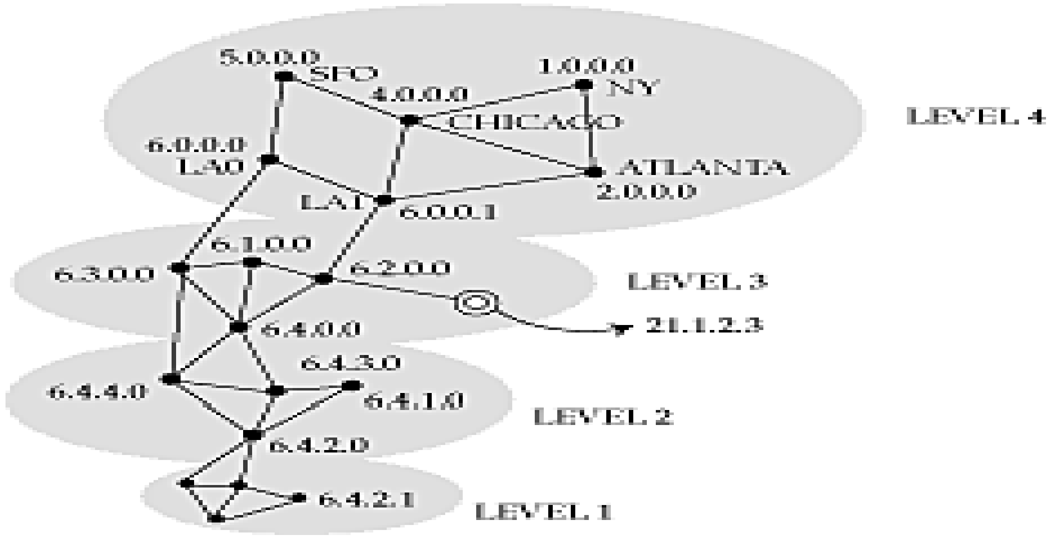
Switch and Router Architectures

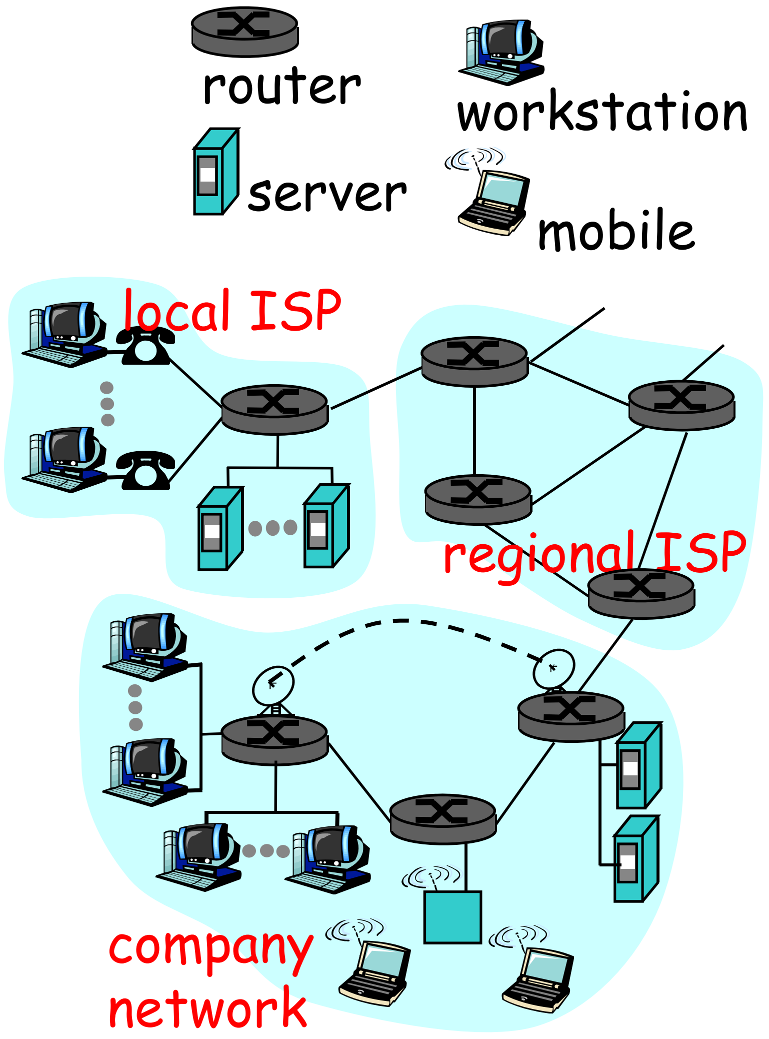
What is it all about?

- How do we move traffic from one part of the network to another?
- Connect end-systems to switches, and switches to each other by links
- Data arriving to an input port of a switch have to be moved to one or more of the output ports

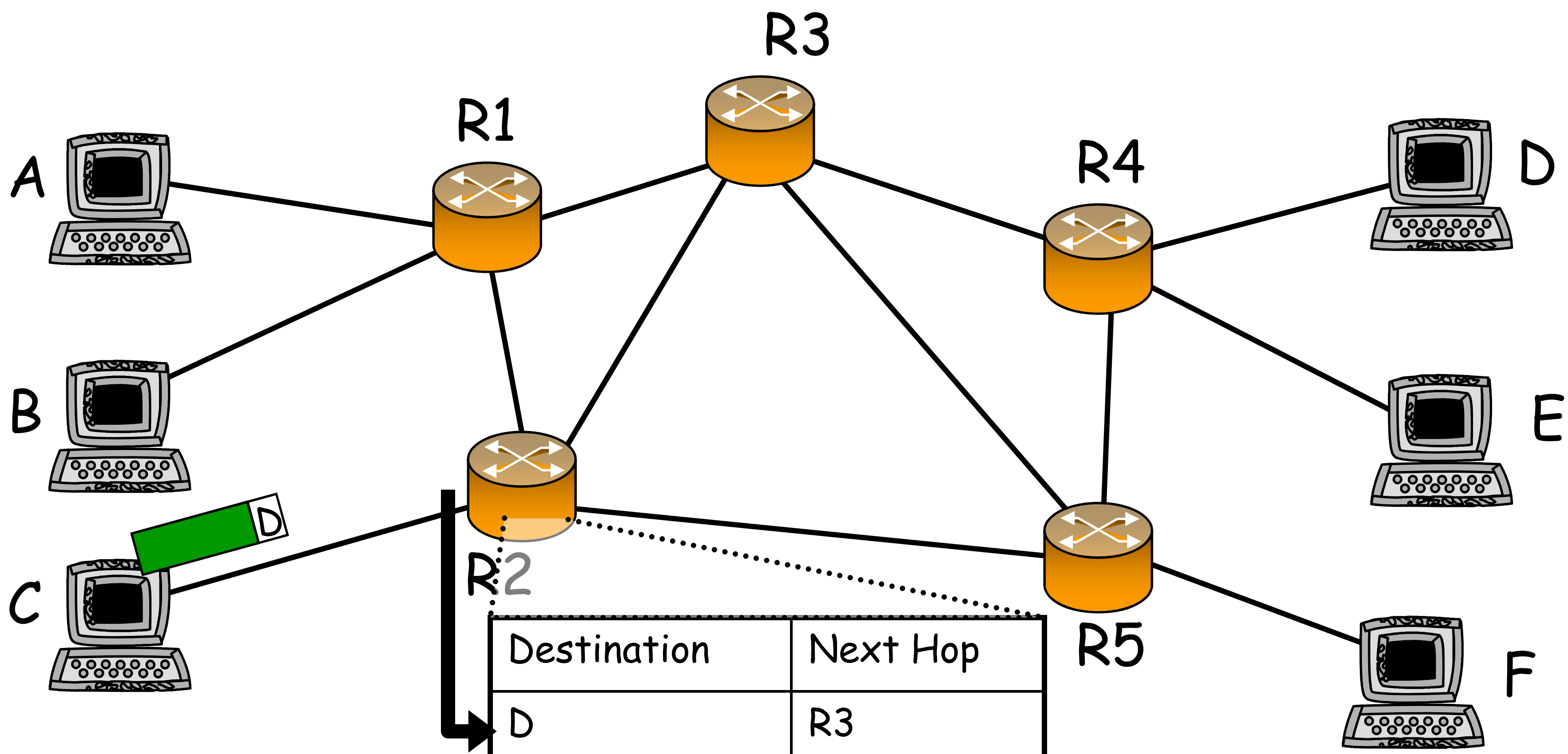


What's the Internet: "nuts and bolts" view

- *Internet:* "network of networks"
  - Any to any reachability
  - but loosely hierarchical
  - Routing protocols populate routing tables in the routers
- Traffic Aggregation
  - Through multiplexing and switching
  - Access Networks
  - Edge
  - Core

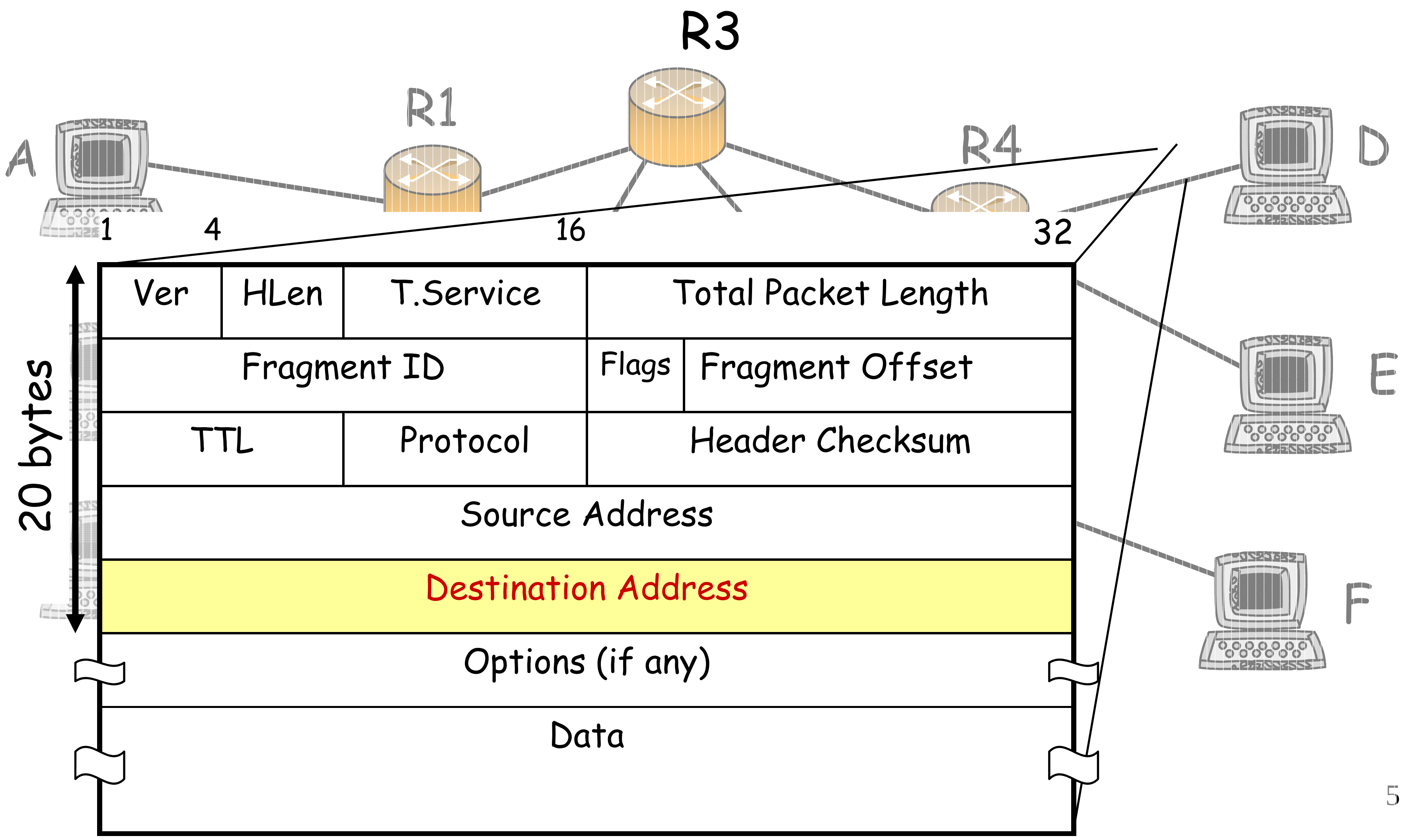


What is Routing?

