Switch structures and fabrics

Keshav Chapter 8

What's really important

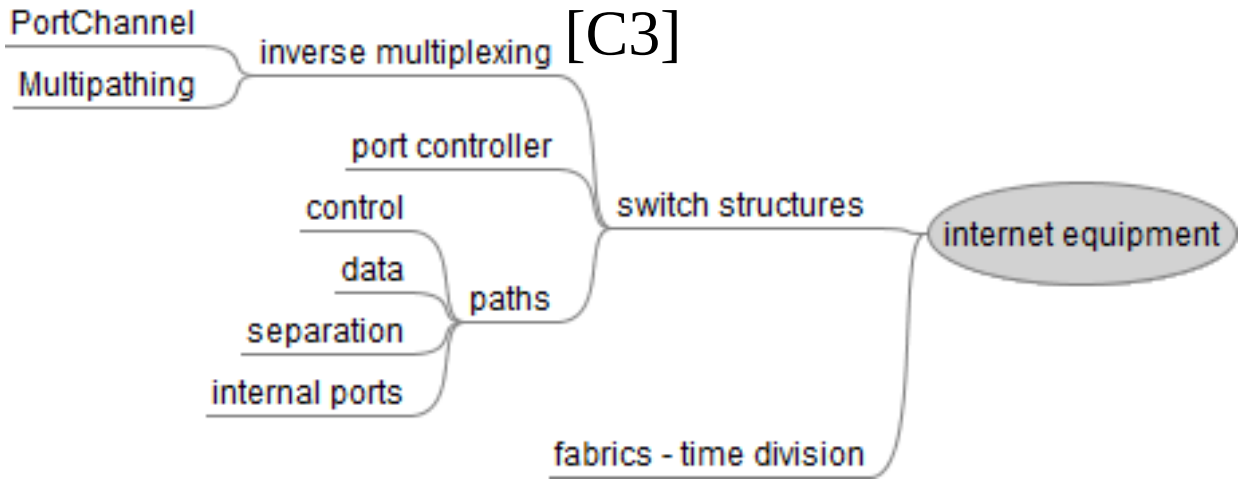
7E Generic switch architecture
0F ZC Port processors
AW Add-Drop Multiplexers (ADMs)
57 Time division switching
FP Shared transmission media switches
RZ Time-Slot-Interchange circuit switch
2X Shared memory packet switches

Relation to Cisco Nexus architecture

"For Ethernet, the destination interface that is returned as a result of a forwarding lookup could be an aggregated interface such as a **PortChannel**"

"the Cisco Nexus 5500 platform can handle packet flows at **wire speed**." [AT>

"The Cisco Nexus 5500 platform **data plane** is primarily implemented with two custom-built ASICs developed by Cisco: a set of unified **port controllers** (UPCs) that provides data-plane processing, and a unified crossbar fabric (UCF) that cross-connects the UPCs."



"The Cisco Nexus 5548P **control plane** runs ... on a dual-core 1.7-GHz Intel Xeon Processor ... The supervisor complex is connected to the **data plane** in-band through two **internal ports**"

Textbook references

- Keshav: Ch. 8.4
- Varghese:
  - Ch. 13

Outline

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

Outline

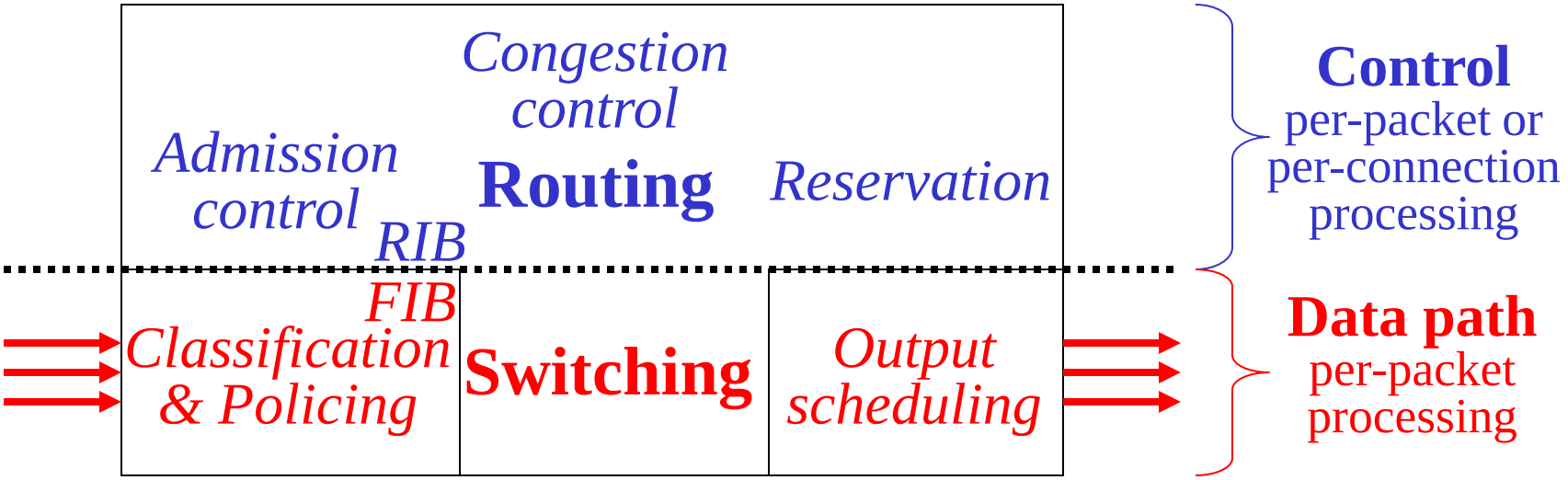
Generic switch architecture
- **Line interfaces**
- **Port processors**
- **Switching fabric**
- **Control processor**

Criteria for evaluating switches

Switch classification: By structure of implementation

Time-division switches

Basic schematic architecture of a "switch"



The division here has nothing to do with layering – i.e. not intending to state that routing occurs at a higher layer than switching.

RIB = Routing Information Base, FIB = Forwarding Information Base [QD>

Slide based on slide 4 of McKeown & Prabhakar's SIGCOMM tutorial: "High Performance Switches and Routers: Theory and Practice"

RIBs vs FIBs

- **Routing Information Base** = knowledge learned from routing protocols, e.g.
  - BGP: all path-vectors for each destination received
  - OSPF: link state information

  Size shown on previous slide
- **Forwarding Information Base** = knowledge used when forwarding, e.g. Packets for D go out on port P
  - Derived from RIB
  - Smaller than RIB, but accessed more frequently
  - Accessed when classifying packets

Software Defined Networking (SDN)

- Traditional devices implement control path & data path together
  - Distributed protocols (e.g. STP, OSPF, BGP) enable device coordination
  - Bundling control&data limits combos avail. to users
- **Software Defined Networking** allows separation of control path (software defined) from data path
  e.g. using Openflow API to data path.
  Allows:
  - More central coordination (=> better optimisation)
  - More (& faster) innovation in control systems
  - Increased abstraction => better control of complicated networks
  - Cheap commodity hardware

For more on SDN, read A Purpose-built Global Network: Google's Move to SDN and/or watch https://www.youtube.com/watch?v=YHeyuD89n1Y

Generic switch architecture

Line interfaces [YP> (input & output)

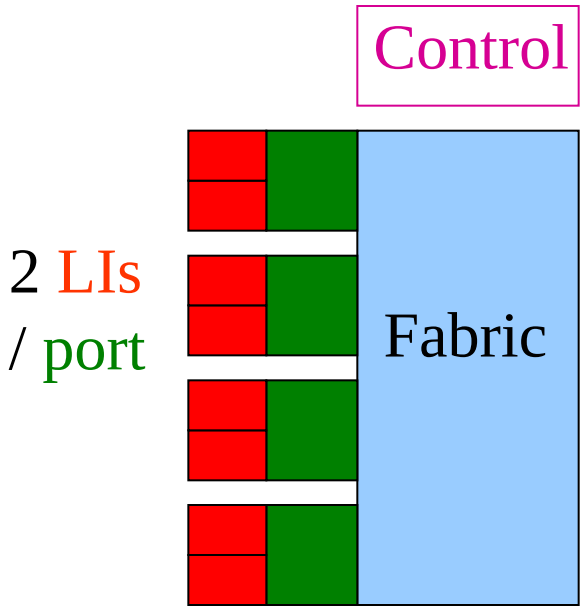
Port processors [0F> (input & output, may be different)

Switching fabric [JJ> (internal to switch)

- We'll consider these in depth shortly.

