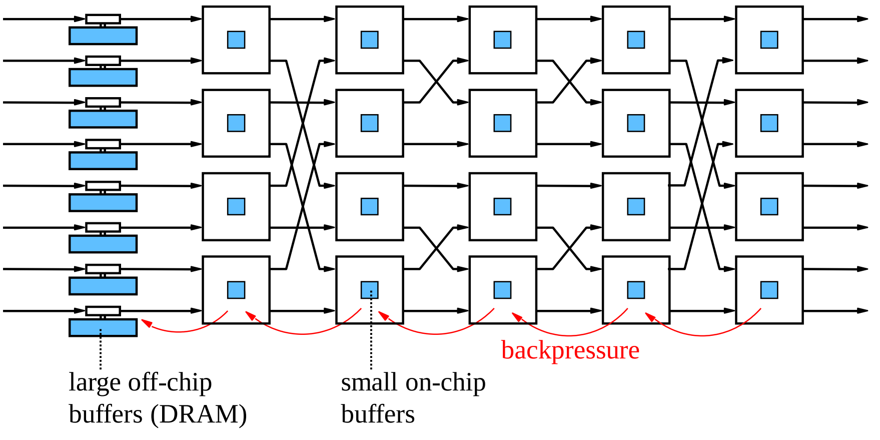
6.3 Buffer Space versus Number of Flows

- What to do when the number of flows is so large that it becomes impractical to allocate a separate flow-control "window" for each one of them
  - Per-destination flow merging: Sapountzis and Katevenis, IEEE Communications Magazine, Jan. 2005, pp. 88-94.
  - Dynamically sharing the buffer space among the flows: the ATLAS I and the QFC flow control protocol, and its evaluation
  - Regional Explicit Congestion Notification (RECN)
  - Request-Grant protocols: N. Chrysos, 2005
  - End-to-end congestion control via request-grant: N. Chrysos, 2006
  - Ethernet Quantized Congestion Control
  - Job-size aware scheduling and buffer management
  - Other buffer sharing protocols of the mid-90's
  - Buffer memory cost versus transmission throughput cost in 1995

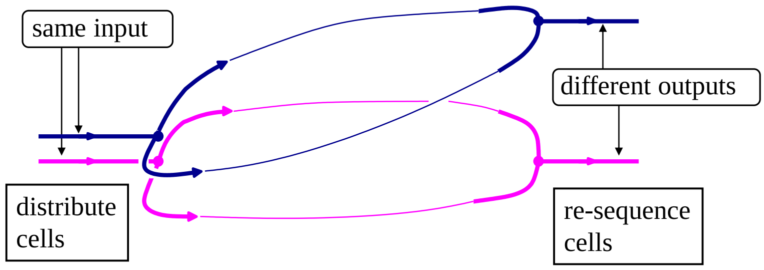
Buffered Switching Fabrics with Internal Backpressure



- Performance of OQ at the cost of IQ,
- Requires per-flow backpressure.

Cell Distribution Methods

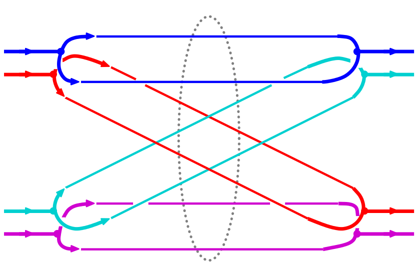
- Aggregate traffic distribution:
  - Randomized routing (no backpressure)
  - Adaptive routing (indiscriminate backpressure)
  - ⇒ load balancing on the long-term only



- Per-flow traffic distribution:
  - Per-flow round-robin (PerFlowRR)
  - Per-flow imbalance up to 1 cell (PerFlowIC)
  - ⇒ accurate load balancing, on a shorter-term basis

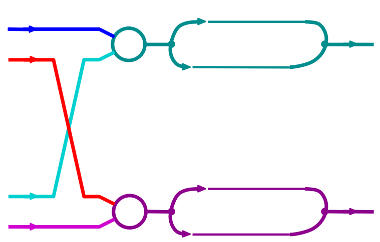
3

Too many Flows



- N^2 per chip in the middle stage

Per-output Flow Merging



- Retains the benefits of per-flow backpressure
- N flows per link, everywhere

  - Re-sequencing needs to consider flows as they were before merging
  - Freedom from deadlock

