Queue and Flow Control Architectures for Interconnection Switches

1. Basic Concepts and Queueing Architectures
2. High-Throughput Multi-Queue Memories
3. Crossbars, Scheduling, & Combination Queueing
4. Flow and Congestion Control in Switching Fabrics

Manolis Katevenis

FORTH and Univ. of Crete, Greece

`http://archvlsi.ics.forth.gr/~kateveni/534`

4. Flow and Congestion Control in Switching Fabrics

Table of Contents:

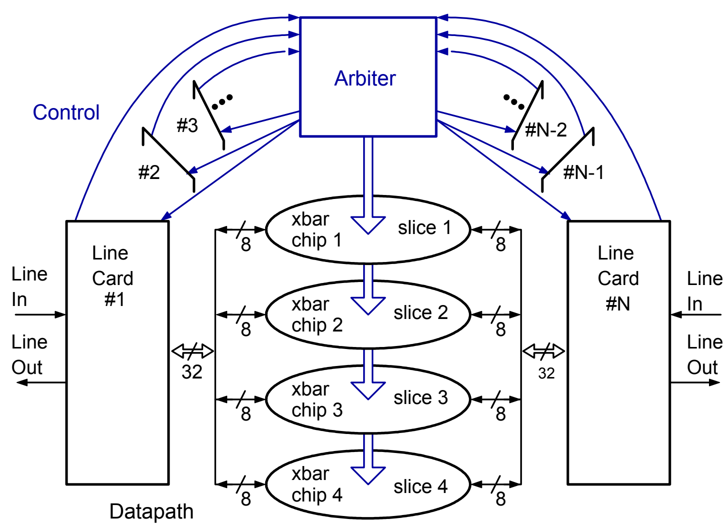
4.1 Parallelism for High-Thruput: Inverse Multiplexing

Bit	Byte-Slice	Packet	Flow
bit 1 of 8	By. 1-8	packet 1	flow 1
bit 2 of 8	By. 9-16	packet 2	flow 2
bit 3 of 8	By. 17-24	packet 3	flow 3
bit 4 of 8	By. 25-32	packet 4	flow 4
bit 5 of 8	By. 33-40	packet 5	flow 5
bit 6 of 8	By. 41-48	packet 6	flow 6
bit 7 of 8	By. 49-56	packet 7	flow 7
bit 8 of 8	By. 57-64	packet 8	flow 8
	of 64B cell	*Inverse Multiplexing*	

same handling for all wires (same time, same destination) — different handling (diff. times & destinations)

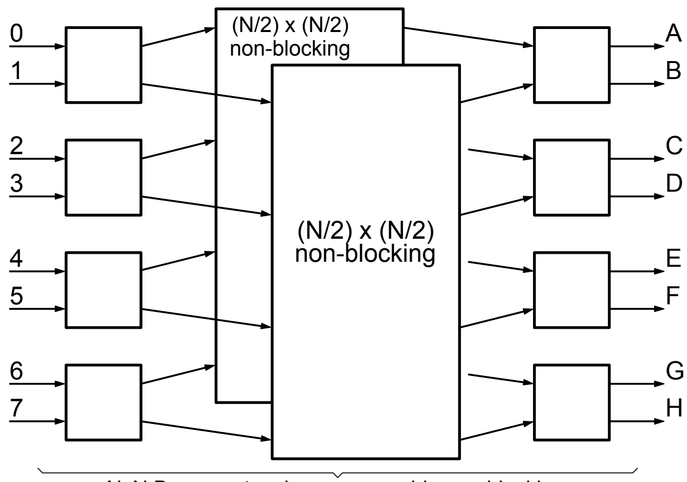
- Parallel wires or network routes for scaling (virtual) “link” throughput up
- Easy: central control, synchronized; Difficult: distributed control, asynch.