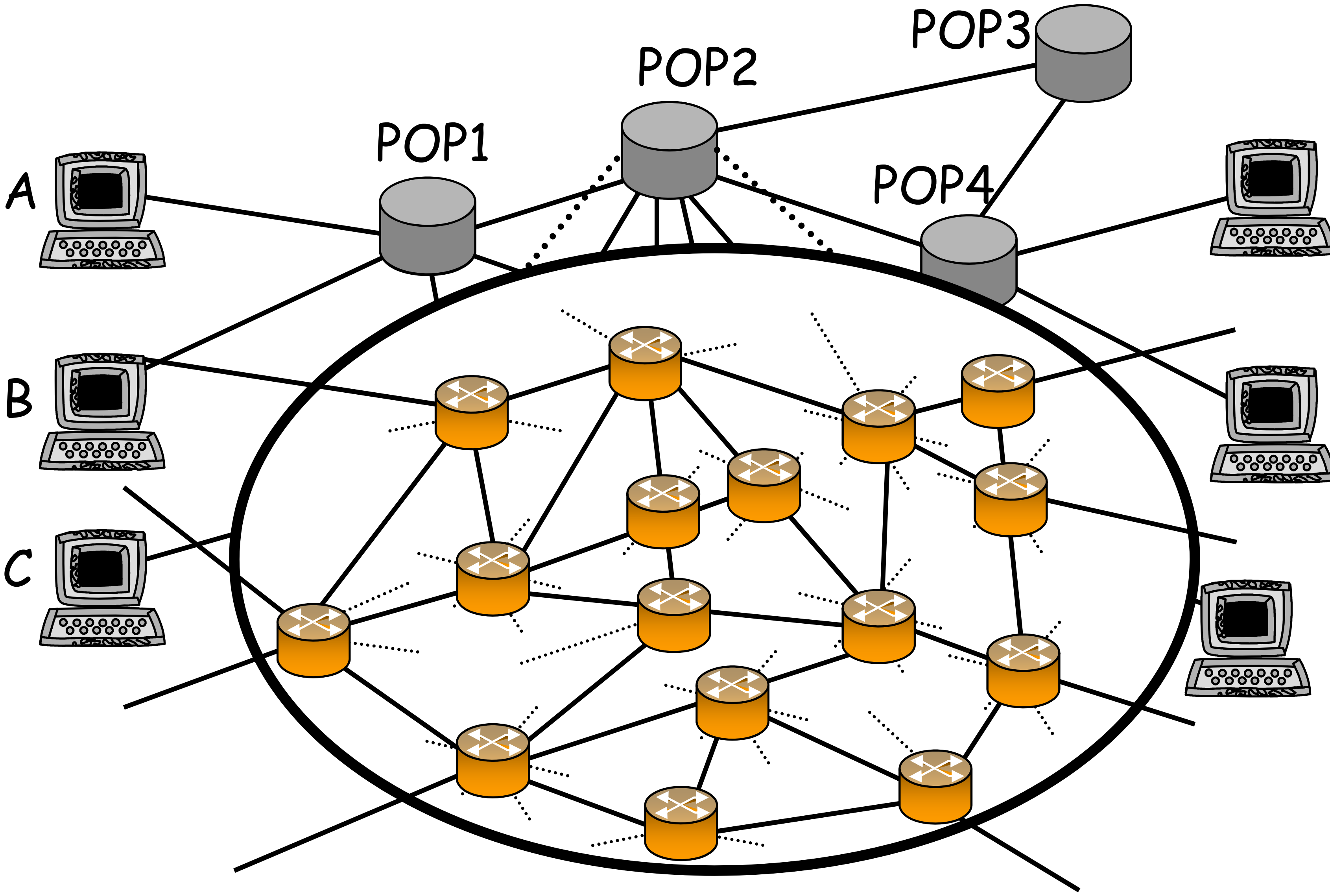


Destination	Next Hop
D	R3
E	R3
F	R5

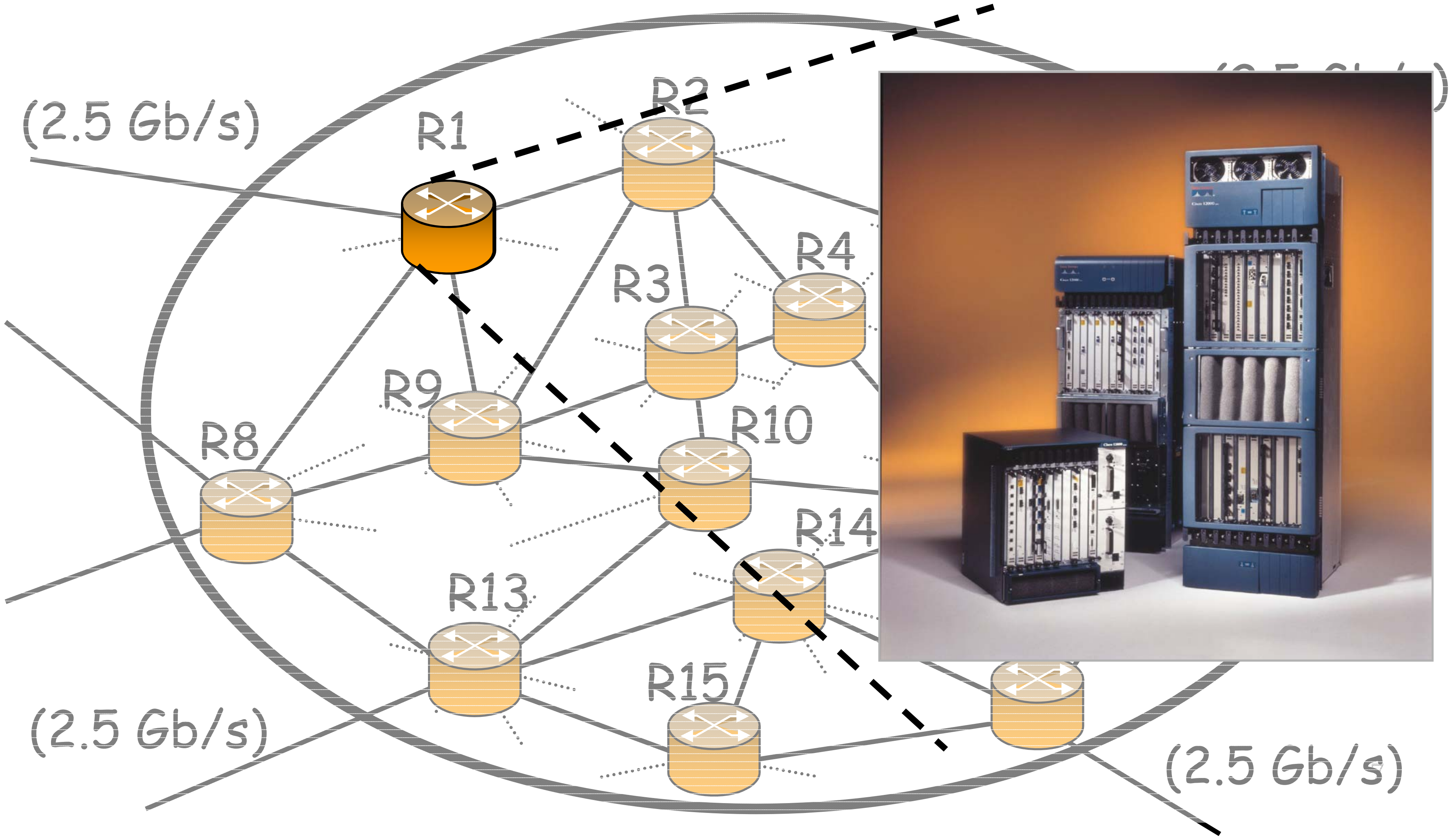
What is Routing?



Points of Presence (POPs)



Where High Performance Routers are Used



What a Router Looks Like

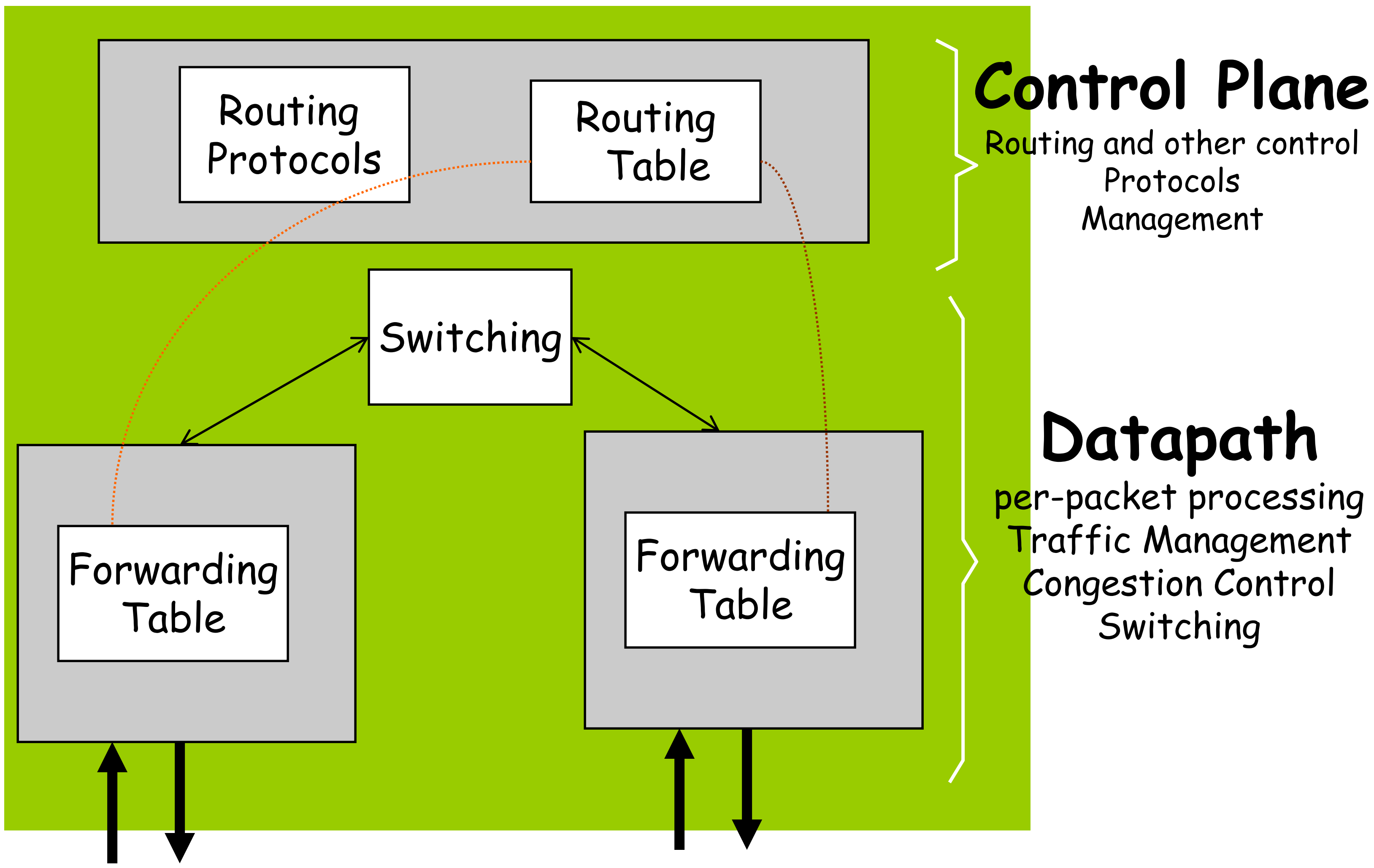
Cisco GSR 12416

Capacity: 160Gb/s
Power: 4.2kW

Juniper M160

Capacity: 80Gb/s
Power: 2.6kW

Basic Architectural Components of an IP Router



Types of switching elements

- Telephone switches
  - switch samples (8 bits)
- Datagram routers
  - route datagrams (variable length 64 bytes minimum)
- ATM (Asynchronous Transfer Mode) switches
  - switch ATM cells (constant length packets = 53 bytes = 5 bytes header + 48 bytes payload)
- MPLS switches
  - switch MPLS packets (variable length)
  - modified routers or ATM switches

What's the difference between routing and switching??

Routing and Switching

- Routing
  - Packet forwarding based on routing tables (established through routing protocols)
  - Longest Prefix Match lookup
  - datagram switching (no circuit setup)

- Switching
  - Pre-establish a circuit (physical or virtual) for communication
  - Packet forwarding is based on cross-connect tables (established through call setup procedures)
  - Uses physical or logical (virtual) circuit identifier (VCI)

Equipment Characteristics

- Switching Fabric Capacity
  - e.g., 1Gb, 10Gb, 320G, 5T
- Number of Interfaces (or ports)
  - 2, 4, 8, 16, 32, 64, 128
- Types of Interfaces (or ports)
  - Ethernet, T1, DS3, OC3, OC48, OC192
- Redundancy
  - Fabric, Port and Power Supply redundancy
- Control Plane (in-band or out-of-band)
  - Protocols supported
  - Management (Command Line Interface CLI, Web based, SNMP)

Classification

- Packet vs. Circuit switches
  - packets have headers (self-routing info) and samples don't
- Connectionless vs. connection oriented
  - connection oriented switches need a call setup
  - setup is handled in *control plane* by *switch controller* using *signaling protocols*
  - connectionless switches deal with *self-contained* datagrams

	Connectionless (router)	*Connection-oriented (switching system)*
Packet switch	Internet router	ATM switching system MPLS Switch
Circuit switch		Telephone switching system

Other switching element functions

- Participate in routing algorithms
    - to build routing tables
- Resolve contention for output trunks
    - scheduling
- Admission control
    - to guarantee resources to certain streams
- We'll discuss these later
- Here we focus on pure data movement (data path)

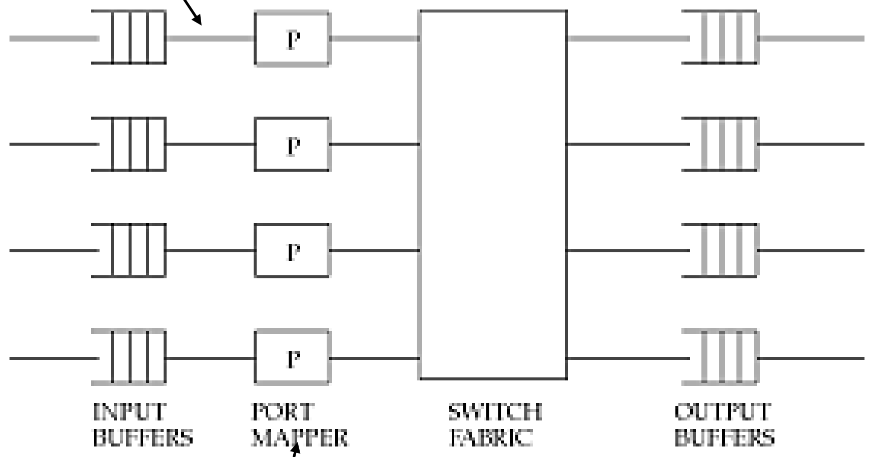
Requirements

- Capacity of switch is the maximum rate at which it can move information, *assuming all data paths are simultaneously active (e.g, 32 ports each at 10G=320G)*
- Primary goal: maximize capacity
  - subject to cost and reliability constraints
- Circuit switch must reject calls if it 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 (why??)

A generic switch

internal links can be serial links or buses

routes the packet to proper output



Ingress

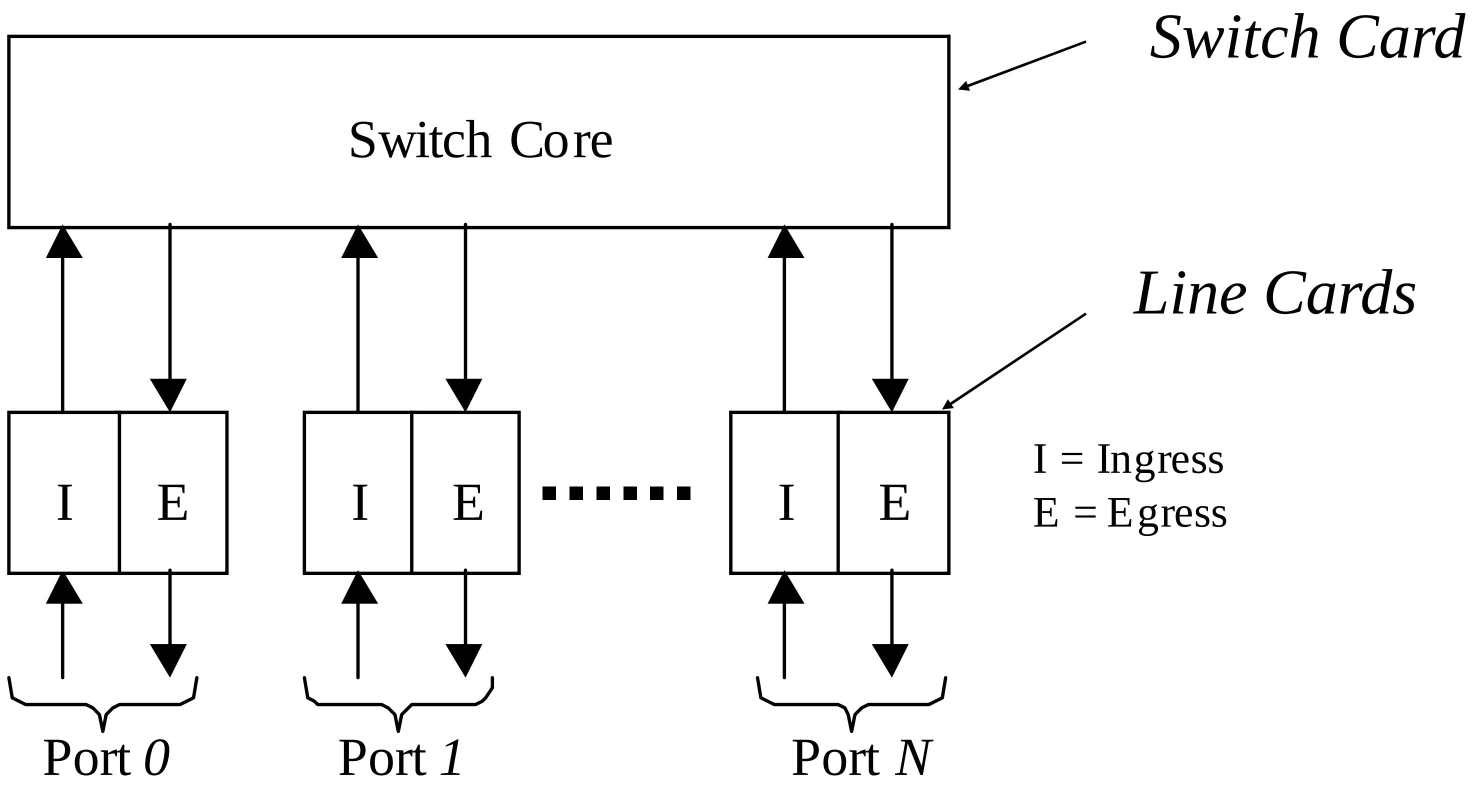
Egress

read header for destination or VCI, index to forwarding table for output port (used only in packet switches)

Ingress, Egress Linecards will host Framing, Traffic Management functions; Not all switches may have all components

Generic Switch - Folded Diagram

- Ports and links are generally bi-directional



Outline

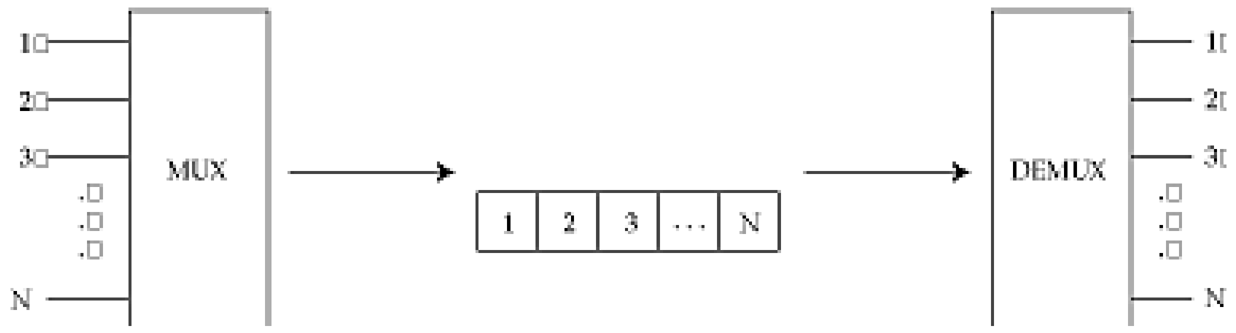
- Circuit switching
- Packet switching
  - Switch generations
  - Switch fabrics
  - Buffer placement
  - Multicast switches

Circuit switching

- Moving 8-bit samples from an input port to an output port
- Recall that samples have no headers
- Destination of sample depends on *time* at which it arrives at the switch
  - actually, relative order within a *frame*
  - *once connection is setup, a time slot assigned, the sample always arrives in that slot*
  - *No other header are necessary*
- We'll first study something simpler than a switch: a multiplexor

