Switch structures and fabrics

Keshav Chapter 8
Outline

Generic switch architecture
Criteria for evaluating switches
Switch classification 4: By structure of implementation
Time-division switches
Space-division switches
Outline

Generic switch architecture
  Line interface cards
  Port processors
  Switching fabric
  Control processor

Criteria for evaluating switches

Switch classification 4: By structure of implementation

Time-division switches
Space-division switches
Basic schematic architecture of a “switch”

<table>
<thead>
<tr>
<th>Admission control</th>
<th>Congestion control</th>
<th>Routing</th>
<th>Reservation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Classification &amp; Policing</td>
<td>Switching</td>
<td>Output scheduling</td>
<td></td>
</tr>
</tbody>
</table>

Control per-packet or per-connection processing

Data path per-packet processing

Slide based on slide 4 of McKeown & Prabhakar’s SIGCOMM 99 tutorial: “High Performance Switches and Routers: Theory and Practice”

Copyright © 2005 Tim Moors
Generic switch architecture

**Line interface cards** (input & output)

**Port processors** (input & output, may be different)

**Switching fabric** (internal to switch)
- We’ll consider these in depth shortly.
- Fabric usually has own clock, and port processors are synchronized to it (e.g. with buffering). Some fabrics include internal buffering.

**Control processor**

---

Copyright © 2005 Tim Moors
Line interfaces

Functions: Physical layer:
- Optoelectronic conversion
- Analog to digital; line coding
- Extract timing from received signal
- Serial-parallel

Terminology: In “wire-speed” switches, the line interfaces form the bottleneck (not port processing, control, or fabric)
Line interfaces: general assumptions

**Same number of input and output interfaces**
- exception: Arrays in Clos switches

**All interfaces have the same speed**
- may want different speeds (sometimes called “asymmetrical” switching)
  - e.g. LAN switch with
    - many hosts connected @ 10Mb/s, and
    - a fast server connected @ 100Mb/s, and
    - a connection to a more central part of the network @ 100Mb/s.
  - achieve through either:
    - variable number of LICs per port processor. e.g. port might have a capacity of 100Mb/s. Provide the option of connecting the port to either 10×10Mb/s line interfaces or 1×100Mb/s interface.
    - inverse multiplexing (defined shortly)

Asymmetrical switching complicates cut-through forwarding (see buffering lecture)
Port processors

Specific processors may provide a subset of these features.

Input processor roles:

- **Validate packets**: check integrity, version #, length, etc
- **Determine class** of service (controls queuing)
- **Determine required output** port using routing tables †
- **Add headers** for use within the switch:
  - self-routing information to reach that port †
  - sequence numbers to protect against mis-sequencing within fabric
- **Maintain stats** about usage (for empirical CAC, security/billing, audit)

† Keshav calls the component performing these actions a “port mapper”.

Copyright © 2005 Tim Moors
Port processors (continued)

Input & output processor roles:
• Packet scheduling, policing and shaping
• Queue information; local switching (between LICs)
• Swap label: input for unicast (with other lookup), output for multicast (differs by output) †

Output processor roles: strip off headers used in switch fabric (e.g. Banyan self-routing headers)

† Keshav calls the component performing these actions a “port mapper”.
Control processor

Contains routing tables, handles signaling, switch management

Often treated (by the fabric) as another node connected to a switch port
so that traffic to it can be switched through the fabric like normal traffic
(possibly with higher priority)
e.g. a switch with 8 bidirectional ports could connect the control processor to one port, leaving 7 for real interconnection. Switch can be controlled (for management, to setup/release connections etc) via 7 interconnection ports by addressing information to 8th port.
Here, control traffic is sent “in-band”, i.e. on the same physical interface (but perhaps a different logical connection) as is used to carry data.
Course outline

1. Fabrics
   1.5: Optical fabrics & interfaces

2. Packet classification

3. Buffer management policies
   (at inputs and outputs)

4. Traffic management
   Policing and shaping
   (at inputs and outputs)
   Scheduling (of outputs)

5. Later: Connecting switches together
“Centralized” vs distributed switches

Switches can be:
• **centralized** (e.g. implemented in a single box) or
• **distributed** (consisting of multiple interconnected boxes)

Potential advantages of centralized switches:
• Regular internal structure, c.f. heterogeneous distributed systems
• Feasible to have central control, c.f. propagation delays in distributed systems

We’ll consider centralized switches first, and later consider distributed switching (using bridges)

Regular “centralized” space-division designs can conceivably be applied to distributed switching, provided component consistency is possible.
Outline
Outline

Generic switch architecture
Criteria for evaluating switches
  Performance
  Blocking
  Cost
Switch classification 4: By structure of implementation
Time-division switches
Space-division switches
Switch performance

Mathematical analysis is often used, but is not covered in this course:

• Complex
• Highly sensitive to traffic characteristics (which are often glossed over):
  • Burstiness of arrivals
  • Focus of traffic (e.g. heavy load to server on one port)
  • Directivity (unicast, multicast, etc)
  • Packet length (e.g. packet processing bottleneck vs bandwidth)

We’ll consider qualitatively the performance impact of unicast and multicast traffic.
Switch performance  (continued)

Important points:

• Switching can change traffic characteristics, e.g. contention causes queuing, increasing burstiness
• Switch performance depends on workload.
• Rarely possible to provide a scalar measure of performance
  
  e.g. a marketing claim of 10Gb/s fabric applicable only to shared-transmission bus & assuming no switching in port processors.
Blocking

*Blocking*: When traffic can’t pass through the switch.