4.1 Byte-Slicing: Tiny Tera & other commercial chips



Mckeown e.a.: “Tiny Tera: a Packet Switch Core”, IEEE Micro, Jan.-Feb.’97

Benes Fabric: Recursive Definition

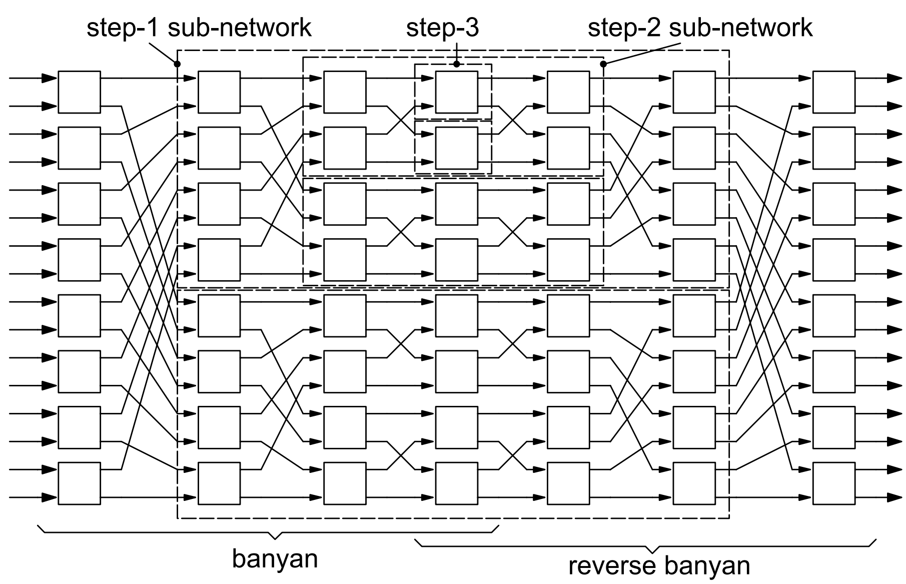


NxN Benes network: rearrangeably non-blocking

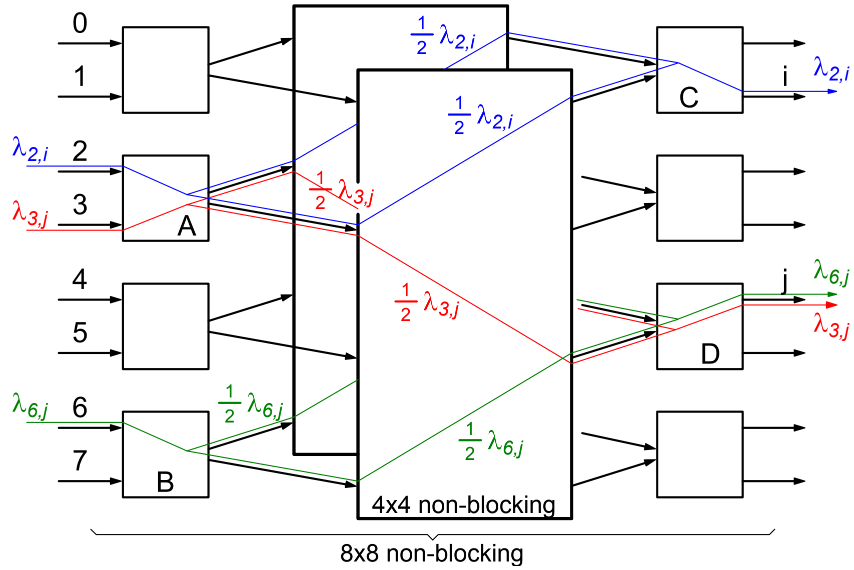
- Goal: reduce switch radix from *N×N* to *(N/2)×(N/2)*: combine ports in pairs
- Port-pairs require links of twice the throughput: use inverse multiplexing

⇒ Use two switches, of half the radix each, in parallel to provide req'd thruput

Full Construction of 16×16 Benes out of 2×2 Switches



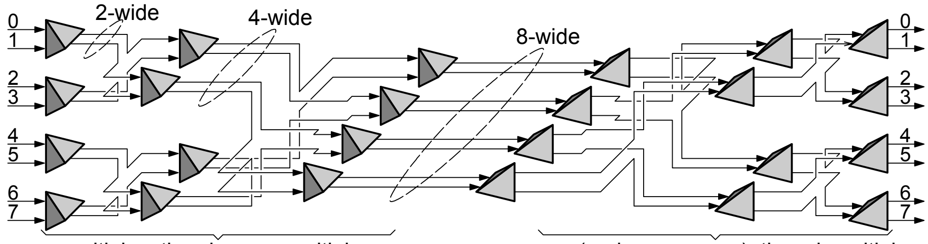
Inverse Multiplexing for Non-Blocking Operation



Per-Flow Inverse Mux'ing for Non-Blocking Operation

- Prove that overall $N \times N$ network is non-blocking, i.e. *any* feasible external traffic ⇒ feasible rates on all internal links
- All traffic entering switch A is feasible, hence of aggregate rate ≤ 1+1 = 2; it is split into two halves ⇒ each of rate ≤ 1 ⇒ traffic entering each *(N/2)×(N/2)* subnetwork is feasible
- It does *not suffice* to balance (equalize) the *aggregate* load out of switch A – must equally distribute *individual* (end-to-end) flows – *per-flow* inverse multiplexing
  - ⇒ each of $\lambda_{2,i}$; $\lambda_{3,j}$; $\lambda_{6,j}$ is individually split in two equal halves
  - ⇒ the sum of $\lambda_{3,j}+\lambda_{6,j}$ is also split in two equal halves
- All traffic exiting switch D is feasible, hence of aggregate rate ≤ 1+1 = 2; it enters D in two equal halves ⇒ each of rate ≤ 1 ⇒ traffic exiting each *(N/2)×(N/2)* subnetwork is also feasible

Conceptual View of 8x8 Benes: Virtual Parallel Links using Inverse Multiplexing



Methods to implement (per-flow) Inverse Multiplexing

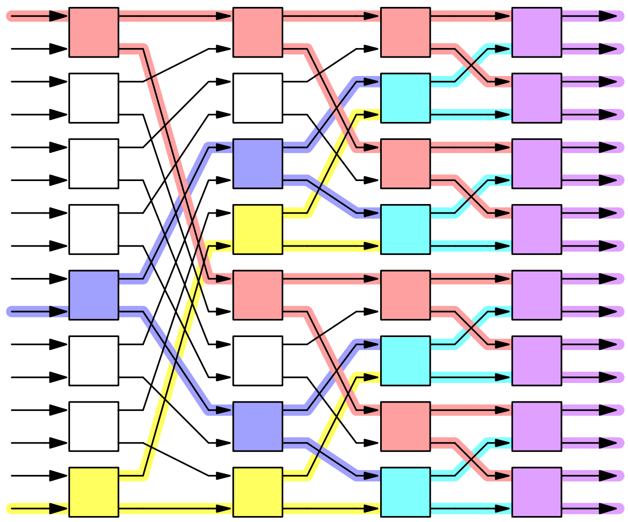
- *Per-Flow Round-Robin*, at packet granularity
  - for each flow, circularly and per-packet alternate among routes
  - requires maintaining per-flow state
  - danger of synchronized RR pointers: pck bursts to same route
  - alternative: arbitrary route selection, provided the (per-flow) imbalance counter has not exceeded upper bound value

Methods to implement (per-flow) inverse multiplexing (continued)

- *Adaptive Routing*, at packet granularity – usu. Indiscriminate
  - chose the route with least-occupied buffer (max. credits)
  - \+ does not maintain or use per-flow state
  - − per-flow load balancing only "after-the-fact", when buffers fill up
- *Randomized* Route Selection, at packet granularity
  - \+ does not require maintaining per-flow state
  - − load balancing is approximate, and long-term
- **Packet Resequencing** (when needed): major cost of inv.mux'ng
  - Chiussi, Khotimsky, Krishnan: IEEE GLOBECOM'98
- Hashed Route Selection *at entire Flow Granularity*
  - route selection based on hash function of flow ID
  - \+ all packets of given flow through same route ⇒ *in-order delivery*
  - − poor load balancing when small number of flows

