High Speed Router Design

Shivkumar Kalyanaraman
Rensselaer Polytechnic Institute
shivkuma@ecse.rpi.edu
http://www.ecse.rpi.edu/Homepages/shivkuma

Also based on slides of S. Keshav (Ensim), Douglas Comer (Purdue), Raj Yavatkar (Intel), Cyriel Minkenberg (IBM Zurich), Sonia Fahmy (Purdue)

Many slides thanks to Nick McKeown (Stanford),
Overview

- Introduction
- Evolution of High-Speed Routers
- High Speed Router Components:
  - Lookup Algorithm
  - Switching
  - Classification, Scheduling
- Multi-Tbps Routers: Challenges & Trends
What do switches/routers look like?

Access routers
e.g. ISDN, ADSL

Core router
e.g. OC48c POS

Core ATM switch
Shivkumar Kalyanaraman
Cisco GSR 12416

- Dimensions: 6ft x 2ft x 2.5ft
- Capacity: 160Gb/s
- Power: 4.2kW

Juniper M160

- Dimensions: 3ft x 19"
- Capacity: 80Gb/s
- Power: 2.6kW
Where high performance packet switches are used

- Carrier Class Core Router
- ATM Switch
- Frame Relay Switch

The Internet Core

Edge Router

Enterprise WAN access & Enterprise Campus Switch
Where are routers?
Ans: Points of Presence (POPs)
Why the Need for Big/Fast/Large Routers?

- **Interfaces**: Price >$200k, Power > 400W
- Space, power, interface cost economics!
- About 50-60% of i/fs are used for interconnection within the POP.
- Industry trend is towards large, single router per POP.

Shivkumar Kalyanaraman

Rensselaer Polytechnic Institute
Job of router architect

- For a given set of features:

\[
\text{Maximize capacity, } C
\]
\[
s.t. \quad \text{Power, } P < 5kW
\]
\[
\text{Volume, } V < 2m^3
\]
Performance metrics

1. **Capacity**
   - “maximize $C$, s.t. volume $< 2m^3$ and power $< 5kW$”

2. **Throughput**
   - Maximize usage of expensive long-haul links.
   - Trivial with work-conserving output-queued routers

3. **Controllable Delay**
   - Some users would like predictable delay.
   - This is feasible with output-queueing plus weighted fair queuing (WFQ).
Relative performance increase

- DWDM Link speed: x2/8 months
- Router capacity: x2.2/18 months
- Internet: x2/yr
- Moore's law: x2/18 m
- DRAM access rate: x1.1/18 m

Memory speed is not keeping up with Moore’s Law.

- **DRAM**: 1.1x / 18 months
- **Moore’s Law**: 2x / 18 months
- **Line Capacity**: 2x / 7 months
- **Router Capacity**: 2.2x / 18 months
An Example: Packet buffers

40Gb/s router linecard

- **Write Rate,** $R$
  - One 40B packet every 8ns

- **Read Rate,** $R$
  - One 40B packet every 8ns

- **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.
Eg: Problems w/ Output Queuing

- Output queued switches are impractical

Can’t I just use $N$ separate memory devices per output?
Packet processing is getting harder

CPU Instructions per minimum length packet since 1996
First-Generation IP Routers

- Most Ethernet switches and cheap packet routers
- Bottleneck can be CPU, host-adaptor or I/O bus
First Generation Routers

Fixed length “DMA” blocks or cells. Reassembled on egress linecard

Fixed length cells or variable length packets

Typically <0.5Gb/s aggregate capacity
First Generation Routers

Queueing Structure: Shared Memory

Numerous work has proven and made possible:
- Fairness
- Delay Guarantees
- Delay Variation Control
- Loss Guarantees
- Statistical Guarantees

Large, single dynamically allocated memory buffer:
- N writes per “cell” time
- N reads per “cell” time.

Limited by memory bandwidth.
Port mapping intelligence in line cards
Higher hit rate in local lookup cache
Second Generation Routers

Typically <5Gb/s aggregate capacity
Second Generation Routers

As caching became ineffective
Second Generation Routers
Queuing Structure: Combined Input and Output Queuing (CIOQ)

Rate of writes/reads determined by bus speed
Third-Generation Switches/Routers

- Third generation switch provides parallel paths (fabric)
- What’s costly? Bus? Memory, CPU?
Third Generation Routers

Switched Backplane

Typically <50Gb/s aggregate capacity
Third Generation Routers

Queueing Structure

1 write per “cell” time

Rate of writes/reads determined by switch fabric speedup

1 read per “cell” time

Switch

Arbiter
Third Generation Routers

Queueing Structure: VOQs

- 1 write per “cell” time
- Rate of writes/reads determined by switch fabric speedup
- 1 read per “cell” time
- Flow-control backpressure

Per-flow/class or per-output queues (VOQs)
Per-flow/class or per-input queues

Switch

Arbiter
• Size-constrained: 19” or 23” wide.
• Power-constrained: ~<8kW.

Supply: 100A/200A maximum at 48V
Fourth Generation:
Clustering/Multi-stage

Switch Core

Linecards

Optical links

100's of feet

100's of feet
**Key: Physically Separating Switch Core and Linecards**

- Distributes power over multiple racks.
- Allows all buffering to be placed on the linecard:
  - Reduces power.
  - Places complex scheduling, buffer mgmt, drop policy etc. on linecard.
Fourth Generation Routers/Switches

Switch Core

The LCS Protocol

Optical links

100’s of feet

Linecards
Physical Separation

1: Req

2: Grant/credit

3: Data

Per Queue Counters

1 RTT

Linecard

Switch Port

Switch Fabric

Switch Scheduler
Physical Separation
Aligning Cells
Fourth Generation Routers/Switches

Queueing Structure

- Virtual Output Queues
- 1 write per “cell” time
- 1 read per “cell” time
- Rate of writes/reads determined by switch fabric speedup

- Switch Fabric
- Switch Arbitration
- Switch Core (Bufferless)

- Lookup & Drop Policy
- Output Scheduling

Typically <5Tb/s aggregate capacity
Basic Ideas: Part II
Forwarding Functions: ATM Switch

- Lookup cell VCI/VPI in VC table.
- Replace old VCI/VPI with new.
- Forward cell to outgoing interface.
- Transmit cell onto link.
Functions: Ethernet (L2) Switch

- Lookup frame destination address (DA) in forwarding table.
  - If known, forward to correct port.
  - If unknown, broadcast to all ports.
- Learn source address (SA) of incoming frame.
- Forward frame to outgoing interface.
- Transmit frame onto link.
Functions: IP Router

- Lookup packet DA in forwarding table.
  - If known, forward to correct port.
  - If unknown, drop packet.
- Decrement TTL, update header Cksum.
- Forward packet to outgoing interface.
- Transmit packet onto link.
Basic Architectural Components

- Policing
- Congestion Control
- Routing
- Admission Control
- Reservation

Switching

Datapath: per-packet processing

Control

Rensselaer Polytechnic Institute

Shivkumar Kalyanaraman
Basic Architectural Components

1. Forwarding Table
   - Forwarding Decision

2. Interconnect

3. Output Scheduling
   - Forwarding Table
   - Forwarding Decision

(Rensselaer Polytechnic Institute)
**Generic Router Architecture**

![Diagram of Generic Router Architecture]

- **Header Processing**
  - Lookup IP Address
  - Update Header
  - Queue Packet

- **Address Table**
  - ~1M prefixes
  - Off-chip DRAM

- **Buffer Memory**
  - ~1M packets
  - Off-chip DRAM

**Key Components**
- Data
- Hdr
- IP Address
  - Next Hop

**Rensselaer Polytechnic Institute**

Shivkumar Kalyanaraman
Generic Router Architecture
Router linecard

OC192c linecard

- 30M gates
- 2.5Gbits of memory
- 2-300W
- 1m²
- $25k cost, $100k price.

40-55% of power in chip-to-chip serial links
Simplest Design: Software Router using PCs!

- Idea: add special-purpose software to general-purpose hardware: Cheap, but slow
- **Measure of speed**: aggregate data rate or aggregate packet rate
- Limits number & type of interfaces, topologies etc
- Eg: 400 Mbps aggregate rate will allow four 100 Mbps ethernet interfaces, but no GbE!
- Eg: MIT’s Click Router
Older Example

- First generation router built with 133 MHz Pentium
  - Mean packet size 500 bytes
  - Interrupt takes 10 microseconds, word access take 50 ns
  - Per-packet processing time takes 200 instructions = 1.504 µs
- Copy loop
  ```
  register <- memory[read_ptr]
  memory [write_ptr] <- register
  read_ptr <- read_ptr + 4
  write_ptr <- write_ptr + 4
  counter <- counter -1
  if (counter not 0) branch to top of loop
  ```
  - 4 instructions + 2 memory accesses = 130.08 ns
- Copying packet takes 500/4 * 130.08 = 16.26 µs; interrupt 10 µs
- Total time = 27.764 µs => speed is **144.1 Mbps**
- Amortized interrupt cost balanced by routing protocol cost
Required: Aggregate Packet vs Bit Rates

- 64 byte pkts
- 1518 byte pkts

Rensselaer Polytechnic Institute

Shivkumar Kalyanaraman
Per-Packet Processing Time Budget

<table>
<thead>
<tr>
<th>Technology</th>
<th>Time per packet for small packets (in μs)</th>
<th>Time per packet for large packets (in μs)</th>
</tr>
</thead>
<tbody>
<tr>
<td>10Base-T</td>
<td>51.20</td>
<td>1,214.40</td>
</tr>
<tr>
<td>100Base-T</td>
<td>5.12</td>
<td>121.44</td>
</tr>
<tr>
<td>OC-3</td>
<td>3.29</td>
<td>78.09</td>
</tr>
<tr>
<td>OC-12</td>
<td>0.82</td>
<td>19.52</td>
</tr>
<tr>
<td>1000Base-T</td>
<td>0.51</td>
<td>12.14</td>
</tr>
<tr>
<td>OC-48</td>
<td>0.21</td>
<td>4.88</td>
</tr>
<tr>
<td>OC-192</td>
<td>0.05</td>
<td>1.22</td>
</tr>
<tr>
<td>OC-768</td>
<td>0.01</td>
<td>0.31</td>
</tr>
</tbody>
</table>

MIT’s Click Router claims 435Kpps with 64 byte packets!


