Load-Balancing and Parallelism for the Internet

Stanford University Ph.D. Oral Examination
Tuesday, Feb 18th 2003

Sundar Iyer (sundaes@cs.stanford.edu)
Department of Computer Science
Stanford University,
http://www.stanford.edu/~sundaes
Motivation

- Build routers with performance guarantees

- Guarantees are
  - Statistical (100% throughput)
  - Deterministic (work-conserving, delay guarantees)

- Hard to build big routers with guarantees

- This talk:
  - Deterministic guarantees
  - Use load-balancing and parallelism
Contents

1. Parallel routers: work-conserving
2. Parallel routers: delay guarantees
3. Parallel packet buffers
4. Summary of contributions
Output-Queued Router

Gives 100% throughput if $\lambda < 1$ and output work conserving

- Work-conservation: $\Rightarrow$ 100% throughput, minimizes delay, delay guarantees possible
- Problem: Memory bandwidth not scalable
- Previous attempts: Very complicated or ad hoc
- My approach: Deterministic parallel work-conserving output-queued router
Parallel Output-Queued Router

Gives 100% throughput if $\frac{\lambda}{k} < \frac{1}{k}$ and output work conserving

Is this work conserving?
Parallel Output-Queued Router
(May not be work-conserving)

Constant size packets

At most two memory operations per time slot: 1 write and 1 read
Problem

Problem:

- How can we design a parallel output-queued work-conserving router from slower parallel memories?

Theorem 1: (sufficiency)

- A parallel output-queued router is work-conserving with $3N - 1$ memories that can perform at most one memory operation per time slot.
Re-stating the Problem

- There are $K$ cages which can contain an infinite number of pigeons.

- Assume that time is slotted, and in any one time slot
  - at most $N$ pigeons can arrive and at most $N$ can depart.
  - at most 1 pigeon can enter or leave a cage via a pigeon hole.
  - the time slot at which arriving pigeons will depart is known.

- For any router:
  - What is the minimum $K$, such that all $N$ pigeons can be immediately placed in a cage when they arrive, and can depart at the right time?
Intuition for Theorem 1

- Only one packet can enter a memory at time $t$
- Only one packet can enter or leave a memory at any time
Proof of Theorem 1

- When a packet arrives in a time slot it must choose a memory not chosen by
  1. The $N - 1$ other packets that arrive at that timeslot.
  2. The $N$ other packets that depart at that timeslot.
  3. The $N - 1$ other packets that can depart at the same time as this packet departs (in future).

- Proof:
  - By the pigeon-hole principle, $3N - 1$ memories that can perform at most one memory operation per time slot are sufficient for the router to be work-conserving.
The Parallel Shared Memory Router

At most **one** operation - a write or a read per time slot

From theorem 1, $k = 7$ memories don't suffice .. but 8 memories do
Distributed Shared Memory Router

- The central memories are moved to distributed line cards and shared
- Memory and line cards can be added incrementally
- From theorem 1, \(3N-1\) memories which can perform one operation per time slot i.e. a total memory bandwidth of \(\approx 3NR\) suffice for the router to be work-conserving
Corollary 1

Problem:

- What is the switch bandwidth for a work-conserving DSM router?

Corollary 1: (sufficiency)

- A switch bandwidth of $4NR$ is sufficient for a distributed shared memory router to be work-conserving

Proof 1:

- There are a maximum of 3 memory accesses and 1 port access
Corollary 2

Problem:

- What is the switching algorithm for a work-conserving DSM router?
  - Bus: No algorithm needed, but impractical
  - Crossbar: Algorithm needed because only permutations are allowed

Corollary 2: (existence)

- An edge coloring algorithm can switch packets for a work-conserving distributed shared memory router

Proof:

- Follows from König’s theorem - Any bipartite graph with maximum degree \( \Delta \) has an edge coloring with \( \Delta \) colors
## Summary