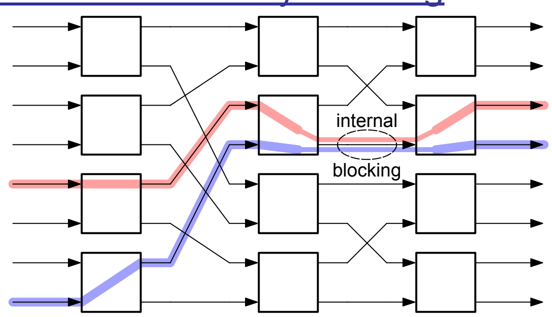
4.2 The Banyan (Butterfly) Network

- Single route from given input to given output
- Each input is the root of a tree leading to all outputs
- Trees share nodes
- (Similarly, outputs are roots of trees feeding each from all inputs)
- for $N \times N$ network made of 2×2 sw.:
- $\log_2 N$ stages, of
- $N/2$ sw. per stage



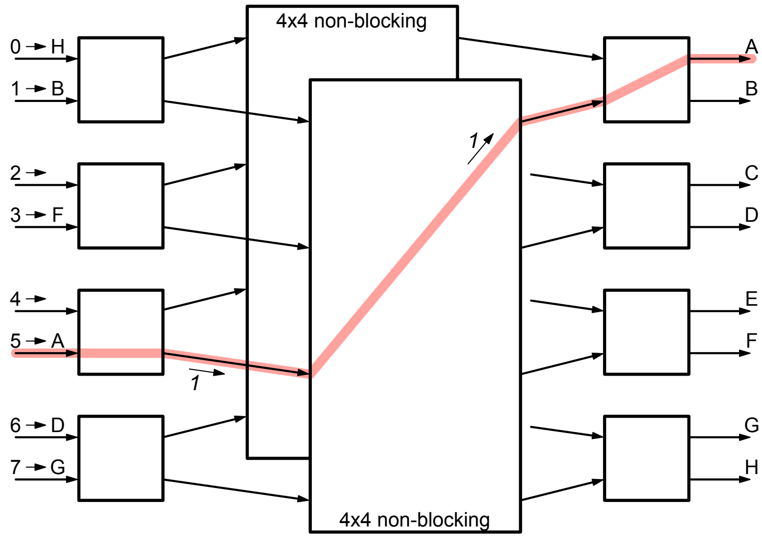
The banyan network is internally blocking

- Consider circuits: each $\lambda_{i,j}$ is either 1 or 0: single connection per port – "telephony" style
- There are $N!$ such circuit connection patterns for a $N \times N$ network – each is a permutation of the numbers (1, 2, …, N)

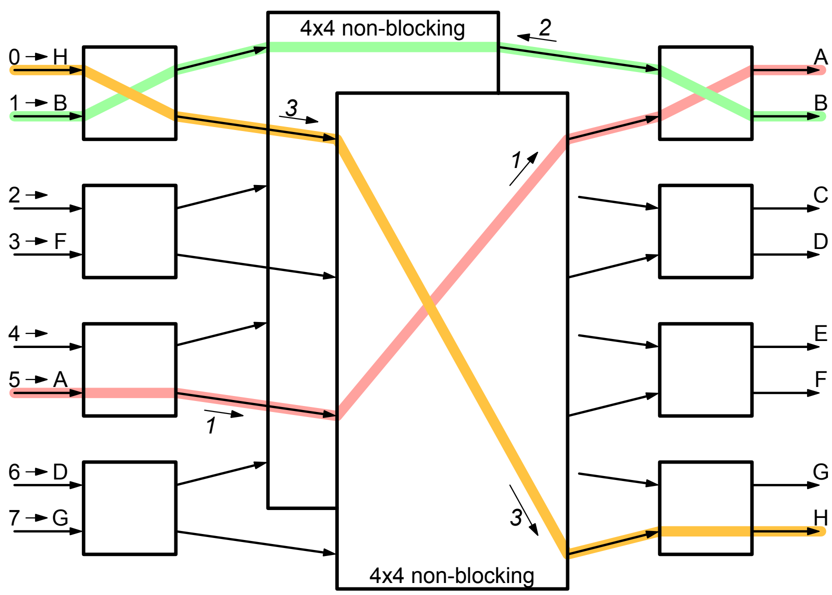


- Any network containing $(N/2) \cdot \log_2 N$ or less 2×2 switches (like the banyan does) has to be internally blocking, because it can only be placed into less than $N!$ states, hence cannot route all $N!$ existing sets of con. req's
- Each 2×2 switch can be placed in 2 different states; a network containing $(N/2) \cdot \log_2 N$ such switches can be placed into $2^{(N/2) \cdot \log N} = N^{(N/2)}$ different states; $N^{(N/2)} = N \cdot (N/2)^{(N/2)-1} \cdot 2^{(N/2)-1} < N \cdot [(N-1) \cdot \ldots \cdot (N/2+1)] \cdot [(N/2) \cdot \ldots \cdot 2] = N!$ $\Rightarrow$ not enough states

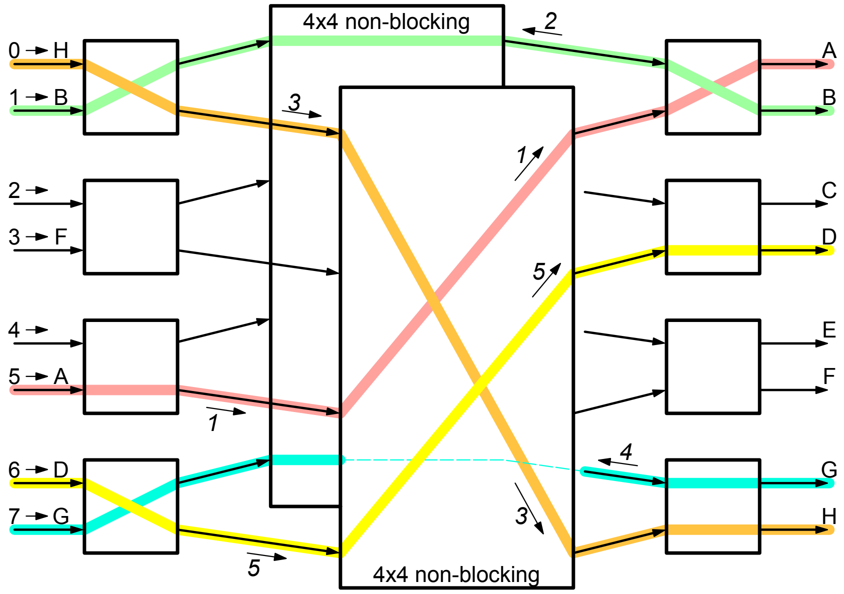
Benes Net under Telephony-Ckt Connection Requests



