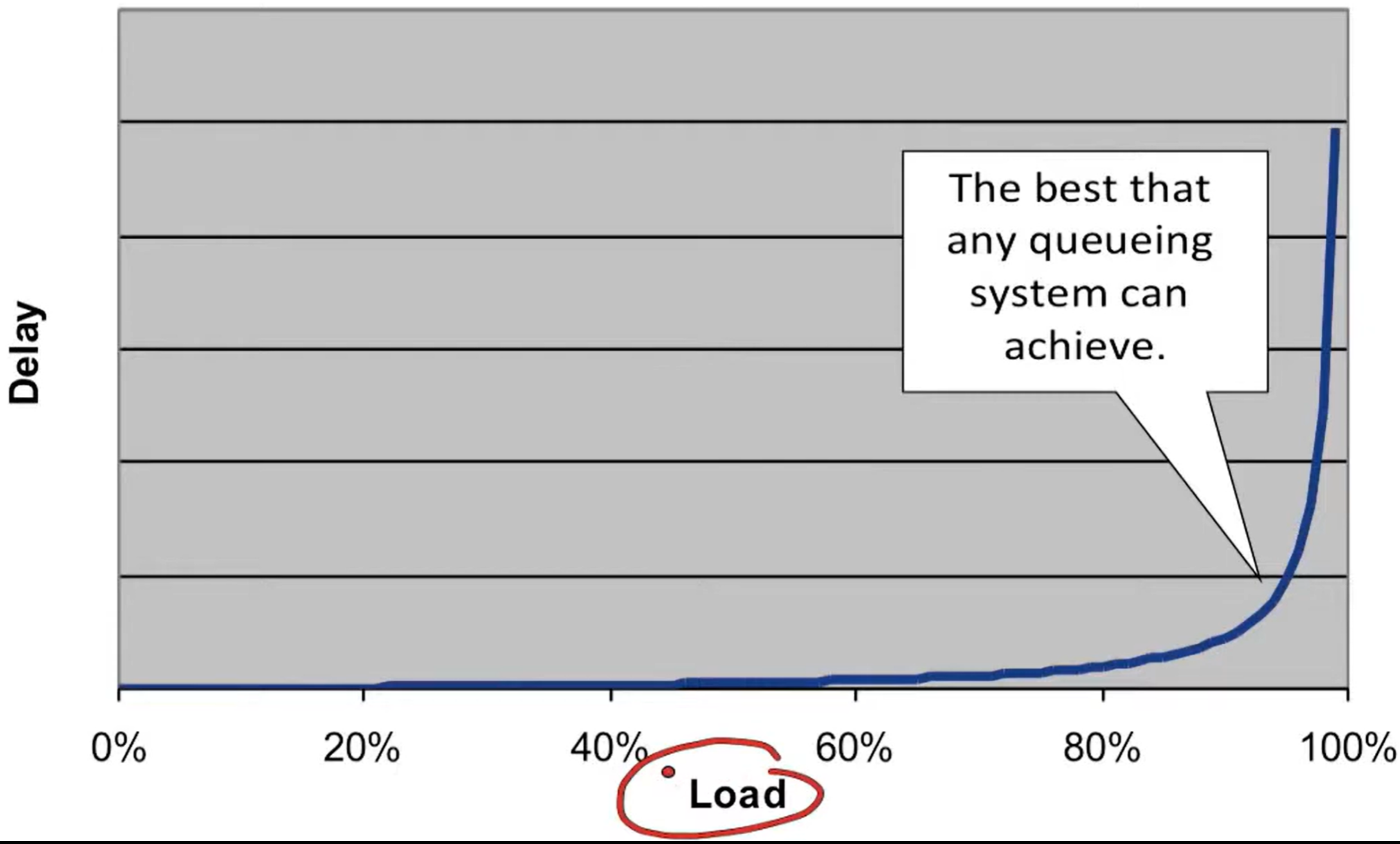
Not be blocked

Black and green are in idle



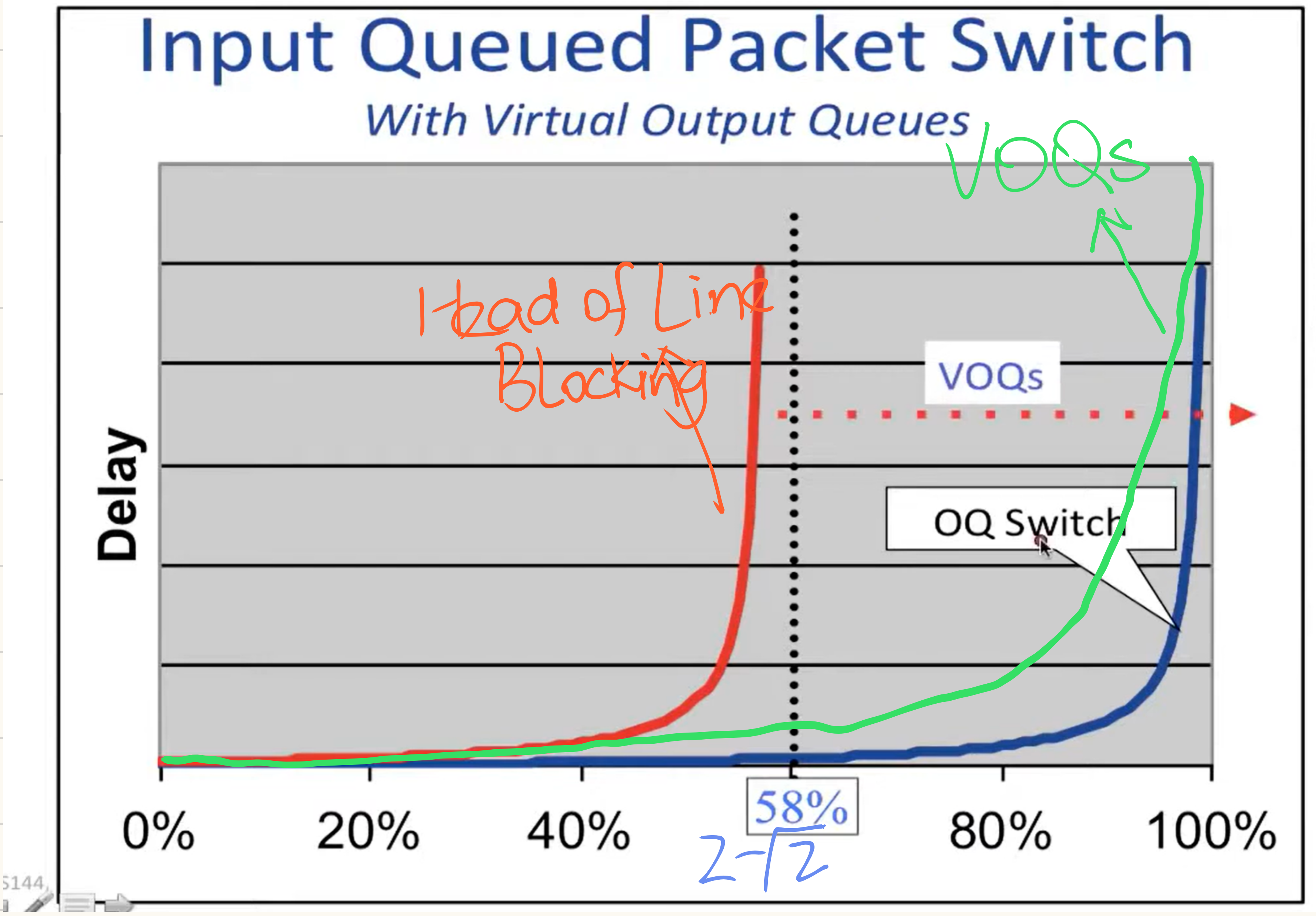
2R

Output Queued Packet Switch



Properties:

① They are "work conserving"
② Throughput is maximized
③ Expected delay is minimized



Packet switches perform 2 basic operations

① Lookup addresses in a forwarding table

② Switching to the correct egress port.