(=> it can do 100 Mbps, but not GbE interfaces!)
Forwarding
a.k.a. Port Mapping
Basic Architectural Components:

1. Forwarding Table
2. Forwarding Decision
3. Output Scheduling

Interconnect
ATM and MPLS Switches

Direct Lookup

VCI

Address

Memory

Data

(Port, VCI)
Bridges and Ethernet Switches

**Associative Lookups**

**Advantages:**
- Simple

**Disadvantages**
- Slow
- High Power
- Small
- Expensive
Bridges and Ethernet Switches

Hashing

Search Data

Hashing Function

16

Address

Memory

Data

Hit?

Associated Data

Address

log_2(N)

Shivkumar Kalyanaraman
Lookups Using Hashing

An example

Hashing Function

CRC-16

Memory

16

Hit?

Address

log₂N

Associated Data

Linked lists

Search Data

48

Hit?
Lookups Using Hashing

Performance of simple example

Where:

\[ ER = \frac{1}{2} \left( 1 + \frac{\alpha}{1 - \left( 1 - \frac{1}{N} \right)^M} \right) \]

- \( ER \) = Expected number of memory references
- \( M \) = Number of memory addresses in table
- \( N \) = Number of linked lists
- \( \alpha = \frac{M}{N} \)
Lookups Using Hashing

**Advantages:**
- Simple
- Expected lookup time can be small

**Disadvantages**
- Non-deterministic lookup time
- Inefficient use of memory
Per-packet processing in an IP Router

1. Accept packet arriving on an incoming link.
2. **Lookup** packet destination address in the forwarding table, to identify outgoing port(s).
3. Manipulate packet header: e.g., decrement TTL, update header checksum.
4. **Send (switch)** packet to the outgoing port(s).
5. **Classify and buffer** packet in the queue.
6. Transmit packet onto outgoing link.
Caching Addresses

- CPU
- Buffer Memory
- DMA
- MAC
- Local Buffer Memory
- Line Card

Slow Path

Fast Path
Caching Addresses

**LAN:**
Average flow < 40 packets

**WAN:**
Huge Number of flows

![Graph showing cache hit rate with different cache sizes]
IP Router: Lookup

IPv4 unicast destination address based lookup

<table>
<thead>
<tr>
<th>Destination</th>
<th>Next Hop</th>
</tr>
</thead>
<tbody>
<tr>
<td>----</td>
<td>----</td>
</tr>
<tr>
<td>----</td>
<td>----</td>
</tr>
<tr>
<td>----</td>
<td>----</td>
</tr>
<tr>
<td>----</td>
<td>----</td>
</tr>
</tbody>
</table>
Lookup and Forwarding Engine

Packet

payload header

Router

Routing Lookup Data Structure

Forwarding Table

<table>
<thead>
<tr>
<th>Dest-network</th>
<th>Port</th>
</tr>
</thead>
<tbody>
<tr>
<td>65.0.0.0/8</td>
<td>3</td>
</tr>
<tr>
<td>128.9.0.0/16</td>
<td>1</td>
</tr>
<tr>
<td>149.12.0.0/19</td>
<td>7</td>
</tr>
</tbody>
</table>

Outgoing Port

Destination Address
### Example Forwarding Table

<table>
<thead>
<tr>
<th>Destination IP Prefix</th>
<th>Outgoing Port</th>
</tr>
</thead>
<tbody>
<tr>
<td>65.0.0.0/8</td>
<td>3</td>
</tr>
<tr>
<td>128.9.0.0/16</td>
<td>1</td>
</tr>
<tr>
<td>142.12.0.0/19</td>
<td>7</td>
</tr>
</tbody>
</table>

**IP prefix: 0-32 bits**

![Diagram showing IP prefix with prefix lengths and 32-bit boundary](image)
IP Routers

Class-based addresses

IP Address Space

Class A | Class B | Class C | D

Routing Table:

212.17.9.0 | Port 4

Exact Match

212.17.9.4

Class A

Class B

Class C

Shivkumar Kalyanaraman
IP Routers
CIDR => “Classless”

Class-based:

Classless:

128.9.0.0
142.12/19
128.9.16.14
IP Routers

CIDR

Most specific route = “longest matching prefix”
CIDR => Prefixes can Overlap

Routing lookup: Find the longest matching prefix (aka the most specific route) among all prefixes that match the destination address.
Difficulty of Longest Prefix Match

2-dimensional search:
- Prefix Length
- Prefix Value

Prefix Length

Prefix Values

128.9.16.14

128.9.16.0/21  128.9.172.0/21
128.9.176.0/24
65.0.0.0/8  128.9.0.0/16  142.12.0.0/19

Rensselaer Polytechnic Institute

Shivkumar Kalyanaraman
### IP Routers

#### Metrics for Lookups

<table>
<thead>
<tr>
<th>Prefix</th>
<th>Port</th>
</tr>
</thead>
<tbody>
<tr>
<td>65/8</td>
<td>3</td>
</tr>
<tr>
<td>128.9/16</td>
<td>5</td>
</tr>
<tr>
<td>128.9.16/20</td>
<td>2</td>
</tr>
<tr>
<td>128.9.19/24</td>
<td>7</td>
</tr>
<tr>
<td>128.9.25/24</td>
<td>10</td>
</tr>
<tr>
<td>128.9.176/20</td>
<td>1</td>
</tr>
<tr>
<td>142.12/19</td>
<td>3</td>
</tr>
</tbody>
</table>

- Lookup time
- Storage space
- Update time
- Preprocessing time
## Lookup Rates Required

<table>
<thead>
<tr>
<th>Year</th>
<th>Line</th>
<th>Line-rate (Gbps)</th>
<th>40B packets (Mpps)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1998-99</td>
<td>OC12c</td>
<td>0.622</td>
<td>1.94</td>
</tr>
<tr>
<td>1999-00</td>
<td>OC48c</td>
<td>2.5</td>
<td>7.81</td>
</tr>
<tr>
<td>2000-01</td>
<td>OC192c</td>
<td>10.0</td>
<td>31.25</td>
</tr>
<tr>
<td>2002-03</td>
<td>OC768c</td>
<td>40.0</td>
<td>125</td>
</tr>
</tbody>
</table>

31.25 Mpps ⇒ 33 ns

DRAM: 50-80 ns, SRAM: 5-10 ns
**Update Rates Required**

- Recent BGP studies show that updates can be:
  - **Bursty:** several 100s of routes updated/withdrawn => insert/delete operations
  - **Frequent:** Average 100+ updates per second

- Need data structure to be efficient in terms of lookup as well as update (insert/delete) operations.
Renewed growth due to multi-homing of enterprise networks!

Source: http://www.telstra.net/ops/bgptable.html
Potential Hyper-Exponential Growth!

Global routing table vs Moore's law since 1999

- Global prefixes
- Moore's law
- Double growth

Prefixes

01/99 04/99 07/99 10/99 01/00 04/00 07/00 10/00 01/01 04/01
Trees and Tries

Binary Search Tree

Binary Search Trie

$N$ entries

$\log_2 N$
Issues with Trees (incl. Patricia trees)

- Tree: each path in the tree from root to leaf corresponds to an entry in the forwarding table.
  - Longest prefix match is the longest path in the tree that matches the destination address of an incoming packet.
  - Algo: start at the root of the tree; recursively matches the children of the current node with the next few bits of the destination address, stop if no match is found.
- Worst case: time proportional to the length of the destination address to find the longest prefix match.
  - => frugal use of memory at the expense of doing more memory lookups => precisely the wrong design!
- Patricia-tree algorithm may need to backtrack to find the longest match, leading to very poor worst-case performance.
Trie: other examples

Setup trie based upon dotted decimal notation instead of bit-representation
Binary search on trie levels

Store a hash table for each prefix length to aid search at a particular trie level.

<table>
<thead>
<tr>
<th>Length</th>
<th>Hash</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td></td>
</tr>
<tr>
<td>24</td>
<td></td>
</tr>
</tbody>
</table>

Example Prefixes
- 10.0.0.0/8
- 10.1.0.0/16
- 10.1.1.0/24
- 10.1.2.0/24
- 10.2.3.0/24

Example Addrs
- 10.1.1.4
- 10.4.4.3
- 10.2.3.9
- 10.2.4.8
Binary search on trie levels

**Disadvantages**
- Multiple hashed memory accesses.
- Updates are complex.

**Advantages**
- Scaleable to IPv6.

33K entries: 1.4MB data structure with 1.2-2.2 Mpps $[O(\log W)]$
Trees and Tries

*Multiway tries*

16-ary Search Trie

```
0000, ptr 1111, ptr
0000, 0 1111, ptr
000011110000 111111111111
```
# Lookup: Multiway Tries

## Tradeoffs

<table>
<thead>
<tr>
<th>Degree of Tree</th>
<th># Mem References</th>
<th># Nodes ((\times 10^6))</th>
<th>Total Memory (Mbytes)</th>
<th>Fraction Wasted (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>48</td>
<td>1.09</td>
<td>4.3</td>
<td>49</td>
</tr>
<tr>
<td>4</td>
<td>24</td>
<td>0.53</td>
<td>4.3</td>
<td>73</td>
</tr>
<tr>
<td>8</td>
<td>16</td>
<td>0.35</td>
<td>5.6</td>
<td>86</td>
</tr>
<tr>
<td>16</td>
<td>12</td>
<td>0.25</td>
<td>8.3</td>
<td>93</td>
</tr>
<tr>
<td>64</td>
<td>8</td>
<td>0.17</td>
<td>21</td>
<td>98</td>
</tr>
<tr>
<td>256</td>
<td>6</td>
<td>0.12</td>
<td>64</td>
<td>99.5</td>
</tr>
</tbody>
</table>

Table produced from \(2^{15}\) randomly generated 48-bit addresses
Routing Lookups in Hardware

Most prefixes are 24-bits or shorter
Routing Lookups in Hardware

Prefixes up to 24-bits

$2^{24} = 16M$ entries

142.19.6.14

142.19.6

14

1

Next Hop
Routing Lookups in Hardware

Prefixes up to 24-bits

128.3.72.44

Prefixes above 24-bits
Routing Lookups in Hardware

Disadvantages
- Large memory required (9-33MB)
- Depends on prefix-length distribution.