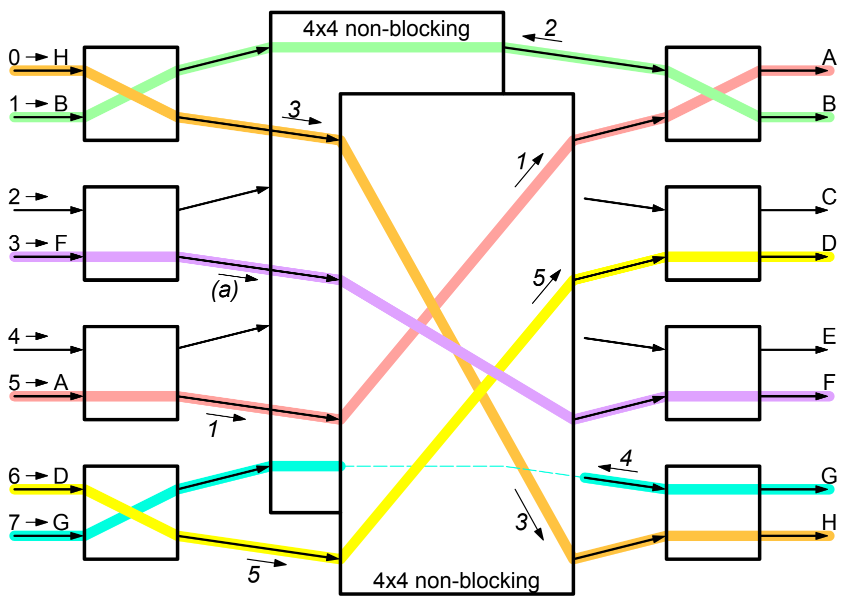
• Circuit Connections: Start from an input, use one of the subnets



- Continue from the brother port of the output, then the brother of the input

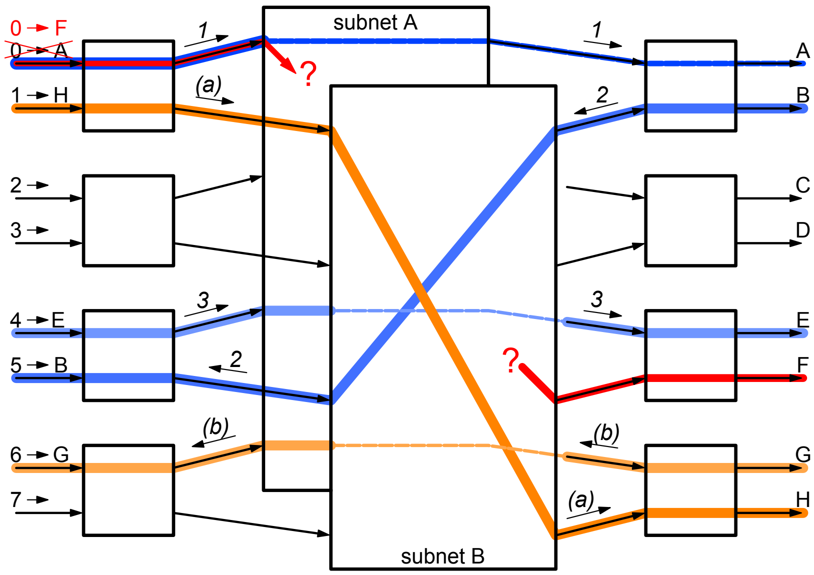


- Keep "threading" output and input switches, till closing or no-connection



- Start a new "thread" (a) from an unconnected input, till completing all conn.

Benes Fabric: *Rearrangeably* Non-Blocking

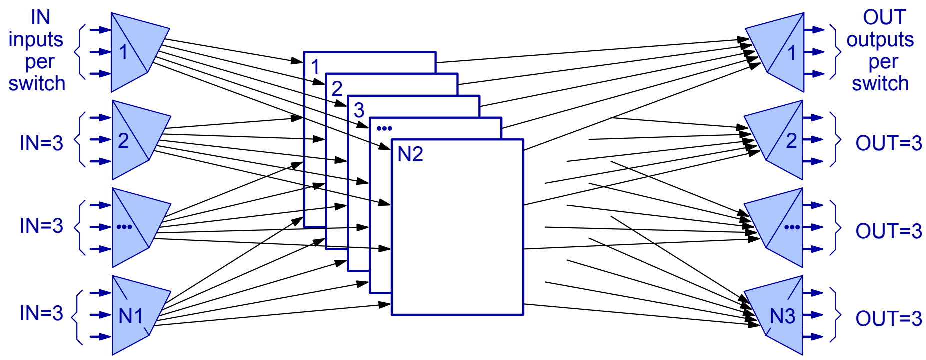


Which is the lowest-cost non-blocking fabric?