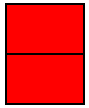
Synchronisation note: Fabric usually has own clock, and port processors are synchronized to it (e.g. with buffering). Some fabrics include internal buffering.

Control processor [GR>



Cisco Catalyst 4006

Line interfaces



Provided through a Line Interface Card (LIC)

Functions: Physical layer:

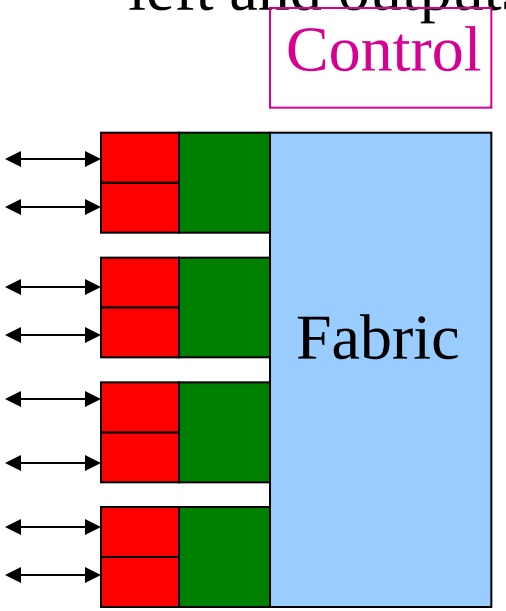
- Optoelectronic conversion
- Analog to digital; line coding
- Extract timing from received signal
- Serial-parallel

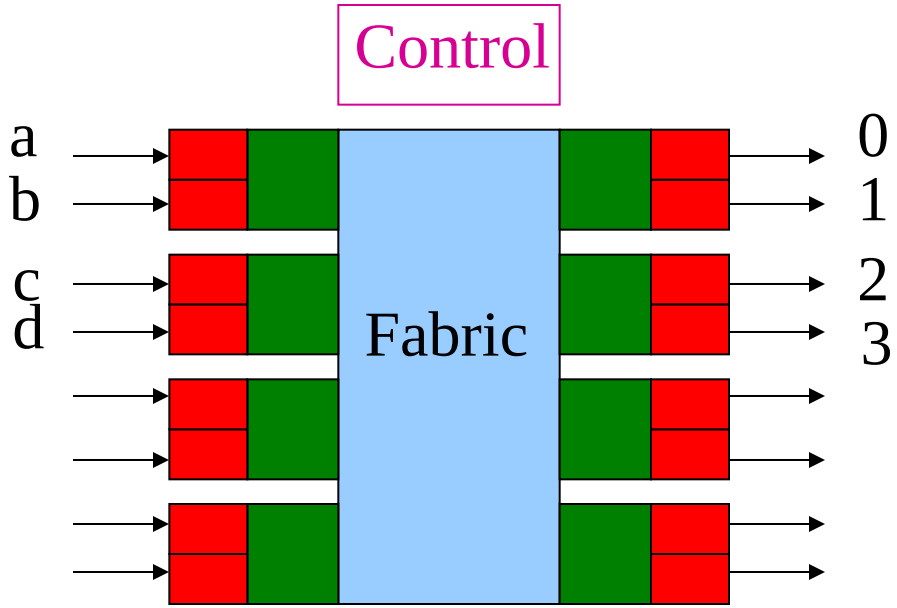
Line interfaces: number ($P^{\dagger}$)

Same number of input and output interfaces / bidirectional interfaces

Exception: Arrays in Clos fabrics [JX>

Often show switch with unidirectional interfaces; inputs on left and outputs on right:





† P for port – see "Interfaces vs ports" [32>

Line interfaces: Speed(s)

Terminology: In "wire-speed" switches, the line interfaces are the bottleneck (not port processing, control, or fabric). Cisco sometimes† uses the term "media-rate" – also covers non-wired media, such as fibre and wireless

Interfaces usually all have the same speed

may want different speeds (sometimes called "asymmetrical switching" [V0>)
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 <DT] @ 100Mb/s.

achieve through either:

- variable number of LIs per port processor. e.g. port might have a capacity of 10Gb/s. Provide the option of connecting the port to either 10×1Gb/s line interfaces or 1×10Gb/s interface.
- inverse multiplexing (defined shortly [C3>)

† And sometimes doesn't, e.g. slide <1YE*]

Interfaces vs ports

Interfaces and ports are basically the same thing: hardware† points of access

- "**Interface**" often used to refer to
  - physical interface, e.g. "Line Interface Card", "WAN Interface Card (WIC)", "Network Interface Card (NIC)"
  - multiple points of access that share a switch fabric port
  - points of access, as perceived by the Internet Protocol (e.g. in a "router")
- "**Port**" often used to refer to
  - Ethernet (e.g. a home router may have 2 interfaces, one of the interfaces may have 4 switched Ethernet ports)
  - Bridge [0W> ports

At the transport layer (e.g. TCP/UDP), ports identify software processes, and are different from switch ports which are physical entities.

Port processors

Port processors provide packet/frame-level processing after(input)/before(output) physical line interface.

Specific processors may provide a subset of these features.

Input port processor roles:

- **Validate packets**: check integrity, version #, length, etc. Send error messages (e.g. ICMP)
- **Packet classification** [16U>
  - **Determine required output** port using forwarding tables†
  - **Determine class** of service (affects buffering)
- **Packet processing:** --TTL/hop count, push/pop headers on/off for tunneling
- **Maintain stats** about usage (for empirical admission control [26>, security/billing, network management, audit).