Routers which give 100% throughput

<table>
<thead>
<tr>
<th>Fabric</th>
<th># Mem.</th>
<th>Mem. BW per Mem</th>
<th>Total Mem. BW</th>
<th>Switch BW</th>
<th>Switch Algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output-Queued</td>
<td>Bus</td>
<td>N</td>
<td>(N+1)R</td>
<td>N(N+1)R</td>
<td>NR</td>
</tr>
<tr>
<td>C. Shared Mem.</td>
<td>Bus</td>
<td>1</td>
<td>2NR</td>
<td>2NR</td>
<td>2NR</td>
</tr>
<tr>
<td>Input Queued</td>
<td>Crossbar</td>
<td>N</td>
<td>2R</td>
<td>2NR</td>
<td>NR</td>
</tr>
<tr>
<td>CIOQ (Cisco)</td>
<td>Crossbar</td>
<td>2N</td>
<td>3R</td>
<td>6NR</td>
<td>2NR</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2N</td>
<td>3R</td>
<td>6NR</td>
<td>3NR</td>
</tr>
<tr>
<td>P Shared Mem.</td>
<td>Bus</td>
<td>k</td>
<td>3NR/k</td>
<td>3NR</td>
<td>2NR</td>
</tr>
<tr>
<td>DSM (Juniper)</td>
<td>Xbar</td>
<td>N</td>
<td>3R</td>
<td>3NR</td>
<td>4NR</td>
</tr>
<tr>
<td></td>
<td></td>
<td>N</td>
<td>3R</td>
<td>3NR</td>
<td>6NR</td>
</tr>
<tr>
<td></td>
<td></td>
<td>N</td>
<td>4R</td>
<td>4NR</td>
<td>4NR</td>
</tr>
<tr>
<td>PPS - OQ</td>
<td>Clos</td>
<td>Nk</td>
<td>2R(N+1)/k</td>
<td>2N(N+1)R</td>
<td>4NR</td>
</tr>
<tr>
<td>PPS — Shared Memory</td>
<td>Clos</td>
<td>Nk</td>
<td>4NR/k</td>
<td>4NR</td>
<td>4NR</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Nk</td>
<td>2NR/k</td>
<td>2NR</td>
<td>2NR</td>
</tr>
</tbody>
</table>

1 Note that lower mem. bandwidth per memory implies higher random access time, which is better
Contents

1. Parallel routers: work-conserving
2. Parallel routers: delay guarantees
3. Parallel packet buffers
4. Summary of contributions
Delay Guarantees

one output, many logical FIFO queues

constrained traffic

$\lambda \to m \to \mu = 1$

Weighted fair queueing sorts packets

PIFO models

- Weighted Fair Queueing
- Weighted Round Robin
- Strict priority etc.

one output, single PIFO queue

constrained traffic

$\lambda \to \text{Push In First Out (PIFO)} \to \mu = 1$
Delay Guarantees

Problem:

- How can we design a parallel output-queued router from slower parallel memories and give delay guarantees?

This is difficult because:

- The counting technique depends on being able to predict the departure time and schedule it.

- The departure time of a cell is not fixed in policies such as strict priority, weighted fair queueing etc.
Theorem 2

- Theorem 2: (sufficiency)

- A parallel output-queued router can give delay guarantees with $4N - 2$ memories that can perform at most one memory operation per time slot.
Intuition for Theorem 2

$N=3$ port router

DT = 3

... 9 8 7

DT = 2

... 6 5 4

DT = 1

Departure Order

FIFO: Window of memories of size N-1 that can't be used

PIFO: 2 windows of memories of size N-1 that can't be used

N-1 packets before cell at time of insertion

2.5

... 8 7 6

... 5 4 3

N-1 packets after cell at time of insertion

2.5 1.5

... 7 6 5

... 4 3 2.5

... 2 1.5 1

Departure Order
Proof of Theorem 2

A packet cannot use the memories:

1. Used to write the $N-1$ arriving cells at $t$.
2. Used to read the $N$ departing cells at $t$.

3. Will be used to read the $N-1$ cells that depart before it.
4. Will be used to read the $N-1$ cells that depart after it.
Summary - Routers which give delay guarantees