Multiplexors and demultiplexors

- Most trunks time division multiplex voice samples
- At a central office, trunk is demultiplexed and distributed to active circuits
- Synchronous multiplexor
    - N input lines
    - Output runs N times as fast as input



More on multiplexing

- Demultiplexor
  - one input line and N outputs that run N times slower
  - samples are placed in output buffer in round robin order
- Neither multiplexor nor demultiplexor needs addressing information (why?)
- Can cascade multiplexors
  - need a standard
  - example: DS hierarchy in the US and Japan
    - DS0 = 64Kbps single voice circuit
    - T1/DS1 = 24 DS0 = 1.544Mbps
    - T3/DS3=28 T1 = 672 DS0

Inverse multiplexing

- Takes a high bit-rate stream and scatters it across multiple trunks
- At the other end, combines multiple streams
  - re-sequencing to accommodate variation in delays
- Allows high-speed virtual links using existing technology

A circuit switch

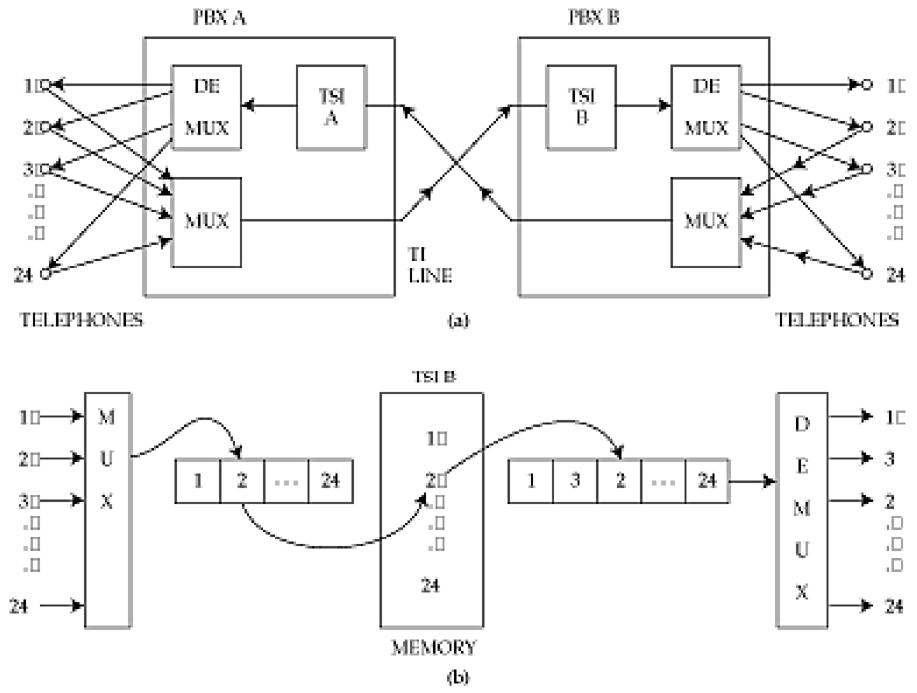
- A switch that can handle N calls has N logical inputs and N logical outputs
  - N up to 200,000
- In practice, input trunks are multiplexed
  - example: DS3 trunk carries 672 simultaneous calls
- Multiplexed trunks carry *frames* = set of samples
- Goal: extract samples from frame, and depending on 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

Call blocking

- Can't find a path from input to output
- Internal blocking
  - slot in output frame exists, but no path
  - Switches are classified as blocking or non-blocking
    - depends upon the architecture
    - a characteristic of the switch architecture
- Output blocking
  - no slot in output frame is available (no resources)
  - independent of switch internal blocking
    - occurs for either blocking or non-blocking switches
  - causes Head of Line (HOL) blocking

Time division switching

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

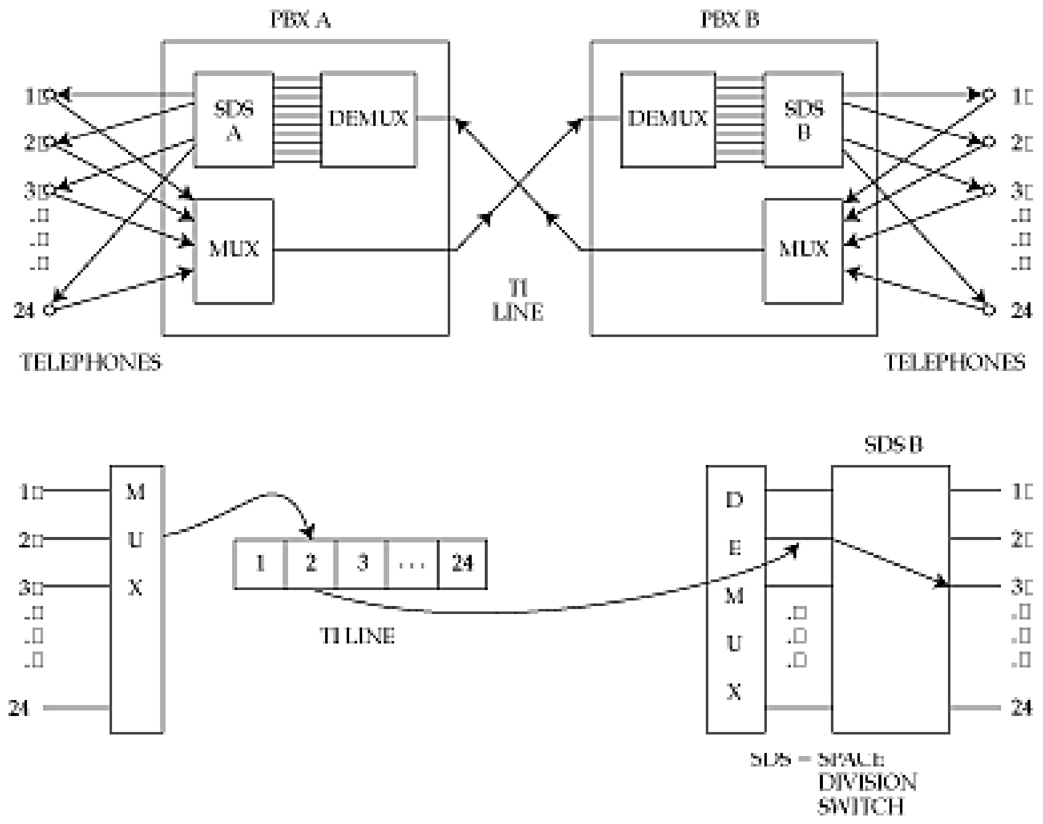


How large a TSI can we build?

- Limit is time taken to read and write to memory
- For 120,000 circuits
  - need to read and write memory (2 operations) once every 125 microseconds
    - Voice = 64Kbps
    - Sample = 8 bytes
    - Rate = 8000 samples / second
    - Time = 1/8000 = 125 microseconds per sample
  - each operation (read or write) takes around 0.5 ns for 120000 circuit TSI
    - impossible with current technology
    - With 40ns memories, we can build 120000/80 = 1500 Circuit TSI
- Need to look to other techniques

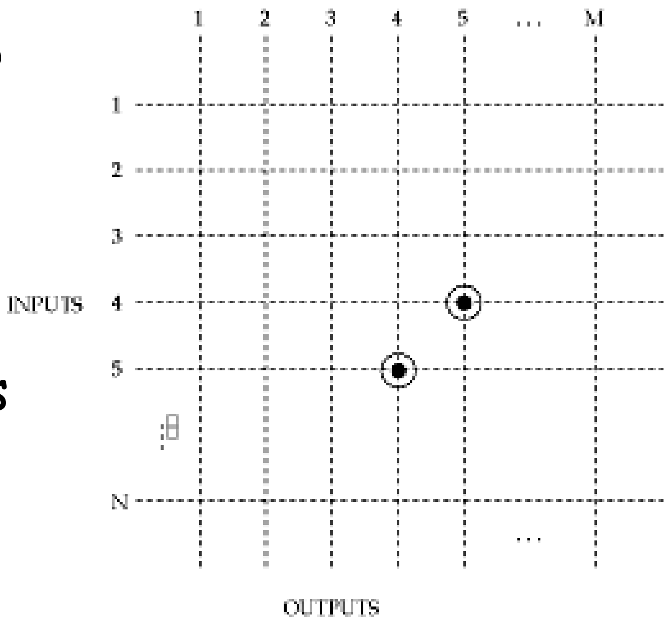
Space division switching

- Each sample takes a different path through the switch, depending on its destination



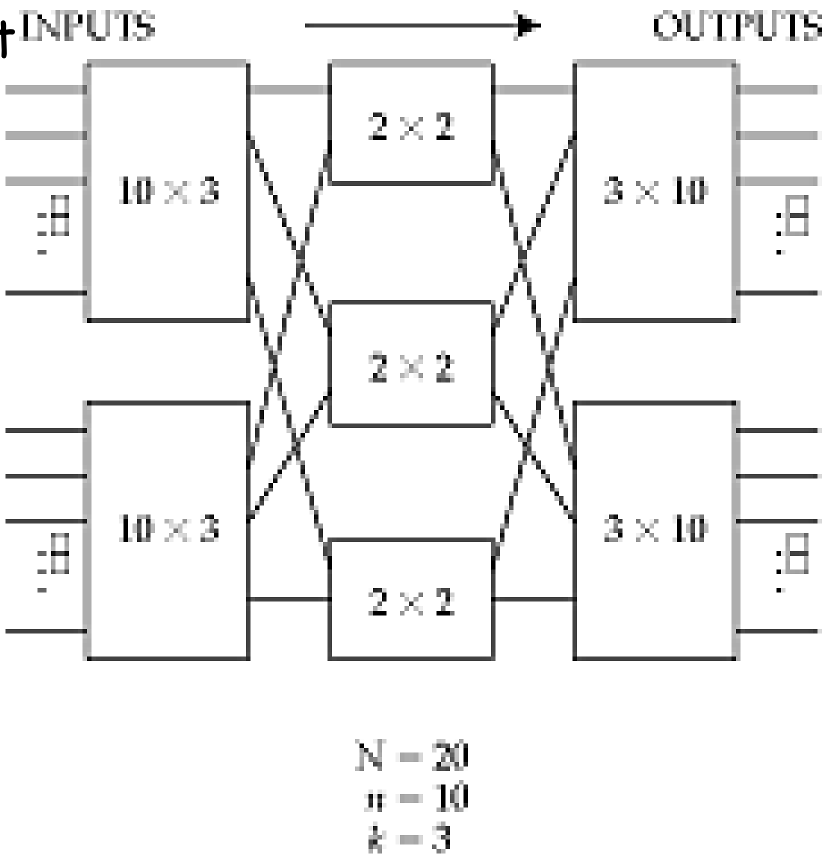
Crossbar

- Simplest possible space-division switch. *NxM* crossbar has *N* inputs and *M* outputs
- *Crosspoints* can be turned on or off (think of a design)
- Need a switching *schedule* (why and what frequency??)
  - Multiplex and non-multiplex signals
- Internally non-blocking (why?)
  - but needs N^2 crosspoints for NxN
  - time taken to set each crosspoint grows quadratically
  - vulnerable to single faults (why?)
- What about output blocking?

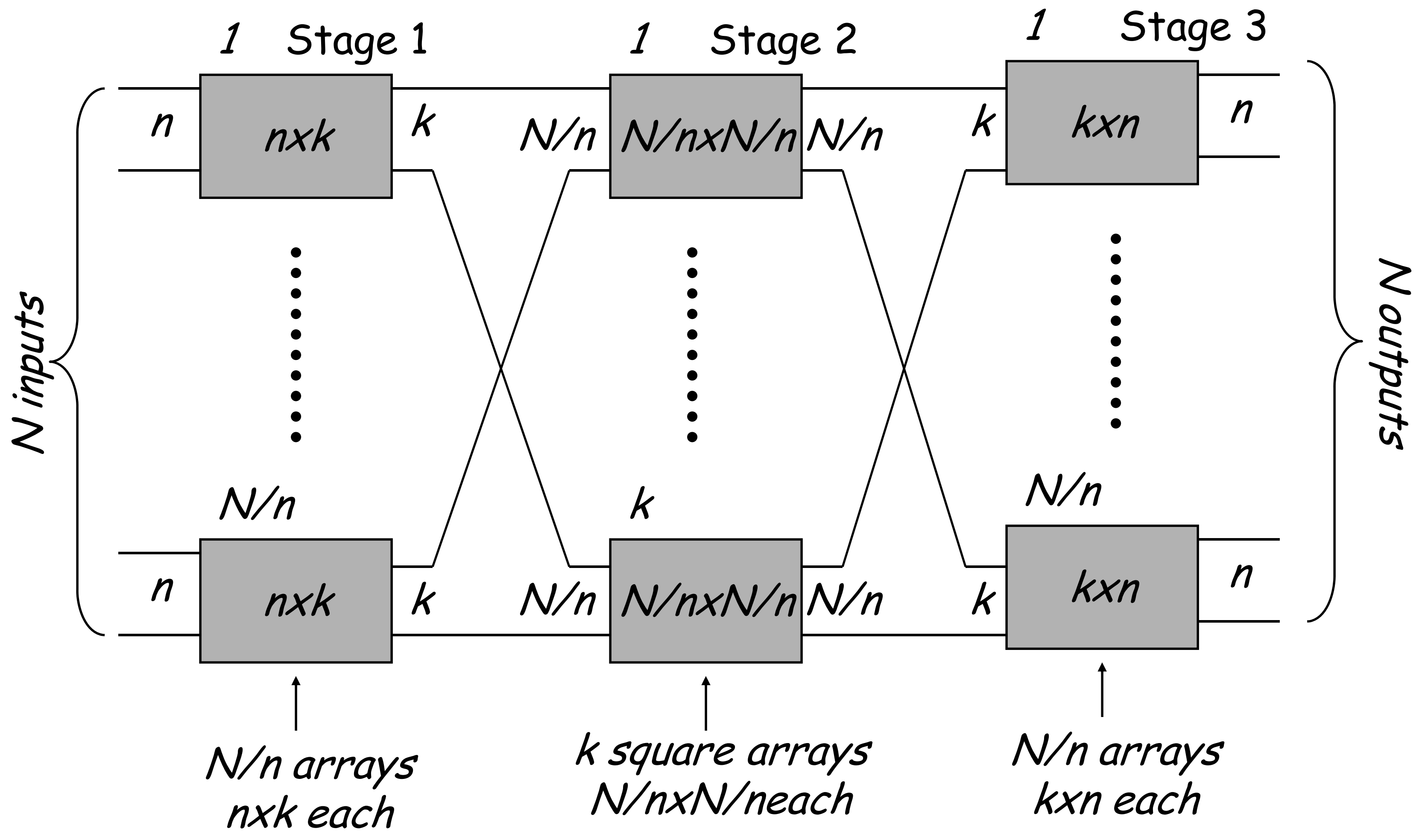


Multistage crossbar

- In a crossbar during each switching time only one crosspoint per row or column is active
- Can save crosspoints if a crosspoint switch can attach to more than one input line (why?)
- This is done in a multistage crossbar
- Inputs are broken into groups (e.g, 20 lines, 2 groups of 10 lines each)
- Multiple paths between inputs and output group share a centre stage switch
- Need to rearrange connections every switching time (switching schedule)



Multistage Switching



$$\boxed{\text{Total Number of Cross Points} = 2Nk+k(N/k)^2}$$

Multistage crossbar

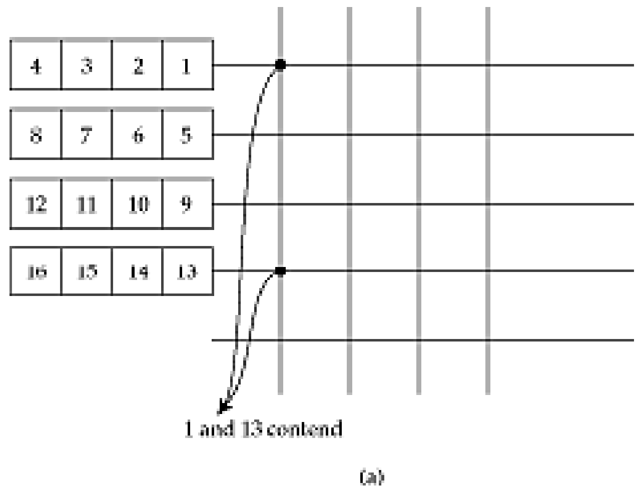
- First stage consists of *N/n* arrays of size *nxk each*
- Second stage consists of *k* arrays of size *N/n x N/n each*
- Third stage consists of *N/n* arrays of size *kxn each*
- *Can suffer internal blocking*
  - *unless sufficient number of second-level stages*
- Number of crosspoints < N^2
- Finding a path from input to output
  - switch controller needs to find a path at the time of call setup
  - uses path search algorithms, such as depth-first-search
  - the path is then stored in the switch schedule
- Scales better than crossbar, but still not too well
  - 120,000 call switch needs ~250 million crosspoints