The simplest and slowest switches use output queueing, which minimizes packet delay

Higher performance switches often use input queueing, with virtual output queue to maximize throughput.

Strict priorities and guaranteed flow rates

$\boxed{\text{FIFO queues}}$ are a free of all: No priorities or guaranteed rates.

$\boxed{\text{Strict Priorities:}}$ High priority traffic "sees" a network without low priority traffic. Useful if we have limited amounts of high priority traffic.

$\boxed{\text{Weighted Fair Queueing (WFQ)}}$ lets us give each flow a guaranteed service rate, by scheduling them in order of their bit-by-bit finishing time (also pkt-by-pkt finishing time)

Guaranteed Delay

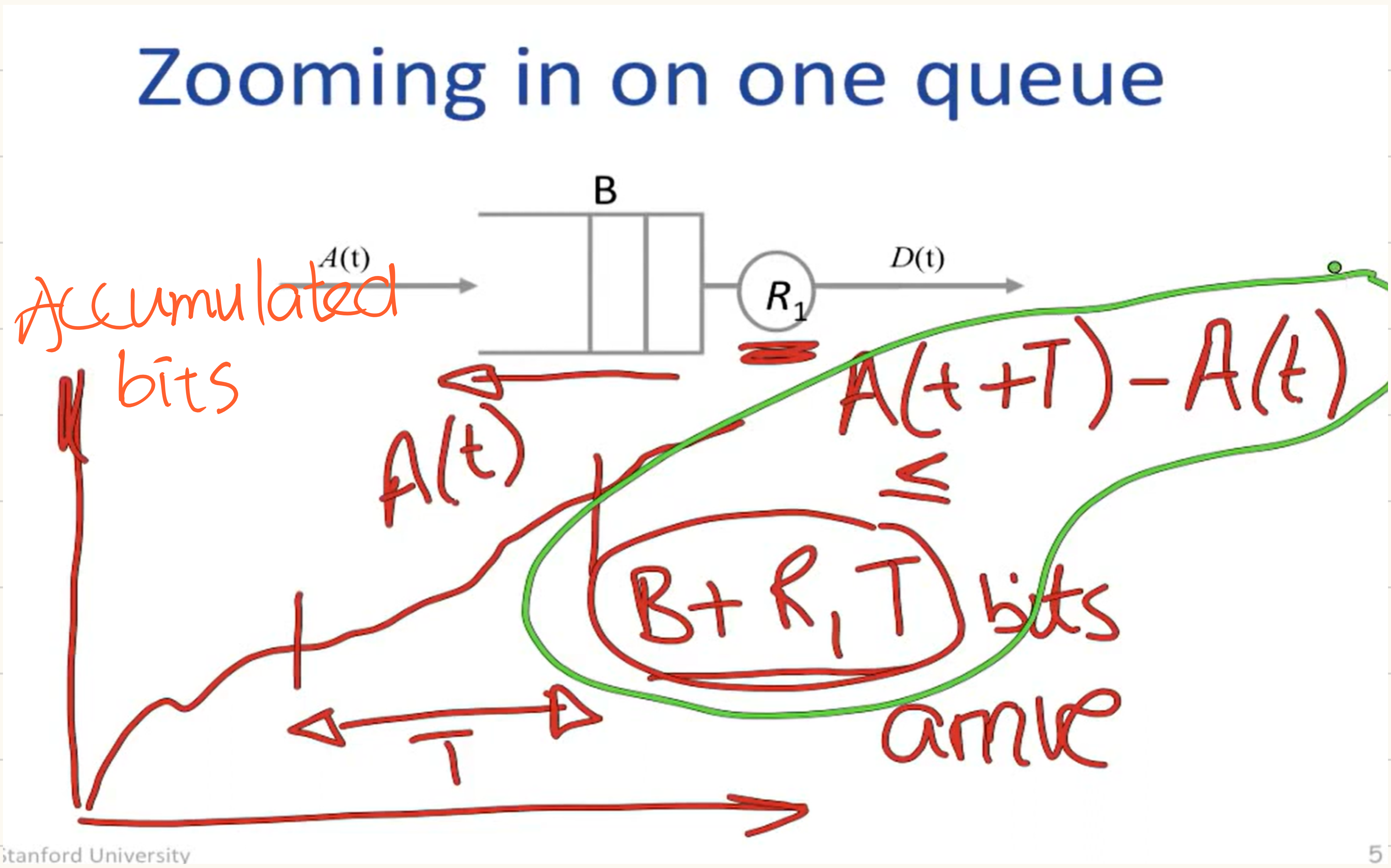
So how can we control the delay of packets?