- $N \times N$ Benes network, made of 2×2 switches:
  - $2 \cdot (\log_2 N) - 1$ stages (2 banyans back-to-back, 1 shared stage)
  - $N/2$ switches per stage $\Rightarrow$ total switches = $N \cdot (\log_2 N) - N/2$
  - number of states that the Benes network can be in = $2^{\#switches}$ = $2^{N \cdot (\log N) - N/2} = (2^{\log N})^N / 2^{N/2} = N^N / 2^{N/2} = [N \cdot \ldots \cdot N] \cdot [(N/2) \cdot \ldots \cdot (N/2)] > N \cdot (N-1) \cdot \ldots \cdot 2 \cdot 1 = N!$ $\Rightarrow$ Benes has more states than the minimum required for a net to be non-blocking
  - Benes was seen to be non-blocking: *(i)* circuits and the "threading" algorithm, *(ii)* packets and inverse multiplexing
  - *"rearrangeably"* non-blocking: in a partially connected network, making a new connection may require re-routing existing ones
- Impossible for any network with about half the switches of the Benes (e.g. banyan) to be non-blocking (# of states)

$\Rightarrow$ Benes is probably the lowest-cost *practical* non-blocking fabric

4.2 Clos Networks (generalization of Benes nets)

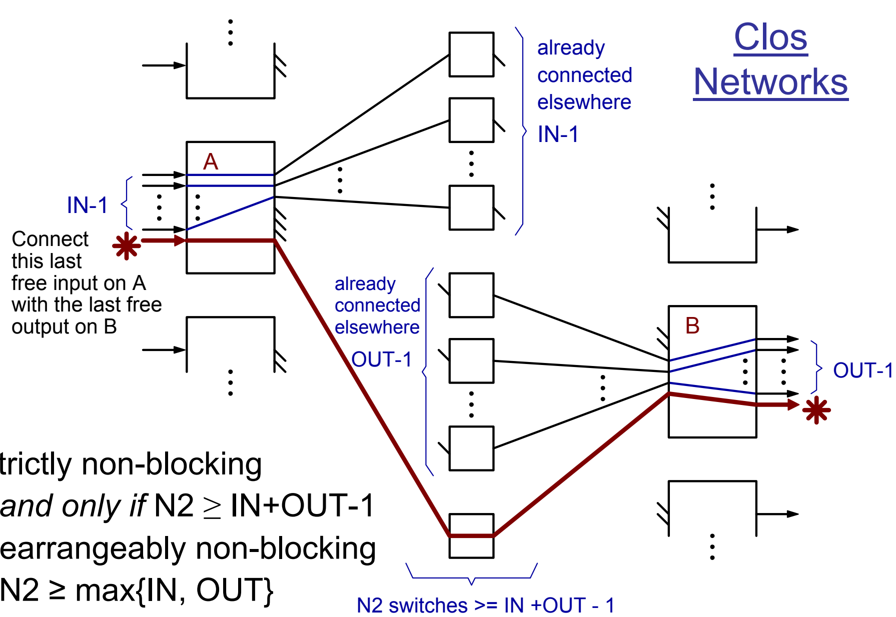


5-parameter Network: (IN, N1, N2, N3, OUT)

this example: the (3, 4, 5, 4, 3) Clos Network

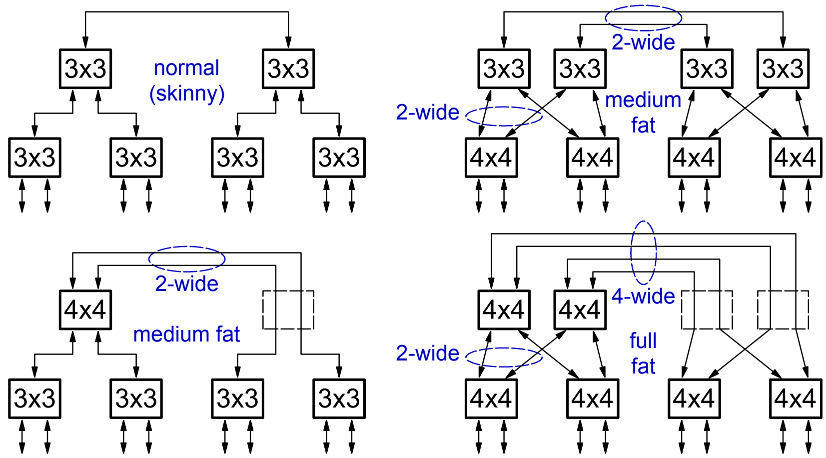
usually: IN = OUT, and N1 = N3

Clos Networks



- Strictly non-blocking *if and only if* N2 $\geq$ IN+OUT-1
- Rearrangeably non-blocking *if* N2 $\geq$ max{IN, OUT}

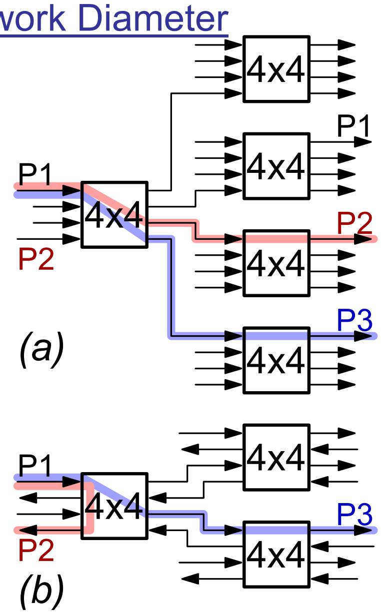
4.2 Fat Trees: customizable local versus global traffic



- Customizable percent fat – configurable amounts of internal blocking
- Bidirectional links, like most practical interconnects
- Skinny trees support local traffic – Full-fat tree is like folded Benes

Switch Radix, Hop Count, Network Diameter

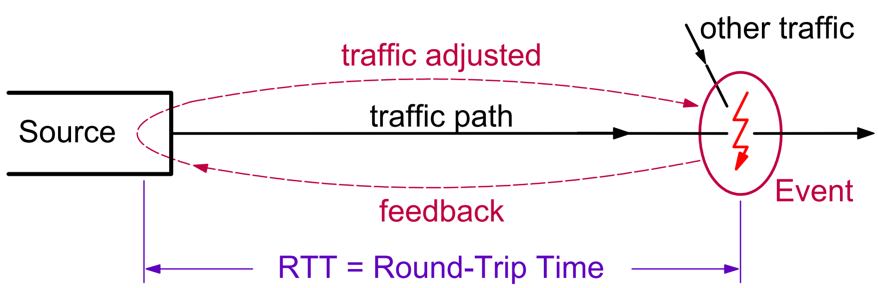
- Most of our examples used unidirectional links – fig. *(a)*
  - "indirect" nets have ports at edges.
- Most practical interconnects use bidirectional links – fig. *(b)*
  - "direct" nets provide external ports on all switches.
- If some destinations are reachable at reduced hop count (P2 in *(b)*), that is at the expense of the total number of destinations reachable at a given hop count – or larger network diameter.
- Energy consumption to cross the net critically depends on the number of chip-to-chip hops, because chip power is dominated by I/O pin driver consum.