Advantages
- 20Mpps with 50ns DRAM
- Easy to implement in hardware

Various compression schemes can be employed to decrease the storage requirements: e.g. employ carefully chosen variable length strides, bitmap compression etc.
Switching
a.k.a. Interconnect

Rensselaer Polytechnic Institute

Shivkumar Kalyanaraman
Basic Architectural Components: *Interconnect*

1. Forwarding Table
2. Interconnect
3. Output Scheduling

- 1. Distributed output scheduling
- 2. Interconnect
- 3. Output scheduling
Switching: what does a switch do?

- Transfers data from an input to an output
  - many ports (density), high speeds
- Eg: Crossbar
Switching goals (telephony & data)

- Capacity of switch is the maximum rate at which it can move information, assuming all data paths are simultaneously active

- Primary goal: **maximize capacity**
  - subject to cost and reliability constraints

- Circuit switch must reject call if can’t find a path for samples from input to output
  - goal: **minimize call blocking**

- Packet switch must reject a packet if it can’t find a buffer to store it awaiting access to output trunk
  - goal: **minimize packet loss**

- Don’t reorder packets
Circuit Switches
Circuit switch

- A switch that can handle $N$ calls has $N$ logical inputs and $N$ logical outputs
  - $N$ up to 200,000
- Moves 8-bit samples from an input to an output port
  - Recall that samples have no headers
  - Destination of sample depends on time at which it arrives at the switch
- In practice, input trunks are multiplexed
  - Multiplexed trunks carry frames = set of samples
- Goal: extract samples from frame, and depending on position in frame, switch to output
  - each incoming sample has to get to the right output line and the right slot in the output frame
Call blocking

- Can’t find a path from input to output
- **Internal** blocking
  - slot in output frame exists, but *no path*
- **Output** blocking
  - *no slot* in output frame is available
- Output blocking is reduced in *transit* switches
  - need to put a sample in one of *several* slots going to the desired next hop
Multiplexors and demultiplexors

- Most trunks time division multiplex voice samples
- At a central office, trunk is demultiplexed and distributed to active circuits
- Addressing not required...
- Synchronous multiplexor: N input lines
  - Output runs N times as fast as input
Circuit Switch

- A switch that can handle $N$ calls has $N$ logical inputs and $N$ logical outputs
  - $N$ up to 200,000
- In practice, input trunks are multiplexed
  - example: DS3 trunk carries 672 simultaneous calls
- Multiplexed trunks carry *frames* = set of samples
- Goal: extract samples from frame, and depending on position in frame, switch to output
  - each incoming sample has to get to the right output line and the right slot in the output frame
  - demultiplex, switch, multiplex
Issue: Call Blocking

- Can’t find a path from input to output
- Internal blocking
  - slot in output frame exists, but no path
- Output blocking
  - no slot in output frame is available
- Output blocking is reduced in transit switches
  - need to put a sample in one of several slots going to the desired next hop
Time division switching

- Key idea: when de-multiplexing, position in frame determines output trunk
- Time division switching interchanges sample position within a frame: time slot interchange (TSI)
Scaling Issues with TSI

- Limit is time taken to read and write to memory
- For 120,000 circuits
  - need to read and write memory once every 125 microseconds
  - each operation takes around 0.5 ns => impossible with current technology
- Need to look to other techniques
Space division switching

- Each sample takes a different path through the switch, depending on its destination.
Switch Fabric Concept

- Any input port can send to any output port

Data path (aka backplane) that provides parallelism
Connects the NICs which have on-board processing

Shivkumar Kalyanaraman
Desired Switch Fabric Properties

- Used inside a single network system
- Interconnection among I/O ports (and possibly CPU)
- Can transfer unicast, multicast, and broadcast packets
- Scales to arbitrary data rate on any port
- Scales to arbitrary packet rate on any port
- Scales to arbitrary number of ports
- Has low overhead
- Has low cost
Space Division Fabric

- Can use multiple paths simultaneously

Asynchronous design arose from multi-processor context
Data can be sent across fabric at arbitrary times
Crossbar

- Simplest possible space-division switch
  - Like 2N buses in parallel
- Crosspoints can be turned on or off, long enough to transfer a packet from an input to an output
  - Arbitration needed to deal with port contention at outputs…
- For fixed-size packets and known arrival pattern, can precompute schedule
- Internally nonblocking
  - but need $N^2$ crosspoints
  - time to set each crosspoint grows quadratically
- Vulnerable to single faults
Multistage crossbar

- In a crossbar during each switching time only one cross-point per row or column is active
- Can **save crosspoints** if a cross-point can **attach to more than one input line**
  - I.e. inputs share crosspoints (expensive resource)
- This is done in a multistage crossbar
- Need to **rearrange connections every switching time**
Even if “internally” non-blocking (i.e. fully inter-connected), port contention can occur!

Need blocking circuits at input and output ports

Arbitration needed to deal with port contention at outputs
Multistage crossbar

- Can suffer internal blocking
  - unless sufficient number of second-level stages
- Number of crosspoints < $N^2$
- Finding a path from input to output requires a depth-first-search
- Scales better than crossbar, *but still not too well*
  - 120,000 call switch needs $\sim 250$ million crosspoints
Time-Space Switching

- Precede each input trunk in a crossbar with a TSI
- Delay samples so that they arrive at the right time for the space division switch’s schedule
Time-Space-Time (TST) switching

- Allowed to flip samples both on input and output trunk
- Gives more flexibility => lowers call blocking probability

Telephone switches like 5ESS use multiple space-stages: eg TSSST etc

Shivkumar Kalyanaraman
Packet Switches
Packet switches

- In a circuit switch, path of a sample is determined at time of connection establishment.
- No need for a sample header—position in frame used.
- In a packet switch, packets carry a destination field or label.
  - Need to look up destination port on-the-fly.
- Datagram switches
  - Lookup based on entire destination address (longest-prefix match).
- Cell or Label-switches
  - Lookup based on VCI or Labels.
- L2 Switches, L3 Switches, L4-L7 switches
  - Key difference is in lookup function (i.e., filtering), not in switching (i.e., not in forwarding).
Shared Memory Switches

- Dual-ported RAM
- Incoming cells converted from serial to parallel
- Elegant, but memory speeds & port counts don’t scale
  - Output buffering
  - 100% throughput under heavy load
  - Minimize buffers
- Eg: CNET Prelude, Hitachi shared buffer s/w, AT&T GCNS-2000
Shared memory fabrics: more…

- Memory interface hardware expensive => many “ports” share fewer memory interfaces
  - Eg: dual-ported memory
- Separate low-speed bus lines for controller
Shared Medium Switches

- Share medium (i.e. bus/ring etc) instead of memory
- Medium has to be N times as fast
  - Address filters & output buffers at the medium speed also!
- TDM + round robin
- Egs: IBM PARIS & plaNET s/w, Fore Forerunner ASX-100, NEC ATOM

AF = address filter
S/P = serial to parallel
P/S = parallel to serial
Fully Interconnected Switches

- Full interconnections
- Broadcast + address-filters
  - Multicasting is natural
- Output queuing
- All hardware same speed => scalable
- Quadratic growth of buffers/filters
- Knockout switch (AT&T) reduced
  - # of buffers: fixed $L (=8)$ buffers per output + a tournament method to eliminate packets
  - Small residual packet loss rate (1/million)
- Egs: Fujitsu bus matrix, GTE SPANet
Crossbar: “Switched” interconnections

- 2N media (i.e. buses), BUT...
  - Use “switches” between each input and output bus instead of broadcasting
- Total number of “paths” required = N+M
- Number of switching points = NxM
- Arbitration/scheduling needed to deal with port contention
Multi-Stage Fabrics

- Compromise between pure time-division and pure space division
- Attempt to combine advantages of each
  - Lower cost from time-division
  - Higher performance from space-division
- Technique: *Limited Sharing*
- Eg: Banyan switch

Features
- Scalable
- Self-routing, i.e. no central controller
- Packet queues allowed, but not required

Note: multi-stage switches share the “crosspoints” which have now become “expensive” resources.
Multi-stage switches: fewer crosspoints

- NxN switch with bxb elements has $\left\lfloor \log_b N \right\rfloor$ elements with $\left\lfloor N/b \right\rfloor$ elements per stage
- Fabric is self routing
- Recursive
- Can be synchronous or asynchronous
- Regular and suitable for VLSI implementation

- Issue: output & internal blocking...
Banyan Switch Fabric (Contd)

- Basic building block = 2x2 switch, labelled by 0/1
- Can be synchronous or asynchronous
  - Asynchronous => packets can arrive at arbitrary times
  - Synchronous banyan offers TWICE the effective throughput!
- Worst case when all inputs receive packets with same label
Switch fabric element

- Goal: “self-routing” fabrics
- Build complicated fabrics from a simple elements

- **Routing rule:** if 0, send packet to upper output, else to lower output
- If both packets to same output, buffer or drop
Multi-stage Interconnects (MINs): Banyan

- Key: reduce the number of crosspoints in a crossbar
- 8x8 banyan: **Recursive design**
  - Use the first bit to route the cell through the first stage, either to the upper or lower 4x4 network,
  - Last 2 bits to route the cell through the 4x4 network to the appropriate output port.
- **Self-routing**: output address completely specifies the route through the network (aka digit-controlled routing)
- Simple elements, scalable, parallel routing, elements at same speed
- Eg: Bellcore Sunshine, Alcatel DN 1100
Banyan Fabric: another view...
Banyan

- Simplest self-routing recursive fabric

- Two packets want to go to the same output => output blocking
- Banyan: packets may block even if they want to go to different outputs => internal blocking!
  - Unlike crossbar: because it has fewer crosspoints
  - However, feasible non-blocking schedules exist => pre-sort & shuffle packets to get to such non-blocking schedules
Non-Blocking Batcher-Banyan

Batcher Sorter

Self-Routing Network

• Fabric can be used as scheduler.
• Batcher-Banyan network is blocking for multicast.
Blocking in Banyan S/ws: Sorting

- Can avoid blocking by choosing order in which packets appear at input ports
- If we can
  - present packets at inputs sorted by output
  - “trap” duplicates (i.e. going to same o/p port)
  - remove gaps
  - precede banyan with a perfect shuffle stage
  - then no internal blocking
- For example: [X, 010, 010, X, 011, X, X, X]:
  - Sort => [010, 011, 011, X, X, X, X, X]
  - Trap duplicates => [010, 011, X, X, X, X, X, X]
  - Shuffle => [010, X, 011, X, X, X, X, X]
- Need sort, shuffle, and trap networks
Sorting using Merging

- Build sorters from merge networks
- Assume we can merge two sorted lists
- Sort pairwise, merge, recurse
Putting together: Batcher-Banyan

- What about trapped duplicates?
  - recirculate to beginning
  - or run output of trap to multiple banyans (dilation)
Scaling Banyan Networks: Challenges

1. Batcher-banyan networks of significant size are physically limited by the possible circuit density and number of input/output pins of the integrated circuit. To interconnect several boards, interconnection complexity and power dissipation place a constraint on the number of boards that can be interconnected.