4

The "ATLAS I" Credit Flow-Control Protocol

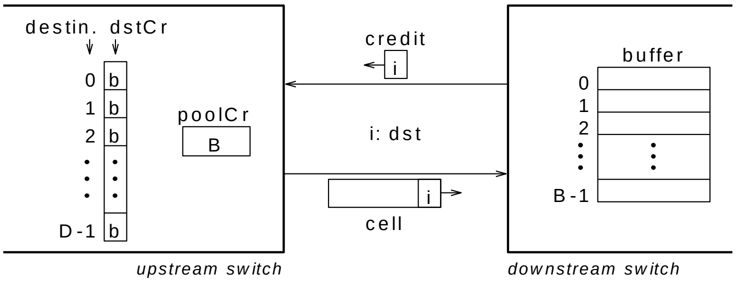
- M. Katevenis, D. Serpanos, E. Spyridakis: "Credit-Flow-Controlled ATM for MP Interconnection: the ATLAS I Single-Chip ATM Switch", Proc. IEEE HPCA-4 (High Perf. Comp. Arch.), Las Vegas, USA, Feb. 1998, pp. 47-56; http://archvlsi.ics.forth.gr/atlasI/atlasI_hpca98.ps.gz

- 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

QFC-like Credit Protocol

⇒ Quantum Flow Control (QFC) Alliance: proposed standard for credit-based flow control over WAN ATM links

⇒ ATLAS I: similar protocol, adapted to
- short links
- hardware implem.



both kinds of credit are needed for a cell to depart

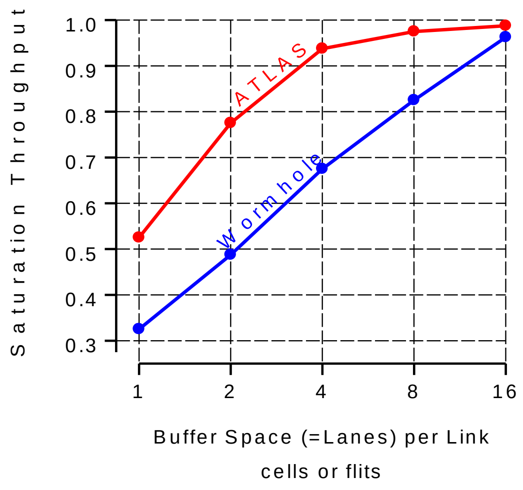
Number of Lanes $L = \frac{B}{b}$

(in ATLAS I: b=1)

Saturation Throughput

64x64 fabric: 6-stage banyan using 2x2 elements

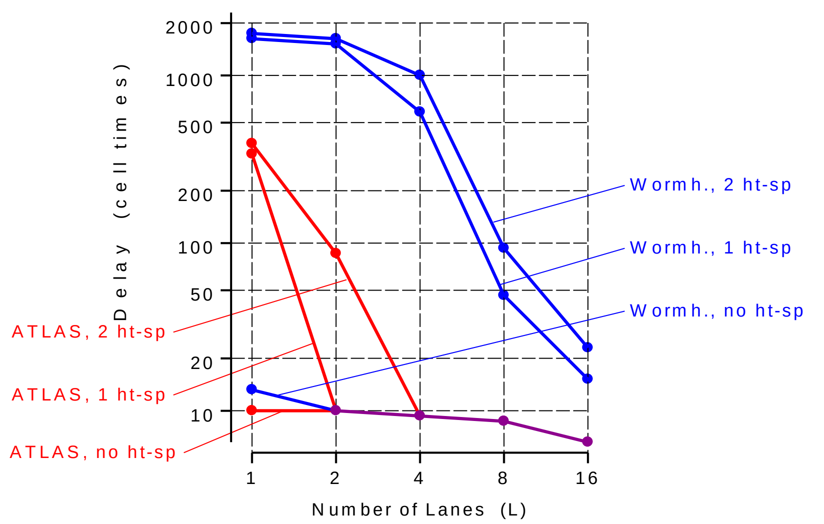
20-cell or 20-flit bursts, uniformly destined



(B=L, with b=1)