What we already know how to control:

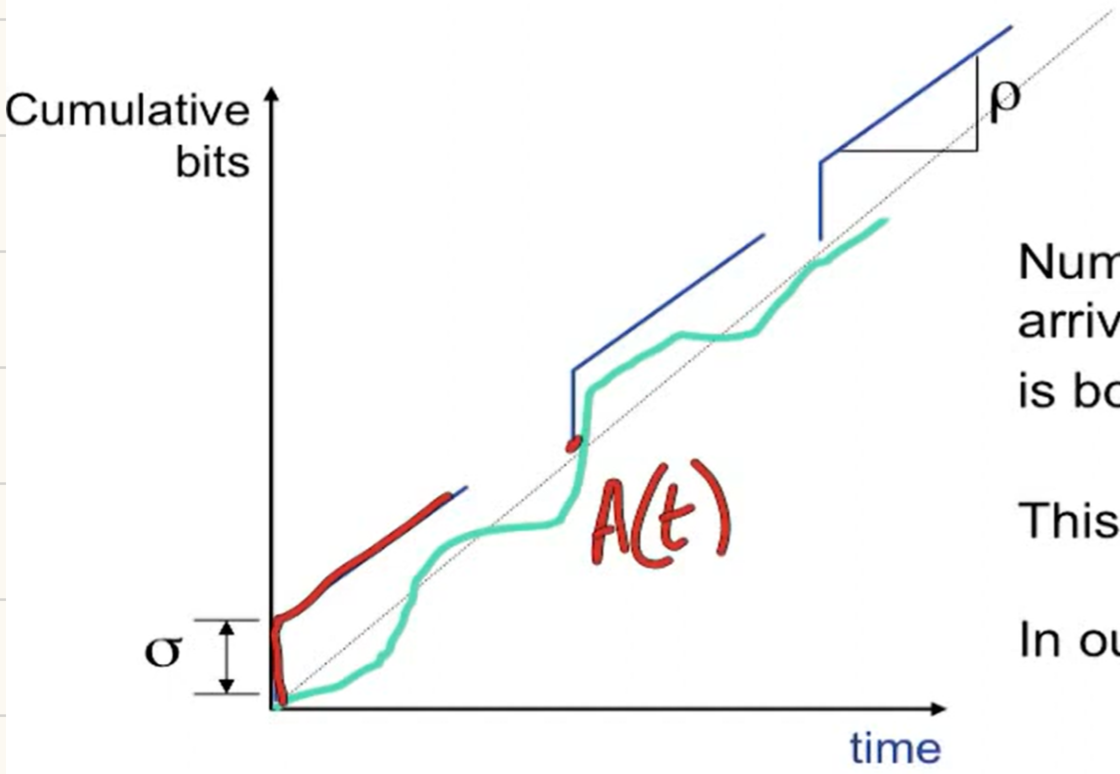
① The rate at which a queue is served (WFQ)

② The size of each queue

And how to we make sure no packets are dropped?



