High Performance Switching and Routing


Nick McKeown
Assistant Professor of Electrical Engineering and Computer Science
nickm@ee.stanford.edu
http://ee.stanford.edu/~nickm
Our Group

Shang-tse “Da” Chuang, Ken Chang, Pankaj Gupta, Youngmi Joo, Steve Lin, Adisak Mekkittikul, Nick McKeown, Rolf Muralt, Kanta Yamamoto (visitor).
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*

The San Jose NAP

Source: [http://www.mfsdatanet.com/MAE/west.stats.html](http://www.mfsdatanet.com/MAE/west.stats.html)
The supply

Router Performance (packets/second)

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

Years:
- 1986
- 1990
- 1994
- 1997
Why we need faster switches/routers

Demand and Supply over time from 1986 to 1997
• Exponential growth in the number of users.
• Exponential growth in traffic per user per hour.
• Linear growth in hours per user per day.
Dialup Demand

Modem usage at U.C. Berkeley

"America on Hold"

% of modems in use

100%

30%

Time of day

5am 8am

High Performance Switching and Routing
Traffic Inversion

10 years ago
Traffic Inversion

Today
Why is this a problem?

 Packet Loss (%)
The race is on...

- Gigabit Routers
- IP Switching
- Tag Switching
- Layer-3 switches
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 Processor:
(e.g. IP Router, ATM Switch, LAN Switch)
Performance of IP Routers

- Min back-to-back packet size
- Packet size
- Time
- Forwarding Decision Time
- Arrival Time
- Copy Time
Most routers do this poorly!

Most routers do this ~ ok

Performance of IP Routers
The Evolution of Routers

The first shared memory routers

- **Routing CPU**
- **Buffer Memory**

- **DMA**
- **Line Card**
- **MAC**
The Evolution of Routers

The first shared memory routers

Diagram showing the components of a router:
- Routing CPU
- Buffer Memory
- Line Card
- DMA
- MAC

Arrows indicate the flow of data between components.
The Evolution of Routers

Reducing the number of bus copies

Routing CPU

Buffer Memory

Line Card

DMA

Route Cache

Buffer Memory

MAC
The Evolution of Routers
Reducing the number of bus copies

Routing CPU
Buffer Memory

Line Card

updates

High Performance Switching and Routing  Page 21 of 52
The Evolution of Routers

Avoiding bus contention

Advantage:
Non-blocking backplane—high throughput

Disadvantage:
Difficult to provide QoS
Multigigabit Routing

BBN’s Multigigabit Router

- 2.4Gb/s
- 5Mpps
- 50Gb/s Crossbar
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 Lookups:
   - Label-Swapping
   - Longest-matching prefixes

2. Switched Backplanes
   - Input Queueing
     — Theory
     — Unicast
     — Multicast
   - Fast Buffering
   - Speedup

3. Our main project: The Tiny Tera
Routing Lookups

Class A  Class B  Class C  D

212.17.9.4

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

Routing Table:

<table>
<thead>
<tr>
<th>212.17.9.0</th>
<th>Port 4</th>
</tr>
</thead>
</table>

Class A
Class B
Class C
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!
Solution 1: Label Swapping

- **Data**
- **Hdr**
- **Label**

**Forwarding Decision** → **Interconnect** → **Output Scheduler**

- **Signaling & Mgmt Processor**

**Direct Lookup**

- **NewLabel**
- **Port**

*IP Switching, Tag Switching, ARIS, Cell-switched Router,...*
Solution 2:
Perform Lookups Faster!

