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
  – 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

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
The “ATLAS I” Credit Flow-Control Protocol


- 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: \( b=1 \))
64x64 fabric: 6-stage banyan using 2x2 elements
20-cell or 20-flit bursts, uniformly destined

Saturation Throughput

Buffer Space (=Lanes) per Link
(cells or flits)

(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.

Delay (cell times)

Number of Lanes (L)

(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

### Table Credit chip control & management
del
del
gen
ser
hdr
pattrn
mm

<table>
<thead>
<tr>
<th>Table</th>
<th>Load Mon.</th>
<th>imp. match</th>
<th>del</th>
<th>ser</th>
<th>hdr</th>
<th>pattrn</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>12</td>
<td>13</td>
</tr>
<tr>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

### Table Load Mon.

<table>
<thead>
<tr>
<th>VPC/VCI Translat. Table</th>
<th>Sched Count</th>
<th>Cell Buffer &amp; Switching</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
</tbody>
</table>

### Core:

- Cell Buffer & Switching
- Header Pr., Rt'n, Transl.
- Credit-based Flow Control
- Queue Pointer Management
- Scheduling, Pop. Counts
- Ctrl/Mgt, Load Mon, Misc.
- Elastic buf., I/O Link Intf.

### Design Effort

<table>
<thead>
<tr>
<th>Cell Buffer &amp; Switching</th>
<th>Gates</th>
<th>FF</th>
<th>SRAM</th>
<th>Area</th>
<th>Power</th>
</tr>
</thead>
<tbody>
<tr>
<td>15%</td>
<td>20%</td>
<td>20%</td>
<td>20%</td>
<td>10%</td>
<td>25%</td>
</tr>
</tbody>
</table>

### Periphery:

- GigaBaud Transceivers
- Pads & Drivers

<table>
<thead>
<tr>
<th>GigaBaud Transceivers</th>
<th>Pads &amp; Drivers</th>
</tr>
</thead>
<tbody>
<tr>
<td>225 mm²</td>
<td>9. W</td>
</tr>
</tbody>
</table>

CS-534 - Copyright University of Crete

6.3 Buffer Space vs. Number of Flows
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

**Switching Fabrics without Backpressure 1: Large Buffers**

Large, off-chip buffers (DRAM); total throughput = \(2N \log N\)

**Switching Fabrics without Backpressure 2: Internal Speedup**

Small, on-chip buffers

Large, off-chip buffers (DRAM); total throughput = \((s+1)N\)

Speedup \(s > 1\): under bursty or non-uniform traffic; \(s \gg 1\)...
6.3 Buffer Space vs. Number of Flows

Backpressure Cost/Benefit 1:
No Backpressure, Large Off-Chip Buffers

Core:

- Cell Buf. & Switching: Gates 20%, FF 20%, SRAM 30%+
- Hdr, R'ng, Transl.: Gates 20%, FF 30%, SRAM 60%
- Credit-b. Flow Ctrl: Gates 20%, FF 10%, SRAM 10%
- Q_Ptr, Sch, Ctrl, etc.: Gates 42%, FF 45%, SRAM 35%

Elastic buf., I/O Intf.

- Elastic buf.: Gates 25%+, FF 10%, SRAM 5%+
- I/O Intf.: Gates 6%+, FF 25%+

Periphery:

- Off-Chip Communication Cost: 35% x 2
- Backpressure Cost/Benefit: 55% x 2

CS-534 - Copyright University of Crete

Backpressure Cost/Benefit 2:
No Backpressure, Internal Speedup, Output Queues

Core:

- Cell Buf. & Switching: Gates 20% x S^2, FF 20% x S^2, SRAM 20%
- Hdr, R'ng, Transl.: Gates 20%, FF 30%, SRAM 60%
- Credit-b. Flow Ctrl: Gates 20%, FF 10%, SRAM 10%
- Q_Ptr, Sch, Ctrl, etc.: Gates 42%, FF 45%, SRAM 35%

Elastic buf., I/O Intf.

- Elastic buf.: Gates 25%+, FF 10%, SRAM 5%+
- I/O Intf.: Gates 6%+, FF 25%+

Periphery:

- Off-Chip Communication Cost: 35% x S
- Backpressure Cost/Benefit: 55% x S

CS-534 - Copyright University of Crete
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

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
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)
  - $fd = Q_{offset} + w \cdot dQ$
  - 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 at outputs

QCN at inputs

6.3 Buffer Space vs. Number of Flows
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 nx “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)

Arrival sampling

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 \(\rightarrow\) versatile/flexible)
  - VM \(\rightarrow\) vNIC \(\rightarrow\) vSwitch \(\rightarrow\) NIC \(\rightarrow\) fabric \(\rightarrow\) NIC \(\rightarrow\) vSwitch \(\rightarrow\) vNIC \(\rightarrow\) VM
- Ping between two remote physical servers: 21 \(\mu\)s; ping between two collocated VMs: 221 \(\mu\)s
- 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 \(\rightarrow\) excessive blocking
  - reserve endpoint vSwitch credits before injecting pkts from host
**Buffer Space for Bounded Peak-to-Average Rate Ratio**

- Assume \( \frac{R_{\text{peak}}(i)}{R_{\text{average}}(i)} \leq PAR \) 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 \( RTT \cdot R_{\text{peak}}(i) \)
- Buffer space for all flows is \( \sum (RTT \cdot R_{\text{peak}}(i)) = RTT \cdot \sum (R_{\text{peak}}(i)) \leq RTT \cdot \sum (PAR \cdot R_{\text{average}}(i)) = PAR \cdot RTT \cdot \sum (R_{\text{average}}(i)) \leq PAR \cdot (RTT \cdot R_{\text{link}}) \)

\[ \Rightarrow \text{Allocate buffer space} = PAR \text{ number of "windows"} \]

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 reallocating it (not obvious how to)


**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.
**Communication Cost versus Buffer/Logic Cost**

- **On-Chip:** millions of transistors - hundreds of pins
- **Off-Chip:**
  - 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?**

- \( \text{windowSize}(VCI) = \text{RTT} \times \text{peakThroughput}(VCI) \)
  \( \Rightarrow \text{windowSize}(VCI) < \text{or} < \text{windowL} := \text{RTT} \times \text{throughput(Link)} \)
- \( \text{cost(Link)} \sim = 330,000 \times \text{cost(windowL)} \)
- 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
  \( \Rightarrow \) 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...