<table>
<thead>
<tr>
<th>Output-Queued</th>
<th>Fabric</th>
<th># Mem.</th>
<th>Mem. BW per Mem.</th>
<th>Total Memory BW</th>
<th>Switch BW</th>
<th>Switch Algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>C. Shared Mem.</td>
<td>Bus</td>
<td>N</td>
<td>(N+1)R</td>
<td>N(N+1)R</td>
<td>NR</td>
<td>None</td>
</tr>
<tr>
<td>Input-Queued</td>
<td>Crossbar</td>
<td>N</td>
<td>2R</td>
<td>2NR</td>
<td>2NR</td>
<td>None</td>
</tr>
<tr>
<td>CIOQ (Cisco)</td>
<td>Crossbar</td>
<td>2N</td>
<td>3R</td>
<td>6NR</td>
<td>2NR</td>
<td>Marriage</td>
</tr>
<tr>
<td></td>
<td>Crossbar</td>
<td>2N</td>
<td>3R</td>
<td>6NR</td>
<td>3NR</td>
<td>Time Reserve</td>
</tr>
<tr>
<td>P. Shared M</td>
<td>Bus</td>
<td>k</td>
<td>4NR/k</td>
<td>4NR</td>
<td>2NR</td>
<td>C. Sets</td>
</tr>
<tr>
<td>DSM (Juniper)</td>
<td>Xbar</td>
<td>N</td>
<td>4R</td>
<td>4NR</td>
<td>5NR</td>
<td>Edge Color</td>
</tr>
<tr>
<td></td>
<td>Xbar</td>
<td>N</td>
<td>4R</td>
<td>4NR</td>
<td>8NR</td>
<td>C. Sets</td>
</tr>
<tr>
<td></td>
<td>Xbar</td>
<td>N</td>
<td>6R</td>
<td>6NR</td>
<td>6NR</td>
<td>C. Sets</td>
</tr>
<tr>
<td>PPS - OQ</td>
<td>Clos</td>
<td>Nk</td>
<td>3R(N+1)/k</td>
<td>3N(N+1)R</td>
<td>6NR</td>
<td>C. Sets</td>
</tr>
<tr>
<td>PPS - Shared Memory</td>
<td>Clos</td>
<td>Nk</td>
<td>6NR/k</td>
<td>6NR</td>
<td>6NR</td>
<td>C. Sets</td>
</tr>
<tr>
<td></td>
<td>Clos</td>
<td>Nk</td>
<td>2NR/k</td>
<td>2NR</td>
<td>2NR</td>
<td>-</td>
</tr>
</tbody>
</table>
Contents

1. Parallel routers: work-conserving
2. Parallel routers: delay guarantees
3. Parallel packet buffers
4. Summary of contributions
Packet Buffering

- **Big**: For TCP to work well, the buffers need to hold one $RTT$ (about 0.25s) of data.

- **Fast**: Clearly, the buffer needs to store (retrieve) packets as fast as they arrive (depart).
An Example
Packet buffers for a 40Gb/s line card

Problem is solved if a memory can be (random) accessed every 4 ns and store 10Gb of data
Available Memory Technology

- **Use SRAM?**
  - Fast enough random access time, but
  - Too low density to store 10Gbits of data.

- **Use DRAM?**
  - High density means we can store data, but
  - Can’t meet random access time.
Problem:

How can we design high speed packet buffers from commodity available memories?
Can’t we just use lots of DRAMs in parallel?

![Diagram]

**Write Rate, \( R \)**

One 40B packet every 8ns

**Buffer Manager**

**Read Rate, \( R \)**

One 40B packet every 8ns

Scheduler Requests
Works fine if there is only one FIFO queue

Buffer Manager (on chip SRAM)

Aggregate 320B for the queue in fast SRAM and read and write to all DRAMs in parallel
In practice, buffer holds many FIFOs

- In an IP Router, $Q$ might be 200.
- In an ATM switch, $Q$ might be $10^6$. 

![Diagram of buffer and FIFOs]

- We don't know which head of line packet the scheduler will request next?

**Write Rate**, $R$

- One 40B packet every 8ns

**Read Rate**, $R$

- One 40B packet every 8ns

Scheduler Requests
Problems

1. A 320B block will contain packets for different queues, which can't be written to, or read from the same location.

2. Suppose we write packets for different queues to different locations. How do we know that the memory will be available for reading when we need to retrieve the packet?

Requirement

1. We will have to aggregate 320B for every queue and read and write to all DRAMs in parallel
Parallel Packet Buffer

Hybrid Memory Hierarchy

Large DRAM memory holds the body of FIFOs

Buffer Manager

(ISIC with on chip SRAM)

Small tail SRAM cache for FIFO tails

Scheduler Requests

Small head SRAM cache for FIFO heads

b = degree of parallelism

Reading b bytes

Writing b bytes

Arriving Packets

Departing Packets

R

R
Problem:

- What is the minimum size of the SRAM needed for the parallel buffer so that every packet is available immediately in the SRAM when requested?

