Router Construction

Outline
- Switched Fabrics
- IP Routers
- Tag Switching

Workstation-Based

- Aggregate bandwidth
  - 1/2 of the I/O bus bandwidth
  - capacity shared among all hosts connected to switch
  - example: 1Gbps bus can support 5 x 100Mbps ports (in theory)

- Packets-per-second
  - must be able to switch small packets
  - 300,000 packets-per-second is achievable
  - e.g., 64-byte packets implies 155Mbps

Switching Hardware

- Design Goals
  - throughput (depends on traffic model)
  - scalability (a function of n)

- Ports
  - circuit management (e.g., map VCIs, route datagrams)
  - buffering (input and/or output)

- Fabric
  - as simple as possible
  - sometimes do buffering (internal)
Buffering

- Wherever contention is possible
  - input port (contend for fabric)
  - internal (contend for output port)
  - output port (contend for link)
- Head-of-Line Blocking
  - input buffering

Crossbar Switches

Knockout Switch

- Example crossbar
- Concentrator
  - select $f$ of $n$ packets
- Complexity: $n^2$
Knockout Switch (cont)

- Output Buffer

Self-Routing Fabrics

- Banyan Network
  - constructed from simple $2 \times 2$ switching elements
  - self-routing header attached to each packet
  - elements arranged to route based on this header
  - no collisions if input packets sorted into ascending order
  - complexity: $n \log_2 n$

Self-Routing Fabrics (cont)

- Batcher Network
  - switching elements sort two numbers
    - some elements sort into ascending (clear)
    - some elements sort into descending (shaded)
  - elements arranged to implement merge sort
  - complexity: $n \log_2 n$

- Common Design: Batcher-Banyan Switch
High-Speed IP Router

- Switch (possibly ATM)
- Line Cards
  - link interface (input, output)
  - router lookup (input)
  - common IP path (input)
  - packet queue (output)
- Control Processor
  - routing protocol(s)
  - exceptional cases

IP Forwarding is Slow

- Problem: classless IP addresses (CIDR)
- Route by variable-length Forwarding Equivalence Classes (FEC)
  - FEC = IP address plus prefix of 1-32 bits; e.g., 172.200.0.0/16
- IP Router
  - forwarding tbl: <FEC> → <next hop, port>
  - match IP address to FEC w/ longest prefix

ATM Forwarding

- Primary goal: fast, cheap forwarding
- 1Gb/s IP router: $187,000
- 5Gb/s ATM switch: $41,000
- Create Virtual Circuit at Flow Setup
  - <in VCI> → <port, out VCI>
- Cell Forwarding
  - index, swap, switch
Cisco: Tag Switching

- Add a VCI-like tag to packets
  - <in tag> <next hop, port, out tag>
- TSR uses ATM switch hardware
- IP routing protocols (OSPF, RIP, BGP)
  - build forwarding table from routing table
- Goal: IP router functionality at ATM switch speeds/costs

Forwarding

- Shim before IP header
- Tag Forwarding Information Base (TFIB)
  - <in tag> <next hop, port, out tag>
- Just like ATM
  - index, swap, switch

Tag Binding

- New FEC from IP routing protocols
  - Select local tag (index in TFIB)
  - <in tag> <next hop, port, ??>
- Need <out tag> for next hop
- Other routers need my <in tag>
- Solution: distribute tags like other routing info
Tag Distribution Protocol

• Send TDP messages to peers
  – <FEC, my tag>
• Upon receiving TDP message, check if sender is next hop for FEC
  – yes, save tag in TFIB
  – no, can discard or save for future use
• ‘Control-driven’ label assignment

The First Tag

• Two kinds of routers: edge vs. interior
  
  E  I  I  E

  • Edge: add shim based on IP lookup, strip at exit
  • Interior: forward by tag only

Robustness Issues

• What if tag fault?
  – try to forward (default route)
  – discard packet
• Forwarding Loops
  – topology changes cause temporary loops
  – TTL field in tag, same as IP
Ipsilon: IP Switching

- Run on ATM switch over ATM network
  - ATM hardware + IP switching software
- Idea: Exploit temporal locality of traffic to cache routing decisions
- Associate labels (VCI) with flows
  - forward packets as usual
  - main difference is in how labels are created, distributed to other routers

IP Switch

- Assume default ATM virtual circuits between routers
- Router runs IP routing protocol, can forward IP packets on default VCs
- Identify flows, assign flow-specific VC
  - flow = port pair or host pair
- ‘Data-driven’ label assignment

Flow Setup on IP Switch

- \(<\text{vci} = x, \text{life}>\)\text{ Port c}\text{ Port i}\text{ Port j}\text{ Controller}\text{ ATM Switch}\text{ IFMP message}<\text{flowID, vci} = x, \text{life}>\text{ IFMP message}<\text{flowID, vci} = y, \text{life}>\text{ vci = x}\text{ vci = y}\text{ Port c}\text{ Port j}\text{ \text{IFMP message}}

- \(\text{<vci = x> \rightarrow \text{<port c, vci = x>}}\)
- Get IFMP, \(\text{<vci = x> \rightarrow \text{<port j, vci = y>}}\)
### Comparison

<table>
<thead>
<tr>
<th>IP Switching</th>
<th>Tag Switching</th>
</tr>
</thead>
<tbody>
<tr>
<td>Switch by flow</td>
<td>Switch by FEC</td>
</tr>
<tr>
<td>Data driven</td>
<td>Control driven</td>
</tr>
<tr>
<td>Soft-state timeout</td>
<td>Route changes</td>
</tr>
<tr>
<td>Between end-hosts</td>
<td>Between edge TSRs</td>
</tr>
<tr>
<td>Every router can do IP lookup</td>
<td>Interior TSRs only do tag switching</td>
</tr>
<tr>
<td>Scalable?</td>
<td></td>
</tr>
</tbody>
</table>

Spring 2002                      
CS 461                            
22