2. The entire set of $N$ cells must be synchronized at every stage.

3. Large sizes increase the difficulty of reliability and repairability.

4. All modifications to maximize the throughput of space-division networks increase the implementation complexity.
Other Non-Blocking Fabrics

Clos Network
Other Non-Blocking Fabrics

Clos Network

Expansion factor required = 2 - 1/N  (but still blocking for multicast)
Blocking and Buffering
Blocking in packet switches

- Can have both internal and output blocking
  - Internal
    - no path to output
  - Output
    - trunk unavailable
- Unlike a circuit switch, *cannot predict if packets will block* (why?)
- If packet is blocked => must *either buffer or drop*
Dealing with blocking in packet switches

- Over-provisioning
  - internal links much faster than inputs
- Buffers
  - at input or output
- Backpressure
  - if switch fabric doesn’t have buffers, prevent packet from entering until path is available
- Parallel switch fabrics
  - increases effective switching capacity
Blocking in Banyan Fabric

- Can avoid with a buffered banyan switch
  - but this is too expensive
  - hard to achieve zero loss even with buffers
- Instead, can check if path is available before sending packet
  - three-phase scheme
  - send requests
  - inform winners
  - send packets
- Or, use several banyan fabrics in parallel
  - intentionally misroute and tag one of a colliding pair
  - divert tagged packets to a second banyan, and so on to k stages
  - expensive
  - can reorder packets
  - output buffers have to run k times faster than input
Buffering: where?

- Input
- Output
- Internal
- Re-circulating
Queuing: input, output buffers

![Diagram of queuing system with input and output buffers]
Switch Fabrics: Buffered crossbar

- What happens if packets at two inputs both want to go to the same output?
- Can defer one at an input buffer
- Or, buffer cross-points: complex arbiter
Queuing:  
*Two basic practical techniques*

**Input Queueing**
- Usually a non-blocking switch fabric (e.g. crossbar)

**Output Queueing**
- Usually a fast bus
Queuing: Output Queueing

**Individual Output Queues**

- Memory b/w = (N+1).R

**Centralized Shared Memory**

- Memory b/w = 2N.R
Output Queuing

- Don’t suffer from head-of-line blocking
- But output buffers need to run much faster than trunk speed (why?)
- Can reduce some of the cost by using the knockout principle
  - unlikely that all N inputs will have packets for the same output
  - drop extra packets, fairly distributing losses among inputs
Input Queuing

- No speedup in buffers or trunks (unlike output queued switch)
- Needs arbiter
- Problem: *head of line blocking*
  - with randomly distributed packets, utilization at most 58.6%
  - worse with *hot spots*
Input Queueing
Head of Line Blocking

Delay

Load

58.6% 100%

Shivkumar Kalyanaraman
Solution: Input Queueing w/ Virtual output queues (VOQ)
Head-of-Line (HOL) in Input Queuing

- Per-output queues at inputs
- Arbiter must choose one of the input ports for each output port
- How to select?
- Parallel Iterated Matching
  - inputs tell arbiter which outputs they are interested in
  - output selects one of the inputs
  - some inputs may get more than one grant, others may get none
  - if >1 grant, input picks one at random, and tells output
  - losing inputs and outputs try again
- Used in DEC Autonet 2 switch
Input Queues

Virtual Output Queues

Delay

Load

100%

Shivkumar Kalyanaraman
Input Queueing

Memory b/w = 2R

Scheduler

Can be quite complex!
Input Queueing

Scheduling

Input 1

Input m

A1(t)

Am(t)

Q(1,1)

Q(1,n)

Q(m,1)

Q(m,n)

Matching, M

Output 1

Output n

D1(t)

Dn(t)
Input Queueing

Scheduling: Example

Request Graph

Bipartite Matching
(Weight = 18)
Input Queueing
Longest Queue First or Oldest Cell First

Weight $= \{\text{Queue Length, Waiting Time}\}$

100%

Maximum weight

Rensselaer Polytechnic Institute

Shivkumar Kalyanaraman
**Input Queueing**

*Scheduling*

- **Maximum Size**
  - Maximizes instantaneous throughput
  - Does it maximize long-term throughput?
- **Maximum Weight**
  - Can clear most backlogged queues
  - But does it sacrifice long-term throughput?
Input Queuing

Why is serving long/old queues better than serving maximum number of queues?

- When traffic is uniformly distributed, servicing the maximum number of queues leads to 100% throughput.
- When traffic is non-uniform, some queues become longer than others.
- A good algorithm keeps the queue lengths matched, and services a large number of queues.
Input Queueing

*Practical Algorithms*

- Maximal Size Algorithms
  - Wave Front Arbiter (WFA)
  - Parallel Iterative Matching (PIM)
  - iSLIP
- Maximal Weight Algorithms
  - Fair Access Round Robin (FARR)
  - Longest Port First (LPF)
Requests

#1
1 → 1
2 → 2
3 → 3
4 → 4

#2
1 → 2
2 → 3
3 → 4

Accept/Match

#1
1 → 1
2 → 2
3 → 3
4 → 4

#2
1 → 2
2 → 3
3 → 4

Grant

Round-Robin Selection

#1
1 → 1
2 → 2
3 → 3
4 → 4

#2
1 → 2
2 → 3
3 → 4

Round-Robin Selection
iSLIP

Properties

- Random under low load
- TDM under high load
- Lowest priority to MRU
- 1 iteration: fair to outputs
- Converges in at most $N$ iterations. On average $\leq \log_2 N$
- Implementation: $N$ priority encoders
- Up to 100% throughput for uniform traffic
iSLIP
Programmable Priority Encoder

\[ \text{State} \rightarrow \text{Grant \ (1)} \rightarrow \text{Accept \ (1)} \rightarrow \text{Decision} \]

\[ \text{State} \rightarrow \text{Grant \ (2)} \rightarrow \text{Accept \ (2)} \rightarrow \text{Decision} \]

\[ \text{State} \rightarrow \text{Grant} \rightarrow \text{Accept} \rightarrow \text{Decision} \]

\[ \log_2 N \]
Throughput results

**Theory:**

- Input Queueing (IQ)
- IQ + VOQ, Maximum weight matching
- IQ + VOQ, Sub-maximal size matching e.g. PIM, iSLIP.

- 58% [Karol, 1987]
- 100% [M et al., 1995]
- 100% [Various]
- 100% [Tassiulas, 1998]

**Practice:**

- Input Queueing (IQ)
- IQ + VOQ, Maximal size matching, Speedup of two.
- Various heuristics, distributed algorithms, and amounts of speedup

100% [Dai & Prabhakar, 2000]
Speedup: Context

A generic switch

The placement of memory gives

- Output-queued switches
- Input-queued switches
- Combined input- and output-queued switches
Output-queued switches

Best delay and throughput performance
- Possible to erect “bandwidth firewalls” between sessions

Main problem
- Requires high fabric speedup \((S = N)\)

Unsuitable for high-speed switching
Input-queued switches

Big advantage
- Speedup of one is sufficient

Main problem
- Can’t guarantee delay due to input contention

Overcoming input contention: use higher speedup
The Speedup Problem

Find a compromise: \(1 < \text{Speedup} \ll N\)
- to get the performance of an OQ switch
- close to the cost of an IQ switch

Essential for high speed QoS switching
Intuition

Bernoulli IID inputs
Fabric throughput = 0.58

Bernoulli IID inputs
Fabric throughput = 1.16
I/p efficiency, $\rho = 1/1.16$
Ave I/p queue = 6.25
Intuition (continued)

Bernoulli IID inputs
Fabric throughput = 1.74
Input efficiency = \frac{1}{1.74}
Ave I/p queue = 1.35

Bernoulli IID inputs
Fabric throughput = 2.32
Input efficiency = \frac{1}{2.32}
Ave I/p queue = 0.75
QoS Support: Review
Recall: What is QoS?

- “Better performance” as described by a set of parameters or measured by a set of metrics.

- **Generic parameters:**
  - Bandwidth
  - Delay, Delay-jitter
  - Packet loss rate (or loss probability)

- **Transport/Application-specific parameters:**
  - Timeouts
  - Percentage of “important” packets lost
What is QoS (contd)?

- These parameters can be measured at several granularities:
  - “micro” flow, aggregate flow, population.

- QoS considered “better” if:
  - a) more parameters can be specified
  - b) QoS can be specified at a fine-granularity.

- QoS vs CoS: CoS maps micro-flows to classes and may perform optional resource reservation per-class

- QoS spectrum:
  - Best Effort
  - Leased Line
Example QoS

- Bandwidth: $r$ Mbps in a time $T$, with burstiness $b$
- Delay: worst-case
- Loss: worst-case or statistical

![Diagram showing QoS parameters and relationships]

$r$ tokens per second

$= R$ bps

regulator

$= b$ tokens

$= \frac{b^*R}{(R-r)}$

slope $r$
In a FIFO service discipline, the performance assigned to one flow is *convoluted* with the arrivals of packets from all other flows!

- Cant get QoS with a “free-for-all”
- Need to use new scheduling disciplines which provide “*isolation*” of performance from arrival rates of background traffic
Fundamental Problems

- **Conservation Law** (Kleinrock): \( \Sigma \rho(i) W_q(i) = K \)

- Irrespective of scheduling discipline chosen:
  - Average **backlog (delay)** is constant
  - Average **bandwidth** is constant

- Zero-sum game => need to “set-aside” resources for premium services
QoS Big Picture: Control/Data Planes

**Control Plane:** Signaling + Admission Control or SLA (Contracting) + Provisioning/Traffic Engineering

**Data Plane:** Traffic conditioning (shaping, policing, marking etc) + Traffic Classification + Scheduling, Buffer management
Eg: Integrated Services (IntServ)

- An architecture for providing QOS guarantees in IP networks for individual application sessions
- Relies on resource reservation, and routers need to maintain state information of allocated resources (eg: g) and respond to new Call setup requests
Call Admission: routers will admit calls based on their R-spec and T-spec and base on the current resource allocated at the routers to other calls.