Clos Network

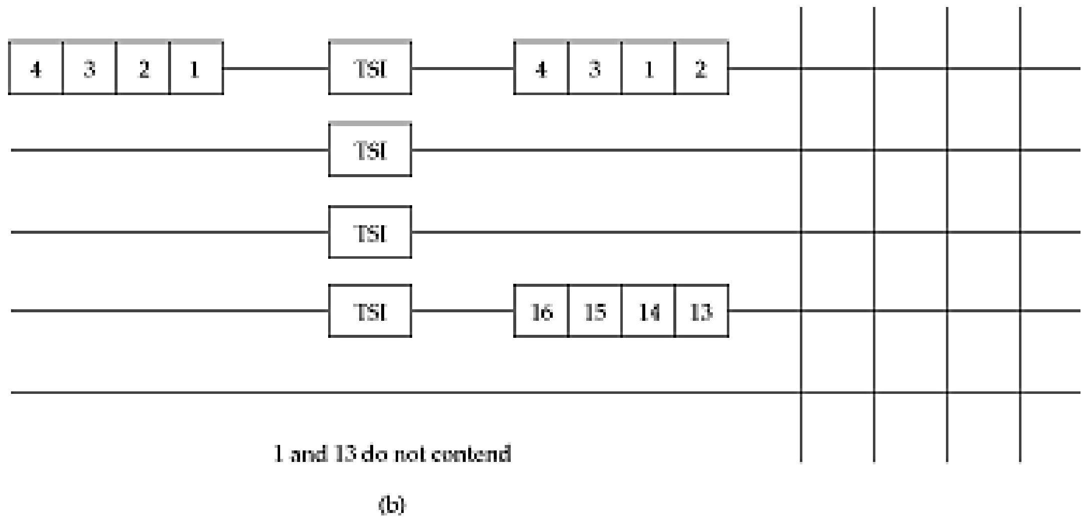
- How large should be *k* (# of center stages) for the switch to be internally non-blocking??
  - Clos [1953 paper] showed that if a switch controller is willing to *rearrange* existing connections when a new call is arrived, the condition is
    - *k≥n* (i.e., the number of center stages must be greater than the number of inputs in a group) (*k=2n-1)*
    - Also called *re-arrangably non-blocking switch*
    - In practice we cannot rearrange live calls (without breaking the circuit) - becomes complex (make before break)
  - Clos network of size *NxN* has $2N(2n-1)+(2n-1)x(N/n)^2$ cross points, way smaller than N^2

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
- Re-orders samples within an input line and switches them to different output if there is output blocking



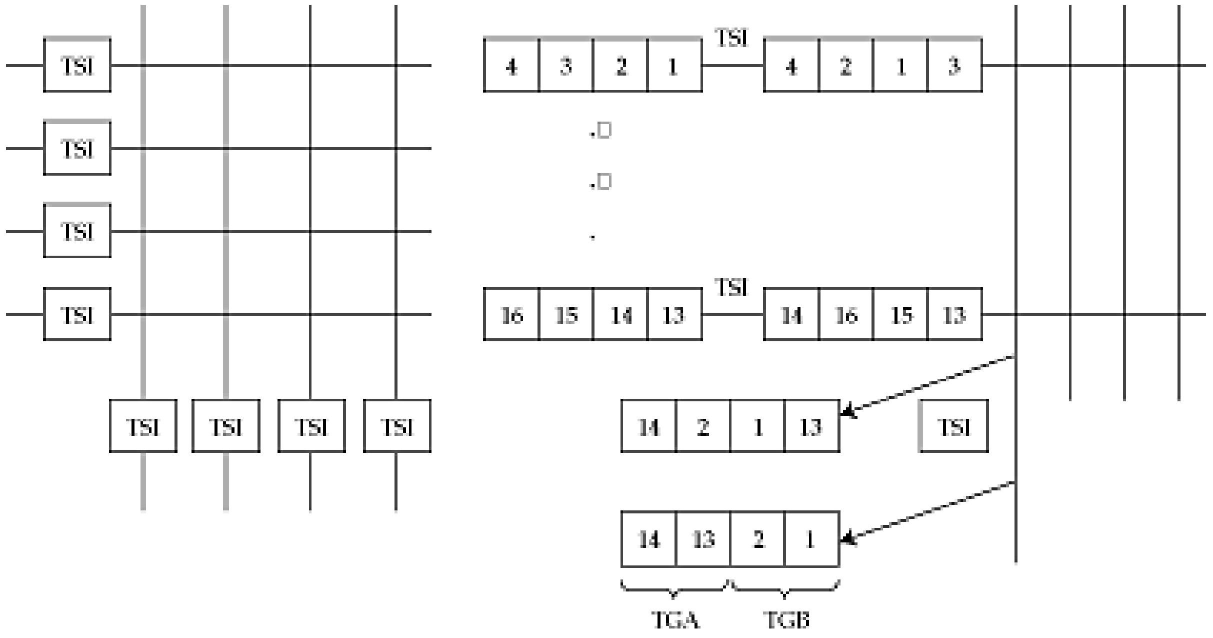
(a)



(b)

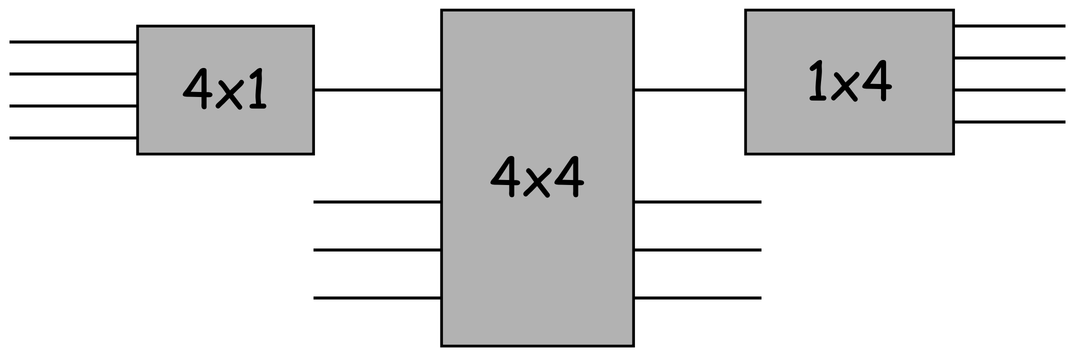
Time-space-time (TST) switching

- Similar to 3-stage crossbar except input and output cross bars use TSI
- Allowed to flip samples both on input and output trunk
  - samples in a TS switch may arrive out of order. Use output TSI to re-order
- Gives more flexibility => lowers call blocking probability

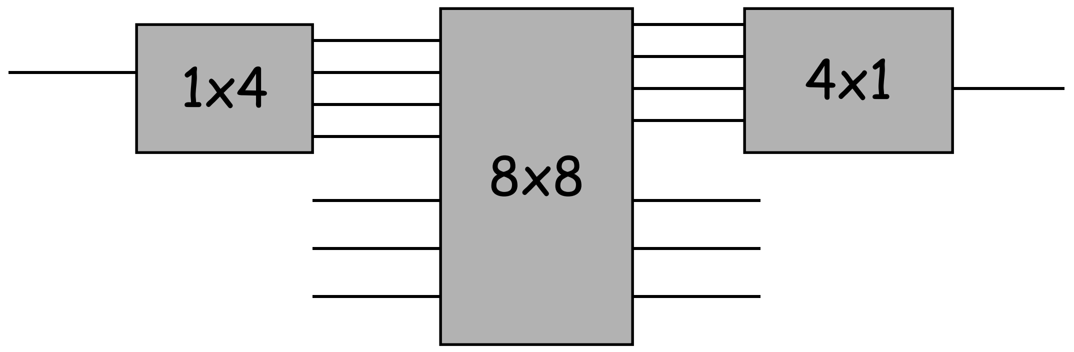


1,2,13,14 all switched to output 1; 1,2 also switched to Trunk Group B and 13,14 are switched to Trunk Group A

Line Heterogeneity



Low speed to High speed



High Speed to Low Speed

Traffic Engineering

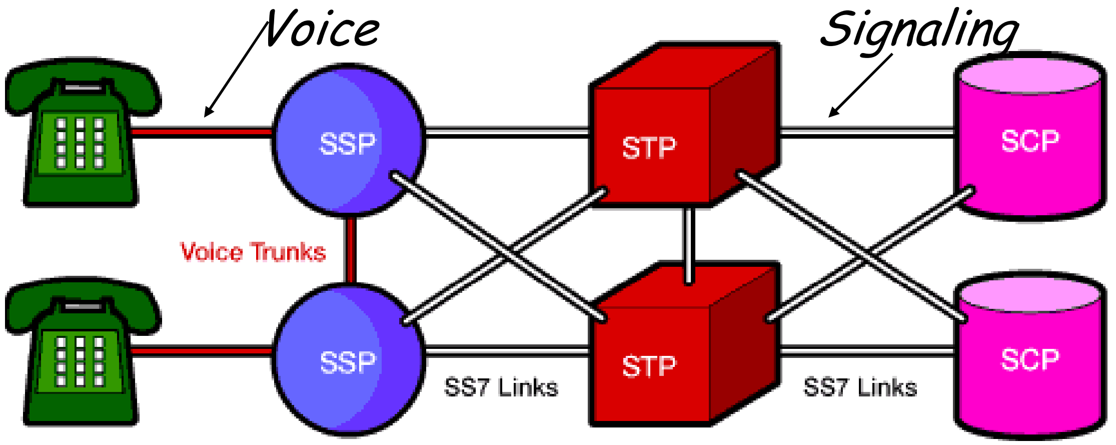
- For *MxN* switch, as M$\rightarrow\infty$, the probability of blocking (i.e., a call is lost) is given by Erlang-B formula

$$P_B = p_N = \frac{A^N / N!}{\sum_{n=0}^{N} A^n / n!}, \qquad \text{where} \quad A = \lambda / \mu$$

- λ is the call arrival rate (calls /sec)
- $1/\mu$ is the call holding time (3 minutes)
- Example: (For *A* = 12 Erlangs)
  - P_B = 1% for *N = 20; A/N = 0.6*
  - P_B = 8% for *N = 18; A/N = 0.8*
  - P_B = 30% for *N = 7; A/N = 1.7*

CCS7 Signaling

- Common channel signaling (out of band) for setup, administration, toll-free management, billing, calling-card, credit-card verification and many others



- SSP: Service Switching Point (Telephone Switches)
- STP: Signal Transfer Point (Routing Management)
- SCP: Service Control Point (Database)

Outline

- Circuit switching
- Packet switching
    - Switch generations
    - Switch fabrics
    - Buffer placement
    - Multicast switches

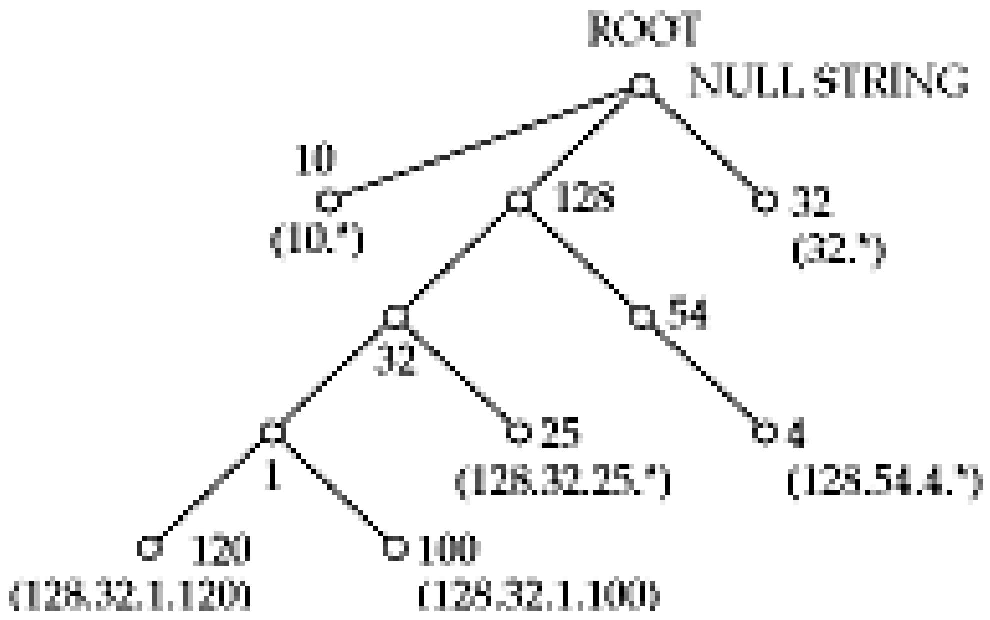
Packet switching

- In a circuit switch, path of a sample is determined at time of connection establishment
- No need for a sample header--position in frame is enough
- In a packet switch, packets carry a destination field
- Need to look up destination port on-the-fly
- Datagram
  - lookup based on entire destination address
- ATM Cell
  - lookup based on VCI
- MPLS Packet
  - Lookup based on label in the packet
- Other than that, very similar

Port mappers

- Look up output port based on destination address
- Easy for VCI: just use a table (Cross Connect)
- Harder for datagrams:
    - need to find *longest prefix match*
        - e.g. packet with address 128.32.1.20
        - entries: (128.32.*, 3), (128.32.1.*, 4), (128.32.1.20, 2)
- A standard solution: trie
    - A tree in which each node corresponds to a string that is defined by the path to that node from the root
    - *Alphabet* is a finite set of elements used to form address strings
    - Children of each node correspond to every element of the alphabet

Tries



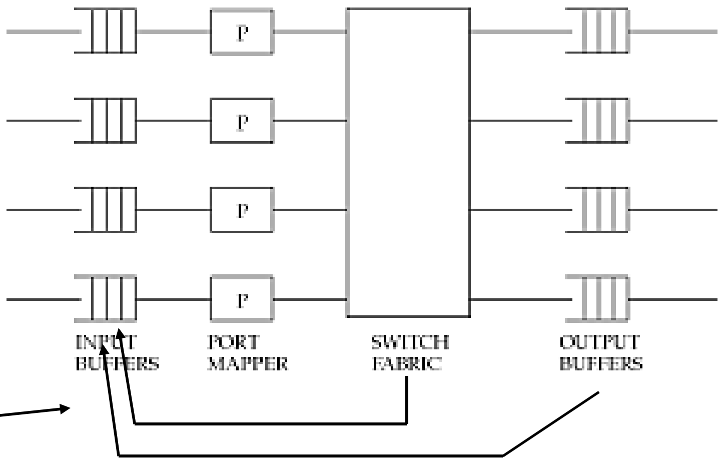
- Two ways to improve performance
  - cache recently used addresses (principle of locality) in a CAM
  - move common entries up to a higher level (match longer strings)

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 it

Dealing with blocking

- Over-provisioning
  - internal links much faster than inputs
  - expensive, waste of resources
- Buffers
  - at input or output
- Backpressure
  - if switch fabric doesn't have buffers, prevent packet from entering until path is available, by sending signals from output to input quickly.
- Sorting and Randomization
  - For certain fabrics, sorting or randomization reduces internal blocking
- Parallel switch fabrics
  - increases effective switching capacity