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

Port processors (ctd)

Input & output port processor roles:

- Queue information; local switching (between LICs that share a port) though that may be unfair to non-local traffic
- Swap label <15]: @ input port for unicast (with other lookup),
  @ output port for multicast (differs by output) †
- Wrapper around fabric:
  - **Headers**: Add (input) or strip off (output) headers for use within the switch:
    - self-routing information to reach that port†
    - sequence numbers to protect against mis-sequencing within the switch fabric
  - **Packet size adjustment**: Segmentation (at input) and reassembly (at output) to match variable-length packets outside switch to fixed-length packets within
- Policing, shaping and scheduling [WH>

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

Control processor

Contains routing tables, handles signaling, switch management

Sometimes treated (by the fabric) as another node connected to a switch port

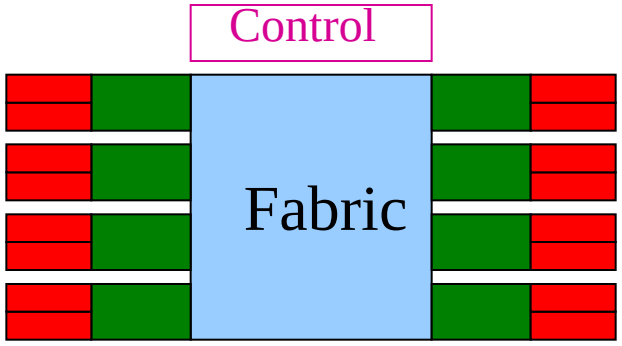
so that traffic to it can be switched through the fabric like normal traffic (possibly with higher priority)

e.g. a switch with 8 bidirectional ports could connect the control processor to one port, leaving 7 for external interconnection. Switch can be controlled (for management, to setup/release connections etc) via 7 interconnection ports by sending 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

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

3. Buffer management policies
(at inputs and outputs)

5. Later: Connecting switches together

Outline

Generic switch architecture

Criteria for evaluating switches

- **Performance**
- **Blocking**
- **Cost**

Switch classification: By structure of implementation

Time-division switches

Switch performance

... is often analysed mathematically, but not in this course:

× Complex

× Highly sensitive to workload = traffic characteristics (see next slide [CH>)

× Switching affects traffic characteristics = workload for downstream switches <<UY]

We'll consider *qualitatively* the performance impact of unicast and multicast traffic.

Traffic characteristics that can affect switch performance

- Burstiness of arrivals <VY]
- Focus of traffic (e.g. heavy load to server on one port)
- Directivity (unicast, multicast, etc)
- Packet length (e.g. packet processing vs transmission capacity as bottlenecks)

(mathematical analysis often requires simplistic models of characteristics, e.g. Poissonian arrivals of packets with exponentially distributed length)

=> 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.

For examples of scalar performance numbers see
http://web.archive.org/web/20100811223337/http://www.cisco.com/web/partners/tools/quickreference/index.html

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:

- **Discard** the request (sometimes called "clearing" blocked calls)
- **Queue** the request, e.g. buffering.
  - A later lecture [8P> 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
- **Schedule**† the request for a time when resources *are* available e.g. known length <74]
  - In queueing, info is buffered @ switch. With scheduling, info buffered @ *sender* which will later transmit when switch is ready.
- **Try again**: Require the initiator to make the request again, e.g. Recirculation [MZ>
  - With scheduling, sender/initiator learns when access will be possible, whereas when trying again, time of future success is unknown

† Scheduling here is of connections, c.f. later lecture on scheduling packets. e.g. the Session Initiation Protocol (SIP) allows a server to tell a client to "Retry-After" some time.

Blocking: Switch types

Switch types (in terms of blocking):

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

  e.g. Crossbar

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

  e.g. Banyan

See also:

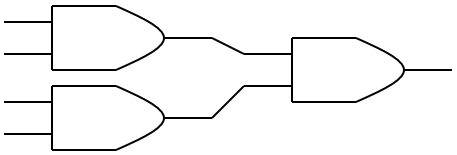
- **(Intermediate) Rearrangeable nonblocking:** Any desired connection can be established, possibly after rerouting existing connections.

  e.g. Clos

Switching cost criteria

- Crosspoint complexity (N_x): Number of gates. Affects:
  - Cost:
    - Historically (1950s) important: crosspoints were implemented as relays, costing a few $ ea.
    - Today:
      - Unimportant for electronic switches (VLSI enables billions of gates)
      - Important for some optical switches (MEMS allow limited mirrors)
  - Reliability: more crosspoints = more failure points
- Interconnect, fan-out, and logical path depth (e.g. 2 below)
- Network control complexity

