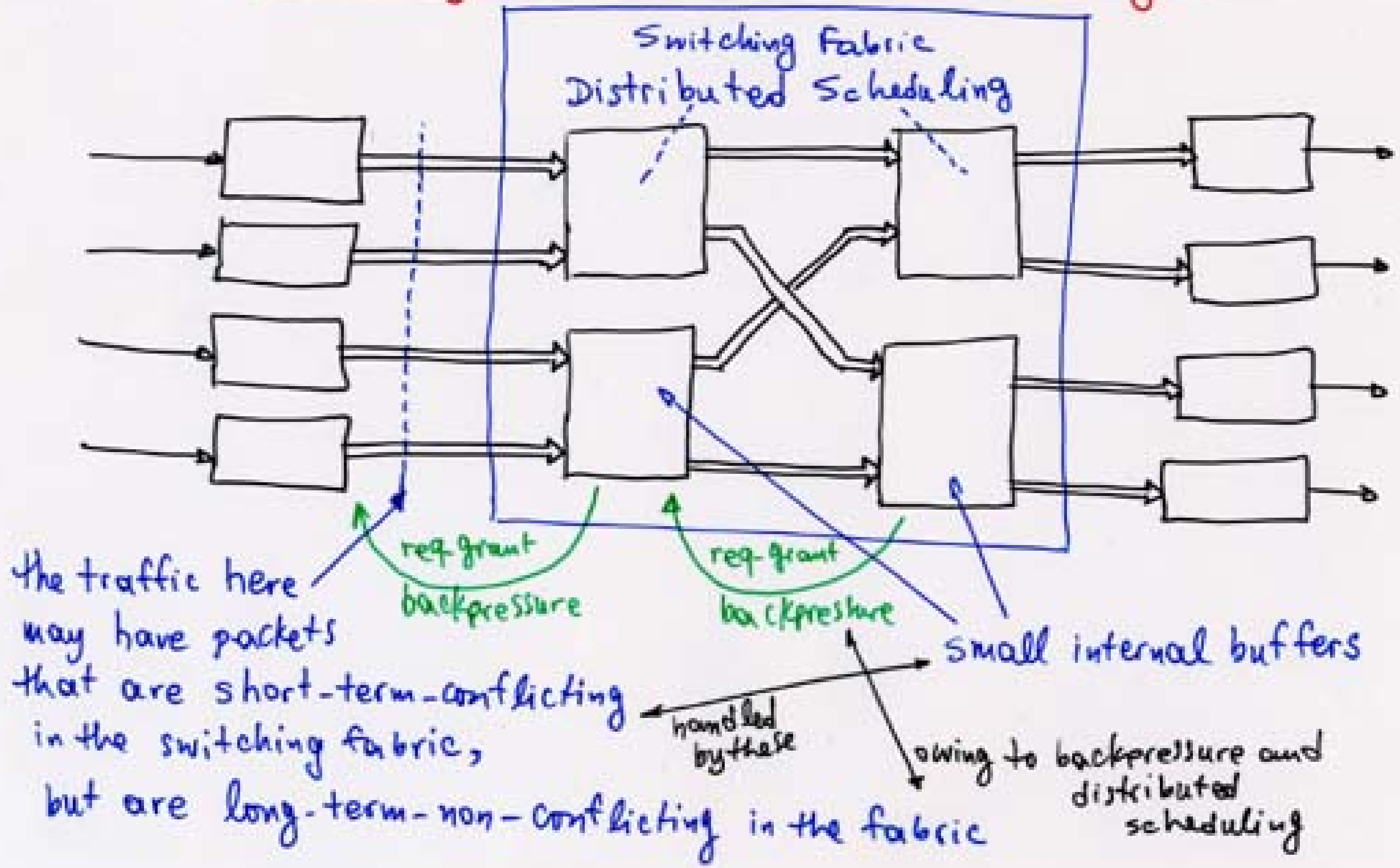
6.1 Credit-Based Flow Control (Backpressure)