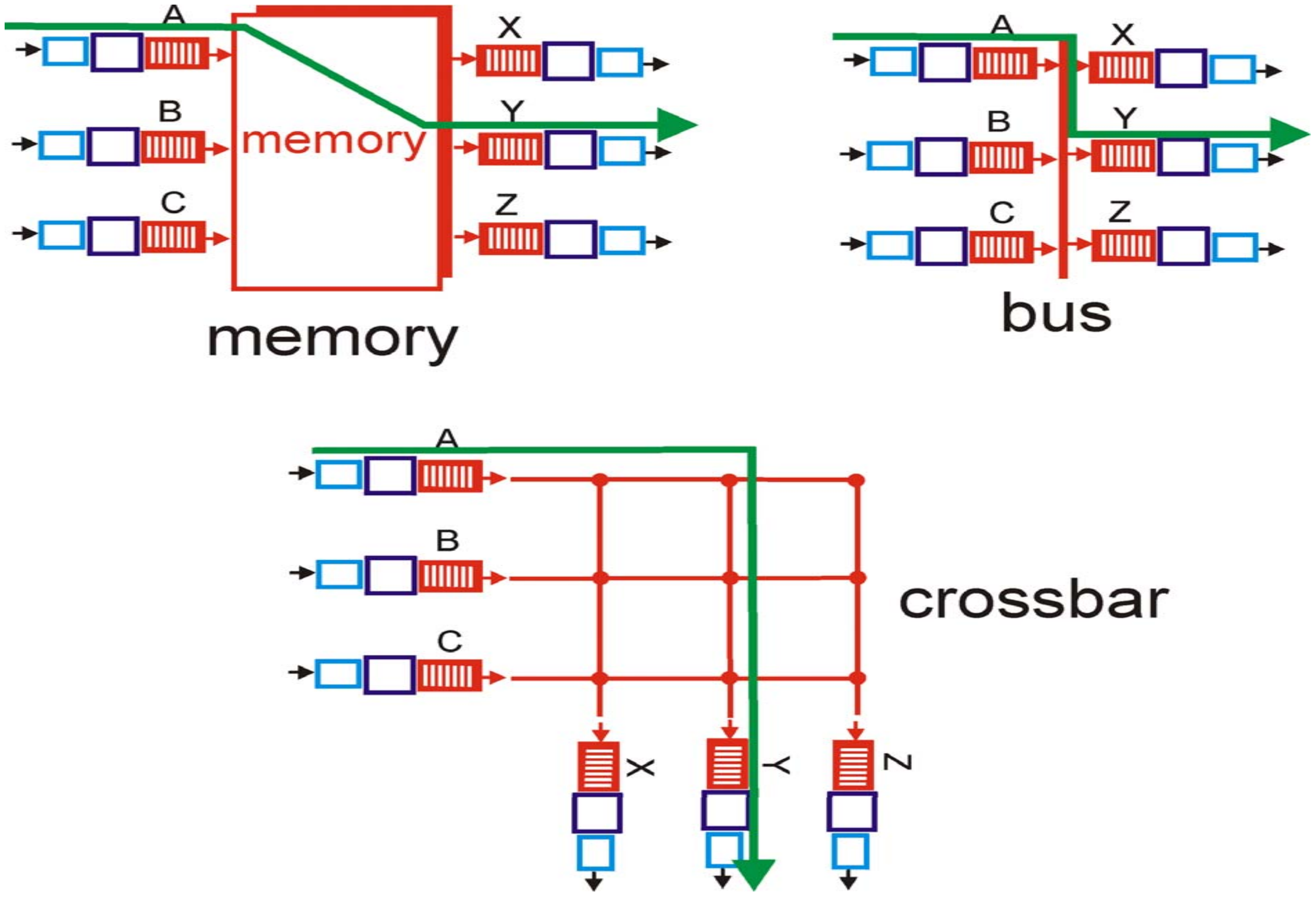
Outline

- Circuit switching
- Packet switching
  - Switch generations
  - Switch fabrics
  - Buffer placement
  - Multicast switches

Three generations of packet switches

- Different trade-offs between cost and performance
- Represent evolution in switching capacity
- All three generations are represented in current products
  - Cisco, Juniper, Nortel, Alcatel and many others

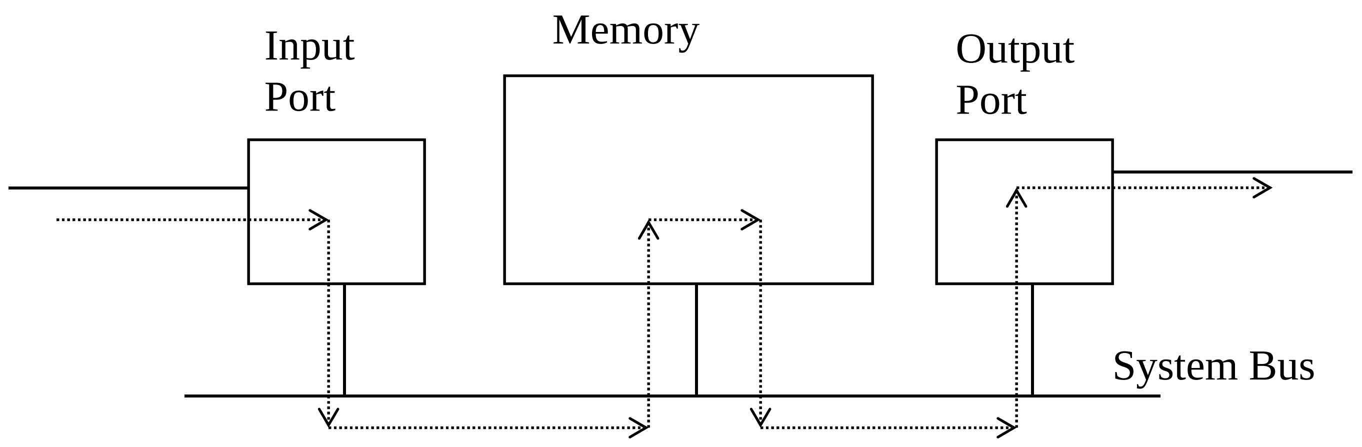
Three types of switching fabrics



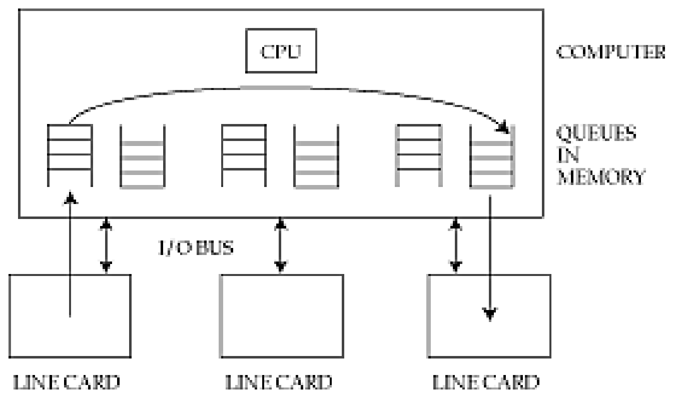
Switching Via Memory

First generation routers:

- traditional computers with switching under direct control of CPU
- packet copied to system's memory
- speed limited by memory bandwidth (2 bus crossings per datagram)

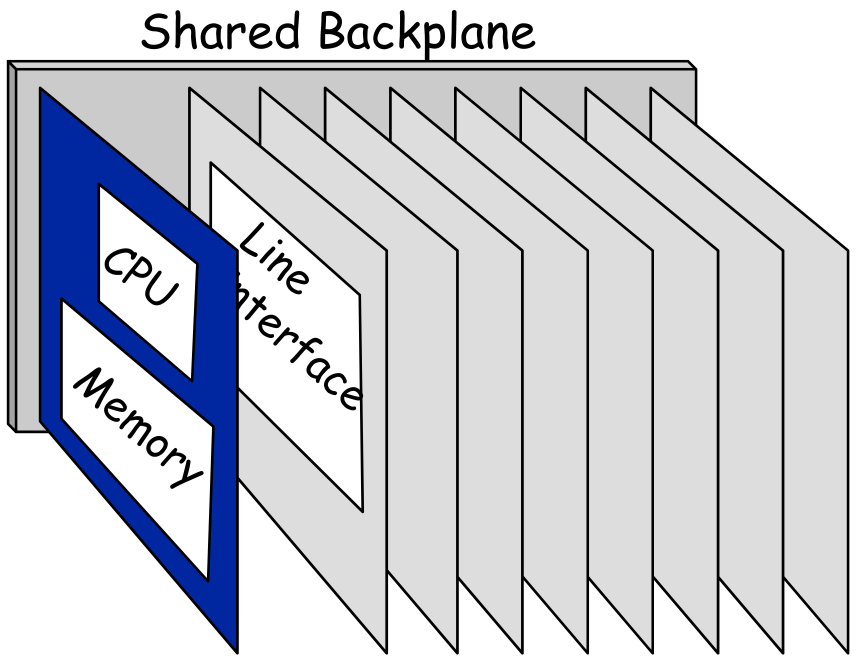


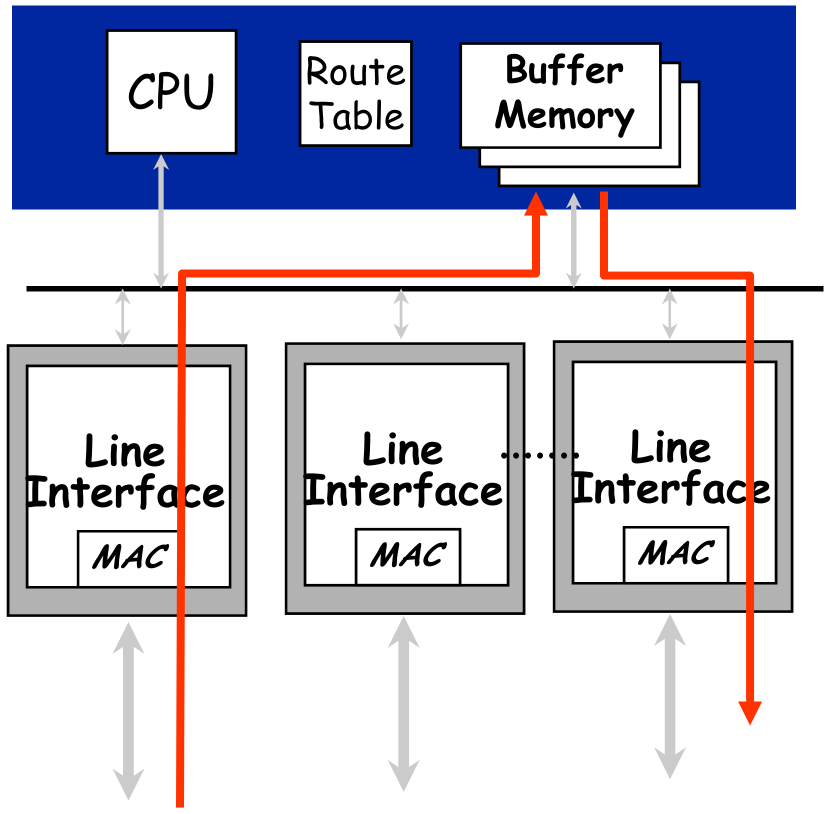
First generation switch



- A computer with multiple line cards
- Processor periodically polls inputs (or is interrupted)
- Most Ethernet switches and cheap packet routers
- Bottleneck can be CPU, host-adaptor or I/O bus, depending on the traffic scenario
- Line card can be cheap (no CPUs on line cards!!)