…



MEMs = Micro Electro Mechanical Systems [E7>

See Section 13.9.1 of Varghese

Outline

Switch fabric classification

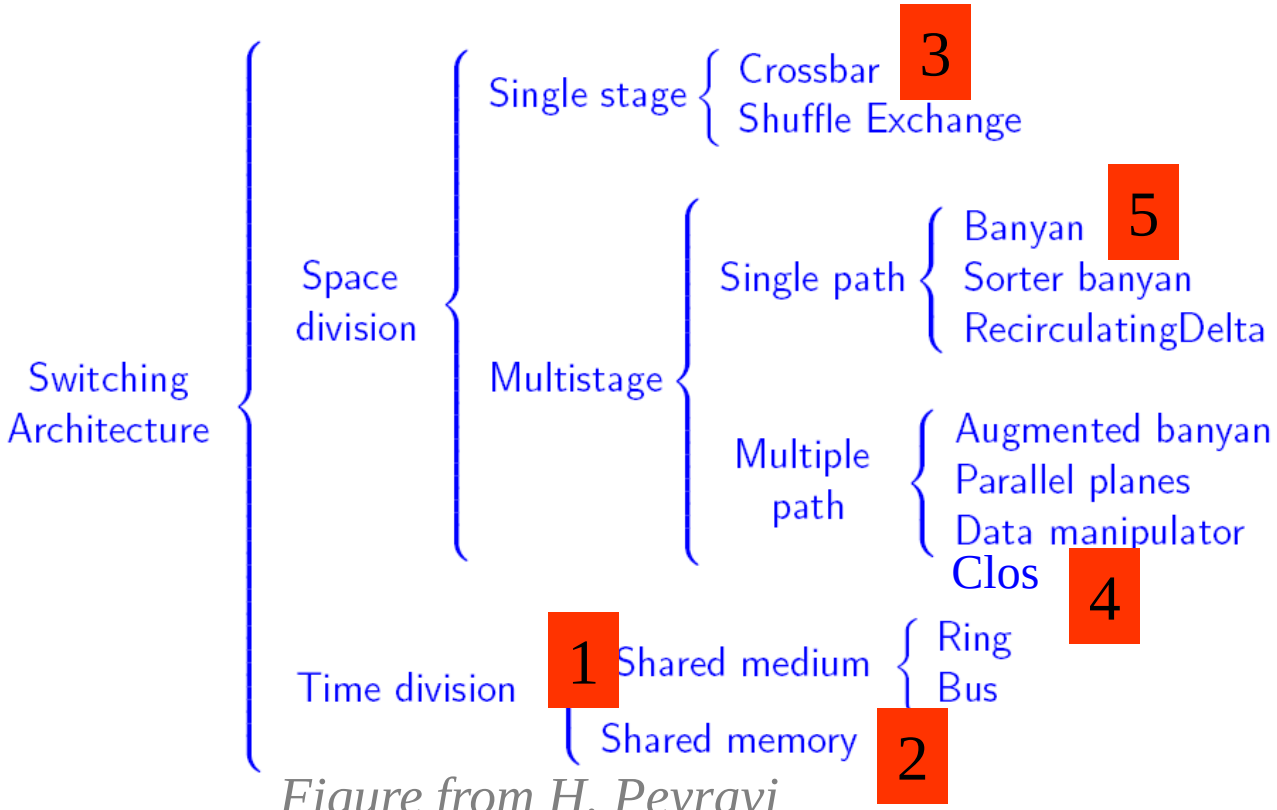


Figure from H. Peyravi

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

Outline

Generic switch architecture

Criteria for evaluating switches

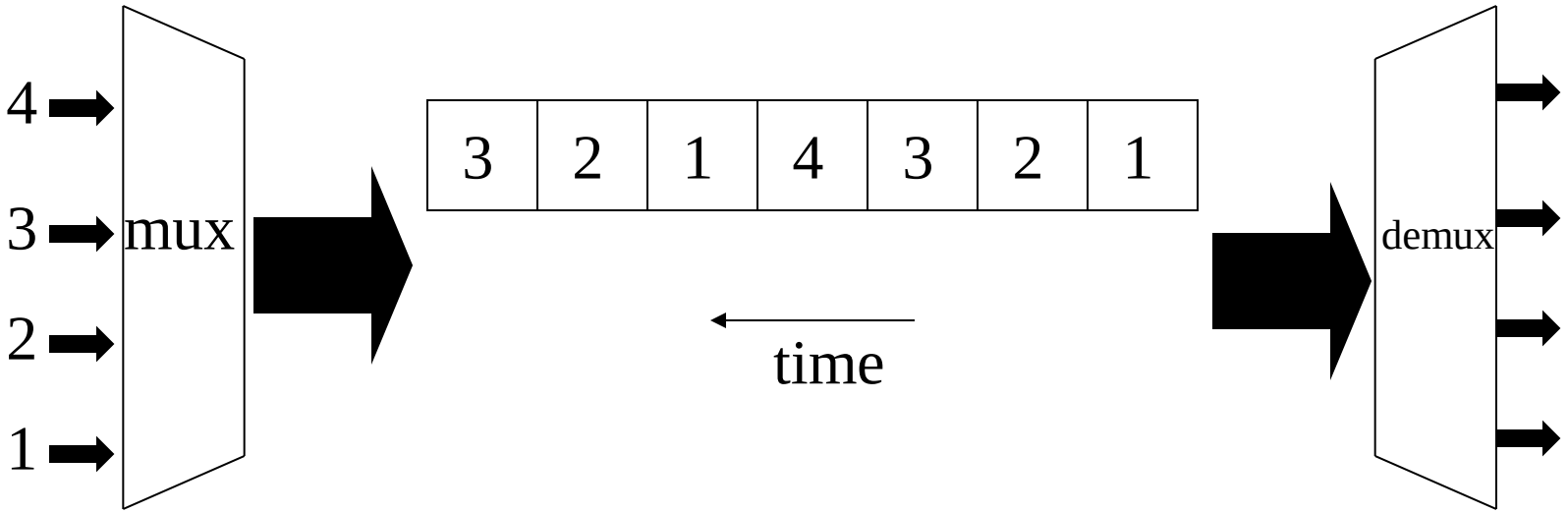
Switch classification: By structure of implementation

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**

Multiplexing and Demultiplexing

Multiplexing (muxing): "The combining of two or more information channels onto a common transmission medium." [ATIS]



[ATIS] http://www.atis.org/glossary/

Multiplexing methods

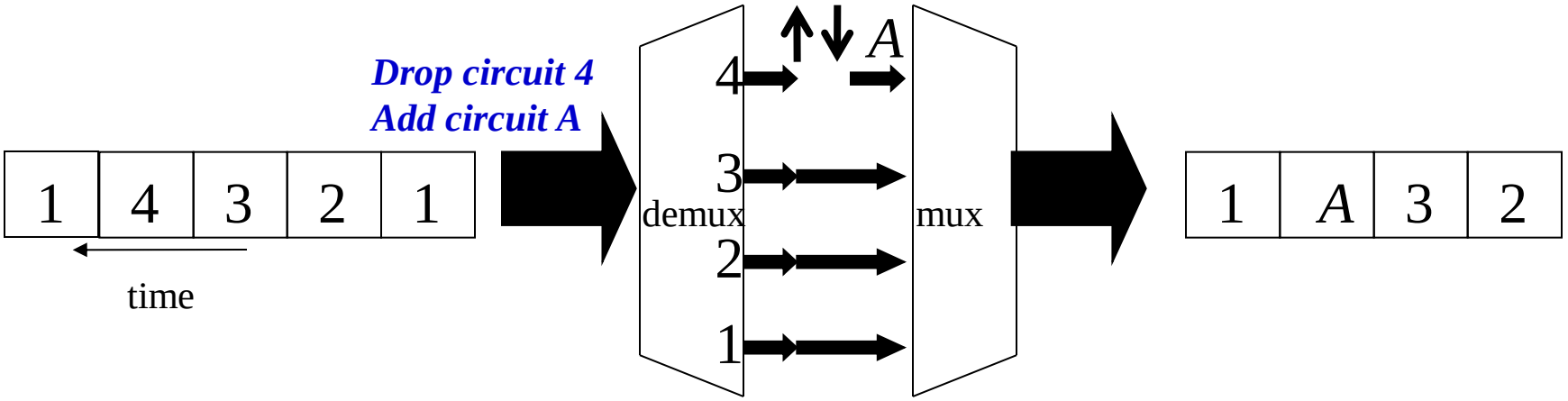
Time-division multiplexing: Different inputs are sent at different times. Alternatives: Frequency & Wavelength division muxing (FDM, WDM)

Synchronous multiplexing: A form of TDM in which the multiplexer alternates between different inputs in round-robin order (c.f. Asynchronous TDM aka ATM)

Add-Drop Multiplexers (ADMs)

Drop one (or some) multiplexed signal(s) from trunk

Add one or more replacement signals, e.g. TDM:



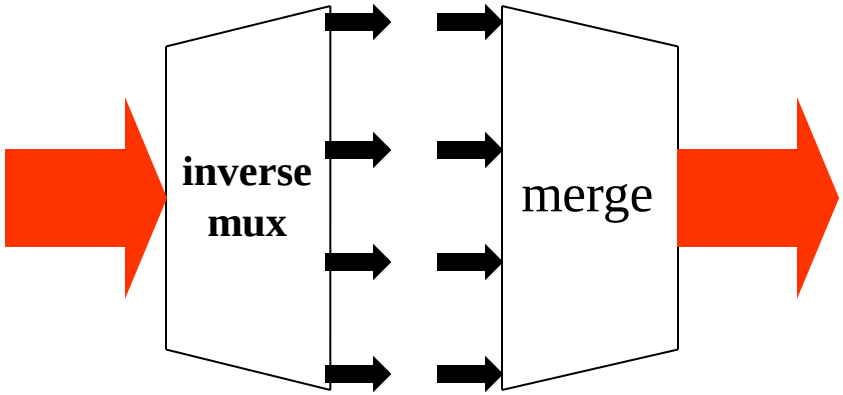
Popular in optical networks (OADM) [QU>, where adding/dropping a wavelength (FDM/WDM) is one of the few functions that is relatively simple to implement all-optically.

Inverse multiplexing†

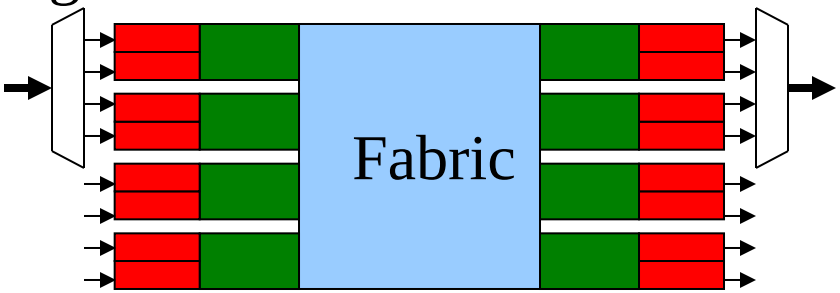
After demultiplexing, outputs usually diverge

Inverse multiplexing: multiple outputs follow same path & later rejoin

Need to deal with potential mis-sequencing.



e.g.