- Summary to be added here

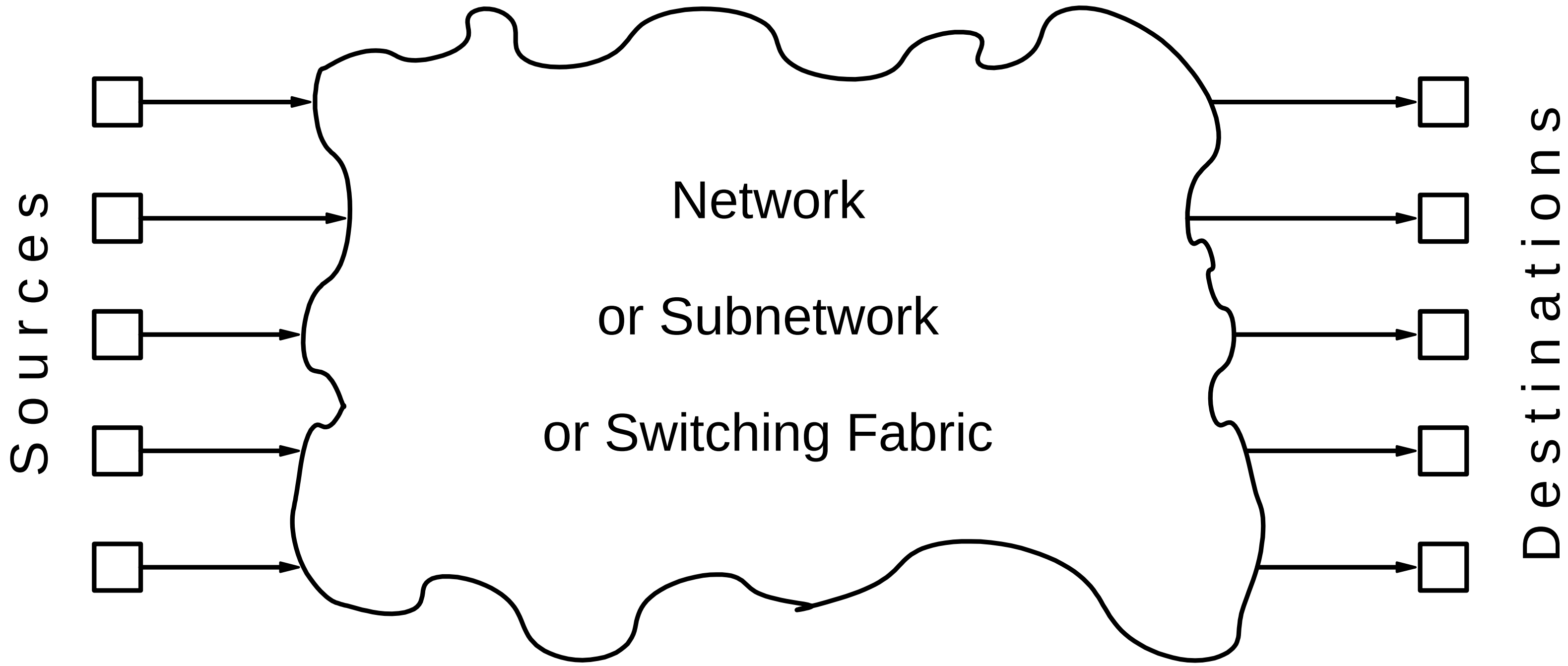
Central Scheduler is Impractical for large N

Solution 2: Switching Fabrics with Internal Buffering & Backpressure



FLOW CONTROL

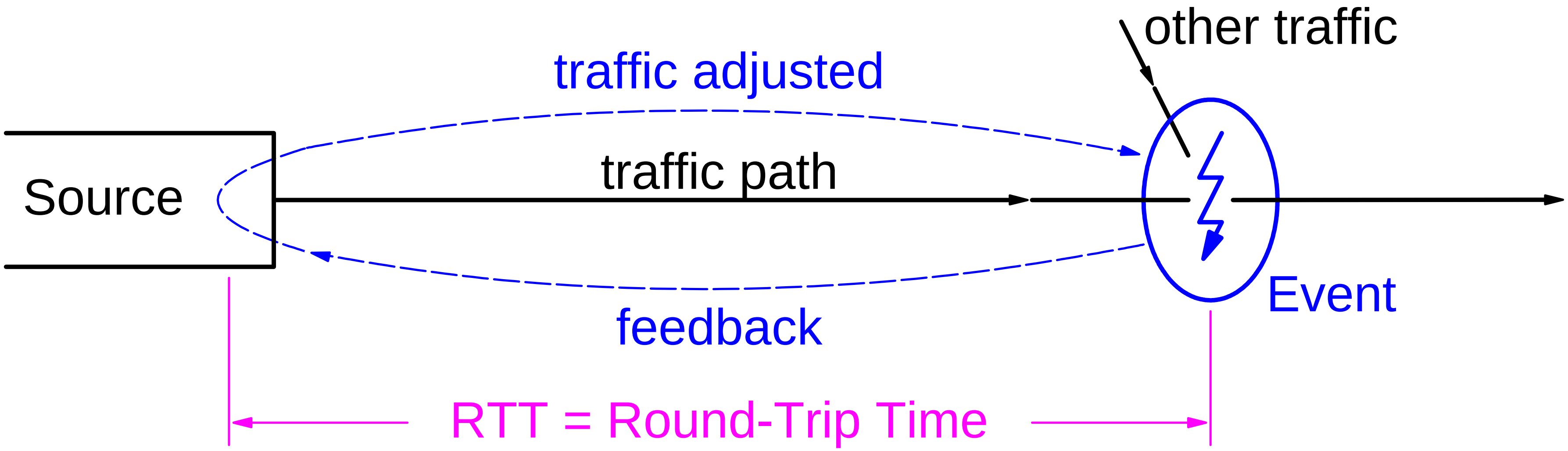
a feedback control problem



- How do the sources know at which rate to transmit?
- How do the sources know when their (collective!) demands exceed the network or the destination capacity?