4.3 Flow Control – Lossy versus Lossless

- *Lossy Flow Control:* may fail to prevent buffer overflows ⇒ packets may be dropped
  - inherited from communications: same as electrical noise
  - \+ simple switches, avoids deadlock danger
  - − need data re-transmissions: additional (and long) delays
  - − wastes communication capacity: “goodput” versus throughput
- *Lossless Flow Control:* guarantees buffers to never overflow
  - inherited from hardware: processors never drop data
  - \+ no wastes – can reach ≈ 100% utilization if properly designed
  - \+ can minimize delay, if properly designed
  - − complex switches – need multilane protocols in order to avoid penomena similar to HOL blocking and deadlocks

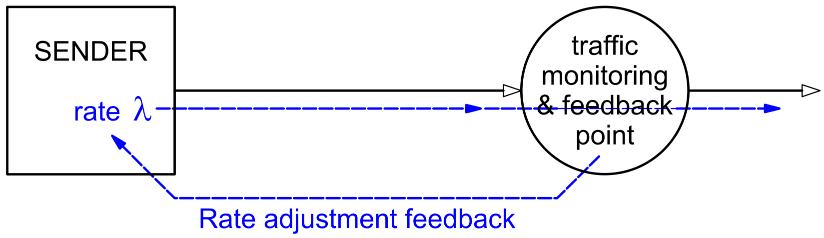
RTT: the fundamental Time-Constant of Feedback



Traffic is “blind” during a time interval of RTT:

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

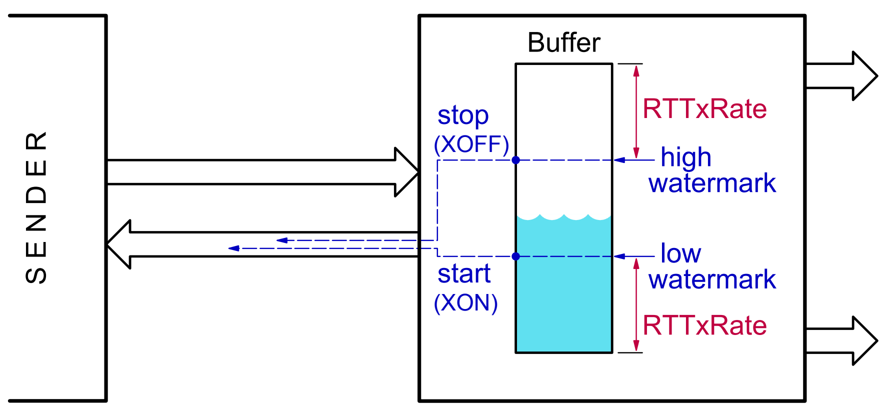
4.3 Rate-based Flow Control



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

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

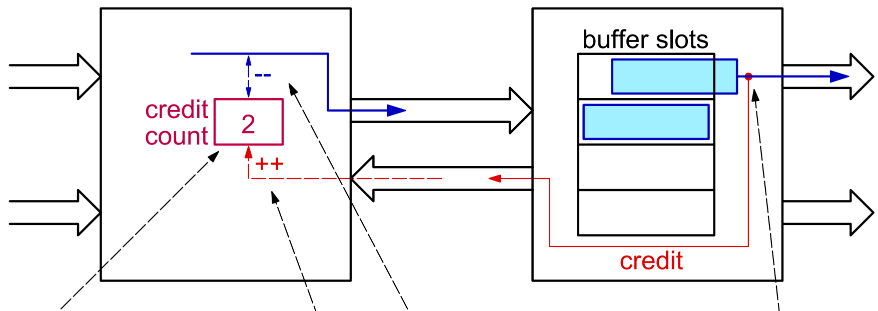
ON/OFF (start/stop): simplistic Rate-based Flow Ctrl



\`\`start'' ≡ (rate := peak;) \`\`stop'' ≡ (rate := 0;)

- Rate-based flow control used for lossless transfers
- Less than half the buffer efficiency of credit-based flow ctrl