1. Request: specify
   - traffic (Tspec)
   - guarantee (Rspec)

2. Element considers
   - unreserved resources
   - required resources

3. Reply: whether or not request can be satisfied
Token Bucket

- Characterized by three parameters (b, r, R)
  - b – token depth
  - r – average arrival rate
  - R – maximum arrival rate (e.g., R link capacity)
- A bit is transmitted only when there is an available token
- When a bit is transmitted exactly one token is consumed

![Diagram](image)

- r tokens per second
- b tokens
- \( \frac{b \times R}{R - r} \) bits
- slope R
- slope r
- \( \leq R \) bps
- regulator
- time
Per-hop Reservation

- Given $b, r, R$ and per-hop delay $d$
- Allocate bandwidth $r_a$ and buffer space $B_a$ such that to guarantee $d$
Use a few bits in header to indicate which queue class a packet goes into (also branded as CoS)

- High $$ users classified into high priority queues, which also may be less populated
  => lower delay and low likelihood of packet drop

- Ideas: priority, round-robin, classification, aggregation, ...

Traffic Sources

$\cdots$ $\cdots$

Class A

Class B

Class C

Traffic Classes
Mechanisms: Buffer Mgmt/Priority Drop

- Drop RED and BLUE packets
  - Ideas: packet marking, queue thresholds, differential dropping, buffer assignments
  - Drop only BLUE packets
Classification
Why Classification?
Providing Value-Added Services

Some examples

- Differentiated services
  - Regard traffic from Autonomous System #33 as `platinum-grade`
- Access Control Lists
  - Deny udp host 194.72.72.33 194.72.6.64 0.0.0.15 eq snmp
- Committed Access Rate
  - Rate limit WWW traffic from sub-interface#739 to 10Mbps
- Policy-based Routing
  - Route all voice traffic through the ATM network
Packet Classification

Forwarding Engine

Packet Classification

Classifier (Policy Database)

<table>
<thead>
<tr>
<th>Predicate</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>-----</td>
<td>-----</td>
</tr>
<tr>
<td>-----</td>
<td>-----</td>
</tr>
<tr>
<td>-----</td>
<td></td>
</tr>
<tr>
<td>-----</td>
<td>-----</td>
</tr>
</tbody>
</table>
Multi-field Packet Classification

Given a classifier with $N$ rules, find the action associated with the highest priority rule matching an incoming packet.

<table>
<thead>
<tr>
<th>Field 1</th>
<th>Field 2</th>
<th>...</th>
<th>Field k</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Rule 1</strong></td>
<td>152.163.190.69/21</td>
<td>152.163.80.11/32</td>
<td>...</td>
<td>UDP</td>
</tr>
<tr>
<td><strong>Rule 2</strong></td>
<td>152.168.3.0/24</td>
<td>152.163.0.0/16</td>
<td>...</td>
<td>TCP</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td><strong>Rule N</strong></td>
<td>152.168.0.0/16</td>
<td>152.0.0.0/8</td>
<td>...</td>
<td>ANY</td>
</tr>
</tbody>
</table>

Shivkumar Kalyanaraman
Prefix matching: 1-d range problem

Most specific route = “longest matching prefix”
Classification: 2D Geometry problem

Field #1 | Field #2 | Data

<table>
<thead>
<tr>
<th>R7</th>
<th>R6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Field #2</td>
<td>e.g. (128.16.46.23, *)</td>
</tr>
<tr>
<td>Field #1</td>
<td>e.g. (144.24/16, 64/24)</td>
</tr>
</tbody>
</table>

Shivkumar Kalyanaraman
Rensselaer Polytechnic Institute
Packet Classification

References

# Proposed Schemes

<table>
<thead>
<tr>
<th>Pros</th>
<th>Cons</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Sequential Evaluation</strong>&lt;br&gt;Small storage, scales well with number of fields</td>
<td>Slow classification rates</td>
</tr>
<tr>
<td><strong>Ternary CAMs</strong>&lt;br&gt;Single cycle classification</td>
<td>Cost, density, power consumption</td>
</tr>
<tr>
<td><strong>Grid of Tries (Srinivasan et al[Sigcomm 98])</strong>&lt;br&gt;Small storage requirements and fast lookup rates for two fields. Suitable for big classifiers</td>
<td>Not easily extendible to more than two fields.</td>
</tr>
</tbody>
</table>
## Proposed Schemes (Contd.)

<table>
<thead>
<tr>
<th></th>
<th><strong>Pros</strong></th>
<th><strong>Cons</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Crossproducting</strong> (Srinivasan et al [Sigcomm 98])</td>
<td>Fast accesses. Suitable for multiple fields.</td>
<td>Large memory requirements. Suitable without caching for classifiers with fewer than 50 rules.</td>
</tr>
<tr>
<td><strong>Bil-level Parallelism</strong> (Lakshman and Stiliadis [Sigcomm 98])</td>
<td>Suitable for multiple fields.</td>
<td>Large memory bandwidth required. Comparatively slow lookup rate. Hardware only.</td>
</tr>
</tbody>
</table>
### Proposed Schemes (Contd.)

<table>
<thead>
<tr>
<th>Pros</th>
<th>Cons</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Hierarchical Intelligent Cuttings</strong> <em>(Gupta and McKeown[HotI 99])</em>&lt;br&gt;Suitable for multiple fields. Small memory requirements. Good update time.</td>
<td>Large preprocessing time.</td>
</tr>
<tr>
<td><strong>Tuple Space Search</strong> <em>(Srinivasan et al[Sigcomm 99])</em>&lt;br&gt;Suitable for multiple fields. The basic scheme has good update times and memory requirements.</td>
<td>Classification rate can be low. Requires perfect hashing for determinism.</td>
</tr>
<tr>
<td><strong>Recursive Flow Classification</strong> <em>(Gupta and McKeown[Sigcomm 99])</em>&lt;br&gt;Fast accesses. Suitable for multiple fields. Reasonable memory requirements for real-life classifiers.</td>
<td>Large preprocessing time and memory requirements for large classifiers.</td>
</tr>
</tbody>
</table>
Scheduling
Output Scheduling

Allocating output bandwidth
Controlling packet delay

scheduler
Output Scheduling

- FIFO
- Fair Queueing
Motivation: Parekh-Gallager theorem

- Let a connection be allocated weights at each WFQ scheduler along its path, so that the least bandwidth it is allocated is $g$
- Let it be leaky-bucket regulated such that # bits sent in time $[t_1, t_2] \leq g(t_2 - t_1) + \sigma$
- Let the connection pass through $K$ schedulers, where the $k$th scheduler has a rate $r(k)$
- Let the largest packet size in the network be $P$

$$\text{end-to-end delay} \leq \sigma / g + \sum_{k=1}^{K-1} \frac{P}{g} + \sum_{k=1}^{K} \frac{P}{r(k)}$$
Motivation

- FIFO is natural but gives poor QoS
  - bursty flows increase delays for others
  - hence cannot guarantee delays

Need round robin scheduling of packets

- Fair Queueing
- Weighted Fair Queueing, Generalized Processor Sharing
Scheduling: Requirements

- An ideal scheduling discipline
  - is easy to implement: VLSI space, exec time
  - is fair: max-min fairness
  - provides performance bounds:
    - deterministic or statistical
    - granularity: micro-flow or aggregate flow
  - allows easy admission control decisions
  - to decide whether a new flow can be allowed
**Choices: 1. Priority**

- Packet is served from a given priority level only if no packets exist at higher levels (*multilevel priority with exhaustive service*)
- Highest level gets lowest delay
- Watch out for starvation!
- Usually map priority levels to delay classes

<table>
<thead>
<tr>
<th>Priority</th>
<th>Low bandwidth urgent messages</th>
<th>Realtime</th>
<th>Non-realtime</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Shivkumar Kalyanaraman

Rensselaer Polytechnic Institute
Scheduling Policies: Choices #1

- **Priority Queuing**: classes have different priorities; class may depend on explicit marking or other header info, eg IP source or destination, TCP Port numbers, etc.
- Transmit a packet from the highest priority class with a non-empty queue. Problem: **starvation**
- **Preemptive and non-preemptive versions**

[Diagram showing high priority queue and low priority queue with arrivals and departures]

[Timeline showing packet arrivals and service departures]
Scheduling Policies (more)

- **Round Robin:** scan class queues serving one from each class that has a non-empty queue
Choices: 2. Work conserving vs. non-work-conserving

- Work conserving discipline is never idle when packets await service
- Why bother with non-work conserving?
Non-work-conserving disciplines

- Key conceptual idea: delay packet till *eligible*
- Reduces delay-jitter => fewer buffers in network
- How to choose eligibility time?
  - rate-jitter regulator
    - bounds maximum outgoing rate
  - delay-jitter regulator
    - compensates for variable delay at previous hop
Do we need non-work-conservation?

