High Performance Switching and Routing

Nick McKeown
Assistant Professor of Electrical Engineering and Computer Science
nickm@stanford.edu
http://www.stanford.edu/~nickm
1. The Demand for Bandwidth
2. The Shortage of Switching/Routing Capacity
3. The Architecture of Switches and Routers
4. Some (of our) solutions
What’s the Problem?

Most things
The demand

The San Jose NAP

Source: http://www.mfsdatanet.com/MAE/west.stats.html
The supply

Router Performance (packets/second)

10^3 10^4 10^5 10^6


High Performance Switching and Routing
Why we need faster switches/routers

Demand

Supply

1986

1992

1997

packets/second
Traffic Inversion
10 years ago
Why is this a problem?

11/01/96 Packet Loss for BBNPlanet (AS1) Between Mae-West and Sprint

Packet Loss (%)

November 1st, 1996
1. The Demand for Bandwidth

2. The Shortage of Switching/Routing Capacity

3. The Architecture of Switches and Routers

4. Some (of our) solutions
The Architecture of Switches and Routers

Generic Packet Switch:
(e.g. IP Router, ATM Switch, LAN Switch)
Performance of IP Routers

- Most routers do this poorly!

Header processing time

Copy rate

Arrival rate

Min back-to-back packet size

Packet size

Time

Most routers do this ~ ok
The Evolution of Routers

The first shared memory routers

Routing CPU

Buffer Memory

Line Card

DMA

MAC

High Performance Switching and Routing
The Evolution of Routers

The first shared memory routers

Routing CPU

Buffer Memory

DMA

Line Card

MAC

DMA

Line Card

MAC

DMA

Line Card

MAC
The Evolution of Routers
Reducing the number of bus copies

High Performance Switching and Routing
The Evolution of Routers

Reducing the number of bus copies

Updates

Routing CPU

Buffer Memory

Line Card

Buffer Memory

Route Cache

DMA

MAC

High Performance Switching and Routing
The Evolution of Routers

Avoiding bus contention

Advantage:
Non-blocking backplane—high throughput

Disadvantage:
Difficult to provide QoS
1. The Demand for Bandwidth
2. The Shortage of Switching/Routing Capacity
3. The Architecture of Switches and Routers
4. Some (of our) solutions
1. Accelerating Forwarding Decisions:
   - Longest-matching prefixes

2. Interconnections: Switched Backplanes
   - Input Queueing
     - Theory
     - Unicast
     - Multicast
   - Fast Buffering
   - Speedup
   - The Tiny Tera Project
Routing Lookups

Routing Table:

Class A  Class B  Class C  D

Exact Match
(hash, cache, pipeline...)

212.17.9.4

Class A

Class B

Class C

Routing Table:

212.17.9.0  Port 4
Routing Lookups with CIDR (“supernetting”)

CIDR uses “longest matching prefix” routing:

212.0.0.0/8
212.17.0.0/16
212.17.9.0/24
212.17.9.4

Hashing, caching and pipelining are hard!
Perform Lookups Faster

Observation #1:

- Size of Routing Tables
- Cost of Memory (per byte)

Time
Performing Lookups Faster

**Observation #2:**

Prefix length

Number in routing table

---

212.17.9.0/24

0 2 32 -1

256

212.17.9.0/24

212.17.9.4

2^{32}-1

0
20 million lookups per second

16Mbytes of 50ns DRAM

212.17.9.1

<table>
<thead>
<tr>
<th>Port 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
</tr>
<tr>
<td>0</td>
</tr>
<tr>
<td>look further</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>Port 4</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>Port 3</td>
</tr>
</tbody>
</table>

<1Mbyte of 50ns DRAM

Port 4
Port 5

Port 4
Port 5

16Mbytes of 50ns DRAM

20 million lookups per second

High Performance Switching and Routing
1. Accelerating Forwarding Decisions:
   • Longest-matching prefixes

2. Interconnections: Switched Backplanes
   • Input Queueing
     — Theory
     — Unicast
     — Multicast
   • Fast Buffering
   • Speedup
   • The Tiny Tera Project
Should we use shared memory or input-queueing?

**Shared Memory:**

- **Advantages:**
  - Highest Throughput.
  - Possible to control packet delay.