⇒ Answer: FEEDBACK from the contention point(s), in the network or at the destinations, to the sources

RTT: the Fundamental Time-Constant of Feedback



The traffic is ``blind'' during a time interval of RTT:

- the source will only learn about the effects of a transmission RTT after this transmission has started (or RTT after a request for such transmission has been issued)
- the (corrective) effects of a contention event will only appear at the contention point RTT after the event occurrence

``Blind Mode'' bits in faster Networks

For faster networks:
- start transmitting earlier
- start transmitting at a higher rate

⇓

For networks to get faster,

an increasing number of bits must be sent in ``blind mode''

initial (``blind mode'') rate of transmission	amount of data xmitted in ``blind mode''	
	distance = 8 km RTT ~= 0.08 ms	distance = 8,000 km RTT ~= 80 ms
1 Mbit/s	10 Bytes	10 KBytes
1 Gbit/s	10 KBytes	10 Mbytes

Lossy versus Lossless Flow Control

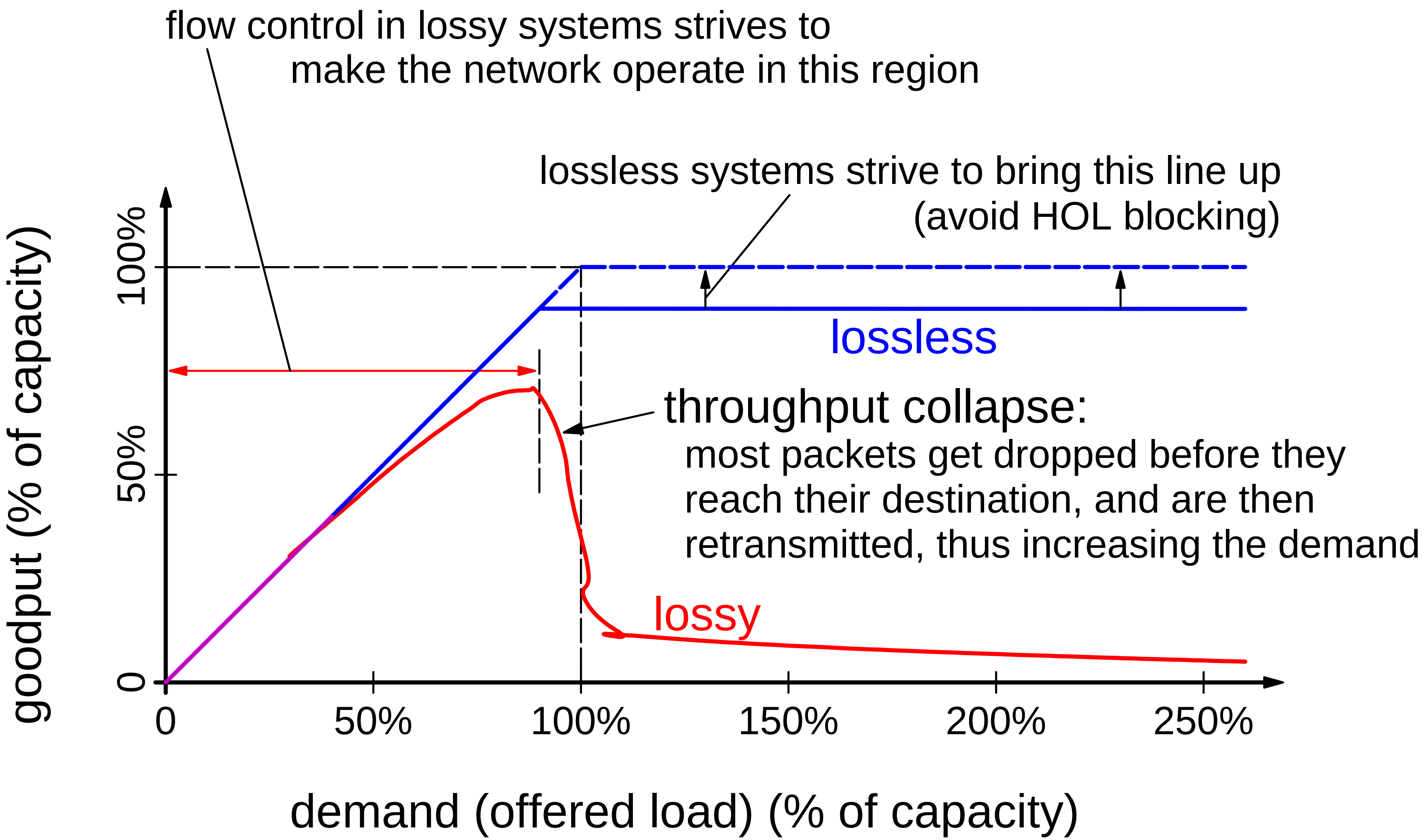
Lossy: *flow control may fail to prevent buffer overflows: packets can be dr*

