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

The traffic here may have packets that are short-term-conflicting in the switching fabric, but are long-term-non-conflicting in the fabric. Owing to backpressure and distributed scheduling.
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
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''

<table>
<thead>
<tr>
<th>initial (``blind mode'') rate of transmission</th>
<th>amount of data xmitted in ``blind mode''</th>
</tr>
</thead>
<tbody>
<tr>
<td>distance = 8 km</td>
<td></td>
</tr>
<tr>
<td>RTT ~= 0.08 ms</td>
<td></td>
</tr>
<tr>
<td>1 Mbit/s</td>
<td>10 Bytes</td>
</tr>
<tr>
<td>1 Gbit/s</td>
<td>10 KBytes</td>
</tr>
<tr>
<td>distance = 8,000 km</td>
<td></td>
</tr>
<tr>
<td>RTT ~= 80 ms</td>
<td></td>
</tr>
<tr>
<td>10 KBytes</td>
<td>10 Mbytes</td>
</tr>
</tbody>
</table>
Lossy versus Lossless Flow Control

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

Flow control in lossy systems strives to make the network operate in this region.

Lossless systems strive to bring this line up (avoid HOL blocking).

Throughput collapse: most packets get dropped before they reach their destination, and are then retransmitted, thus increasing the demand.

Diagram: Graph showing goodput (%) of capacity versus demand (offered load) (%) of capacity. The graph illustrates the differences between lossy and lossless systems in terms of goodput and demand.
Rate-Based Flow Control

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" ≡ (rate := peak);  "stop" ≡ (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

credits
0

3 cells

delay = 3 cell-times

credits
0

delay = 3 cell-times

6-cell buffer

suddenly stops

suddenly starts
Theorem: Infinite Queue Push-Back

0 ≤ service rate ≤ Rpk
arbitrary schedule

is equivalent to:

Upstream Departures
flow control algorithm

Downstream Arrivals
capacity ≥ Rpk

size = RTT*Rpk

Feedback Information

Downstream Departures
Cumulative Amount of Traffic
(bits, or bytes, or cells, or...)

- Downstream Arrivals: DA(t)
- Upstream Departures: UD(t)
- Feedback Information: FI(t)
- Downstream Departures: DD(t)
- Buffer occupancy (downstream)

RTT*Rpk

time
• **Downstream Departures DD(t) (cumulative):**

  arbitrary function of time, provided that its slope satisfies:
  
  \[ 0 \leq \text{service rate of } S \leq R_{pk} \tag{1} \]
  
  \[ \Rightarrow 0 \leq DD(t+d) - DD(t) \leq d*R_{pk} \tag{2} \]

• **Upstream Departures UD(t) (cumulative):**

  \[ UD(t) = DD(t-tf) + RTT*R_{pk} ; \text{ this is always feasible, since:} \]
  
  \[ \text{link capacity } \geq R_{pk} \geq \text{ service rate of } S \]

• **Downstream Arrivals: DA(t) = DD(t-RTT) + RTT*R_{pk}** (3)

• **Buffer Occupancy (downstream):** \[ BO(t) = DA(t) - DD(t) \]

  \[ (3) \Rightarrow BO(t) = RTT*R_{pk} - [ DD(t) - DD(t-RTT) ] \]

  with (2) \[ \Rightarrow 0 \leq BO(t) \leq RTT*R_{pk} \]

  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

**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:

\[
DD(t) \mod 2^{28} \quad \text{(or modulo } 2^8 \text{ for short links)}
\]

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

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 $DD(t-t_f)$, $UD(t)$; similarly, variations in $t_f$ (delay of feedback messages) lead to $UD(t)$ value errors.