† aka "striping", "splitting" or "link aggregation", e.g. the IEEE 802.3ad standard.
See Chapter 9 of Seifert for details about link aggregation.

With demultiplexing, since flows diverge they usually (except when transit switching) must go in *specific* directions, whereas with inverse multiplexing they need only be spread amongst outputs.
=> lower chance of blocking when inverse multiplexing.

Inverse muxing: Applications

e.g. Construct high-speed interface from multiple lower-speed interfaces:

- 1×40Gb/s interface from 4×10Gb/s switch interfaces
- Interconnect routers through 64kb/s phone lines, with rate of connection (# of lines) varying according to demand.

"Etherchannel", "PortChannel" and "Dial-on-demand Routing (DDR)" are Cisco-proprietary implementations of inverse multiplexing.

The Link Aggregation song

A link is a link, that's my instinct,
And each of the links is distinct, I think.
But aggregate the links, and blink –
You'll see they work as one.

The frames source and sink, like any link,
But now you'll have bandwidth galore, I think.
And even with a failing link,
The load can shift around.

If statistics tell you that the network's running slow,
Use aggregated links to make the data really flow!

A link is a link, that's my instinct,
And each of the links is distinct, I think.
But aggregate the links, and blink –
You'll see they work as one.

From Seifert, p. 355

Outline

Time division switching

- **In time-division switches, all information passes via some single part of the switch at different times**.
  - c.f. space-division switches: spatially separated paths inside the switch can concurrently carry different info: different info flows through different parts, possibly at same time
- "some single part" may be
  - transmission medium
  - memory

× that part tends to form a bottleneck

Most commonly produced form of switch

(c.f. academic emphasis on space-division switching)

Shared transmission media switches

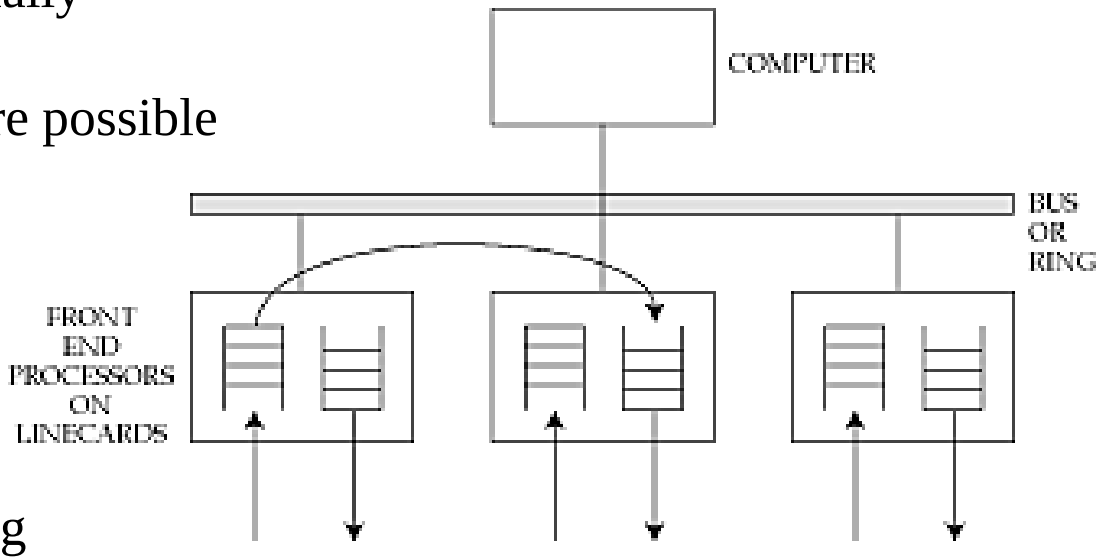
A single transmission medium is shared by all input and output ports,
i.e. broadcast-and-select <2Y] within the switch.

Access from input ports is usually
fixed round-robin TDM;
but asynchronous schemes are possible
(e.g. Ethernet or DQDB†)

Output ports select
packets destined to them,
based on address (async),
time of arrival (TDM), etc.

Limitation: high-speed filtering
(details in packet classification lecture)

Switch capacity often equals sum of input port capacities



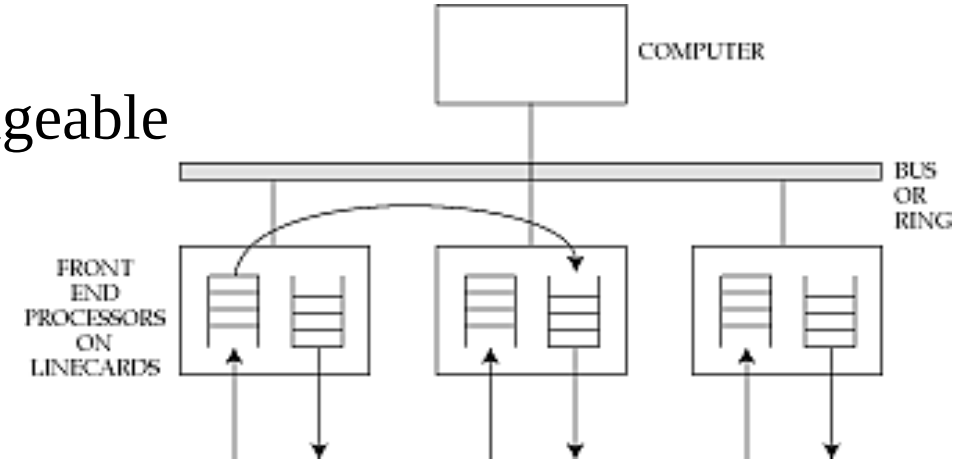
Keshav Fig. 8.12

Varieties of shared transmission media

Often a bus, but can transmit unidirectionally to reduce fanout: ring, dual ring/bus, folded bus

Dispersion is (relatively) manageable due to limited physical size

=>Wide buses to achieve high transmission capacity, e.g. 424b (53B=ATM cell)



Keshav Fig. 8.12

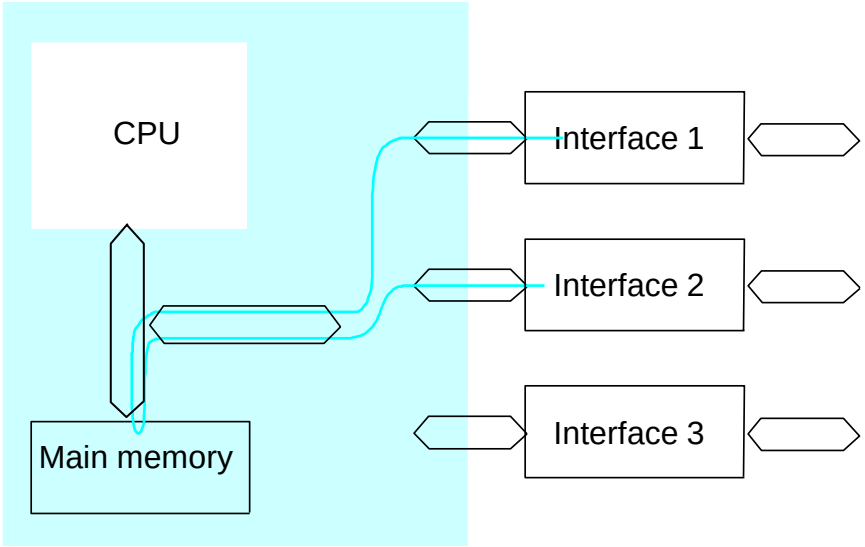
† DQDB = Distributed Queue Dual Bus – a MAC protocol designed for unidirectional buses

Implementing time-division switches on a general-purpose computer

General-purpose computer with an open bus like "Personal Computer"s of the 1990s

How:

- Use general-purpose NICs for line interfaces
- PC's bus & memory for switching fabric & buffering
  - Packet is read in from a port into memory
  - CPU decides where to forward the packet
  - CPU sends packet to appropriate outgoing NIC



The CPU here implements the controller and some port processing (rest by NICs), and bus implements the fabric.