- inherited from ``communications engineers'': same as electrical noise
- simple switches
- for data: need retransmissions => long delays, complex if in H/W
- wastes communication capacity: ``goodput'' versus throughput
- need carefully designed protocols to sustain satisfactory goodput

Lossless: *flow control guarantees that buffers will never overflow*

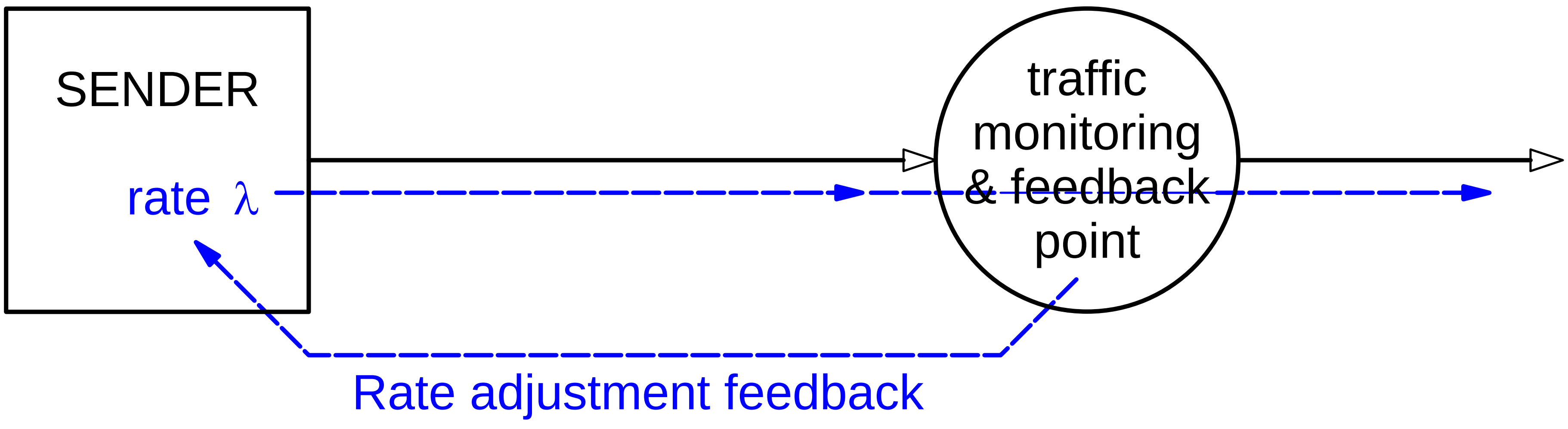
- inherited from ``hardware engineers'': processors never drop data
- no wasted communication capacity, minimizes delay
- need multilane protocols to avoid HOL blocking & deadlocks
- switches are more complex, need H/W support for high speed

Goodput versus Demand in Lossy and Lossless Flow Control



(this slide intentionally left blank)