Non-Hot-Spot Delay, in the Presence of Hot-Spot Destinations

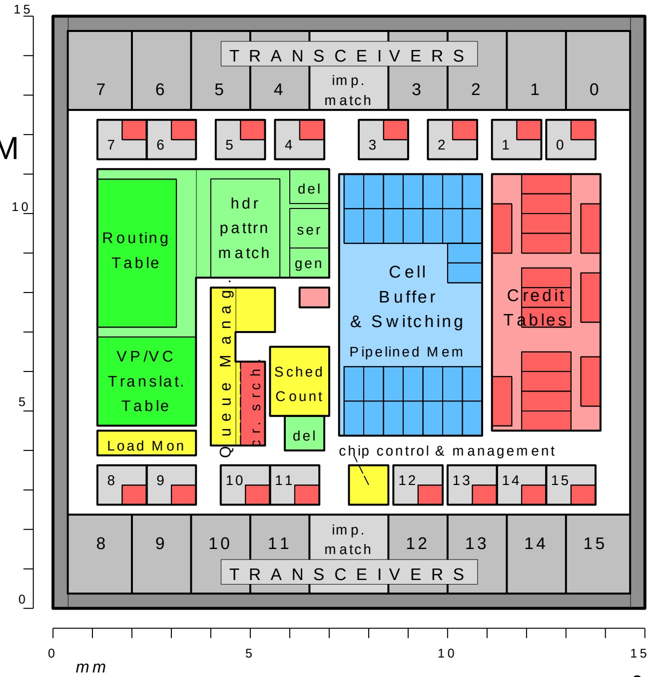
non-hot-spot load = 0.2; 20-cell/flit bursts; 64x64 fabric: 6-stg banyan w. 2x2 el.



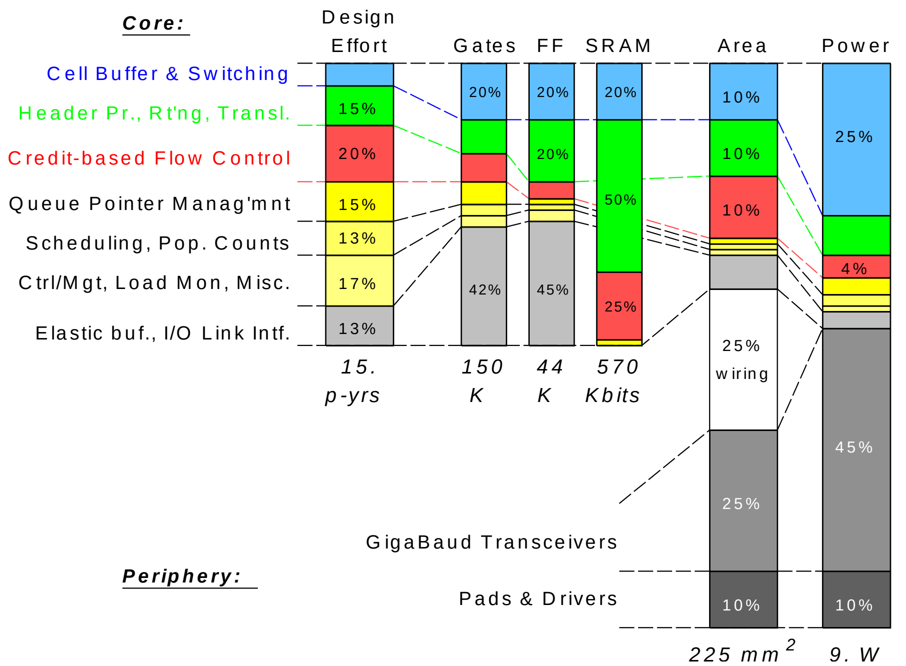
(with buffer space B=16 cells or flits per link)

ATLAS I

- Single-chip ATM Switch with Multilane Backpressure
- 10 Gbit/s = 16×16 @ 622 Mb/s/port
- Shared Buffer
- 0.35 µm CMOS
- 1996-98, FORTH-ICS, Crete, GR