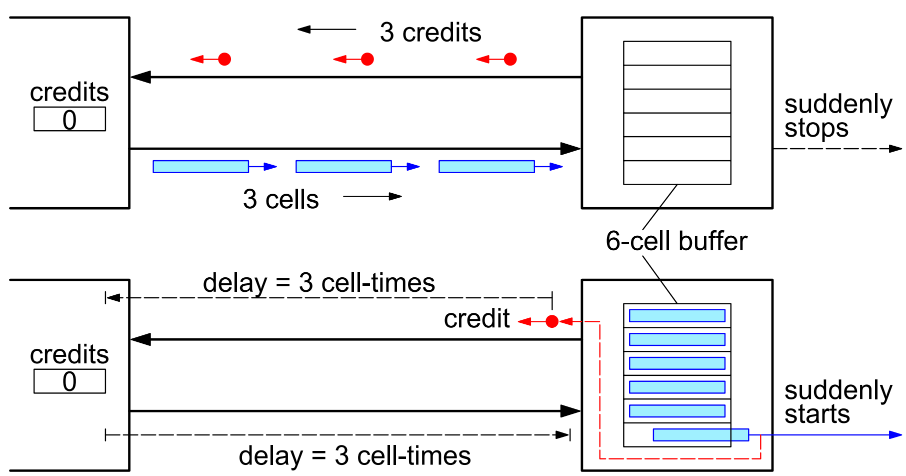
4.3 Credit-Based (Window) Flow Control



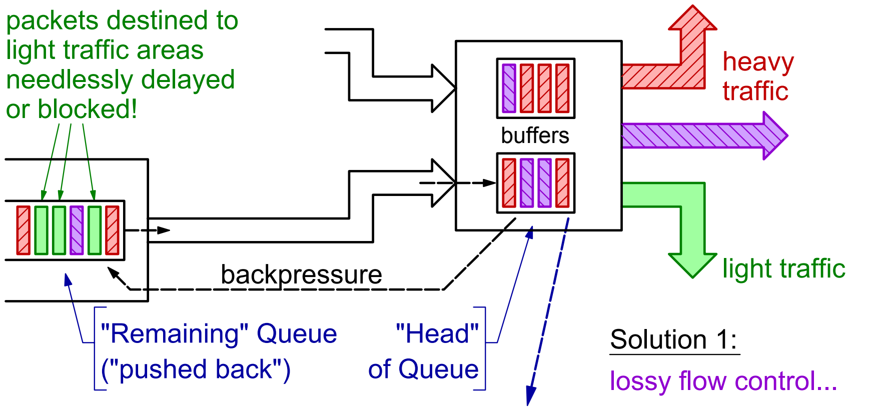
- count of buffer slots known to be available at the downstream site (not allowed to go negative)
- traffic can only depart if and when it acquires (decrements) the credit(s) that correspond to the buffer slot(s) needed
- arriving credits increment the credit count
- when new buffer slots are made available,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

⇒ ***Lossless Flow Control***

(necess. & suffic'nt) Buffer Space = Peak Thruput × RTT



4.4 Indiscriminate Lossless FC ⇒ HOL Blocking



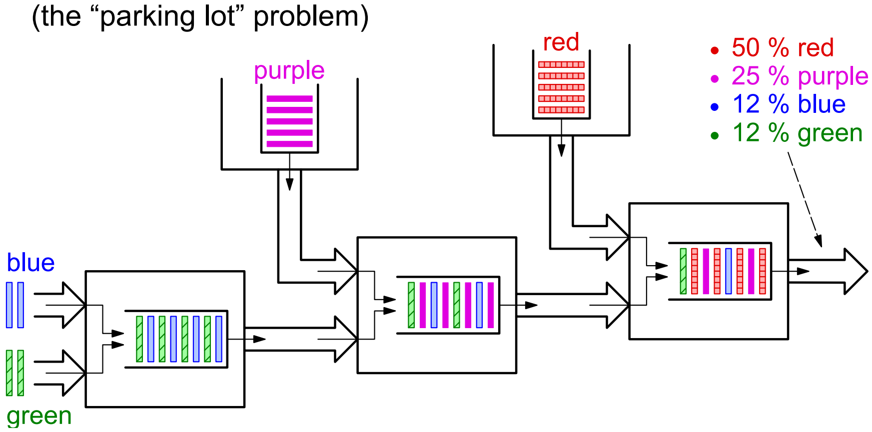
With any queueing discipline (FIFO or not) this switch has only access to and can only schedule packets in this limited buffer space
⇒ similar to Head-of-Line (HOL) Blocking!

Solution 1:
lossy flow control...

Solution 2:
Per-Flow
queueing & flow contro

Indiscriminate (shared-queue) Queueing is Unfair

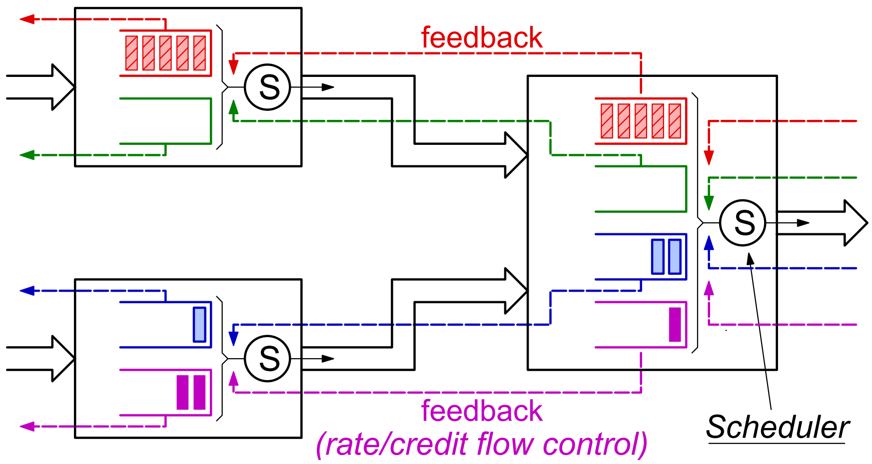
(the “parking lot” problem)



- 50 % red
- 25 % purple
- 12 % blue
- 12 % green

- *Solution:*
  Per-Flow queueing and (weighted) round-robin scheduling

4.4 *Solution:* Per-Flow Queueing & Flow Control

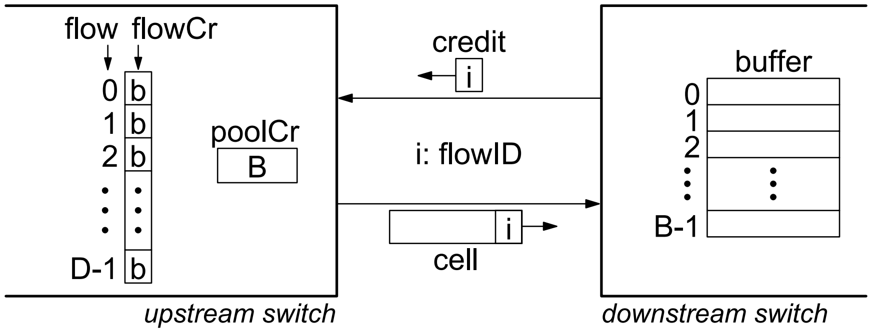


- Congested flows (e.g. red) become stopped, yet other flows can bypass them –since each flow uses its own, separate queue– hence not feeling any negative effects from the congestion in other parts of the network