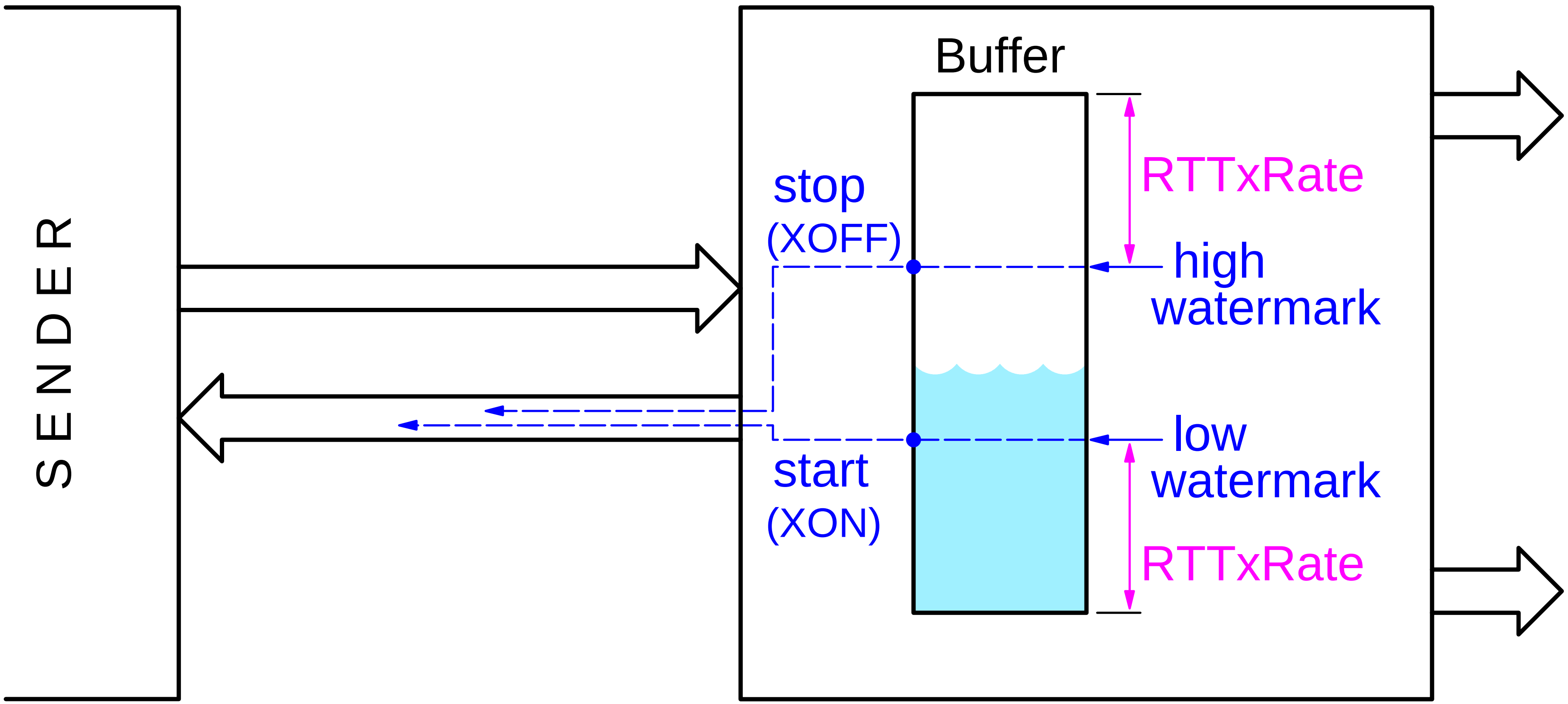
Rate-Based Flow Control



- differential (speed-up / slow-down), or
- absolute (new rate := value)

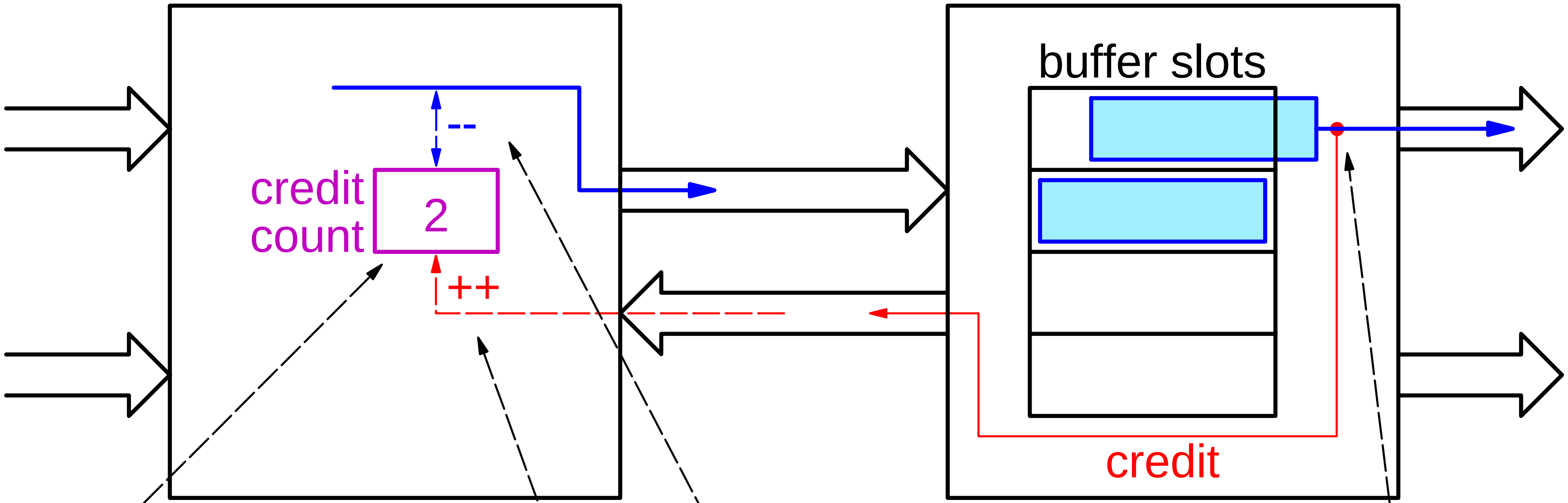
Note: oftentimes, the sender uses a variable-size window mechanism in order to control its rate

ON/OFF (start/stop) (XON/XOFF): simplistic Rate-based FC



- ``start'' $\equiv$ (rate := peak);   ``stop'' $\equiv$ (rate := 0)
- rate-based flow control used for lossless transfers
- less than half the buffer efficiency of credit-based flow control