9



10

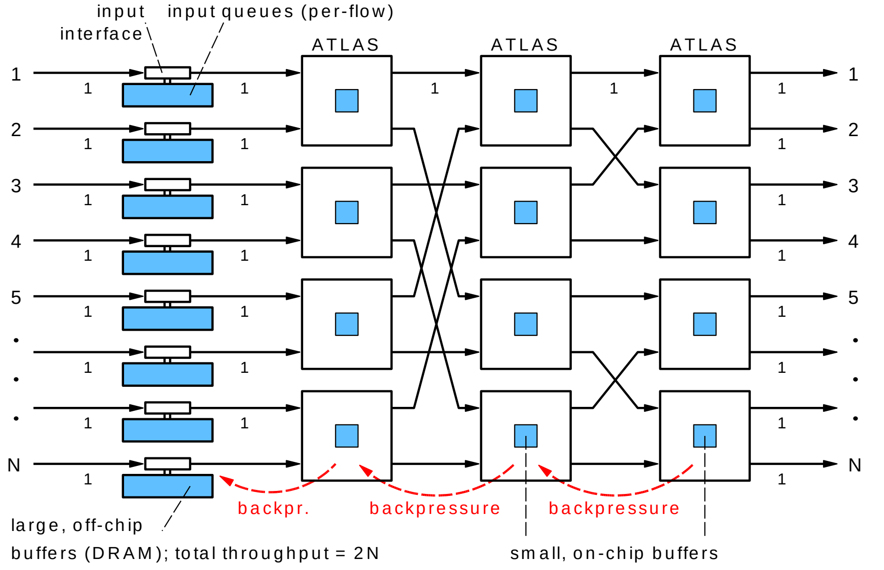
Backpressure Cost Evaluation, versus Alternatives

- Measure the cost of credit flow control in ATLAS & compare to:
- Alternatives, without internal backpressure in the fabric:
  - large buffers (off-chip DRAM) in all switches throughout the fabric, or
  - internal speedup in the fabric and output buffers

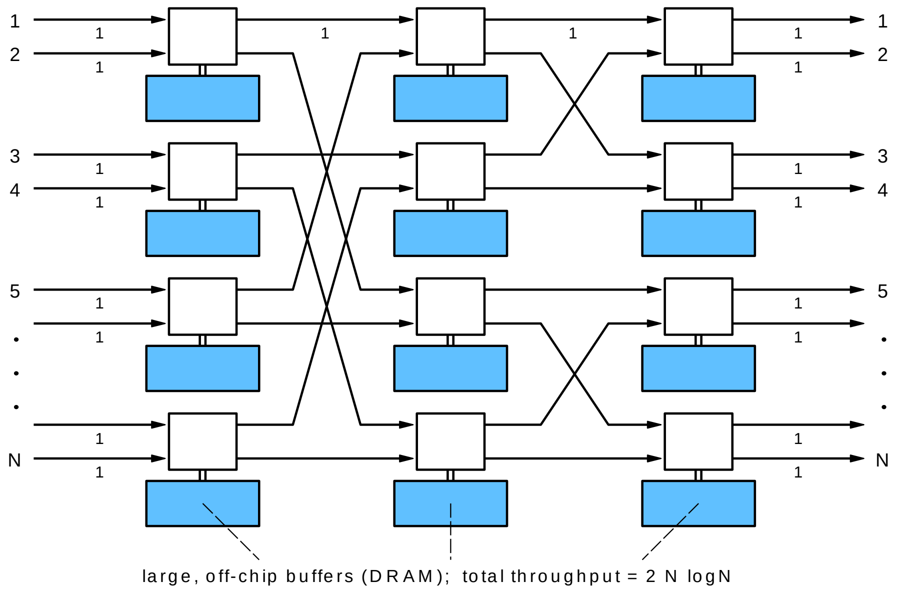
- Kornaros, Pnevmatikatos, Vatsolaki, Kalokerinos, Xanthaki, D. Mavroidis, Serpanos, Katevenis: “ATLAS I: Implementing a Single-Chip ATM Switch with Backpressure”, IEEE Micro Magazine, Jan/Feb. 1999, http://archvlsi.ics.forth.gr/atlasI/hoti98/

Making Large ATM Switches:

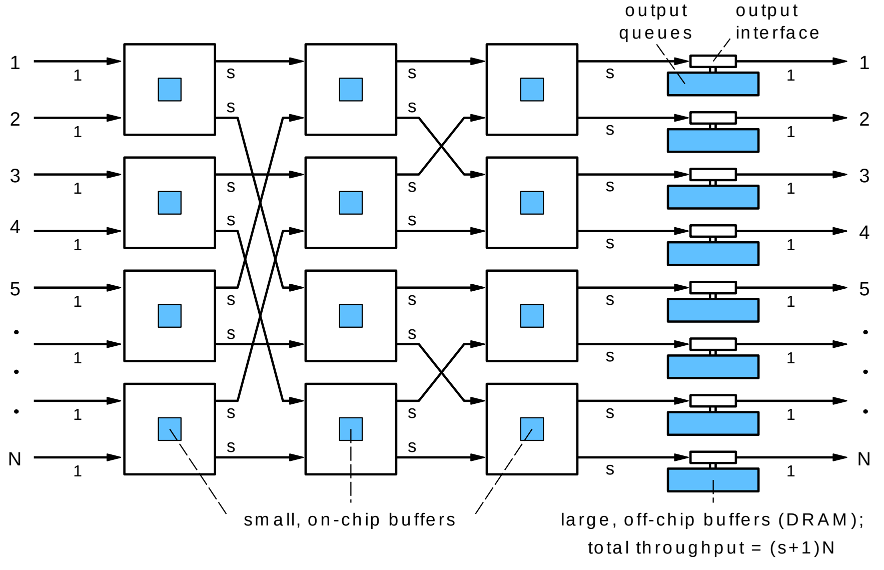
Switching Fabrics with Internal Backpressure



Switching Fabrics without Backpressure 1: Large Buffers



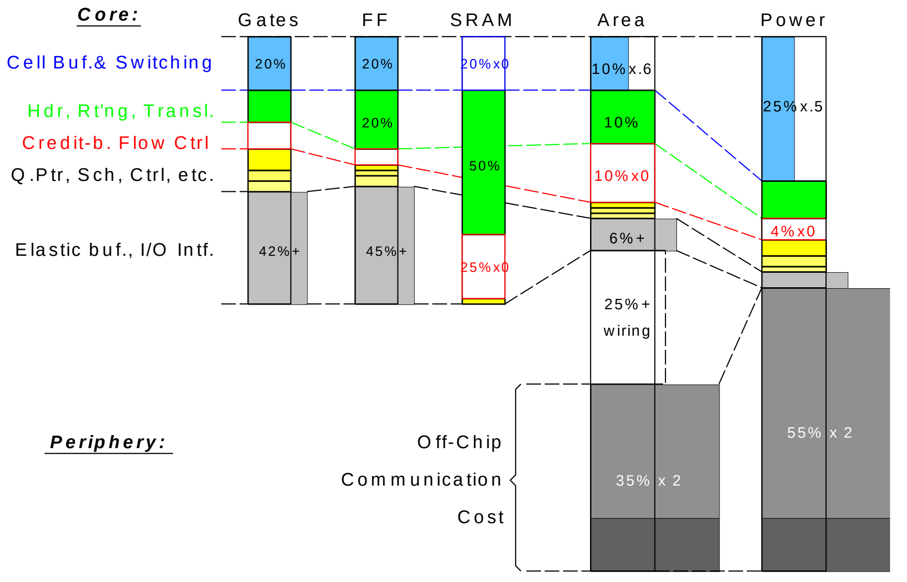
Switching Fabrics without Backpressure 2: Internal Speedup



speedup s>1; under bursty or non-uniform traffic: s>>1...

Backpressure Cost/Benefit 1:

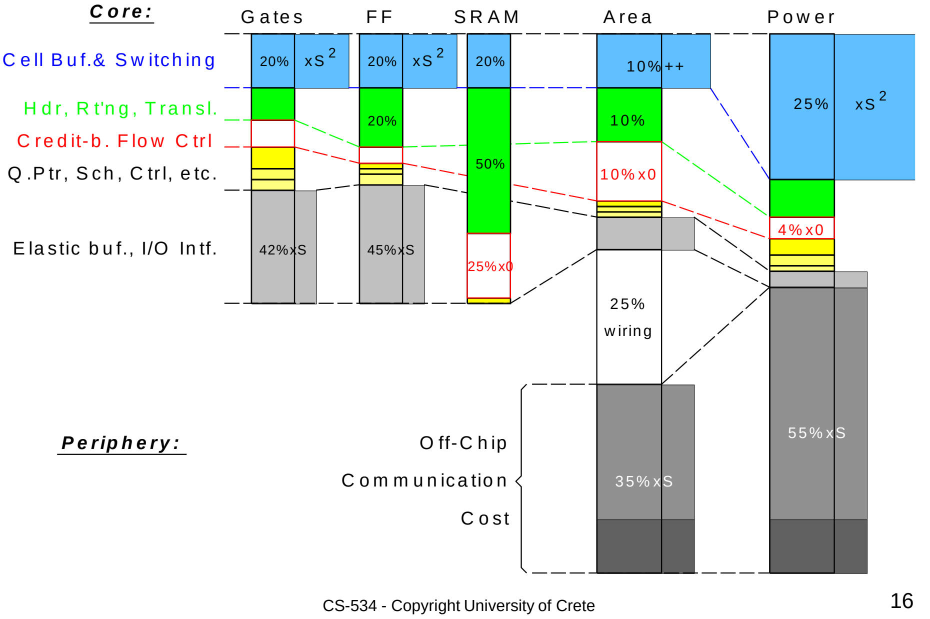
No Backpressure, Large Off-Chip Buffers



 15

Backpressure Cost/Benefit 2:

No Backpressure, Internal Speedup, Output Queues



 16

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
    $\Rightarrow$ 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.

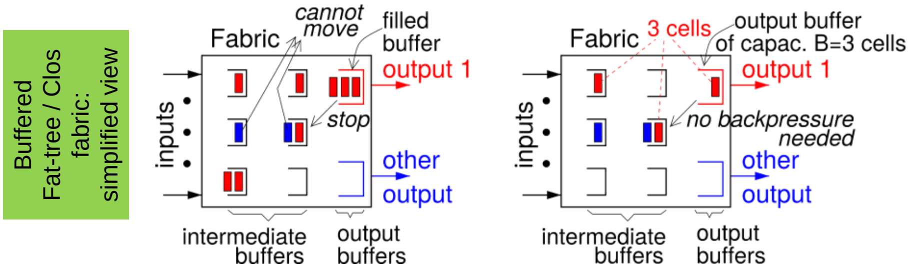
Request-Grant Protocols

- 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!

  $\Rightarrow$ 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;

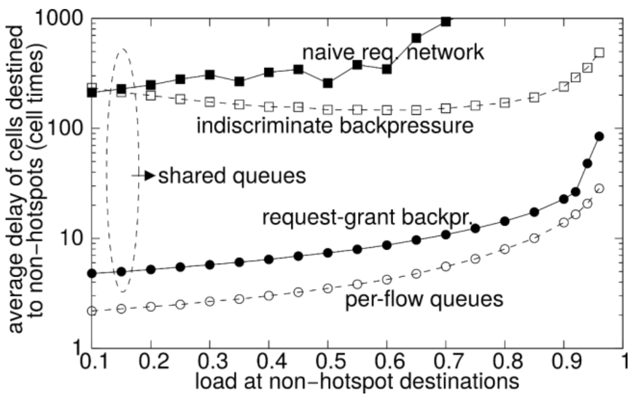
  N. Chrysos, M. Katevenis: "Scheduling in Non-Blocking Buffered Three-Stage Switching Fabrics", IEEE Infocom 2006, Barcelona, Spain, Apr. 2006; http://archvlsi.ics.forth.gr/bpbenes/

Congestion elimination using proactive admissions



- Ensure all injected packets can fit in fabric-output buffers
  - fabric output buffers never exert backpressure
- Output buffer credits requested by inputs – scheduled by output arbiter
  - injection rates are fair end2end – no “parking lot” problem
- With good traffic distribution (inverse-multiplexing)
  - intermediate buffers never fill up as well → backpressure eliminated!
- Minimal in-fabric packet delay (queues usually empty) & no packet drops
- → low flow completion times

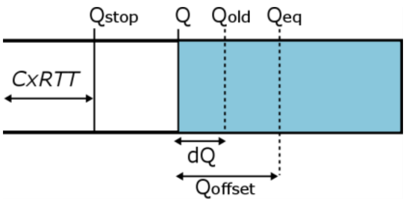
Hotspot Traffic: the load for two outputs is 1.1 x C