First Generation Routers





Typically <0.5Gb/s aggregate capacity

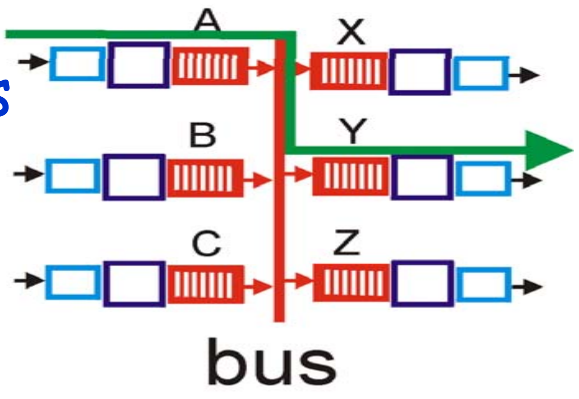
Example

- First generation router built with 133 MHz Pentium
  - Assume instruction takes one clock cycle (=7.52 ns)
  - Mean packet size 500 bytes
  - Interrupt takes 10 microseconds, word (4 bytes) access takes 50 ns
  - Per-packet processing time (routing table lookup and others) takes 200 instructions = 1.504 *μs*
- Copy loop (one word copy)

```
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 = 16.26+10+1.504=27.764 *μs* => speed is 144.1 Mbps
- How many Ethernet ports (10Mbps, 100Mbps) can be support??
  - Linux, Windows can do this now!!

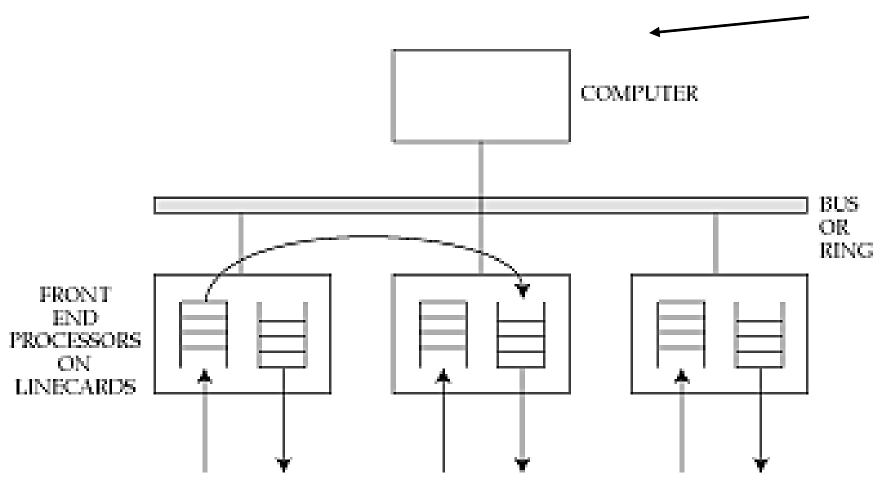
Switching Via a Bus



- datagram from input port memory

  to output port memory via a shared bus
- bus contention: switching speed limited by bus bandwidth
- 1 Gbps bus, Cisco 1900: sufficient speed for access and enterprise routers (not regional or backbone)

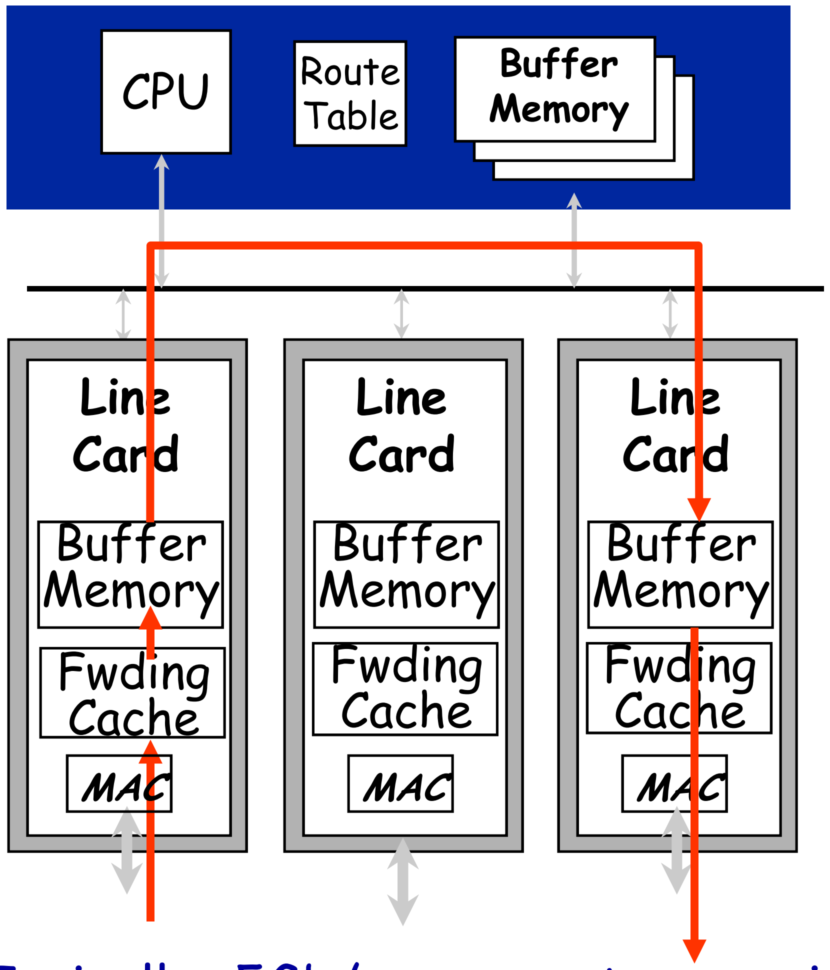
Second generation switch



Control plane
Routing Protocols
Routing Tables

- Port mapping intelligence in line cards (processor based)
- Ring or Bus based backplane to connect line cards
  - Bottleneck -> performance impact (discuss bus and ring)
- Lookup cache on line cards (for better performance)
- For switch, cross connect table (port mapping entries) only changed when calls are setup / torn down
- For datagram router, ask the control processor if the entry is not found in local forwarding table or automatically updated.

Second Generation Routers



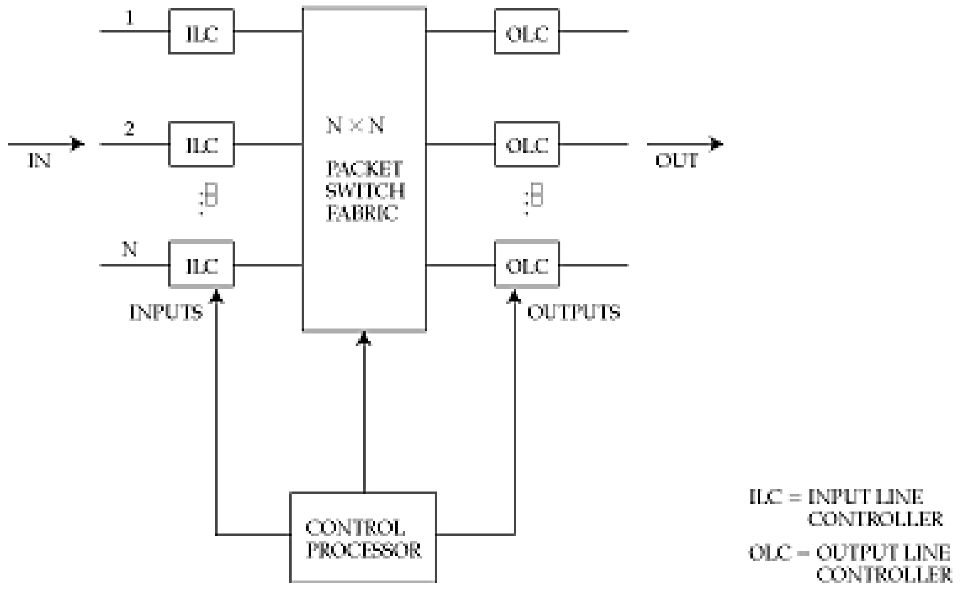
Typically <5Gb/s aggregate capacity

Switching Via An Interconnection Network

- overcome bus bandwidth limitations
- Banyan networks, other interconnection nets initially developed to connect processors in multiprocessor
- Advanced design: fragmenting datagram into fixed length cells, switch cells through the fabric.
  - Segmentation and Reassembly (SAR)
    - Discuss overheads
- Cisco 12000: switches Gbps through the interconnection network

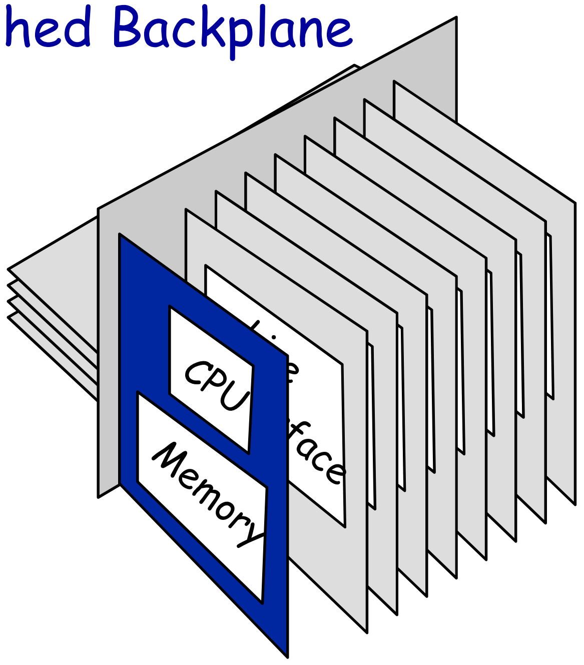
Third generation switches

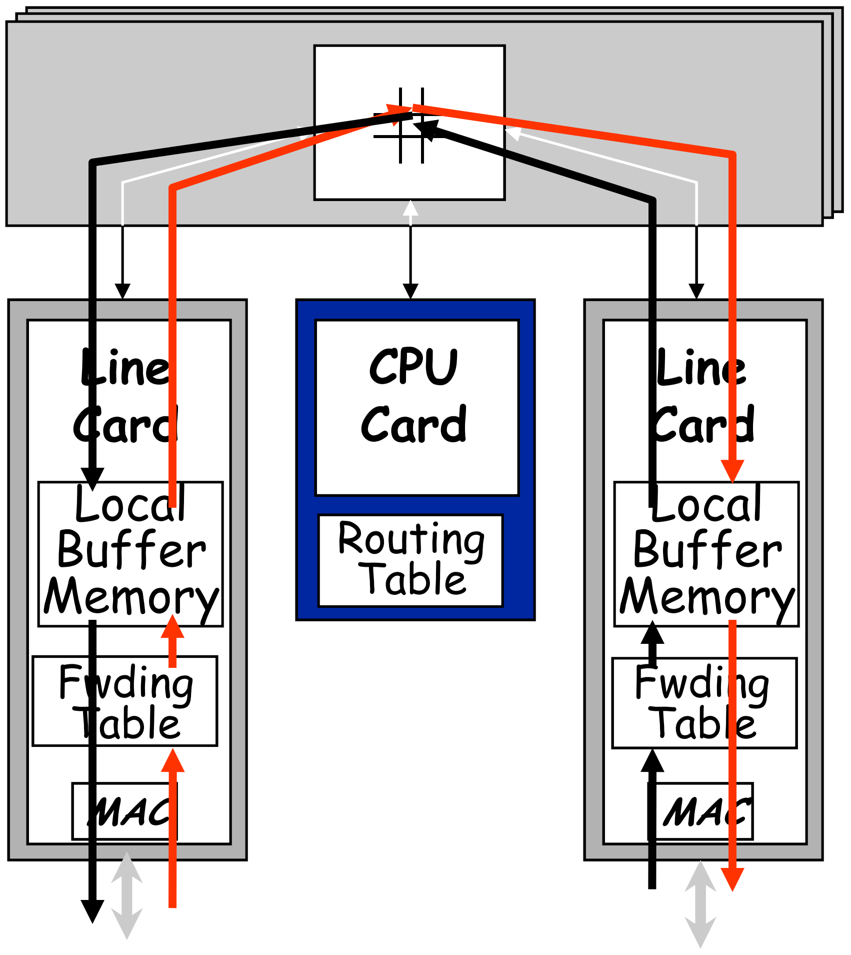
- Bottleneck in second generation switch is the bus (or ring)
- Third generation switch provides parallel paths using a switch fabric



Third Generation Routers

Switched Backplane



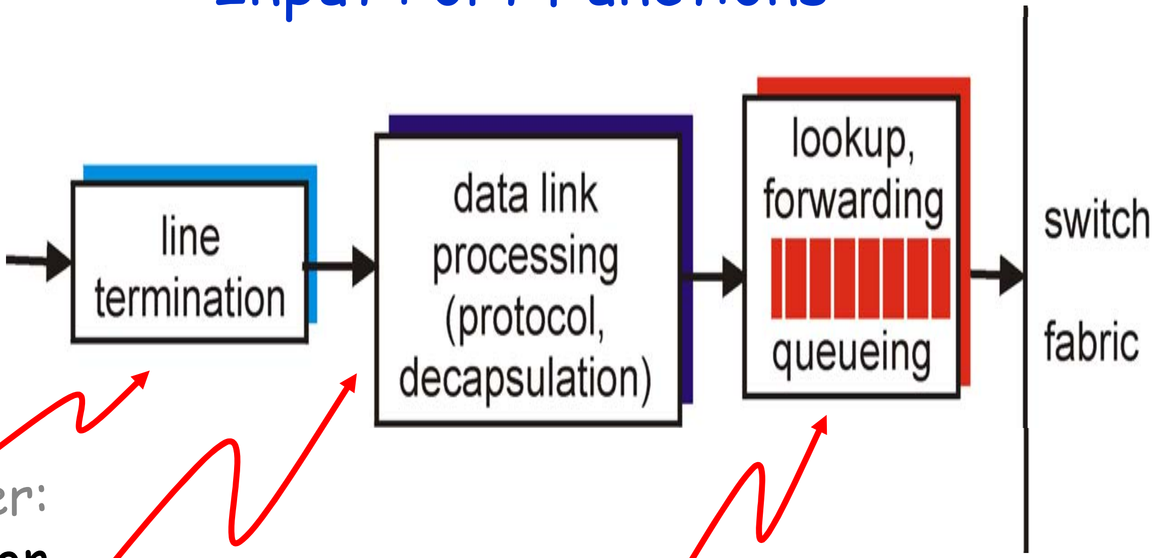


Typically <50Gb/s aggregate capacity

Third generation (contd.)

- Features
  - self-routing fabric
  - output buffer is a point of contention
    - unless we *arbitrate* access to fabric
  - potential for unlimited scaling, as long as we can resolve contention for output buffer

Input Port Functions



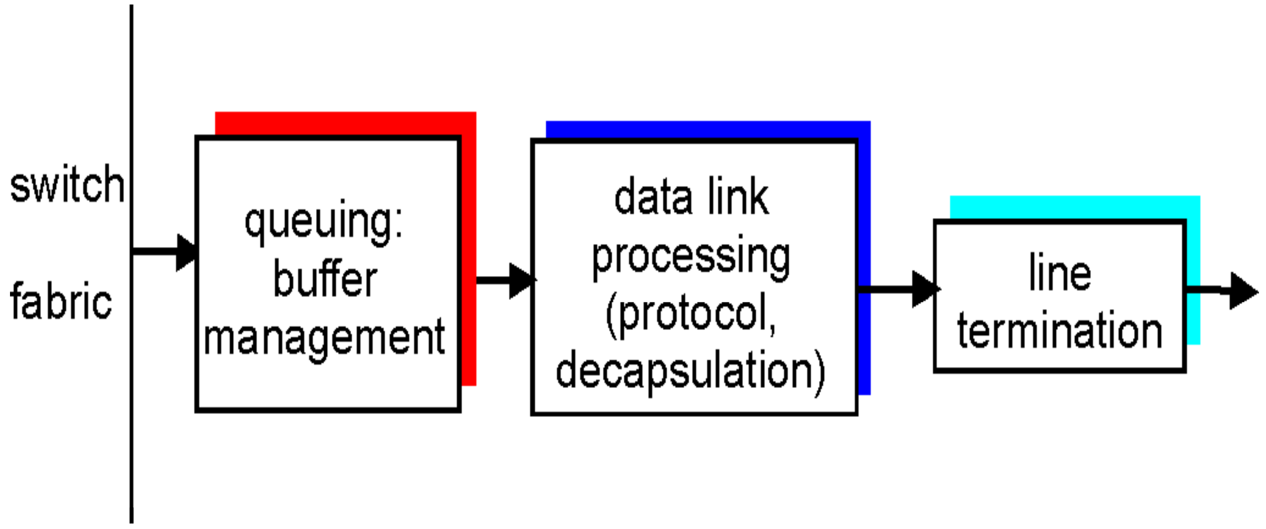
Physical layer:
bit-level reception

Data link layer:
e.g., Ethernet
see chapter 5

Decentralized switching:

- given datagram dest., lookup output port using forwarding table in input port memory
- goal: complete input port processing at 'line speed'
- queuing: if datagrams arrive faster than forwarding rate into switch fabric

Output Ports


switch
fabric
queuing: buffer management
data link processing (protocol, decapsulation)
line termination

- *Buffering* required when datagrams arrive from fabric faster than the transmission rate
- *Scheduling discipline* chooses among queued datagrams for transmission

Outline

- Circuit switching
- Packet switching
    - Switch generations
    - Switch fabrics
    - Buffer placement
    - Multicast switches

Switch fabrics

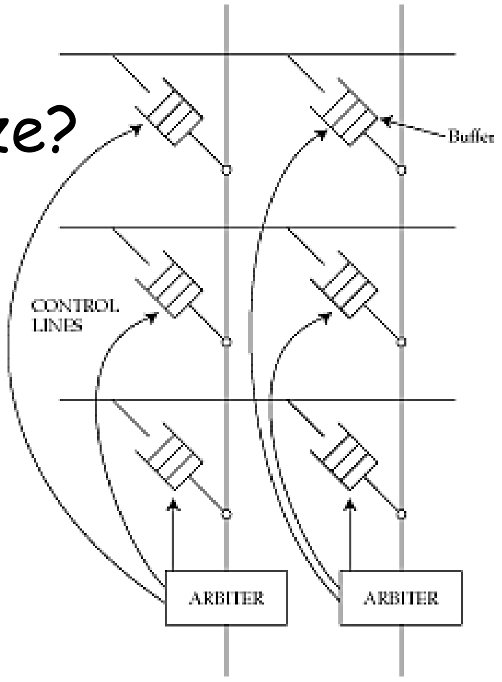
- Transfer data from input to output, ignoring scheduling and buffering
- Usually consist of links and *switching elements*

Crossbar

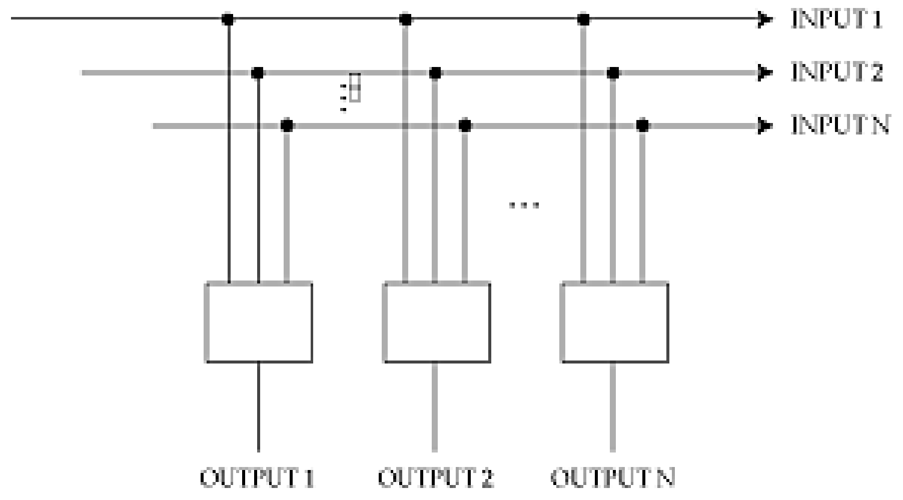
- Simplest switch fabric
  - think of it as 2N buses in parallel
- Used here for *packet* routing: cross-point is left open long enough to transfer a packet from an input to an output
- For fixed-size packets and known arrival pattern, can compute schedule in advance (e.g., circuit switching)
- Otherwise, need to compute a schedule on-the-fly (what does the schedule depend on?)

Buffered crossbar

- What happens if packets at two inputs both want to go to same output?
  - Output blocking
- Can defer one at an input buffer
- Or, buffer cross-points
- How large is the buffer size?
- Overflow in the switch
  - Can we afford?
  - Solutions?
    - Backpressure



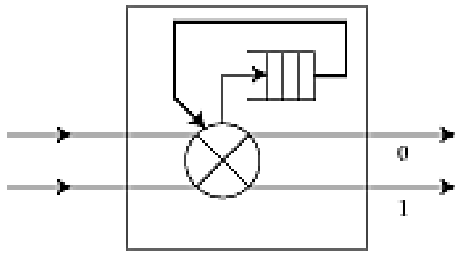
Broadcast



- Packets are tagged with output port #
- Each output matches tags
- Need to match N addresses in parallel at each output
- Useful only for small switches, or as a stage in a large switch

Switch fabric element

- Can build complicated fabrics from a simple element consisting of two inputs, two outputs and an optional buffer