Credit-based (window) (backpressure) Flow Control

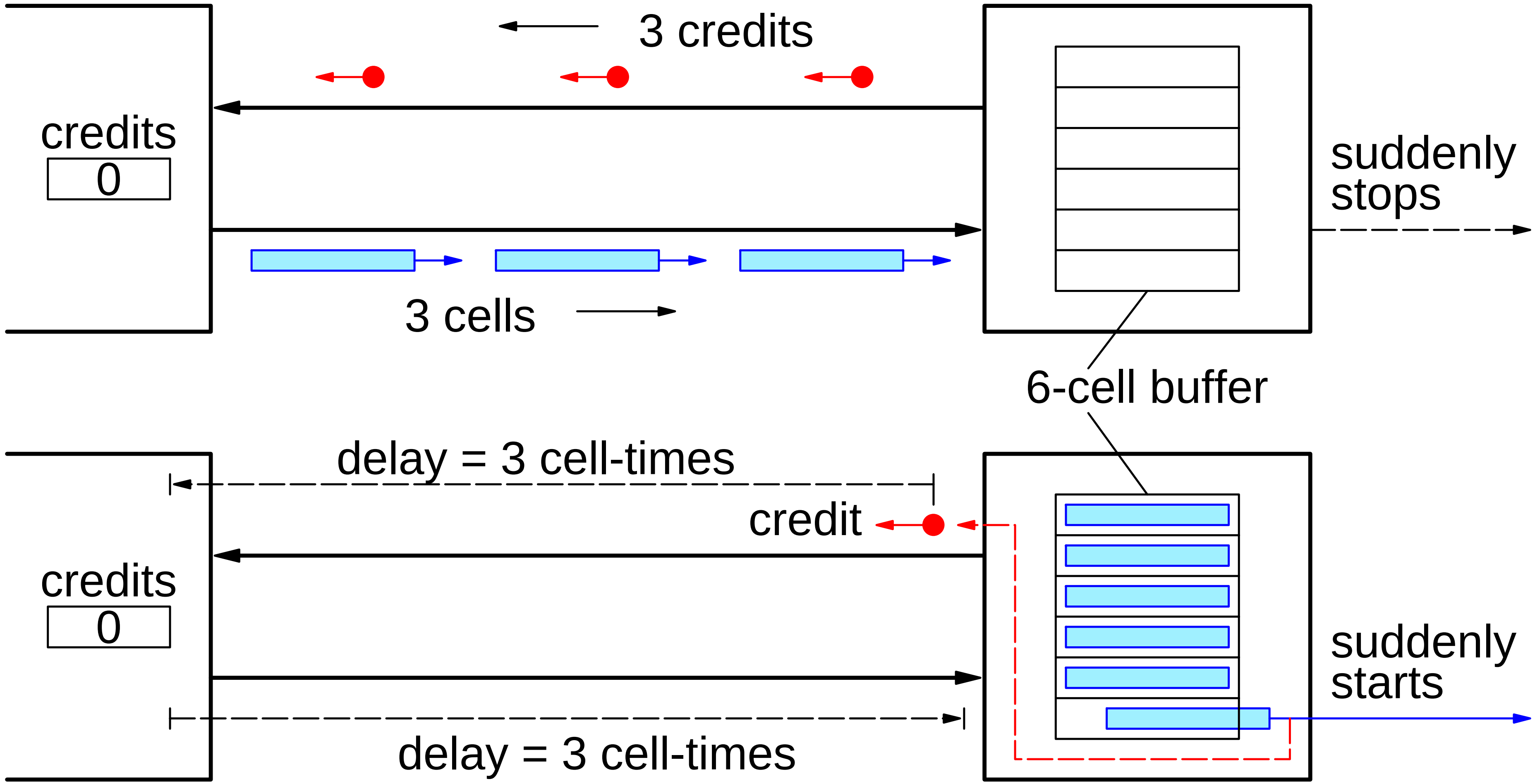


- count of buffer slots known to be available at the downstream site (not allowed to go negative)
- traffic can only depart if and when it acquires (decrements) the credit(s) that correspond to the buffer slot(s) needed
- arriving credits increment the credit count
- when new buffer slots are made available, corresponding credits are sent upstream