Constraining traffic

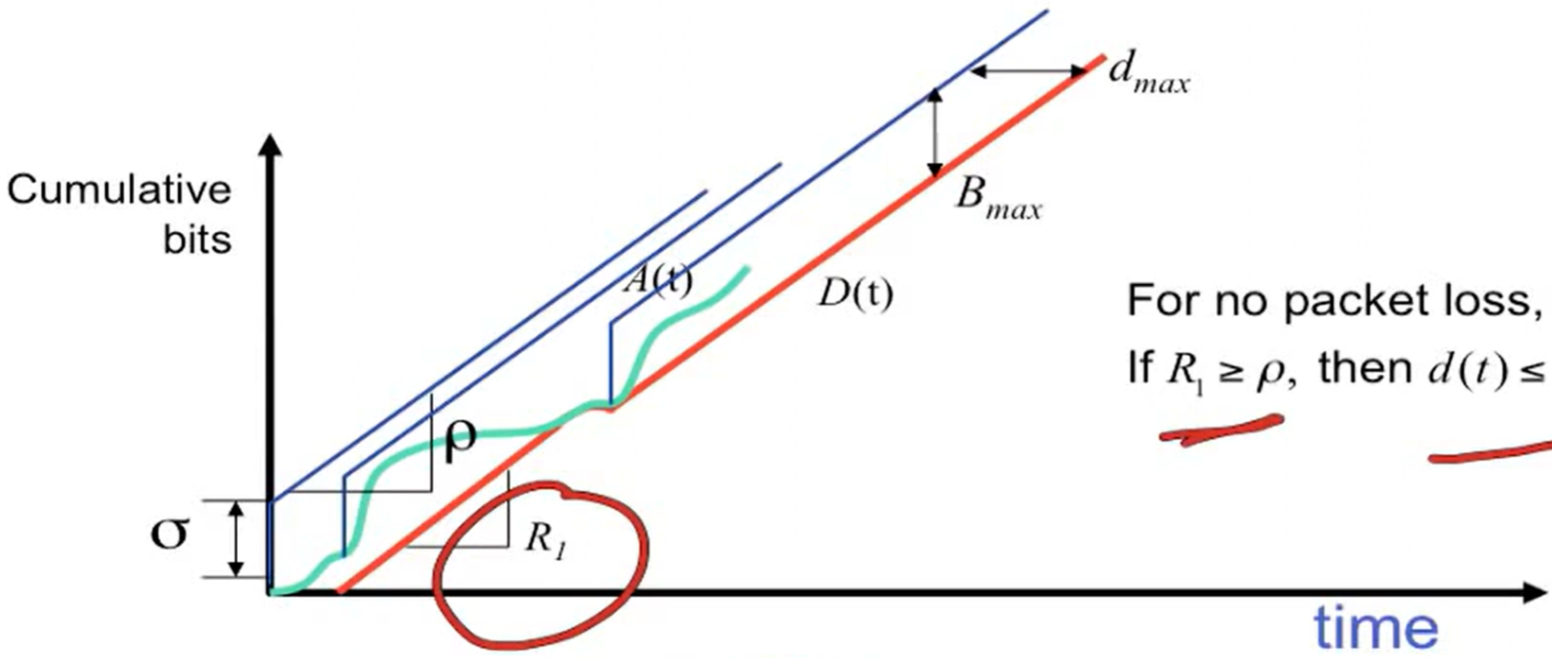


Number of bits that can arrive in any period of length t is bounded by: $\sigma + \rho t$

This is called "(σ,ρ) regulation"