Memory here only holds one packet at a time, while CPU processes it. Memories in shared memory switches can hold multiple packets.

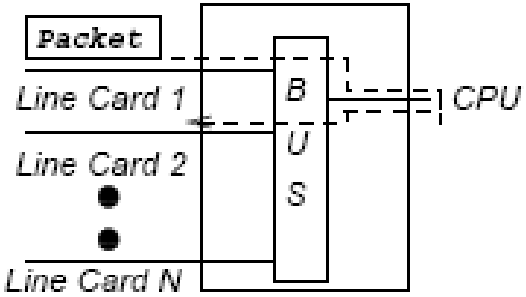
Disadvantages of PC switching: Centralisation

- High bus load: Each packet is sent across the bus twice (unless DMA and non-standard NICs);
  Really doesn't scale well with increasing numbers of ports
- 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.

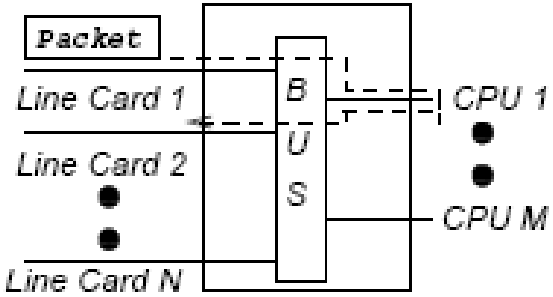
Disadvantages of PC switching: Wrong function

- Excessive NIC functionality: General-purpose NICs don't make the frame available immediately after header received (only after integrity check) => limits forwarding modes [FF>.
- General-purpose computer:
  - Unnecessary features (=> cost): video, disk, etc
  - Isn't optimised for communications processing
    - memories: Video RAM [3K> in wrong place, no CAMs [1DW> for address matching
    - interrupt processing overheads

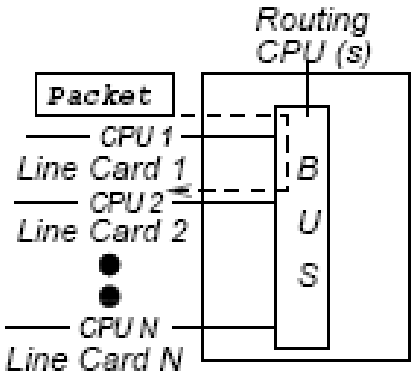
Router evolution



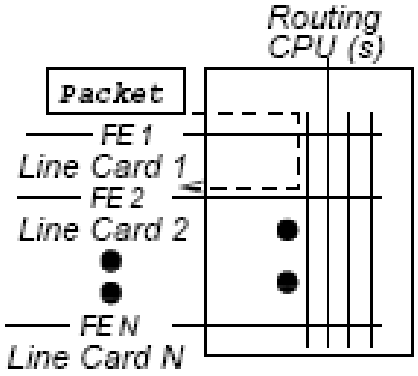
A) PALEOZOIC: BUS, SHARED CPU



B) PALEOLITHIC: BUS, SHARED CPUs



C) NEOLITHIC: BUS,
PER LINE CARD CPUs

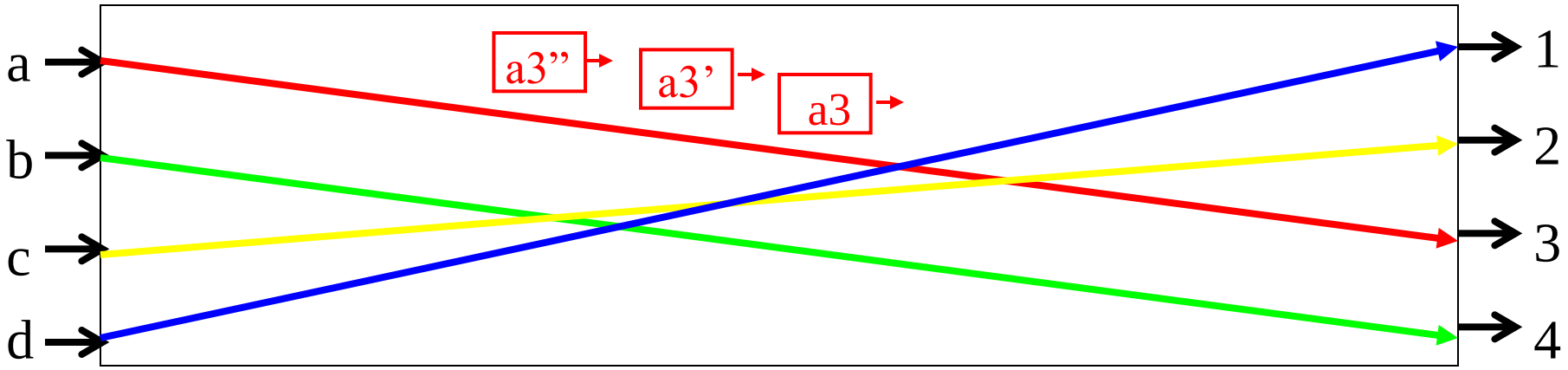


D) MODERN: CROSSBAR,
PER LINE CARD FORWARDING ENGINES

Fig. 13.3 of Varghese

Outline

Time-Slot-Interchange circuit switch



Information flows periodically through ports, e.g. fixed-size slots (e.g. PCM samples) arrive at instants t, t’, t”

“Slots” carry payload (information to be switched) and flow left-to-right through the figures

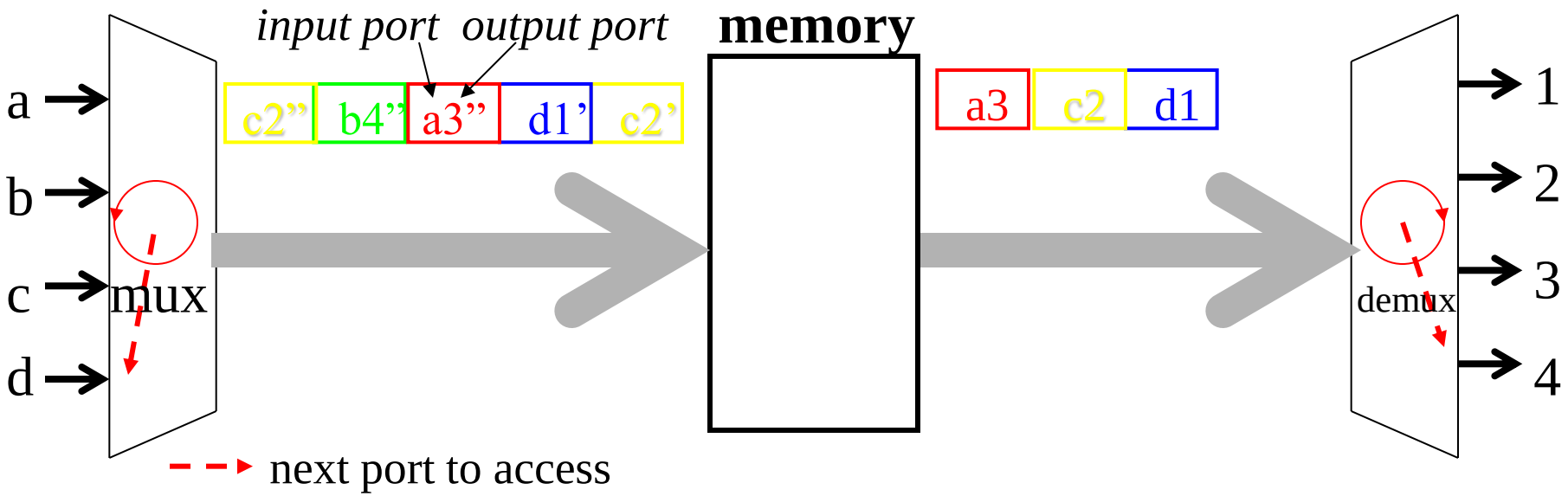
Circuit switched: Slots don’t contain addressing information