- Packets arrive simultaneously; Look at the header;



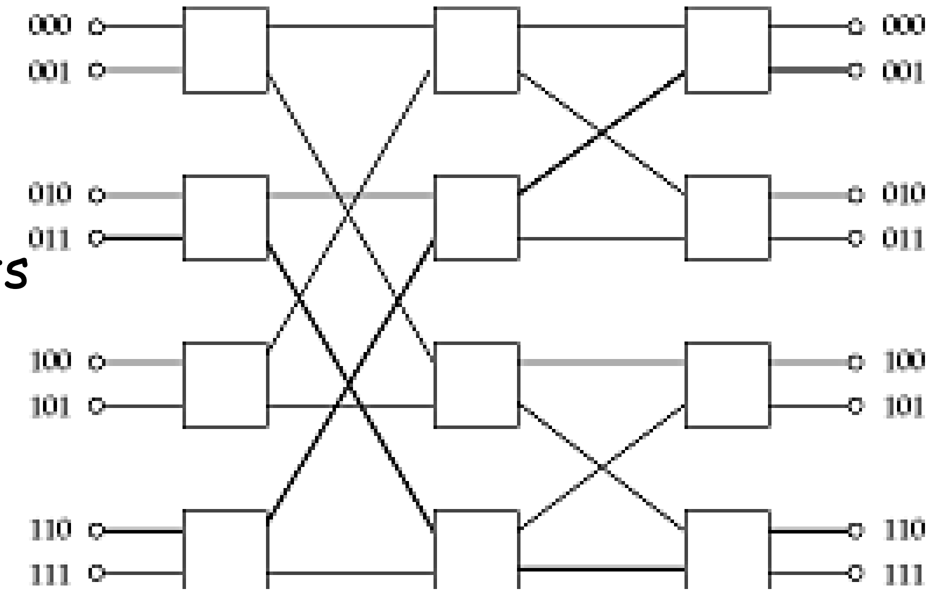
- Routing rule: if 0, send packet to upper output, else to lower output
- If both packets to same output, buffer or drop

Features of fabrics built with switching elements

- NxN switch with bxb elements has $\lceil \log_b N \rceil$ stages with $\lceil N/b \rceil$ elements per stage
  - e.g., 8x8 switch with 2x2 elements has 3 stages of 4 elements per stage
  - e.g., 4096x4096 switch built with 8x8 blocks has four stages with 512 elements in each stage
- Fabric is *self routing*
  - Once a packet is labeled to a correct output, it will automatically makes its way
- Recursive
  - composed of smaller components that resemble larger network
- Can be synchronous or asynchronous (permits variable length packets)
- Regular and suitable for VLSI implementation

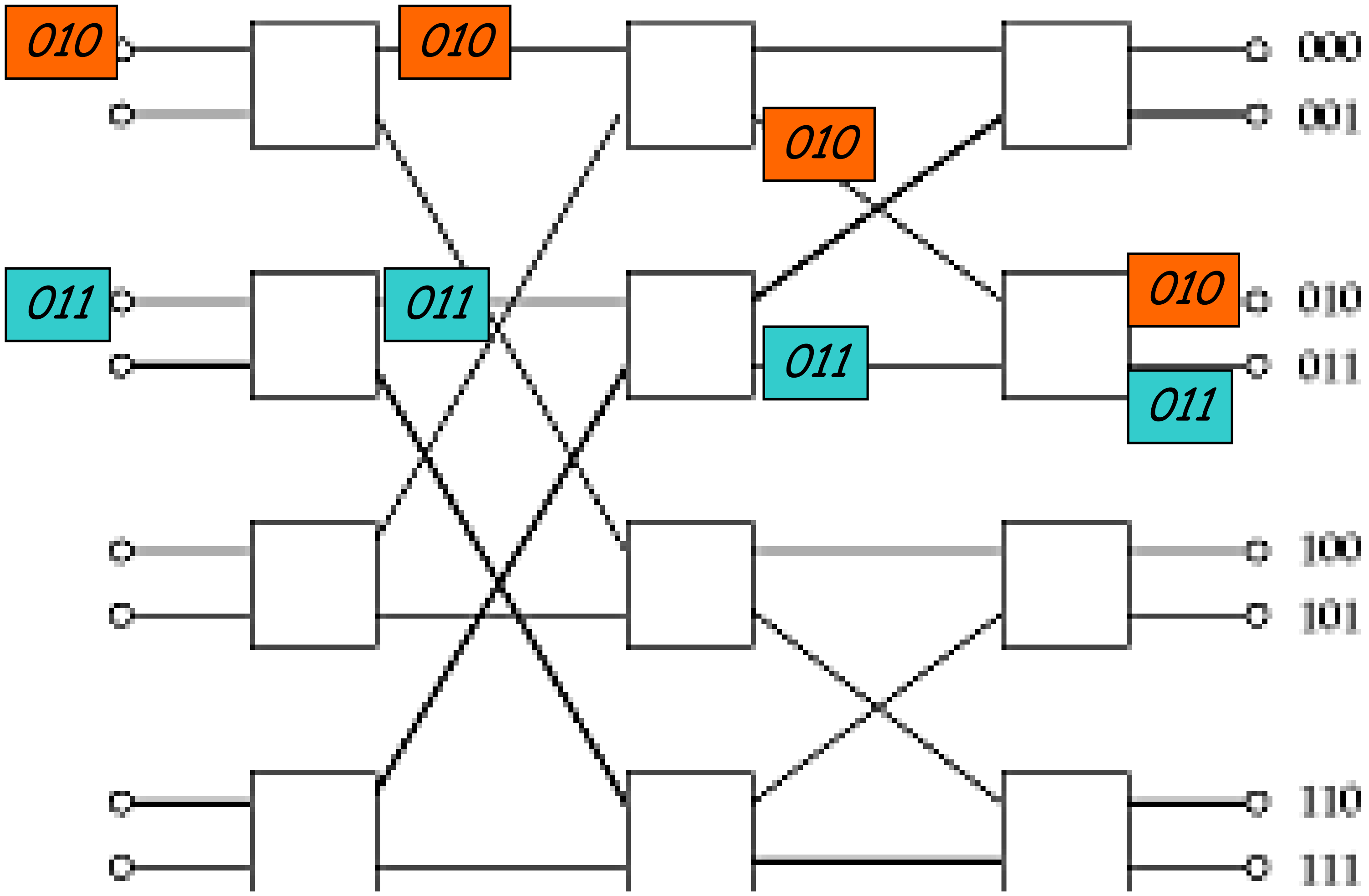
Banyan

- Simplest self-routing recursive fabric. Packets are tagged with output port in binary
- Made of 2x2 switches
- Fabric needs *n* stages for 2^n outputs with 2^{n-1} elements in each stage



- (why does it work?) Each switching element at the *i*th stage looks at the *i*th bit to make a forwarding decision
- What if two packets both want to go to the same output?
  - output blocking

Banyan (Example)



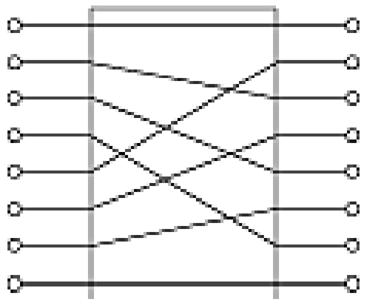
Blocking

- Can avoid with a buffered banyan switch
  - but this is too expensive; how much buffer at each element?
  - 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 (e.g., 32x32 switch with 9 banyans can achive 10^{-9} loss)
  - can reorder packets
  - output buffers have to run k times faster than input

Sorting

- Or we can avoid blocking by choosing order in which packets appear at input ports
- If we can
  - present packets at inputs sorted by output
  - similar to TSI
  - remove duplicates
  - remove gaps
  - precede banyan with a perfect shuffle stage
  - then no internal blocking
- For example

  [X, 011, 010, X, 011, X, X, X] -(sort)->
  [010, 011, 011, X, X, X, X, X] -(remove dups)->
  [010, 011, X, X, X, X, X, X] -(shuffle)->
  [010, X, 011, X, X, X, X, X] ⟶ *This input when presented to Banyan Network is non-blocking*
- Need sort, shuffle, and trap networks



Shuffle Exchange

Sorting

- Build sorters from merge networks
- Assume we can merge two sorted lists to make a larger sorted list
  - Called Batcher Network
  - Needs $\lceil \log N \rceil \lceil \log N+1/2 \rceil$ stages
- Sort pairwise, merge, recurse
- Divide list of N elements into pairs and sort each pair (gives $N/2$ lists)
- Merge pair wise to form $N/4$ and recurse to form $N/8$ etc to form one fully sorted list
- All we need is way to sort two elements and a way to merge sorted lists

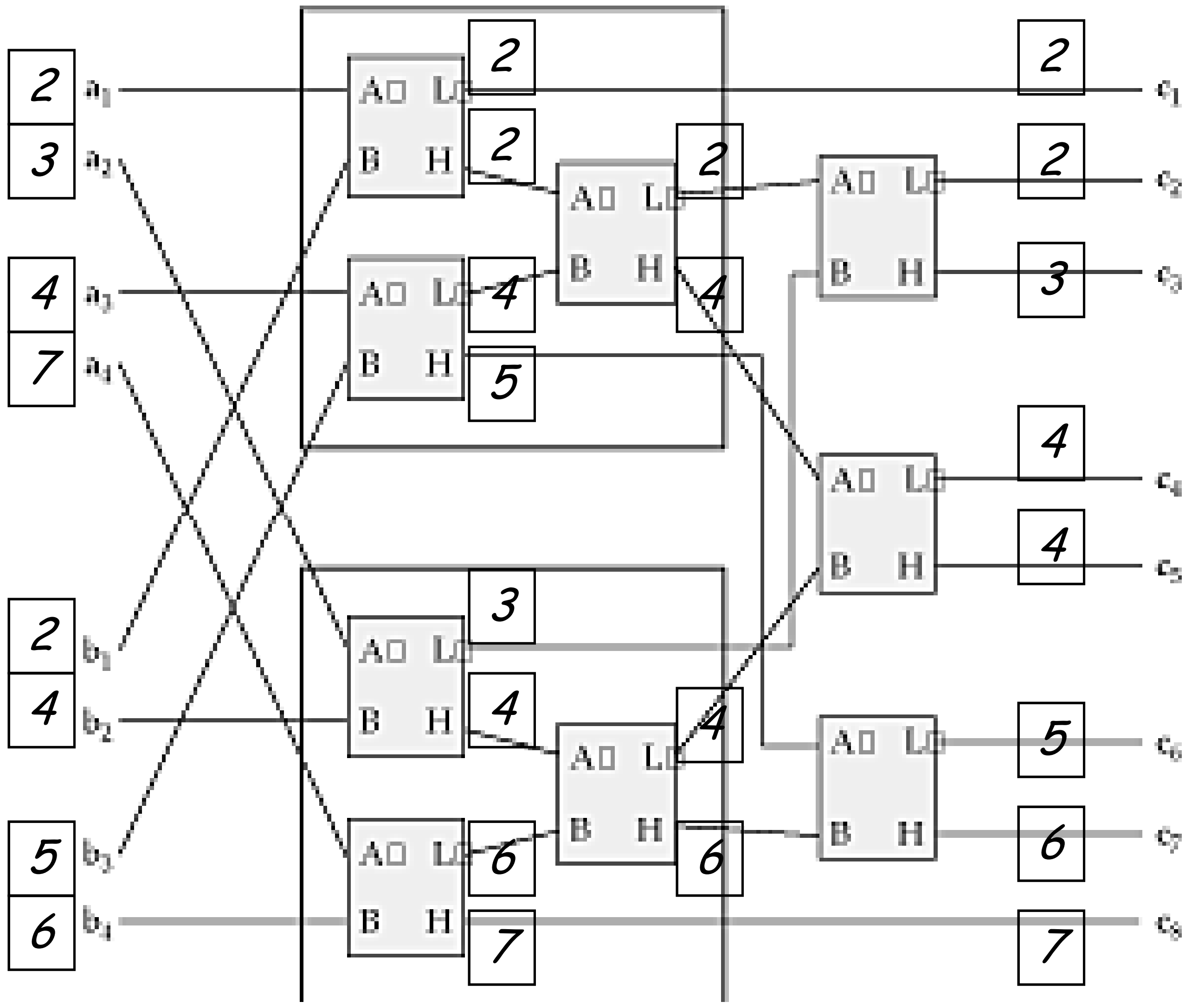
Sorting (Example)

- Sort the list 5,7,2,3,6,2,4,5 by merging
- Solution:
  - Sort elements two-by-two to get four sorted lists {5,7}, {2,3}, {2,6}, {4,5}
  - Second step is to merge adjacent lists to get four element sorted lists {2,3,5,7}, {2,4,5,6}
  - In the third step, we merge two lists to create a fully sorted list {2,2,3,4,5,5,6,7}
- Sorter is easy to build
  - Use a comparator
- Merging needs a separate network

Merging Networks

- A merging network of size 2N takes two sorted lists of size N as inputs and creates a merged list of size 2N
- Consists of two N-sized merging networks
- One of them merges all the even elements of the two inputs and the other merges all the odd elements
- The outputs of the mergers are handed to a set of 2x2 comparators

Merging

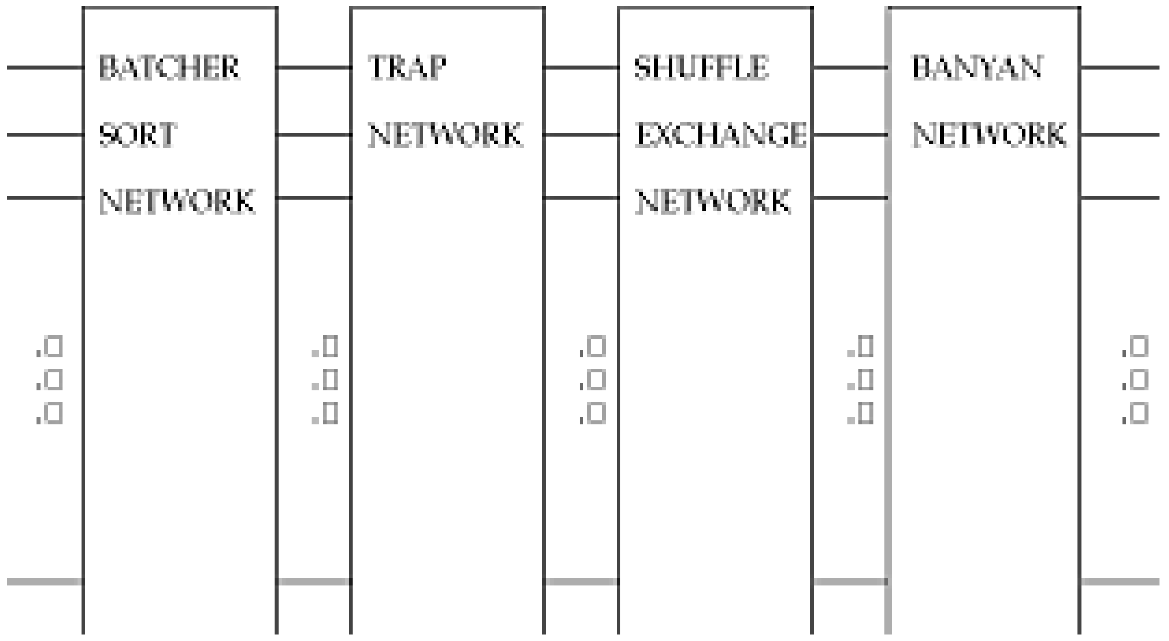


2
3
4
7
2
4
5
6
a_1
a_2
a_3
a_4
b_1
b_2
b_3
b_4
A L
B H
2
2
4
5
3
4
6
7
2
4
4
6
2
2
3
4
4
5
6
7
c_1
c_2
c_3
c_4
c_5
c_6
c_7
c_8

Merging Example

- Merge the two sorted lists {2,3,4,7} and {2,4,5,6}
- Solution:
    - First stage, we merge even elements from the two lists {2,4} with {2,5}
    - Recursing we need to merge {2} with {2} and {4} with {5} then compare them
    - Results of the two merges are {2,2} and {4,5}
    - Comparing higher element of the first list with lower element of the second list, we determine the merged list is {2,2,4,5}
    - Next merge odd elements {3,7} with {4,6} with result {3,4} and {6,7}
    - Comparing the high and low elements we get merged list {3,4,6,7}
    - Carrying out the comparisons we get {2,2,3,4,4,5,6,7}

Putting it together- Batcher Banyan



- What about trapped duplicates?
  - re-circulate to beginning
  - or run output of trap to multiple banyans (*dilation*)

Effect of packet size on switching fabrics

- A major motivation for small fixed packet size in ATM is ease of building large parallel fabrics
- In general, smaller size => more per-packet overhead, but more preemption points/sec