4.4 Buffer Space versus Number of Flows

- For proper congestion management, flows are defined per priority level and end-to-end in the network – not per switch (not the mere VOQ's inside individual switches)
  - optimization: merge flows from diff. sources to same destination
- What to do when the number of flows is so large that it becomes impractical to allocate a separate queue and space for a flow-control window for each flow?
  - Dynamically share the available buffer space among the flows
    - ATLAS I and QFC protocols: share a number of lanes among flows
    - H.T.Kung protocol: bounded acceleration per RTT
  - Regional Explicit Congestion Notification (RECN)
  - Credit allocation by Request-Grant, rather than pre-allocation

Share a number of Lanes among Flows: QFC, ATLAS



- *Two kinds* of Credits – a packet must secure *one of each* for departure:
  - *Pool Credits* (init: B), to contol overall downstream buffer occupancy
  - *Per-Flow Credits* (init: b), to limit the buffer space occupied per flow
- *Number of Lanes* = *B / b*
  - the buffer is guarantted to fit packets from at least that many flows
  - if # congested destinations < # Lanes ⇒ others can bypass congested

The QFC / ATLAS I Credit Flow-Control Protocol

- Quantum Flow Control (QFC) Alliance (1995): proposed a standard for credit-based flow control over WAN ATM links.
- ATLAS I switch chip (FORTH, 1995-98): implemented a similar protocol at 32 K flow granularity.
- Features:
  - identically destined traffic is confined to a single “lane”
  - each “lane” can be shared by cells belonging to multiple packets
- As opposed to Wormhole Routing, where:
  - a “virtual circuit” (VC) is allocated to a packet and dedicated to it for its entire duration, i.e. until all “flits” (cells) of that packet go through
  - identically destined packets are allowed to occupy distinct VC’s
- Reference: Katevenis e.a.: “Credit-Flow-Controlled ATM for MP Interconnection: the ATLAS I Single-Chip ATM Switch”, IEEE HPCA-4, Feb. 1998

Buffer Space for Bounded Peak-to-Average Rate Ratio

- Assume $R_{peak}(i) / R_{average}(i) \leq PAR$ for all flows *i* on a link
  - $R(i)$ is the rate (throughput) of flow *i*
  - *PAR* is a constant: peak-to-average ratio bound
  - interpretation: rate fluctuation is bounded by *PAR*
- Each flow *i* needs a credit window of $RTT \cdot R_{peak}(i)$
- Buffer space for all flows is $\sum (RTT \cdot R_{peak}(i)) =$

  $= RTT \cdot \sum(R_{peak}(i)) \leq RTT \cdot \sum(PAR \cdot R_{average}(i)) =$

  $= PAR \cdot RTT \cdot \sum(R_{average}(i)) \leq PAR \cdot (RTT \cdot R_{link})$

$\Rightarrow$ Allocate buffer space = *PAR* number of "windows"

When individual flow rates change, rearrange the allocation of buffer space between flows –but must wait for the buffer of one flow to drain before rallocating it (not obvious how to)

- H.T. Kung, T. Blackwell, A. Chapman: "Credit-Based Flow Control for ATM Networks: Credit Update Protocol, Adaptive Credit Allocation, and Statistical Multiplexing", SIGCOMM '94, pp. 101-114.

Dynamically Sharing the Buffer Space among Flows

- In order to depart, a packet must acquire both:
  - a per-flow credit (to guard against "buffer hogging"), and
  - a per-link credit (to ensure that the shared buffer does not overflow)
- Properly manage (increase or decrease) the per-flow window allocation based on traffic circumstances:
  - ATLAS and QFC protocols never change the per-flow window
  - H.T.Kung protocol moves allocations between flows (unclear how)
  - other idea: use two window sizes –a "full" one and a "small" one; use full-size windows when total buffer occupancy is below a threshold, use small-size windows (for all flows) above that point (flows that had already filled more than a small window will lose their allocation on packet departure) – C. Ozveren, R. Simcoe, G. Varghese: "Reliable and Efficient Hop-by-Hop Flow Control", IEEE JSAC, May 1995.
- Draining Rate Theorem: M. Katevenis: "Buffer Requirements of Credit-Based Flow Control when a Minimum Draining Rate is Guaranteed", HPCS'97; ftp://ftp.ics.forth.gr/tech-reports/1997/1997.HPCS97.drain_cr_buf.ps.gz

Regional Explicit Congestion Notification (RECN)

- Generalization & evolution of the ATLAS/QFC protocol
- Source-routing header describes path through fabric
- Intermediate or final link congestion sends back-notification
- All packets to congested link confined to a single lane
  - intermediate links identified via path component in header
    - ⇒ entire trees of destinations in single lane (improvement over QFC)
  - equivalent of lane here called *"Set-Aside Queue"* (SAQ)
- VOQ's replaced by Single (!) Input Queue + SAQ's
  - dynamically create/delete SAQ's
  - CAM assumed to match incoming pck header versus current SAQ's
- Duato, Johnson, Flich, Naven, Garcia, Nachiondo: "A New Scalable and Cost-Effective Congestion Management Strategy for Lossless Multistage Interconnection Networks", HPCA-11, San Francisco, USA, Feb. 2005.

Request-Grant Protocols

- Consider a buffer feeding an output link, and receiving traffic from multiple sources:
- If credits are pre-allocated to each source, the buffer needs to be as large as one RTT-window per source;
- If credits are "held" at buffer and only allocated to requesting source(s) when these have something to transmit, then a single RTT-window suffices for all sources!

  ⇒ economize on buffer space at the cost of longer latency
- N. Chrysos, M. Katevenis: "Scheduling in Switches with Small Internal Buffers", IEEE Globecom 2005, St. Louis, USA, Nov. 2005;

  N. Chrysos, M. Katevenis: "Scheduling in Non-Blocking Buffered Three-Stage Switching Fabrics", IEEE Infocom 2006, Barcelona, Spain, Apr. 2006; http://archvlsi.ics.forth.gr/bpbenes/