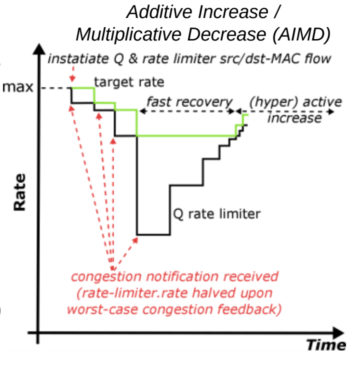
- Indiscriminate bckpr: cells to non-hotspot dests have very large delays
- Req-grant bckpr. performs as per-flow queues, but uses shared queues
  - O(N) *less* queues
- Higher latency at low loads due to request-grant msgs
- Need to handle contention among requests using per-flow req. cnts

Ethernet Quantized Congestion Control (QCN)

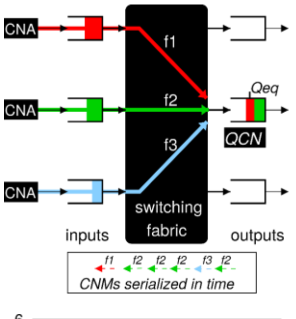
- Congestion point @ sw. queues: every ~ 100 new frames, compute congestion feedback value (fd)
  - fd = Q_{offset} + w • dQ
  - if fd > 0, issue congestion notification msg (CNM) to src NIC, identifying MAC-pair flow



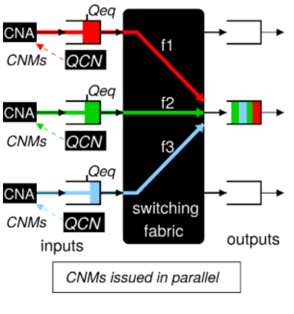
- Non-congested flows share a single NIC queue
- Upon receiving a CNM for a dest MAC:
  - alloc. flow queue & rate limiter (RL)
  - target = current RL
  - new RL = *f* (RL, fd)
- Recovery upon absence of CNMs:
  - fast recovery: RL = (RL+ target) / 2
  - active increase: increment target (+x Mbps)

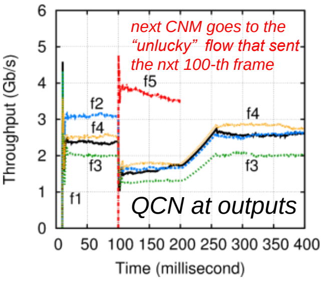


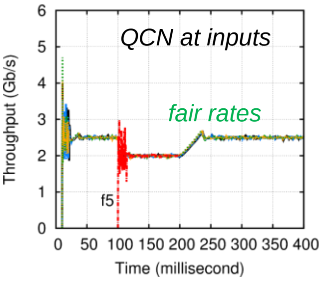
QCN at outputs is unfair: who's to get the next CNM?



QCN CPs at outputs versus. at inputs under fan-in scenario

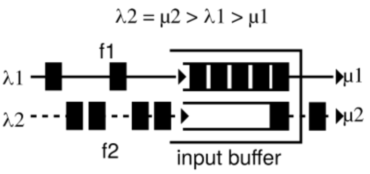


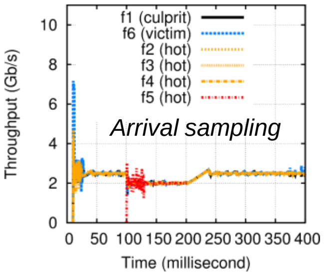


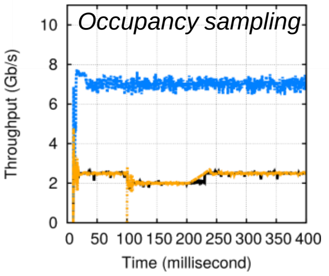


Arrival-sampling QCN at inputs does not protect non-congested flows

- Departures from switch input buffer (VOQs) not FIFO (order enforced by arbiter)
- Arrival sampling: CNM flow of nxt “100th” frame
  – flows CNM’ed proportionally to arrival rates
  – unfair with non-FIFO service out of buffer
- Occupancy sampling: CNM the flow w. largest “backlog” (~ arrivals rate *minus* departures rate)







Chrysos e.a. “Unbiased QCN for scalable server-rack fabrics”, IBM Technical Report: http://domino.research.ibm.com/library/cyberdig.nsf/papers/EF857B498A21B3EF85257D6A002E2235