Observation #1:

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

Time
Performing Lookups Faster

Observation #2:

Number in routing table

Prefix length

0 2 32 -1

256

212.17.9.0/24

212.17.9.4

2^{32}-1
Solution 2 (cont): 20 million lookups per second

16Mbytes of 50ns DRAM

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

<1Mbyte of 50ns DRAM

Port 4
Port 5

212.17.9.1
Port 3
0
look further
Port 3
1
1. Accelerating Lookups:
   - Label-Swapping
   - Longest-matching prefixes

2. Switched Backplanes
   - Input Queueing
     - Theory
     - Unicast
     - Multicast
   - Fast Buffering
   - Speedup

3. Our main project: *The Tiny Tera*
Should we use shared memory or input-queueing?

**Shared Memory:**

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

**Disadvantages:**
- N-fold internal speed-up

**Input Queueing:**

**Advantages:**
- Simplicity
- High Bandwidth

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

Time

Memory Size

SRAM

DRAM

Memory Speed

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

A Solution....

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

\[ \rho_{\text{max}} = 100\% \]

"Virtual Output Queueing"
.....but requires scheduling...

Input 1

\[ A_{1}(t) \rightarrow Q(1,1) \rightarrow A_{1,1}(t) \]
\[ Q(1,1) \rightarrow Q(1,n) \rightarrow D_{1}(t) \]

Input m

\[ A_{m}(t) \rightarrow Q(m,1) \rightarrow A_{m,1}(t) \]
\[ Q(m,1) \rightarrow Q(m,n) \rightarrow D_{n}(t) \]

Matching, M

Output 1

\[ D_{1}(t) \]

Output n

\[ D_{n}(t) \]
....which is equivalent to graph matching

Request Graph

Bipartite Matching
(Weight = 18)
## Practical Algorithms

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Weight</th>
<th>Description</th>
<th>Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>iSLIP</td>
<td>Weight = 1</td>
<td>Iterative round-robin</td>
<td>Simple, fast, efficient</td>
</tr>
<tr>
<td>iLQF</td>
<td>Weight = Occupancy</td>
<td></td>
<td>Good for non-uniform traffic.</td>
</tr>
<tr>
<td>iOCF</td>
<td>Weight = Cell Age</td>
<td></td>
<td>Complex!</td>
</tr>
<tr>
<td>MCFF</td>
<td>Weight = Backlog</td>
<td></td>
<td>Good for non-uniform traffic. Simple!</td>
</tr>
</tbody>
</table>
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
Multicast Traffic

1. Residue Concentration

2. Tetris-based schedulers
1. **Accelerating Lookups:**
   - Label-Swapping
   - Longest-matching prefixes

2. **Switched Backplanes**
   - Input Queueing
     - Theory
     - Unicast
     - Multicast
   - Fast Buffering
   - Speedup

3. **Our main project:** *The Tiny Tera*
Fast Buffering

Ping-pong Memory

Buffer Memory

Buffer Memory

Buffer Memory
Fast Buffering

Ping-pong Memory

Occupancy

\[ M \]

\[ M/2 \]

Maximum “cost” = \( M/2 \)
In practise, cost <5%

**Fast Buffering**

**Ping-pong Memory**

Loss rate with ping-pong: (M/2, M/2)

Loss rate with single memory: M

In practise, cost <5%

**Loss Rate**

**Buffer size, M**
1. Accelerating Lookups:
   • Label-Swapping
   • Longest-matching prefixes

2. Switched Backplanes
   • Input Queueing
     — Theory
     — Unicast
     — Multicast
   • Fast Buffering
   • Speedup

3. Our main project: *The Tiny Tera*
Matching Output Queueing with Input- and Output-Queueing

How much speedup is enough?

Combined Input- and Output-Queueing:

$k$ 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:

\[ k = N \]

Fact To match output queueing, with virtual output queues:

\[ k = 4 \text{ is sufficient} \]

Conjecture: To match output queueing, with VOQs:

\[ k = 2 \text{ is sufficient} \]
1. Accelerating Lookups:
   • Label-Swapping
   • Longest-matching prefixes

2. Switched Backplanes
   • Input Queueing
     — Theory
     — Unicast
     — Multicast
   • Fast Buffering
   • Speedup

3. Our main project: The Tiny Tera