- Can remove delay-jitter at an endpoint instead
  - but also reduces size of switch buffers…
- Increases mean delay
  - not a problem for *playback* applications
- Wastes bandwidth
  - can serve best-effort packets instead
- Always punishes a misbehaving source
  - can’t have it both ways
- Bottom line: not too bad, implementation cost may be the biggest problem
Choices: 3. Degree of aggregation

- More aggregation
  - less state
  - cheaper
    - smaller VLSI
    - less to advertise
  - BUT: less individualization

- Solution
  - aggregate to a *class*, members of class have same performance requirement
  - no protection within class
Choices: 4. Service within a priority level

- In order of arrival (FCFS) or in order of a service tag
- Service tags => can arbitrarily reorder queue
  - Need to sort queue, which can be expensive
- FCFS
  - bandwidth hogs win (no protection)
  - no guarantee on delays
- Service tags
  - with appropriate choice, both protection and delay bounds possible:
    - eg: differential buffer management, packet drop
Weighted round robin

- Serve a packet from each non-empty queue in turn
- Unfair if packets are of different length or weights are not equal
- Different weights, fixed packet size
  - serve more than one packet per visit, after normalizing to obtain integer weights
- Different weights, variable size packets
  - normalize weights by mean packet size
    - e.g. weights \{0.5, 0.75, 1.0\}, mean packet sizes \{50, 500, 1500\}
    - normalize weights: \{0.5/50, 0.75/500, 1.0/1500\} = \{0.01, 0.0015, 0.000666\}, normalize again \{60, 9, 4\}
Problems with Weighted Round Robin

- With variable size packets and different weights, need to know mean packet size in advance
- Can be unfair for long periods of time
- E.g.
  - T3 trunk with 500 connections, each connection has mean packet length 500 bytes, 250 with weight 1, 250 with weight 10
  - Each packet takes $500 \times \frac{8}{45}$ Mbps = 88.8 microseconds
  - Round time $= 2750 \times 88.8 = 244.2$ ms
Generalized Processor Sharing (GPS)

- Assume a fluid model of traffic
  - Visit each non-empty queue in turn (RR)
  - Serve infinitesimal from each
  - Leads to “max-min” fairness
- GPS is un-implementable!
  - We cannot serve infinitesimals, only packets
Fair Queuing (FQ)

- Idea: serve packets in the order in which they would have finished transmission in the fluid flow system
- Mapping *bit-by-bit schedule onto packet transmission* schedule
- Transmit packet with the lowest $F_i$ at any given time
- Variation: *Weighted Fair Queuing (WFQ)*
FQ Example

Cannot preempt packet currently being transmitted
WFQ: Practical considerations

- For every packet, the scheduler needs to:
  - classify it into the right flow queue and maintain a linked-list for each flow
  - schedule it for departure

- Complexities of both are $o(\log [\# \ of \ flows])$
  - first is hard to overcome (studied earlier)
  - second can be overcome by DRR
Deficit Round Robin

200 600 400

500 500

250 500 400

750 1000

Good approximation of FQ
Much simpler to implement

500 Quantum size
WFQ Problems

- To get a delay bound, need to pick $g$
  - the lower the delay bounds, the larger $g$ needs to be
  - large $g$ => exclusion of more competitors from link
  - $g$ can be *very large*, in some cases 80 times the peak rate!
- Sources must be leaky-bucket regulated
  - but choosing leaky-bucket parameters is problematic
- WFQ *couples* delay and bandwidth allocations
  - low delay requires allocating more bandwidth
  - wastes bandwidth for low-bandwidth low-delay sources
Delay-Earliest Due Date (EDD)

- Earliest-due-date: packet with earliest deadline selected
- Delay-EDD prescribes how to assign deadlines to packets
- A source is required to send slower than its *peak rate*
- Bandwidth at scheduler reserved at peak rate
- Deadline = expected arrival time + delay bound
  - If a source sends faster than contract, delay bound will not apply
- Each packet gets a hard delay bound
- Delay bound is *independent* of bandwidth requirement
  - but reservation is at a connection’s peak rate
- Implementation requires per-connection state and a priority queue
Rate-controlled scheduling

- A class of disciplines
  - two components: regulator and scheduler
  - incoming packets are placed in regulator where they wait to become eligible
  - then they are put in the scheduler
- Regulator shapes the traffic, scheduler provides performance guarantees
- Considered impractical; interest waning after QoS decline
Examples

- Recall
  - rate-jitter regulator
    - bounds maximum outgoing rate
  - delay-jitter regulator
    - compensates for variable delay at previous hop
- Rate-jitter regulator + FIFO
  - similar to Delay-EDD
- Rate-jitter regulator + multi-priority FIFO
  - gives both bandwidth and delay guarantees (RCSP)
- Delay-jitter regulator + EDD
  - gives bandwidth, delay, and delay-jitter bounds (Jitter-EDD)
Stateful Solution Complexity

- Data path
  - Per-flow classification
  - Per-flow buffer management
  - Per-flow scheduling
- Control path
  - install and maintain per-flow state for data and control paths
Differentiated Services Model

- **Edge routers**: traffic conditioning (policing, marking, dropping), SLA negotiation
  - Set values in DS-byte in IP header based upon negotiated service and observed traffic.
- **Interior routers**: traffic classification and forwarding (near stateless core!)
  - Use DS-byte as index into forwarding table
Diffserv Architecture

**Edge router:**
- per-flow traffic management
- marks packets as in-profile and out-profile

**Core router:**
- per class TM
- buffering and scheduling based on marking at edge
- preference given to in-profile packets
- Assured Forwarding
Diff Serv: implementation

- Classify flows into classes
  - maintain only **per-class** queues
  - perform FIFO within each class
  - avoid “curse of dimensionality”
Diff Serv

- A framework for providing differentiated QoS
  - set Type of Service (ToS) bits in packet headers
  - this classifies packets into classes
  - routers maintain per-class queues
  - condition traffic at network edges to conform to class requirements
- May still need queue management inside the network
Network Processors (NPUs)
What makes a CPU appealing for a PC

- **Flexibility**: Supports many applications
- **Time to market**: Allows quick introduction of new applications
- **Future proof**: Supports as-yet unthought of applications
- No-one would consider using fixed function ASICs for a PC
Why NPUs seem like a good idea

- What makes a NPU appealing
  - **Time to market:** Saves 18 months building an ASIC. Code re-use.
  - **Flexibility:** Protocols and standards change.
  - **Future proof:** New protocols emerge.
  - **Less risk:** Bugs more easily fixed in s/w.
- Surely no-one would consider using fixed function ASICs for new networking equipment?
The other side of the NPU debate...

- Jack of all trades, master of none
  - NPUs are difficult to program
  - NPUs inevitably consume more power,
  - ...run more slowly and
  - ...cost more than an ASIC
- Requires domain expertise
  - Why would a/the networking vendor educate its suppliers?
- Designed for computation rather than memory-intensive operations
NPU Characteristics

- NPUs try hard to hide memory latency
  - Conventional caching doesn’t work
    - Equal number of reads and writes
    - No temporal or spatial locality
    - Cache misses lose throughput, confuse schedulers and break pipelines
  - Therefore it is common to use multiple processors with multiple contexts
Incoming packets dispatched to:
1. Idle processor, or
2. Processor dedicated to packets in this flow (to prevent mis-sequencing), or
3. Special-purpose processor for flow, e.g. security, transcoding, application-level processing.
Network Processors
Pipelining

Processing broken down into (hopefully balanced) steps,
Each processor performs one step of processing.
NPUs and Memory

- Network processors and their memory
  - Packet processing is all about getting packets into and out of a chip and memory.
  - Computation is a side-issue.
  - Memory speed is everything: Speed matters more than size.
NPUs and Memory

Typical NPU or packet-processor has 8-64 CPUs, 12 memory interfaces and 2000 pins.
Intel IXP Network Processors

- **Microengines**
  - RISC processors optimized for packet processing
  - Hardware support for multi-threading
- **Fast path**
- **Embedded StrongARM/Xscale**
  - Runs embedded OS and handles exception tasks
  - Slow path, Control plane

Shivkumar Kalyanaraman
NPU Building Blocks: Processors

The Pyramid Of Processor Scale

- Lower levels need the most increase
Division of Functions

- **General purpose CPU**
  - Highest level functionality
  - Administrative interface
  - System control
  - Overall management functions
  - Routing protocols

- **Embedded processor**
  - Intermediate functionality
  - Higher-layer protocols
  - Control of I/O processors
  - Exception and error handling
  - High-level ingress (e.g., reassembly)
  - High-level egress (e.g., traffic shaping)

- **I/O processor**
  - Basic packet processing
  - Classification
  - Forwarding
  - Low-level ingress operations
  - Low-level egress operations
NPU Building Blocks: Memory

The Pyramid Of Memory Scale

- Largest memory is least expensive
Memory Scaling

- Access latency
  - Raw read/write access speed
  - SRAM 2 - 10 ns
  - DRAM 50 - 70 ns
  - External memory takes order of magnitude longer than onboard
- Memory cycle time
  - Measure of successive read/write operations
  - Important for networking because packets are large
  - Read Cycle time (tRC) is time for successive fetch operations
  - Write Cycle time (tWC) is time for successive store operations
# Memory Types

<table>
<thead>
<tr>
<th>Memory Technology</th>
<th>Abbreviation</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Synchronized DRAM</td>
<td>SDRAM</td>
<td>Synchronized with CPU for lower latency</td>
</tr>
<tr>
<td>Quad Data Rate SRAM</td>
<td>QDR-SRAM</td>
<td>Optimized for low latency and multiple access</td>
</tr>
<tr>
<td>Zero Bus Turnaround SRAM</td>
<td>ZBT-SRAM</td>
<td>Optimized for random access</td>
</tr>
<tr>
<td>Fast Cycle RAM</td>
<td>FCRAM</td>
<td>Low cycle time optimized for block transfer</td>
</tr>
<tr>
<td>Double Data Rate DRAM</td>
<td>DDR-DRAM</td>
<td>Optimized for low latency</td>
</tr>
<tr>
<td>Reduced Latency DRAM</td>
<td>RLDRAM</td>
<td>Low cycle time and low power requirements</td>
</tr>
</tbody>
</table>
NPU Building Blocks: CAM and Ternary CAM

- Given
  - Pattern for which to search
  - Known as *key*
- CAM returns
  - First slot that matches key, or
  - All slots that match key

**Ternary CAM (T-CAM):**

- Allows masking of entries
- Good for network processor

**CAM Operation:**

```plaintext
for each slot do {
    if ((key & mask) == (slot & mask)) {
        declare key matches slot;
    } else {
        declare key does not match slot;
    }
}
```
Memory Caching vs CAM

**CACHE:**
- General-purpose technique
- May not work well in network systems
  - Low temporal locality
  - Large cache size (either more entries or larger granularity of access)

**Content Addressable Memory (CAM):**
- Combination of mechanisms
  - Random access storage
  - Exact-match pattern search
- Rapid search enabled with parallel hardware
**Ternary CAMs**

**Associative Memory**

<table>
<thead>
<tr>
<th>Value</th>
<th>Mask</th>
<th>Next Hop</th>
</tr>
</thead>
<tbody>
<tr>
<td>10.0.0.0</td>
<td>255.0.0.0</td>
<td>R1</td>
</tr>
<tr>
<td>10.1.0.0</td>
<td>255.255.0.0</td>
<td>R2</td>
</tr>
<tr>
<td>10.1.1.0</td>
<td>255.255.255.0</td>
<td>R3</td>
</tr>
<tr>
<td>10.1.3.0</td>
<td>255.255.255.0</td>
<td>R4</td>
</tr>
<tr>
<td>10.1.3.1</td>
<td>255.255.255.255</td>
<td>R4</td>
</tr>
</tbody>
</table>