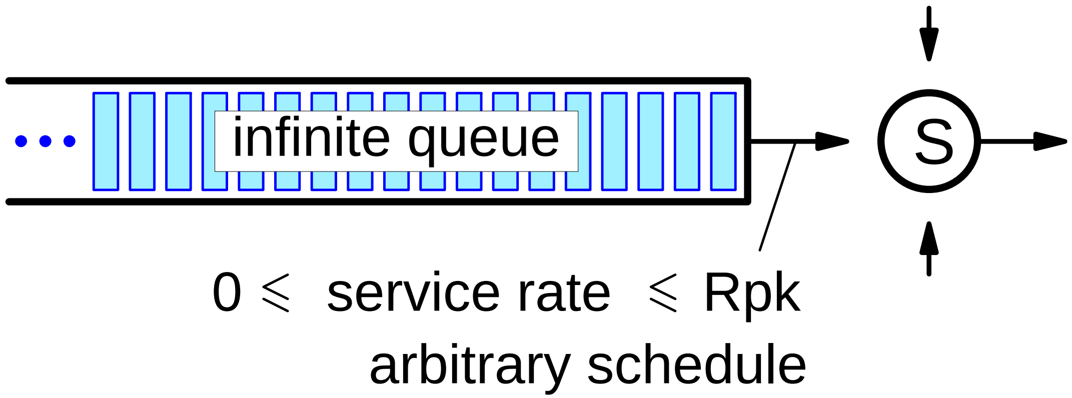
⇨ ***Lossless Flow Control***

Buffer Space = Peak Throughput x Round-Trip Time

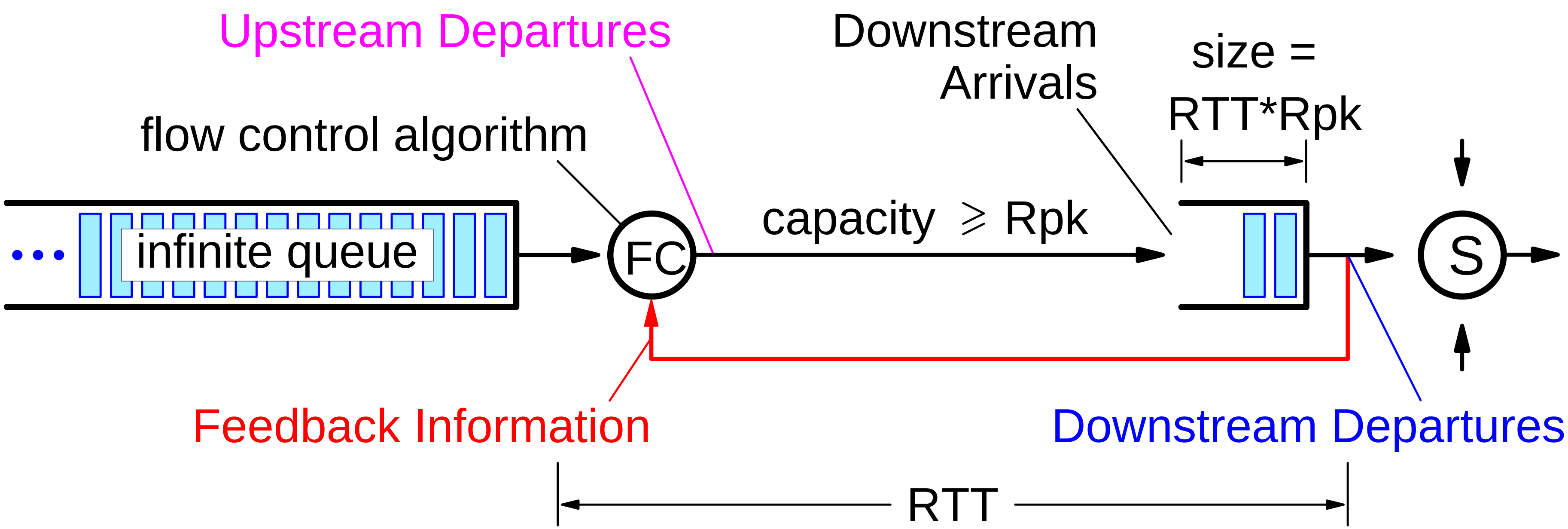
necessary & sufficient

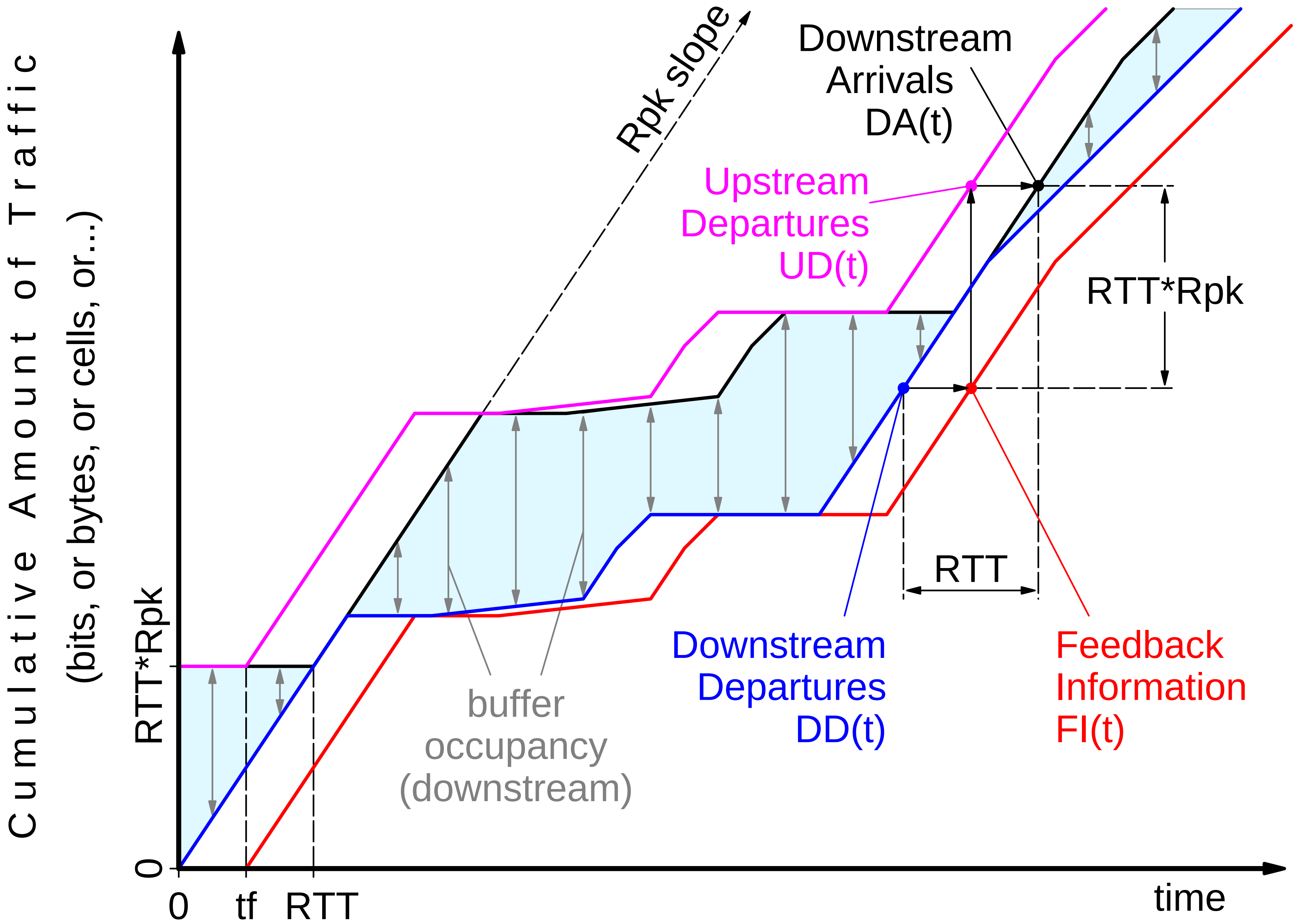


Theorem: Infinite Queue Push-Back



is equivalent to:




Cumulative Amount of Traffic
(bits, or bytes, or cells, or...)
RTT*Rpk
0
Rpk slope
Downstream
Arrivals
DA(t)
Upstream
Departures
UD(t)
RTT*Rpk
RTT
buffer
occupancy
(downstream)
Downstream
Departures
DD(t)
Feedback
Information
FI(t)
0
tf
RTT
time

- Downstream Departures DD(t) (cumulative):

  arbitrary function of time, provided that its slope satisfies:

$$0 \leqslant \text{service rate of S} \leqslant \text{Rpk} \qquad (1)$$

$$\Rightarrow \quad 0 \leqslant DD(t+d) - DD(t) \leqslant d*Rpk \qquad (2)$$

- Upstream Departures UD(t) (cumulative):

  UD(t) = DD(t-tf) + RTT*Rpk; this is always feasible, since:

  link capacity $\geqslant$ Rpk $\geqslant$ service rate of S

- Downstream Arrivals: DA(t) = DD(t-RTT) + RTT*Rpk (3)

- Buffer Occupancy (downstream): BO(t) = DA(t) - DD(t)

  (3) $\Rightarrow$ BO(t) = RTT*Rpk - [DD(t) - DD(t-RTT)]

  with (2) $\Rightarrow$ $0 \leqslant BO(t) \leqslant RTT*Rpk$

  ↑ feasibility of arbitrary departure schedule (provided (1) holds)

  ↑ downstream buffer never overflows

Feedback Format Options:

how to make the function DD(t) known to the upstream neighbor

① QFC Credit-Based Flow Control

② Classical (incremental) Credit-Based Flow Control

③ Rate-Based Flow Control

① Quantum Flow Control (QFC) *http://www.qfc.org*