Identifiers are for illustrative purposes only:

colours: flow of information | rank (’, ” etc): order of arrival

letters: input port | numbers: output port

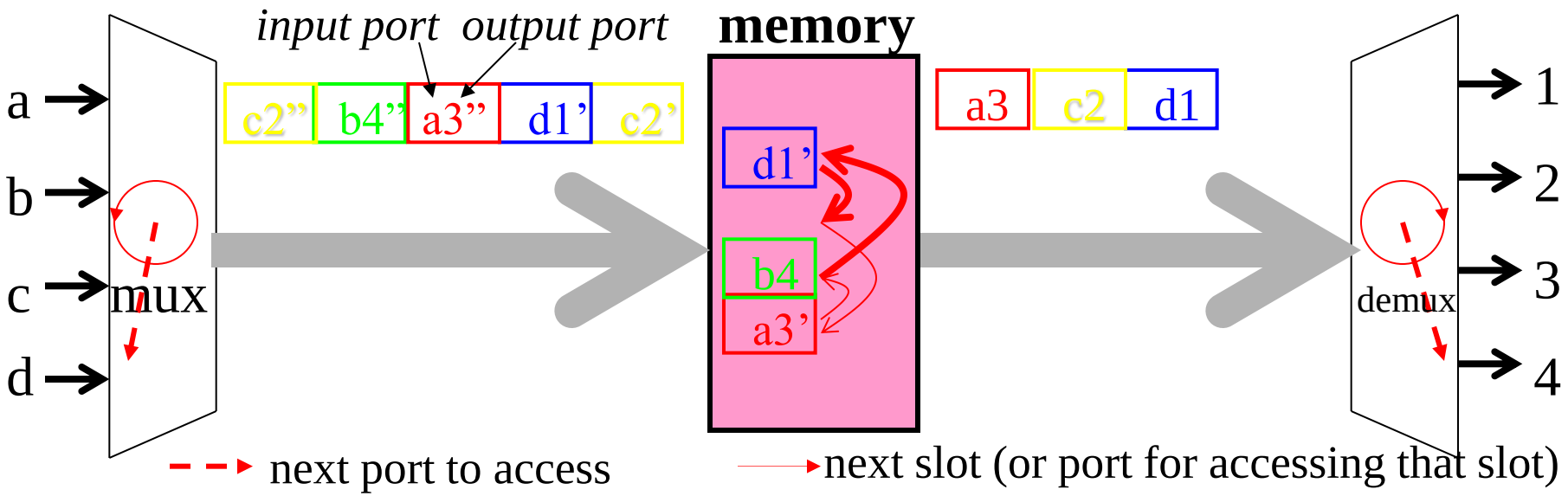
Time-Slot-Interchange circuit switch



Time-Slot-Interchange (TSI) switch:

1. Multiplexes inputs onto one medium (e.g. write port of memory)
2. Memory interchanges slot order
3. Demultiplexes output of memory to output ports.

Time-Slot-Interchange circuit switch



- Slots change order as they pass through the memory
  - Incoming slots are written to memory
  - Outgoing slots are read out of the memory, *but in a different order*
- Preprogrammed (before data arrives) order of linked list determines switching pattern, e.g.
  - every 4th slot (a3, a3’, a3”) is destined to port 3
  - the slot after that (b4, b4’, b4”) is destined to port 4

Notes about TSI circuit switch

Used in old PBXs & new optical switches [79>.

Scalability:

× Speed: Can multiplex multiple physical inputs to TSI input, but memory access speed (shared medium) increases in proportion to aggregate input access speed.

√ Size: Memory size is proportional to number of inputs (if allowing switching between arbitrary sets of ports)

This circuit switch *does* have buffering (albeit small, since arrivals are deterministic), i.e. it is not just packet switches that have buffers.

Outline

Shared memory packet switches

Memory provides a queue (e.g. linked lists) for each output port

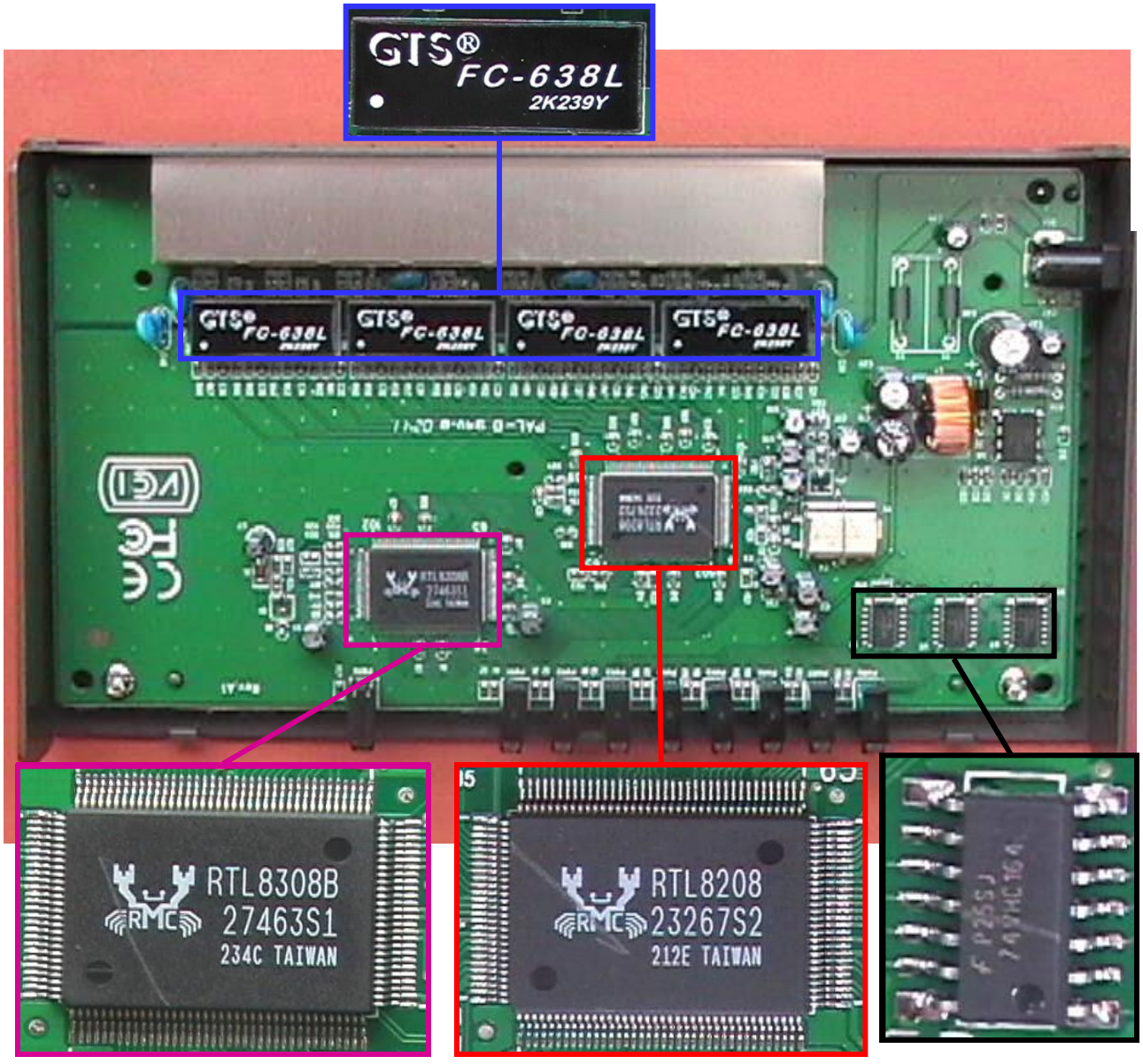
(Linked lists are useful in other switches to implement scheduling disciplines, e.g. fair queuing [WH>)