- **Disadvantages:**
  - N-fold internal speedup

**Input Queueing:**

- **Advantages:**
  - Simplicity
  - High Bandwidth

- **Disadvantages:**
  - HOL Blocking
  - Less efficient
  - Difficult to control packet delay.
Memory Bandwidth

- Memory Size
- Memory Speed

SRAM: memory speed increasing with time
DRAM: memory speed relatively unchanged with time

High Performance Switching and Routing
An aside: How fast can shared memory operate?

How fast can a 16 port switch run with this architecture?

5ns per packet $\times$ 2 memory operations per cell time

$\Rightarrow$ aggregate bandwidth is 160Gb/s
Because of a shortage of memory bandwidth, most multigigabit and terabit switches and routers use either:

1. Input Queueing, or
2. Combined Input and Output Queueing.
Head of Line Blocking

The Problem

\[ \rho_{max} = 2 - \sqrt{2} = 58\% \]

A Solution....

"Virtual Output Queueing"

\[ \rho_{max} = 100\% \]
High Performance Switching and Routing

...but requires scheduling...
which is equivalent to graph matching

Request Graph

Bipartite Matching
(Weight = 18)
1. **iSLIP** — Weight = 1
   — Iterative round-robin
   — Simple to implement

2. **iLQF** — Weight = Occupancy

3. **iOCF** — Weight = Cell Age

4. **LPF** — Weight = Backlog
   - Simple, fast, efficient
   - Good for non-uniform traffic. Complex!
   - Good for non-uniform traffic. Simple.
Achieving 100% Throughput

Longest Queue First & Oldest Cell First

\[
\text{Weight} = \frac{\text{Queue Length}}{\text{Waiting Time}} \rightarrow 100\%
\]

Maximum weight

High Performance Switching and Routing
Theorem:

Both LQF and OCF can achieve 100% throughput for independent traffic both uniform and non-uniform.

Proof:

\[ E \left[ \sum_{i,j} L_{i,j}(n) \right] < \infty, \forall n \]

Def \[ \Rightarrow \]

100% throughput

Lyapunov Stability Criterion:

\[ E \left[ V(L(n+1)) - V(L(n)) \mid L(n) \right] \leq 0, \forall |L(n)| > k \]

http://tiny-tera.stanford.edu/~adisak/research.html
Approximating LQF and OCF

*iLQF & iOCF*

Iteration steps

Step 1. Request

Step 2. *Grant to the largest request*

Step 3. *Accept grant to the largest request*
Problem is in Comparators

**iLQF and iOCF**

**Grant Arbiter**, 1

**Accept Arbiter**, 1

**Clear Requests**

**Requests**

**Matches**

**Decoder**

Requests

L1,1

L2,1

LN,1

LogN

Input i

Accept Arbiter, 1

L1,1

LN,1

LogN

Input i

Grant Arbiter N

L1,N

L2,N

LN,N

Requests

Clear Requests

Accept Arbiter N

L1,N

L1,N

L1,N

Matches
Solution to Complexity Problem

- Longest Port First (LPF)
- Oldest Port First (OPF)

**Advantages**

- SIMPLER.
  - Can use maximum size matching — O(N^{2.5}).

- FASTER.
  - Move magnitude comparator out of the critical path.
  - Lends itself well to pipelining.
LPF Algorithm

Using Port Occupancy

\[ w_{i, j} = \sum_j L_{i, j} + \sum_i L_{i, j} \]

i.e. \( w_{1, 1} = L_{1,1} + L_{1,2} + L_{1,1} + L_{2,1} \)

Input occupancy

Output occupancy
On The Theorems

**Theorem:**

An LPF match is of both maximum weight and maximum size.

**Theorem:**

LPF can achieve 100% throughput for independent traffic both uniform and non-uniform.