**Using T-CAMs for Classification:**

- Extract values from fields in headers
- Form values in contiguous string
- Use a key for T-CAM lookup
- Store classification in slot
IXP: A Building Block for Network Systems

- Example: IXP2800
  - 16 micro-engines + XScale core
  - Up to 1.4 Ghz ME speed
  - 8 HW threads/ME
  - 4K control store per ME
  - Multi-level memory hierarchy
  - Multiple inter-processor communication channels

- NPU vs. GPU tradeoffs
  - Reduce core complexity
    - No hardware caching
    - Simpler instructions \( \rightarrow \) shallow pipelines
  - Multiple cores with HW multi-threading per chip

Rensselaer Polytechnic Institute
IXP2800 Features

- Half Duplex OC-192 / 10 Gb/sec Ethernet Network Processor
- XScale Core
  - 700 MHz (half the ME)
  - 32 Kbytes instruction cache / 32 Kbytes data cache
- Media / Switch Fabric Interface
  - 2 x 16 bit LVDS Transmit & Receive
  - Configured as CSIX-L2 or SPI-4
- PCI Interface
  - 64 bit / 66 MHz Interface for Control
  - 3 DMA Channels
- QDR Interface (w/Parity)
  - (4) 36 bit SRAM Channels (QDR or Co-Processor)
  - Network Processor Forum LookAside-1 Standard Interface
  - Using a “clamshell” topology both Memory and Co-processor can be instantiated on same channel
- RDR Interface
  - (3) Independent Direct Rambus DRAM Interfaces
  - Supports 4i Banks or 16 interleaved Banks
  - Supports 16/32 Byte bursts
Hardware Features to ease packet processing

- Ring Buffers
  - For inter-block communication/synchronization
  - Producer-consumer paradigm
- Next Neighbor Registers and Signaling
  - Allows for single cycle transfer of context to the next logical micro-engine to dramatically improve performance
  - Simple, easy transfer of state
- Distributed data caching within each micro-engine
  - Allows for all threads to keep processing even when multiple threads are accessing the same data
XScale Core processor

- Compliant with the ARM V5TE architecture
  - support for ARM’s thumb instructions
  - support for Digital Signal Processing (DSP) enhancements to the instruction set
  - Intel’s improvements to the internal pipeline to improve the memory-latency hiding abilities of the core
- does not implement the floating-point instructions of the ARM V5 instruction set
Microengines – RISC processors

- IXP 2800 has 16 microengines, organized into 4 clusters (4 MEs per cluster)
- ME instruction set specifically tuned for processing network data
- 40-bit x 4K control store
- Six-stage pipeline in an instruction
  - On an average takes one cycle to execute
- Each ME has eight hardware-assisted threads of execution
  - can be configured to use either all eight threads or only four threads
- The non-preemptive hardware thread arbiter swaps between threads in round-robin order
Registers available to each ME

- Four different types of registers
  - general purpose, SRAM transfer, DRAM transfer, next-neighbor (NN)
- 256, 32-bit GPRs
  - can be accessed in thread-local or absolute mode
- 256, 32-bit SRAM transfer registers.
  - used to read/write to all functional units on the IXP2xxx except the DRAM
- 256, 32-bit DRAM transfer registers
  - divided equally into read-only and write-only
  - used exclusively for communication between the MEs and the DRAM
- Benefit of having separate transfer and GPRs
  - ME can continue processing with GPRs while other functional units read and write the transfer registers
### Different Types of Memory

<table>
<thead>
<tr>
<th>Type of Memory</th>
<th>Logical width (bytes)</th>
<th>Size in bytes</th>
<th>Approx unloaded latency (cycles)</th>
<th>Special Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Local to ME</td>
<td>4</td>
<td>2560</td>
<td>3</td>
<td>Indexed addressing post incr/decr</td>
</tr>
<tr>
<td>On-chip scratch</td>
<td>4</td>
<td>16K</td>
<td>60</td>
<td>Atomic ops 16 rings w/at. get/put</td>
</tr>
<tr>
<td>SRAM</td>
<td>4</td>
<td>256M</td>
<td>150</td>
<td>Atomic ops 64-elem q-array</td>
</tr>
<tr>
<td>DRAM</td>
<td>8</td>
<td>2G</td>
<td>300</td>
<td>Direct path to/from MSF</td>
</tr>
</tbody>
</table>
IXA Software Framework

External Processors
- Control Plane Protocol Stacks

XScale™ Core
- Control Plane PDK
- Core Components
  - Core Component Library
  - Resource Manager Library
- C/C++ Language

Microengine Pipeline
- Microblock Library
  - Microblock
  - Microblock
  - Microblock
- Microengine C Language
- Protocol Library
- Utility Library
- Hardware Abstraction Library
Micro-engine C Compiler

- C language constructs
  - Basic types,
  - Pointers, bit fields
- In-line assembly code support
- Aggregates
  - Structs, unions, arrays

Microengine C

```c
This code is used to populate CRC16 table for CRC calculation unsigned
crc_table (unsigned int l) int
{
    unsigned int j, k, crc = 0;
    k = i << 8;
    for(j = 0; j < 8; j++)
        if((crc ^ k) & 0x8000)
            crc = (crc << 1) ^ 0x1021;
        else crc <<= 1;
        k <<= 1;
    return crc;
}
```

Optimized code generated by the Microengine C compiler

Microcode

**crc_table#:**
- `alu_shl[B1, --, B, A0, <<8]`
- `immed[A0, 0, 0]`
- `immed[B0, 0, 0]`

**l_3#:**
- `alu[A2, B1, XOR, A0]`
- `br_bset[A2, 15, l_4#]`
- `alu_shl[A0, --, B, A0, <<1]`
- `br[l_5#]`

**l_4#:**
- `alu_shl[B2, --, B, A0, <<1]`
- `immed[A0, 4129, 0]`
- `alu[A0, B2, XOR, A0]`

**l_6#:**
- `alu_shl[B1, --, B, B1, <<1]`
- `alu[B0, B0, +, 1]`
- `alu[--, B0, -, 8]`
- `brl=cout[l_3#]`
- `alu[B0, --, B, A0]`
- `rtn[A1]`

Example shows:
50% fewer lines of code required than with assembly

For simplified, portable, microengine programming!

Shivkumar Kalyanaraman
What is a Microblock

- Data plane packet processing on the microengines is divided into *logical functions called microblocks*.
- Coarse Grain and stateful
- Example
  - 5-Tuple Classification, IPv4 Forwarding, NAT
- Several microblocks running on a microengine thread can be combined into a *microblock group*.
  - A microblock group has a *dispatch loop* that defines the dataflow for packets between microblocks.
  - A microblock group runs on each thread of one or more microengines.
- Microblocks can send and receive packets to/from an associated Xscale Core Component.
Core Components and Microblocks

XScale™ Core

Core Component Library
Resource Manager Library

Microengines

Microblock Library

Microblock
Microblock
Microblock

Microblock Library
Intel/3rd party blocks
User-written code
Core Libraries
Debate about network processors

The nail:

Data Hdr

Context

The hammer:

Data cache(s)

Characteristics:

1. Stream processing.
2. Multiple flows.
3. Most processing on header, not data.
4. Two sets of data: packets, context.
5. Packets have no temporal locality, and special spatial locality.
6. Context has temporal and spatial locality.

Characteristics:

1. Shared in/out bus.
2. Optimized for data with spatial and temporal locality.
3. Optimized for register accesses.
Challenges in Modern Multi-Tera-bit Class Switch Design
Tbps System Architecture

- **Switch Core**
- **Switch Fabric**
- **Input Line 1**
- **Output Line 1**
- **OC-x iff**
- **Network Processor**
- **Ingress Buffer**
- **Egress Buffer**
- **Data**
- **Ingress FC**
- **Egress FC**
- **iRT**
- **eRT**

- **Line Card 1**
- **Line Card N**

Rensselaer Polytechnic Institute
Trend: Single POP routers

- Very high capacity (10+Tb/s)
- Line-rates T1 to OC768

Reasons:
- Big multi-rack router more efficient than many single-rack routers,
- Easier to manage fewer routers.
- Power requirements easier to meet
Multi Tbps Systems: Goals...

- Design of a terabit-class system
  - Several Tb/s aggregate throughput
    - 2.5 Tb/s: 256x256 OC-192 or 64x64 OC-768
  - OEM
    - Achieve wide coverage of application spectrum
- Single-stage
- Electronic fabric
Trends & Consequences

1. CPU Instructions per minimum length packet

2. Disparity between traffic and router growth

Consequences:
1. Per-packet processing is getting harder.
2. Efficient, simple processing will become more important.
3. Routers will get faster, simpler and more efficient.
(Weren’t they supposed to be simple in the first place?)
Power consumption is out of control

Disparity between line-rate and memory access time

Consequences:
1. Power efficiency will continue to be important.
2. Memories will seem slower and slower.
   Are we just going to keep adding more parallelism?
What’s hard, what’s not

- Linerate forwarding:
  - Linerate LPM was an issue for while.
  - Commercial TCAMs and algorithms available up to 100Gb/s.
  - 1M prefixes fit in corner of 90nm ASIC.
  - $2^{32}$ addresses will fit in a $10$ DRAM in 8 years

- Packet buffering:
  - Not a problem up to about 10Gb/s; big problem above 10Gb/s.

- Header processing:
  - For basic IPv4 operations: not a problem.
  - If we keep adding functions, it will be a problem.
  - More on this later…
What’s hard, what’s not (2)

- Switching
  - If throughput doesn’t matter:
    - Easy: Lots of multistage, distributed or load-balanced switch fabrics.
  - If throughput matters:
    - Use crossbar, VOQs and centralized scheduler
    - Or multistage fabric and lots of speedup.
  - If throughput guarantee is required:
    - Maximal matching, VOQs and speedup of two [Dai & Prabhakar ‘00]; or
    - Load-balanced 2-stage switch [Chang 01; Sigcomm 03].
Memory: Buffers

- Memory speed will matter more than size
  - Memory speed will remain a problem.
  - Waiting for slow off-chip memory will become intolerable.
  - Memory size will become less of an issue.

- Memory Size
  - Packet buffers: Today they are too big; they’ll get smaller.
Switching: Myths about CIOQ-based crossbar switches

1. “Input-queued crossbars have low throughput”
   - An input-queued crossbar can have as high throughput as any switch.