- Incoming traffic is appended to the end of the list for the required output port
- Sharing memory reduces overall memory size:
  - √ Allows statistical gain: Busy port can use memory not being used by an idle port
  - √ Helps multicasting: Stored once in memory for multiple outputs (multiple links to payload)

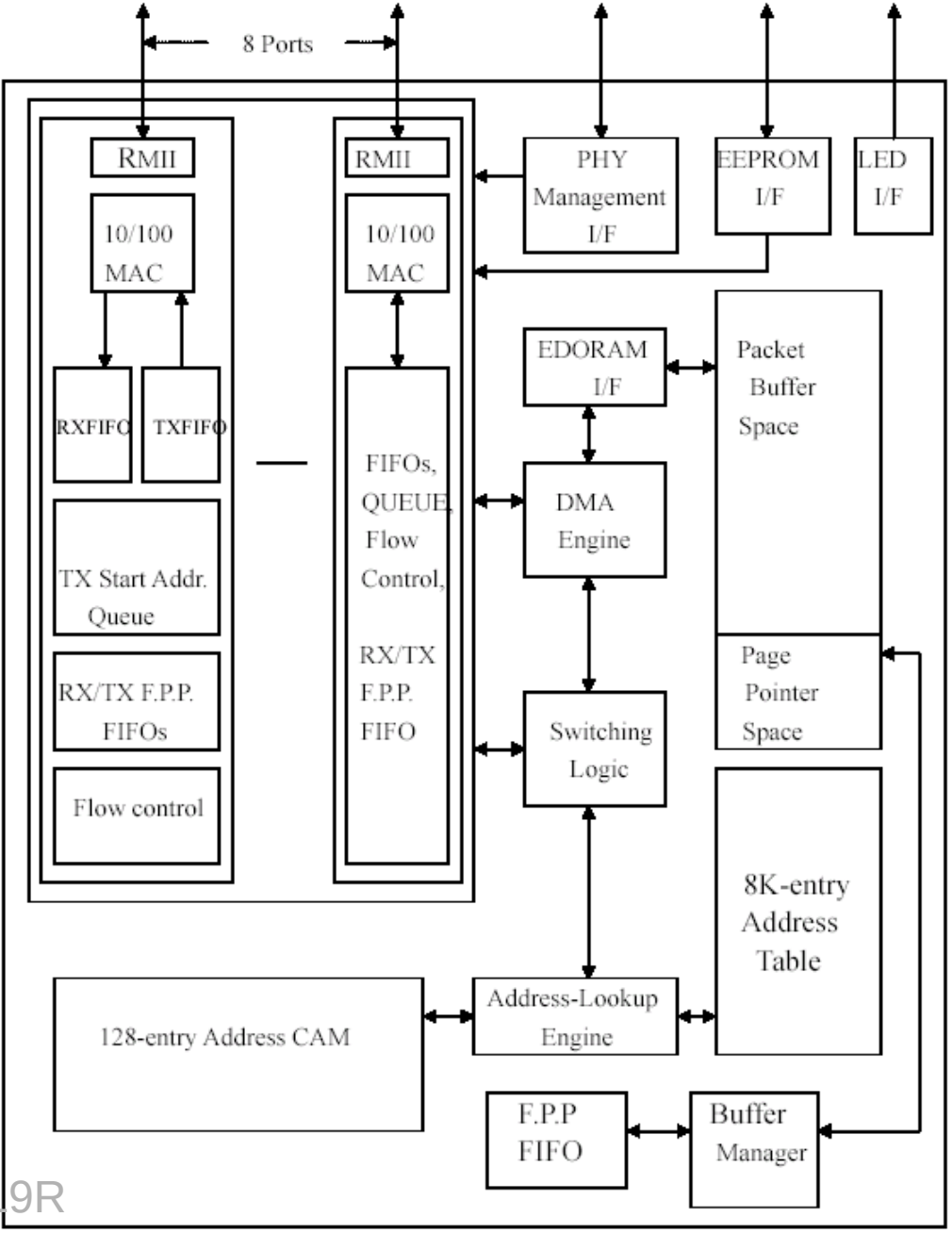
Example of a shared memory switch D-link DES-1008D 8-Port Switch

Sales blurb:

- AUD$50 approx
- unmanaged
- 8 port, 10/100Mb/s Ethernet (NWay auto-negotiated). Full or half duplex. RJ-45 connectors.
- LEDs to indicate port activity (power, speed, collision)



RTL8308 switch + glue:
RTL8208 transceiver
FC-638L transformers
74VHC164 8b SIPO



RTL8308

2MiB DRAM
32Ki x 64b @ 50MHz
256B pages

Store-and-forward + cut-through operation

"packet" classification:

1. Addresses are hashed to index the 8Ki lookup table.
2. Hash bash handled by storing colliding addresses in CAM

Outline

Multicasting in shared media switches

Readily supported since all output ports are naturally exposed to the traffic.

- For shared memory, multicasting requires
  - multiple reads from memory, reducing *throughput* in proportion to number of members of multicast group.
  - no change to the *size* of the buffer.
- For shared transmission, traffic will likely† propagate past all output ports in any case, so *multicast can be switched as fast as unicast*.

† "likely": For a bidirectional bus. When the medium is unidirectional receivers could terminate the propagation, allowing other traffic to be transmitted concurrently over non-overlapping paths.

Time vs space-division switching

Time-division advantage: **Fabric cost**

e.g. adding one extra port may require that a space-division switch double its port capacity

Space-division advantage: ***Internal* port cost**

Time-division switches require the switch internals to speed up in proportion to the number of ports.

Port costs can dominate system costs, but ideally it is the *external* port costs, which are inevitable, that dominate.

Things to think about

- Critical thinking:
  - Time-division switches have a core bottleneck; consider how you might avoid that problem before we consider space-division switches next week.
- Engineering methods:
  - This lecture presented criteria for evaluating switches <1R]. Engineering design choices are usually made after a cost/benefit analysis according to such criteria.
  - "Open the hood" to explore inside devices to see how they are built (e.g. shared memory switch)
- Links to other areas:
  - This lecture linked switch design to computer architecture
- Independent learning:

For more on SDN, read A Purpose-built Global Network: Google's Move to SDN and/or watch https://www.youtube.com/watch?v=YHeyuD89n1Y