**Proof:**

\[ V(L(n)) = L^T(n)TL(n) \]

\[ \mathbb{E} [V(L(n + 1)) - V(L(n)) \mid L(n)] \leq 0, \forall |L(n)| > k \]
Presorting Inputs & Outputs

Weight request

Permute

High Performance Switching and Routing
<table>
<thead>
<tr>
<th>20</th>
<th>19</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>25</td>
<td>45</td>
<td>44</td>
</tr>
<tr>
<td>22</td>
<td>42</td>
<td>41</td>
</tr>
<tr>
<td>1</td>
<td>21</td>
<td>20</td>
</tr>
</tbody>
</table>

Remove Weights

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

Matching

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

0 1 1 0
Implementation

Input Occup
\{10, 20, 30\}

Output Occup
\{20, 25, 15\}

Sorter

X Bar

Raw Requests
\[
\begin{bmatrix}
1 & 1 & 0 \\
1 & 1 & 0 \\
0 & 0 & 1 \\
1 & 0 & 0 \\
0 & 1 & 0 \\
0 & 0 & 1 \\
\end{bmatrix}
\]

Match

Permuted Requests

\[
\begin{bmatrix}
10 & 0 \\
01 & 0 \\
00 & 1 \\
\end{bmatrix}
\]

\[
\begin{bmatrix}
3 & 2 & 1 \\
2 & 1 & 3 \\
\end{bmatrix}
\]

Maxsize Matching

Input permutation

Output permutation
Multicast Traffic

Queue Architecture

1. Making use of the crossbar
2. Why treat multicast differently?
3. Why maintain a single FIFO queue?
4. Fanout-splitting
Fanout-Splitting

The graph shows the relationship between offered load and average cell latency. Two lines are plotted: one for 'Fanout-splitting' and another for 'No Fanout-splitting'. As the offered load increases, the average cell latency also increases, with the 'Fanout-splitting' line showing a more gradual increase compared to the 'No Fanout-splitting' line.

High Performance Switching and Routing
Multicast Traffic

1. Residue Concentration
2. Tetris-based schedulers
1. Accelerating Forwarding Decisions:
   • Longest-matching prefixes

2. Interconnections: Switched Backplanes
   • Input Queueing
     — Theory
     — Unicast
     — Multicast
   • Fast Buffering
   • Speedup
   • The *Tiny Tera* Project
Fast Buffering
Ping-pong Memory

Buffer Memory

M

Buffer Memory

M/2

Buffer Memory

M/2
memory 1

memory 2

\[ X_1 X_2 = X_1 \]

memory 1

memory 2

\[ x_2 \]

\[ x_1 = x_2 \]

\[ M/2 \]

\[ M/2 \]

\[ x_1 = x_2 \]

\[ M/2 \]

\[ M/2 \]
Fast Buffering

Ping-pong Memory

Occupancy

$M$

$t$

Maximum “cost” = $M/2$
Fast Buffering

Ping-pong Memory

- Single memory: $M$
- Single memory: $M/2$
- Ping-pong: $(M/2, M/2)$

Log(Overflow Rate) vs. Buffer size, $M$

In practice, cost <5%
Some Results

Input Queued Switch

Wastage Factor, \( \omega(R) \equiv \frac{M(R) - \tilde{M}(R)}{M(R)} \)

- \( \omega(R) \) decreases with \( M \)
- \( \omega(R) \) decreases with burstiness
- \( \omega(R) \) decreases with load
- \( \omega(R) \) decreases with number of ports
1. Accelerating Forwarding Decisions:
   • Longest-matching prefixes

2. Interconnections: Switched Backplanes
   • Input Queueing
     — Theory
     — Unicast
     — Multicast
   • Fast Buffering
   • Speedup
   • The Tiny Tera Project
Matching Output Queueing with Input- and Output-Queueing

*How much speedup is enough?*

**Combined Input- and Output-Queueing:**

![Diagram of combined input- and output-queueing](image)

$k$ reads and writes
Matching Output Queueing with Input- and Output-Queueing

*How much speedup is enough?*

Conventional wisdom suggests:

*A speedup \( k = 2 - 4 \) leads to high throughput*
Fact. To match output queueing, with FIFO input queues:

\[ k = N \]

is necessary and sufficient.

Fact. To match output queueing, with virtual output queues:

\[ k = \left( 2 - \frac{1}{N} \right) \]

is necessary and sufficient.
1. **Accelerating Forwarding Decisions:**
   - Longest-matching prefixes

2. **Interconnections: Switched Backplanes**
   - Input Queueing
     - Theory
     - Unicast
     - Multicast
   - Fast Buffering
   - Speedup
   - The *Tiny Tera* Project