2. “Crossbars don’t support multicast traffic well”
   - A crossbar inherently supports multicast efficiently.

3. “Crossbars don’t scale well”
   - Today, it is the number of chip I/Os, not the number of crosspoints, that limits the size of a switch fabric. Expect 5-10Tb/s crossbar switches.
Packet processing gets harder

What we’d like: (more features)
QoS, Multicast, Security, ...

What will happen

Instructions per arriving byte

time
Packet processing gets harder

Clock cycles per minimum length packet since 1996
Power

Requirement
- Do not exceed the per shelf (2 kW), per board (150W), and per chip (20W) budgets
- Forced-air cooling, avoid hot-spots
- More throughput at same power: Gb/s/W density is increasing
- I/O uses an increasing fraction of power (> 50%)
- Electrical I/O technology has not kept pace with capacity demand
- Low-power, high-density I/O technology is a must
- CMOS density increases faster than W/gate decreases
- Functionality/chip constrained by power rather than density

Power determines the number of chips and boards
- Architecture must be able to be distributed accordingly
Packaging

- Requirement
  - NEBS compliance
  - Constrained by
    - Standard form factors
    - Power budget at chip, card, rack level
  - Switch core
    - Link, connector, chip packaging technology
      - Connector density (pins/inch)
      - CMOS density doubles, number of pins +5-10% per generation
      - This determines the maximum per-chip and per-card throughput
  - Line cards
    - Increasing port counts
    - Prevalent line rate granularity OC-192 (10 Gb/s)
    - 1 adapter/card

> 1 Tb/s systems require multi-rack solutions
- Long cables instead of backplane (30 to 100m)
- Interconnect accounts for large part of system cost
2.5 Tb/s, 1.6x speedup, 2.5 Gb/s links 8b/10b: 4000 links (diff. pairs)

Shivkumar Kalyanaraman
Switch-Internal Round-Trip (RT)

- Physical system size
  - Direct consequence of packaging
- CMOS technology
  - Clock speeds increasing much slower than density
  - More parallelism required to increase throughput
- Shrinking packet cycle
  - Line rates have up drastically (OC-3 through OC-768)
  - Minimum packet size has remained constant

**Large round-trip (RT) in terms of min. packet duration**
- Can be (many) tens of packets per port
- Used to be only a node-to-node issue, now also inside the node
- System-wide clocking and synchronization

### Evolution of RT

<table>
<thead>
<tr>
<th>Line rate</th>
<th>OC-12</th>
<th>OC-48</th>
<th>OC-192</th>
<th>OC-768</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interconnect distance</td>
<td>1 m</td>
<td>1 m</td>
<td>6 m</td>
<td>30 m</td>
</tr>
<tr>
<td>Interconnect type</td>
<td>backplane</td>
<td>backplane</td>
<td>cable</td>
<td>fiber</td>
</tr>
<tr>
<td>Packet duration</td>
<td>512 ns</td>
<td>128 ns</td>
<td>32 ns</td>
<td>8 ns</td>
</tr>
<tr>
<td>Round Trip</td>
<td>&lt;&lt; 1 cell</td>
<td>~ 1 cell</td>
<td>16 cells</td>
<td>64 cells</td>
</tr>
</tbody>
</table>
Switch-Internal Round-Trip (RT)

Consequences
- Performance impact?
- All buffers must be scaled by RT
- Fabric-internal flow control becomes an important issue
Physical Separation
Separating Control & Data

Linecard

Switch Port

Data Channel
Control Channel

1: Req
2: Grant/credit
3: Data

Linecard measures RTT to ~1 cell time

Switch Fabric
Buffer or Guard-Band

Switch Scheduler
Speed-Up

Requirement
- “Industry standard” 2x speed-up
- Three flavors
  - Utilization: compensate SAR overhead
  - Performance: compensate scheduling inefficiencies
  - OQ speed-up: memory access time

Switch core speed-up $S$ is very costly
- Bandwidth is a scarce resource: COST and POWER
- Core buffers must run $S$ times faster
- Core scheduler must run $S$ times faster
- Is it really needed?
- SAR overhead reduction
  - Variable-length packet switching: hard to implement, but may be more cost-effective
  - Performance: does the gain in performance justify the increase in cost and power?
    - Depends on application
    - Low Internet utilization
Multicast

Requirement
- Full multicast support
  - Many multicast groups, full link utilization, no blocking, QoS
- Complicates everything
  - Buffering, queuing, scheduling, flow control, QoS

Sophisticated multicast support really needed?
- Expensive
- Often disabled in the field…
  - Complexity, billing, potential for abuse, etc.
- Again, depends on application
Packet size

- Requirement
  - Support very short packets (32-64B)
    - 40B @ OC-768 = 8 ns
  - Short packet duration
    - Determines speed of control section
      - Queues and schedulers
    - Implies longer RT
    - Wider data paths

- Do we have to switch short packets individually?
  - Aggregation techniques
    - Burst, envelope, container switching, “packing”
  - Single-stage, multi-path switches
    - Parallel packet switch
Increase Payload Size

Packing Packets in “Cells”

Packets all in same VOQ:

100B  4

40B  3

178B  2

80B  1

Cell Hdr

Cell Hdr

Cell Hdr

Cell Hdr

Cell Hdr

128B cell payload
Can optics help in switching?
Can optics help?

Cynical view:

1. A packet switch (e.g. an IP router) must have buffering.
2. Optical buffering is not feasible.
3. Therefore, optical routers are not feasible.
4. Hence, “optical switches” are circuit switches (e.g. TDM, space or Lambda switches).
Can optics help?

Open-minded view:

- Optics seem ill-suited to processing intensive functions, or where random access memory is required.
- Optics seems well-suited to bufferless, reconfigurable datapaths.
100Tb/s optical router

Stanford University Research Project

- Collaboration
  - 4 Professors at Stanford (Mark Horowitz, Nick McKeown, David Miller and Olav Solgaard), and our groups.

- Objective
  - To determine the best way to incorporate optics into routers.
  - Push technology hard to expose new issues.
    - Photonics, Electronics, System design
  - Motivating example: The design of a 100 Tb/s Internet router
    - Challenging but not impossible (~100x current commercial systems)
    - It identifies some interesting research problems
100Tb/s optical router

Electronics Linecard #1
- Line termination
- IP packet processing
- Packet buffering

Optical Switch

Electronics Linecard #625
- Line termination
- IP packet processing
- Packet buffering

160Gb/s

160-320Gb/s

(100Tb/s = 625 * 160Gb/s)
Research Problems

- Linecard
  - Memory bottleneck: Address lookup and packet buffering.
- Architecture
  - Arbitration: Computation complexity.
- Switch Fabric
  - Optics: Fabric scalability and speed,
  - Electronics: Switch control and link electronics,
  - Packaging: Three surface problem.
Problem

Packet buffer needs density of DRAM (40 Gbits) and speed of SRAM (2ns per packet)

Solution

- Hybrid solution uses on-chip SRAM and off-chip DRAM.
- Identified optimal algorithms that minimize size of SRAM (12 Mbits).
- Precisely emulates behavior of 40 Gbit, 2ns SRAM.
The Arbitration Problem

- A packet switch fabric is reconfigured for every packet transfer.
- At 160Gb/s, a new IP packet can arrive every 2ns.
- The configuration is picked to maximize throughput and not waste capacity.
- Known algorithms are too slow.
100Tb/s Router

Optical Switch Fabric

Optical links

Racks of 160Gb/s Linecards
Racks with 160Gb/s linecards
Passive Optical Switching

Integrated AWGR or diffraction grating based wavelength router

- Ingress Linecard 1: $\lambda_1, \ldots, \lambda_n$
- Ingress Linecard 2: $\lambda_1, \ldots, \lambda_n$
- Ingress Linecard n: $\lambda_1, \ldots, \lambda_n$
- Midstage Linecard 1: $\lambda_1, \ldots, \lambda_n$
- Midstage Linecard 2: $\lambda_1, \ldots, \lambda_n$
- Midstage Linecard n: $\lambda_1, \ldots, \lambda_n$
- Egress Linecard 1: $\lambda_1, \ldots, \lambda_n$
- Egress Linecard 2: $\lambda_1, \ldots, \lambda_n$
- Egress Linecard n: $\lambda_1, \ldots, \lambda_n$

Integrated AWGR or diffraction grating based wavelength router
Question

- Can we use an **optical** fabric at 100Tb/s with 100% throughput?

- Conventional answer: **No.**
  - Need to **reconfigure** switch too often
  - 100% throughput requires complex electronic scheduler.
Two-stage load-balancing switch

100% throughput for weakly mixing, stochastic traffic.

[C.-S. Chang, Valiant]
Optical two-stage router

Phase 1
- Lookup
- Buffer

Phase 2
- Lookup
- Buffer
- Lookup
- Buffer

Linecards
100Tb/s Load-Balanced Router

Linecard Rack 1

$L = 16$
160Gb/s linecards

Linecard Rack 2

$L = 16$
160Gb/s linecards

Linecard Rack $G = 40$

$L = 16$
160Gb/s linecards

40 x 40 MEMS

Switch Rack < 100W

55 56
Predictions: Core Internet routers

- The need for more capacity for a given power and volume budget will mean:
- Fewer functions in routers:
  - Little or no optimization for multicast,
  - Continued over-provisioning will lead to little or no support for QoS, DiffServ, …,
- Fewer unnecessary requirements:
  - Mis-sequencing will be tolerated,
  - Latency requirements will be relaxed.
- Less programmability in routers, and hence no network processors (NPs used at edge…).
- Greater use of optics to reduce power in switch.
Likely Events

The need for capacity and reliability will mean:

- Widespread replacement of core routers with transport switching based on circuits:
  - Circuit switches have proved simpler, more reliable, lower power, higher capacity and lower cost per Gb/s. Eventually, this is going to matter.
  - Internet will evolve to become edge routers interconnected by rich mesh of WDM circuit switches.
Summary

- High speed routers: lookup, switching, classification, buffer management
- **Lookup**: Range-matching, tries, multi-way tries
- **Switching**: circuit s/w, crossbar, batcher-banyan,
- **Queuing**: input/output queuing issues
- **Classification, Scheduling**: …
- Road ahead to 100 Tbps routers…

Shivkumar Kalyanaraman