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

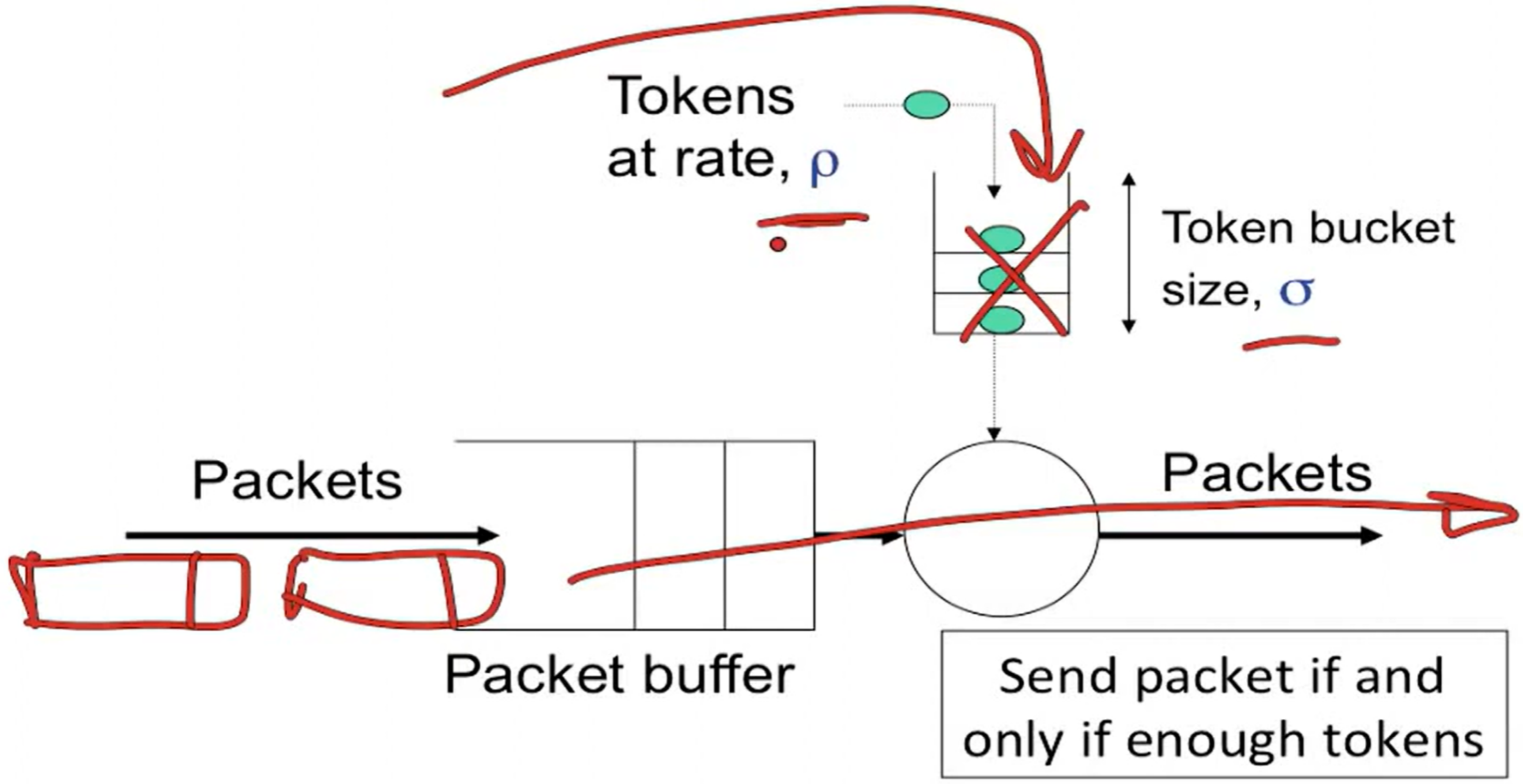
(σ,ρ)-constrained Arrivals and Minimum Service Rate



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



This will make sure we're allowing for bursts of up to σ but over on the long term rate is only ρ.

While it's technically possible to do so, very few networks actually control end to end delay

Why?

① It is complicated to make work, requiring coordination

② In most networks, a combination of over-provisioning and priorities work well enough.