Every time DD(t) changes by more than a given threshold N relative to the last time a feedback message was sent, transmit the current value of:

DD(t) modulo 2^{28} (or modulo 2^{8} for short links)

- credit-based flow control: lossless
- robust: even if a feedback message is corrupted (lost), the next one will restore the error

② Classical (incremental) Credit-Based Flow Control

every N downstream departures (DD(t1) = DD(t0) + N),
transmit a credit back (N is an implicit parameter);
the upstream node maintains a credit count CC equal to:

$$CC = RTT*Rpk + DD(t-tf) - UD(t);$$

this is incremented by N on every credit arrival, and
decremented by 1 on every departure of a unit of traffic

- shorter feedback (credit) messages than QFC
- non-robust: loss of a credit leads to buffer and
  transmission capacity underutilization;
  accumulated losses of credits lead to deadlock!

Equivalence of Rate and Credit Based Flow Control

③ Rate-Based Flow Control

On every change of the slope of DD(t) (rate of downstream departures), send back the new value of the rate (slope); upon reception of such feedback, the upstream node adjusts its rate of transmission (slope of UD(t)) to the value received; thus, UD(t) is (almost!) a delayed and shifted-up copy of DD(t).

- Rate-based flow control
- Could be made lossless, but this would not be robust:
  slight mismatches between real & measured rate accumulate to large differences between the values of DD(t-tf), UD(t); similarly, variations in tf (delay of feedback messages) lead to UD(t) value errors.