*Why* blocking may occur:

- **output blocking**: output port is not available – in use by other traffic
- **internal blocking**: there is no path through the switch to get to the output

*Where*: output blocking may occur internally within the switch, e.g. in Banyan
Blocking: Response

Possible responses to blocking:

- **Queue** the request, *e.g.* buffering.
  - A later lecture will examine buffering at input and output ports to deal with such blocking.
  - This lecture will examine some fabrics that use buffering for the switching process
- **Discard** the request (sometimes called “clearing” blocked calls)
- **Schedule** the request for a time when resources are available
- **Try again**: Require the initiator to make the request again, *e.g.* recirculation
Blocking: Types

Types of switch blocking:

• **Nonblocking**: Any desired connection can be established immediately.

• **Rearrangeable nonblocking**: Any desired connection can be established, possibly after rerouting existing connections.

• **Blocking**: There exist connection sets that prevent additional connections from being established.

• “Head-of-line blocking” covered later under buffering examples to come when discussing specific fabrics...
Switching cost criteria

- Crosspoint complexity ($N_x$)
- Interconnect, fan-out, and logical path depth (e.g. 2 below)
- Network control complexity
Outline
Switch classification 4: By structure of implementation

We’ll consider some of these structures (e.g. crossbar, Banyan, time-division) shortly...

Copyright © 2005 Tim Moors
Multiplexing and Demultiplexing

**Multiplexing** (muxing): “The combining of two or more information channels onto a common transmission medium.” [TG2k]
Multiplexing methods

**Time-division multiplexing**: Different inputs are sent at different times. Alternatives: Frequency & Wavelength division muxing (FDM, WDM)

**Synchronous multiplexing**: A form of TDM in which the multiplexer alternates between different inputs in round-robin order (c.f. Asynchronous TDM)
Add-drop multiplexers (ADMs)

Drop one (or some) multiplexed signal(s) from trunk
Add one or more replacement signals

Popular in optical networks (OADM), where adding/dropping a wavelength is one of the few functions that is relatively simple to implement all-optically.
Inverse multiplexing†

After demuxing, outputs usually diverge

**Inverse multiplexing**: multiple outputs follow same path & later rejoin

† aka “splitting” or “link aggregation”, e.g. the IEEE 802.3ad standard. See Chapter 9 of Seifert for details about link aggregation.
Inverse muxing: Application

e.g. Construct high-speed link from multiple lower-speed links:

- $1 \times 100$Mb/s port from $10 \times 10$Mb/s switch ports
- Interconnect routers through PSTN, with rate of connection varying according to demand.

Need to deal with potential mis-sequencing.
Outline
Outline

Generic switch architecture
Criteria for evaluating switches
Switch classification 4: By structure of implementation

Time-division switches
• Multiplexing and demultiplexing
  • add-drop multiplexers
  • inverse multiplexing
• Shared transmission medium
• Shared storage medium
  • Time-slot-interchange switch
  • other shared memory switches

Space-division switches
Shared transmission media switches

Most commonly produced form of switch
(c.f. academic emphasis on space-division switching)
A single transmission medium shared by all input and output ports, i.e. broadcast-and-select within the switch.

Output ports pick off packets destined to them, based on address, time of arrival, etc.

Limitation: high-speed filtering

Keshav Fig. 8.12
Implementing shared transmission switches

Often a bus, but can use unidirectional transmission to reduce fanout: ring, dual ring/bus, folded bus

Dispersion is (relatively) manageable in a centralized switch
⇒ Wide buses to achieve high bandwidth, e.g. 424b (53B=ATM cell)
   (utilisation may be low for small packets)
Bandwidth often equals aggregate of input ports

Access from input ports is usually fixed round-robin TDM; but other schemes are possible (e.g. Ethernet or Distributed Queuing MAC)
Implementing time-division switches on a PC

How:
• Use general-purpose NICs for line interfaces
• PC’s bus & memory for switching fabric & buffering
  o Packet is read in from a port into memory
  o Processor decides where to forward the packet
  o Processor sends packet to appropriate outgoing NIC
Disadvantages of time-division switching on a PC

- **Excessive NIC functionality**: General-purpose NICs don’t make the frame available immediately (only after integrity check) ⇒ limits forwarding modes (see buffering lecture).

- **High bus load**: Each packet is sent across the bus twice (unless DMA and non-standard NICs); Doesn’t scale well with increasing numbers of ports
Disadvantages of time-division switching on a PC (continued)

- General-purpose computer:
  - Has features that are wasted (unnecessary cost): video, disk, etc
  - Isn’t optimised for communications processing (e.g. Video RAM† in wrong place, CAMs‡ for address matching); interrupt processing overheads

- PC architecture is designed to feed information to a Central Processing Unit. Centre = bottleneck  
⇒ prefer to distribute processing amongst switch port processors, and switch dataflow directly between ports, rather than through central bottleneck.

† To be described shortly. ‡ See packet classification lecture