Theorem 3: (Necessity)

- An SRAM size of size $Q_w = Q(b - 1)(2 + \ln Q)$ bytes is necessary.
  
  - $w$ is the size of each queue
Why do we need a large SRAM?

\[ Q = 4, \ w = 3, \ b = 3 \]

At \( t = 6 \) if the scheduler requests a black packet it will not be in the SRAM.
Theorem 4

- **Theorem 4: (sufficiency)**
  
  - An SRAM cache of size $Q_w = Q_b(2 + \ln Q)$ bytes is sufficient so that every packet is available immediately in the SRAM when requested.

- **Discussion:**
  
  - The algorithm replenishes the queue with the most urgent need of replenishment.

  - It is almost optimal.
Intuition for Theorem 4

- The maximum number of un-replenished requests for any $i$ queues $w_i$, is the solution of the difference equation -

$$w_i \geq (w_{i-1} - b) + \frac{(w_{i-1} - b)}{i-1}; \quad i \in \{2, 3, ... Q\}$$

- with boundary conditions $w_q < Qb$

---

Examples:

1. 40Gb/s line card, $b=640$, $Q=128$: SRAM = 560kBytes
2. 160Gb/s line card, $b=2560$, $Q=512$: SRAM = 10MBytes
Theorem 5

Problem:

- What is the minimum size of the SRAM so that every packet is available in the SRAM within a bounded pipeline latency when requested?

Theorem 5: (necessity and sufficiency)

- An SRAM cache of size $Q_w = Q(b - 1)$ bytes is both necessary and sufficient if the pipeline latency is $Q(b - 1) + 1$ time slots.
Intuition for Theorem 5

- The algorithm replenishes the queue which is going to be in the most urgent need of replenishment
  - If we use a lookahead buffer to know the requests “in advance”, we can identify the queue which will empty earliest
  - This increases the pipeline latency from when a request is made until the byte is available.

Example:

1. 160Gb/s line card, \( b=2560, Q=128 \): SRAM = 160kBytes, latency is 8\(\mu s\).
Theorem 6

- **Problem:**

  - What is the minimum size of the SRAM needed so that every packet is available in the SRAM within a bounded pipeline latency $x$ in the range $(0, Q(b - 1) + 1)$ when requested?

- **Theorem 6: (sufficiency)**

  - An SRAM cache of size $Q_w \approx Q_b(2 + \ln Q_b/x)$ bytes is sufficient.
Discussion of Theorem 6

\( Q=1000, \ b = 10 \)

\[
\frac{dw}{dx} \approx -\frac{1}{x}
\]
Contents

1. Parallel routers: work-conserving
2. Parallel routers: delay guarantees
3. Parallel packet buffers
4. Summary of contributions
Summary of Contributions
Covered in this talk

1. **Parallel routers:**

2. **Parallel packet buffers:**
Summary of Contributions

Not covered in this talk

3. **Other switches (load-balancing):**
   
   
   
   

4. **Parallel packet switch (multicast):**
   
Summary of Contributions
Not covered in this talk

5. Parallel architecture (statistics counters):


6. Parallel architecture (packet state):

- S. Iyer, N. McKeown, “Maintaining State in Router Line Cards”. In preparation for *IEEE Communication Letters*.

7. Parallel packet classification:


Summary of Contributions
Not covered in this talk

8. **Parallel switching algorithms** (Buffered crossbars):


9. **Load-balancing** (IP backbone):


10. **Greedy switching algorithms**:

Acknowledgements

1. Nick
2. Balaji
3. Members of my committee - Sunil, Mark and Rajeev
4. Industry:
   • Cisco - Flavio
   • Sprint Labs - Christophe, Supratik, Nina and the IP group
   • SwitchOn/PMC-Sierra - Ajit, Moti and the whole team
5. Research “co-conspirators”
6. Members of the research group
7. Members of network-group
8. Department staff
9. Friends, friends & friends from all over the globe …
10. The couch on the 5th floor, the basement vending machine, …
11. Mom

- I was told: “A good way to judge how well you are doing, is to measure the quality of people with whom you interact.
- By that criteria, I could not have asked for more ….