Small & flat FCTs for scale-out latency-sensitive apps

- Traditionally, congestion control cares about tput, fairness of congested link b/w & latency of non-congested pkts – congested pkts → high latency
- Main perf. metric of interest when a group of servers collectively work on latency-sensitive (datacenter) applications
  – flow completion time (FCT) -- small for as many flows as possible
- Measures/actions
  – TCP variants: keep in-fabric queues empty (Alizadeh, SIGCOM 2010)
  – datacenter networks rethink lossy (current) vs. lossless flow control
    - avoid delayed TCP retransmission: up to many millisecond to recover pkts
  – shift also from deterministic routing to flow-level (and more recently also to packet-level) multipathing M. Al-Fares, SIGCOMM’08, Zats SIGCOMM’12)
    - what about flows crossing temporarily congested paths ?
  – proactive (scheduled injections) also drawing renewed interest
    - don’t blindly inject packets--scheduled injections reduce in-fabric backlogs
  – host (network-stack) latencies: many tens of μs;
    - ok for previous netw. (latency 100s μs) – but new nets only fewμs
    - avoid excessive packet copies (RDMA?), TCP offloading

Shortest-job-first scheduling: let small go first

Alizadeh, e.a., “pFabric: Minimal Near-Optimal Datacenter Transport”

- Switches store packets in per-input priority queues sorted based on the remaining size of the corresponding TCP flows
  - but how to find the remaining size of a TCP flow?
    - heuristic: use the already transmitted size as indication
  - schedule first packets from the flow with the smallest remaining size
  - when buffer exceeds threshold, drop packets from the flow with the largest remaining size

- Cost of priority queues
  - commodity (cheap) switches have small on-chip buffers (a few hundreds of 1500B frames per input)
  - h/w comparators for a few-hundred values is feasible

Virtual switches in cloud (computing) datacenters

- Another layer of switching inside hosts (likely in s/w → versatile/flexible)
  - VM → vNIC → vSwitch → **NIC** → **fabric** → **NIC** → vSwitch → vNIC → VM
- Ping between two remote physical servers: 21 μs; ping between two collocated VMs: 221 μs

- vSwitches have “soft” capacity links: transfer pkts when “on” CPU
  - excessive packet drops inside dest host (Crisan, e.a., SIGCOMM 2013)
  - offloaded vSwitch functions inside hardware network adapters (SRIOV)
  - good but not fully SDN compatible

- Lossless vSwitches (Crisan, e.a. SIGCOM 2013)
  - vSwitch backpr. can propagate inside physical network → excessive blocking
  - reserve endpoint vSwitch credits before injecting pkts from host
    - Crisan, Birke, Chrysos, Minkenberg, Gusat, “zFabric: how to virtualized lossless Ethernet”, IEEE Cluster 2014

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

Communication Cost versus Buffer/Logic Cost

- On-Chip: millions of transistors - hundreds of pins
- Off-Chip: data from Hot Interconnects '95 keynote speech, by A. Fraser, VP, AT&T Bell Labs:

speed of transmission line	45	Mb/s
cost of long distance xmission	45	$/mile/month
speed of signal propagation	7	microsec/mile
round-trip window size	79	bytes/mile
cost of 16 MByte DRAM	1000	$
cost of window size memory	0.5	cents/mile
investment write-down period	36	months
cost of queue mem. per month	0.014	cents/mile/month
ratio: transmission/memory cost	330,000 to 1	

Per-Connection Queueing & FC:

How many ``Windows'' of Buffer Space?

- windowSize(VCi) = RTT * peakThroughput(VCi)

  ⇒ windowSize(VCi) < or << windowL := RTT * throughput(Link)
- cost(Link) ~= 330,000 * cost(windowL)
- lossy flow control usually operates the network with goodput reaching up to 70 - 80 % of link throughput
- lossless flow control operates up to 98 - 100 % link utilization
- the 20-30 % extra utilization with lossless FC is worth approx. 10 to 100 thousand windowL's worth of extra buffer memory

⇒ if lossless flow control can yield its link utilization advantage with less than a few tens of thousands of windowL's of extra buffer memory, then lossless flow control is a clear win

- indeed, lossless FC can do that, even with quite less buffer space...