  - At high speeds, overhead dominates!
- Fixed size packets helps build synchronous switch
  - But we could fragment at entry and reassemble at exit
  - Or build an asynchronous fabric
  - Thus, variable size doesn't hurt too much
- Maybe Internet routers can be almost as cost-effective as ATM switches

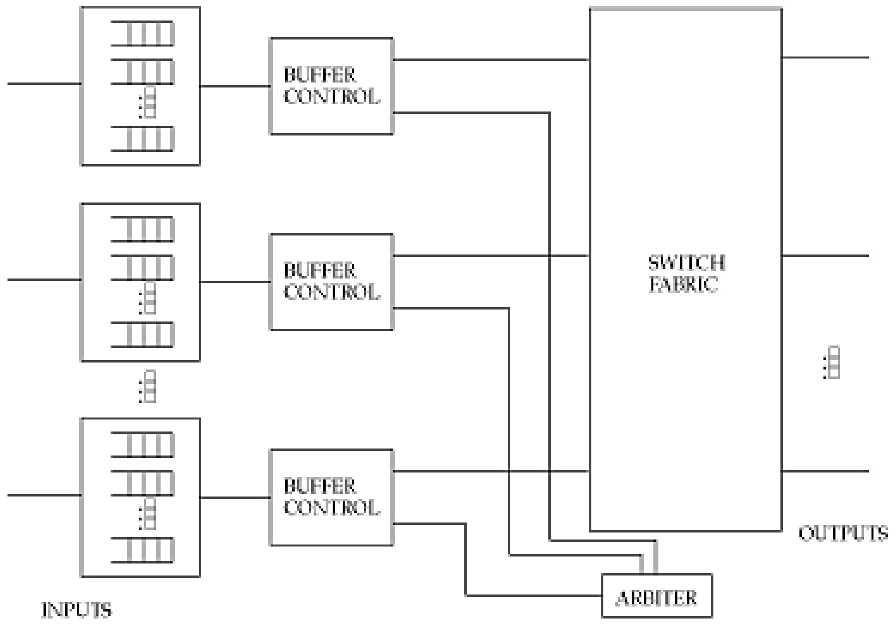
Outline

- Circuit switching
- Packet switching
  - Switch generations
  - Switch fabrics
  - Buffer placement
  - Multicast switches

Buffering

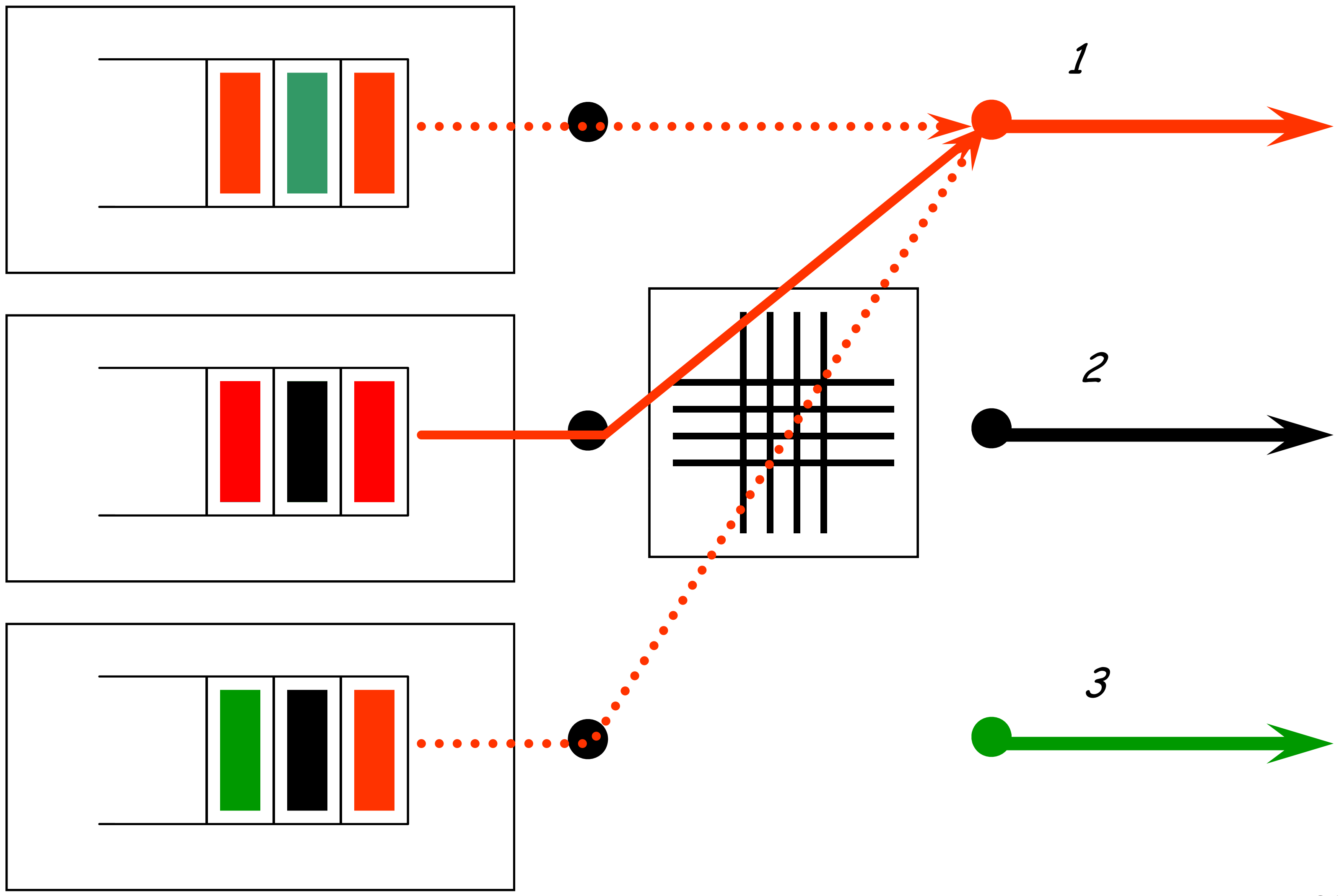
- All packet switches need buffers to match input rate to service rate
  - or cause heavy packet loses
- Where should we place buffers?
  - input
  - in the fabric
  - output
  - shared

Input buffering (input queueing)



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

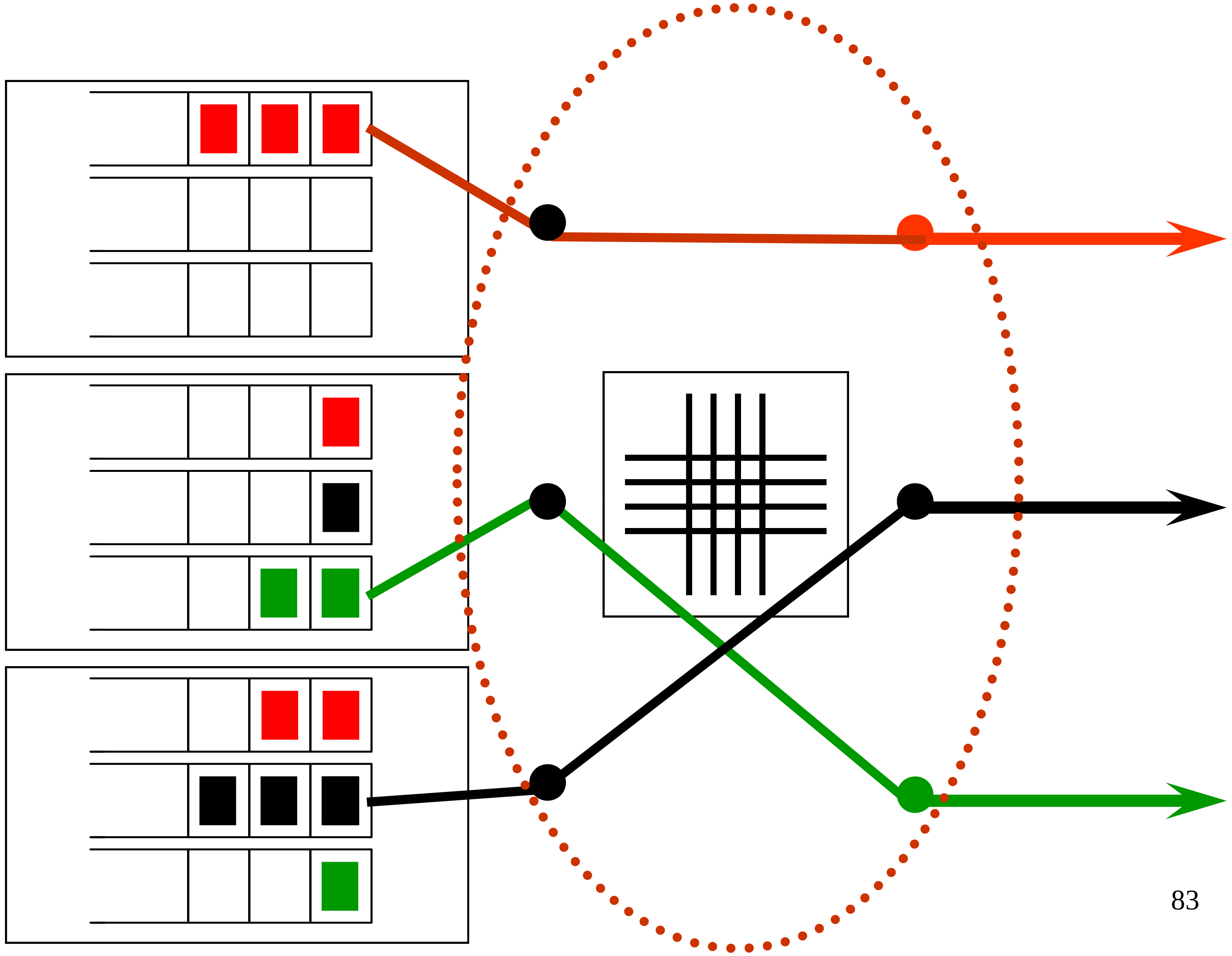
Head of Line blocking



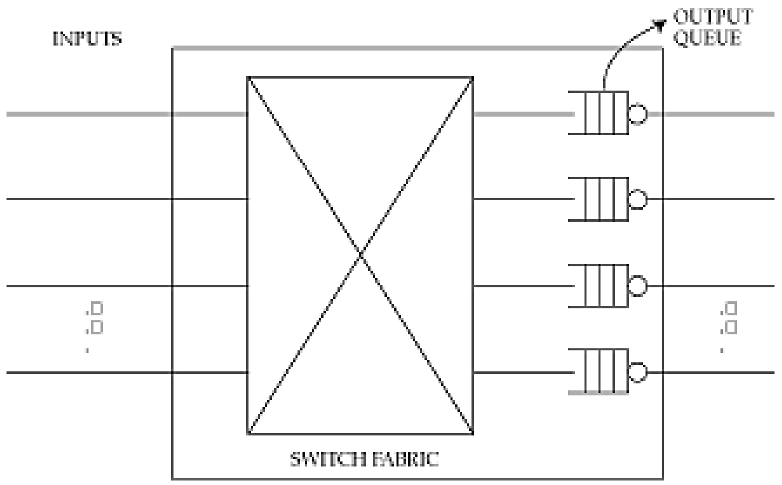
Dealing with HOL blocking

- Per-output queues at inputs (Virtual Input Queueing)
- 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 many large switches

Virtual Input (Output) Queueing

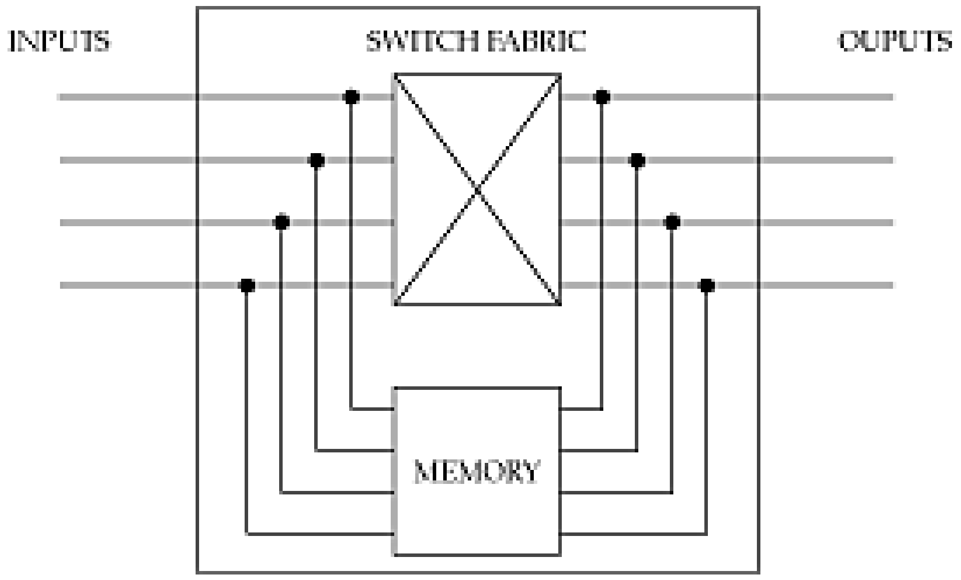


Output queueing



- Doesn'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

Shared memory

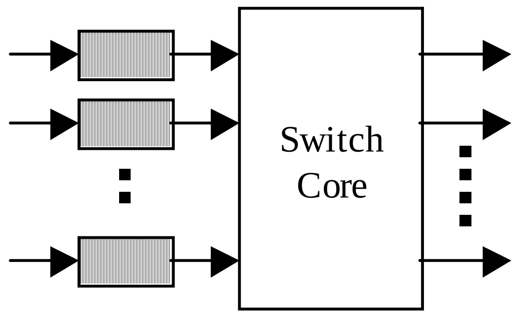


- Route only the header to output port
- Bottleneck is time taken to read and write multiported memory
- Doesn't scale to large switches
- But can form an element in a multistage switch

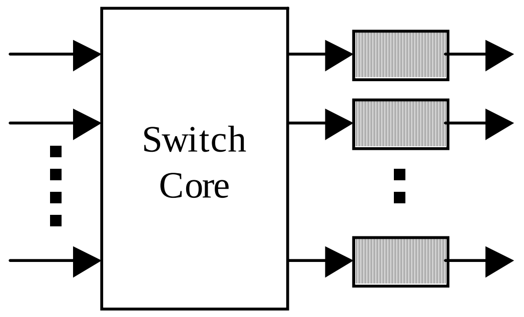
Buffered fabric

- Buffers in each switch element
- Pros
  - Speed up is only as much as fan-in
  - Hardware backpressure reduces buffer requirements
- Cons
  - costly (unless using single-chip switches)
  - scheduling is hard

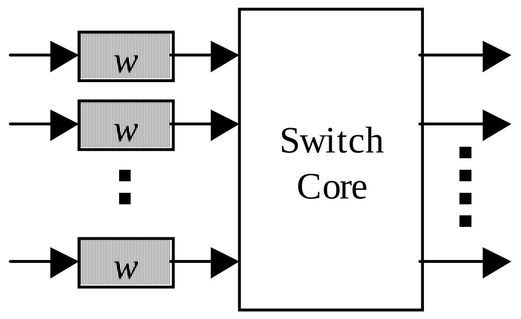
Summary of Buffer Placement



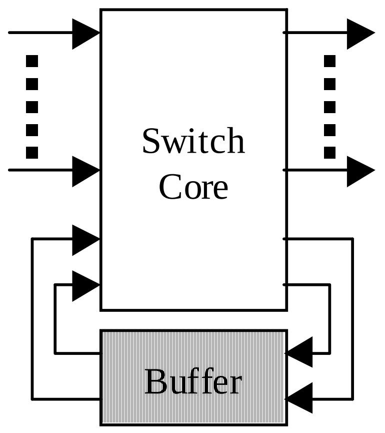
a) Input Queueing



c) Output Queueing



b) Window Selection



d) Shared Buffering

Non-blocking Switch Performance

- Non-blocking Switch with no buffers
  - If output contention occurs, only one among *n* contending packets transmitted, all other dropped
  - Throughput = 63.2%; But remaining is all packet loss!!
- Non-blocking Switch FIFO input buffers
  - Throughput = 58.6%
  - Packet loss is a function of buffer size
  - For a Bernoulli packet arrival process (with a probability *p*)

$$P_{loss} < \frac{p(2-p)}{2(1-p)}\left[\frac{p^2}{2(1-p)^2}\right]^B$$

Switch Performance (contd ..)

- Non-blocking switch with non-FIFO buffers
  - packets are selected from a window (*w*) of buffer to minimize contention

Size	FIFO	Window Size (*w*)							
N		1	2	3	4	5	6	7	8
2	75.0%	75%	84%	89%	92%	93%	94%	95%	96%
4	65.5%	66%	76%	81%	85%	87%	89%	94%	92%
8	61.8%	62%	72%	78%	82%	85%	87%	88%	89%
16		60%	71%	77%	81%	84%	86%	87%	88%
32		59%	70%	76%	80%	83%	85%	87%	88%
64		59%	70%	76%	80%	83%	85%	86%	88%
∞	58.6%								

Switch Performance (contd ..)

- Non-blocking Switch with Output buffers
  - Best performance (100% Throughput) as there is no HOL blocking
  - Delay performance depends on the output queueing
- Non-blocking Switch with Shared buffers
  - Packets lost in contention are stored in a separate buffer that feeds as direct input (depending upon the number of extra inputs)
  - Performance can be close to 100% with large shared buffer
  - Switch size grows

Hybrid solutions

- Buffers at more than one point
- Becomes hard to analyze and manage
- But common in practice

Outline

- Circuit switching
- Packet switching
  - Switch generations
  - Switch fabrics
  - Buffer placement
  - Multicast switches

Multicasting

- Useful to do this in hardware
- Assume port-mapper knows list of outputs
- Incoming packet must be copied to these output ports
- Two subproblems
  - generating and distributing copies
  - VCI translation for the copies

Generating and distributing copies

- Either implicit or explicit
- Implicit
  - suitable for bus-based, ring-based, crossbar, or broadcast switches
  - multiple outputs enabled after placing packet on shared bus
  - used in Paris and Datapath switches
- Explicit
  - need to copy a packet at switch elements
  - use a *copy* network
  - place # of copies in tag
  - element copies to both outputs and decrements count on one of them
  - collect copies at outputs
- Both schemes increase blocking probability

Header translation

- Normally, in-VCI to out-VCI translation can be done either at input or output
- With multicasting, translation easier at output port (why?)
- Use separate port mapping and translation tables
- Input maps a VCI to a set of output ports
- Output port swaps VCI
- Need to do two lookups per packet

Packet Size Impacts

- Fixed Length Packets
- Variable Length Packets