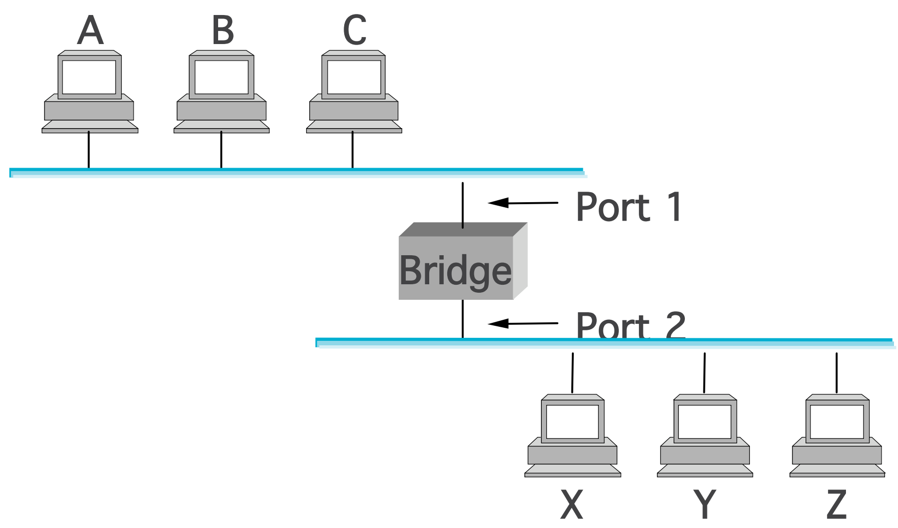
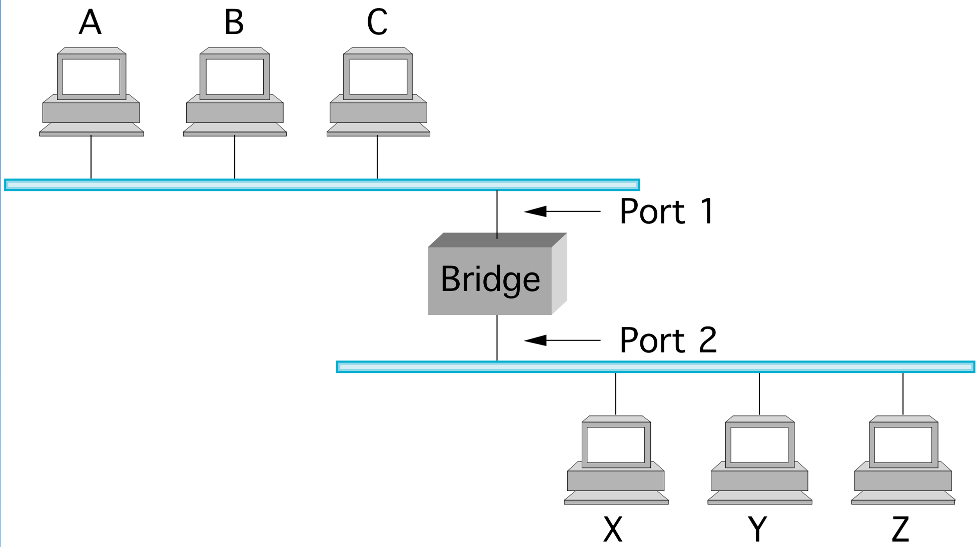
Bridges and Extended LANs

- LANs have physical limitations (e.g., 2500m)
- Connect two or more LANs with a bridge
  - Bridges use "accept and forward" strategy
  - level 2 connection (<u>does not add packet header</u>)



Learning Bridges

- Do not forward when unnecessary
- Maintain forwarding table

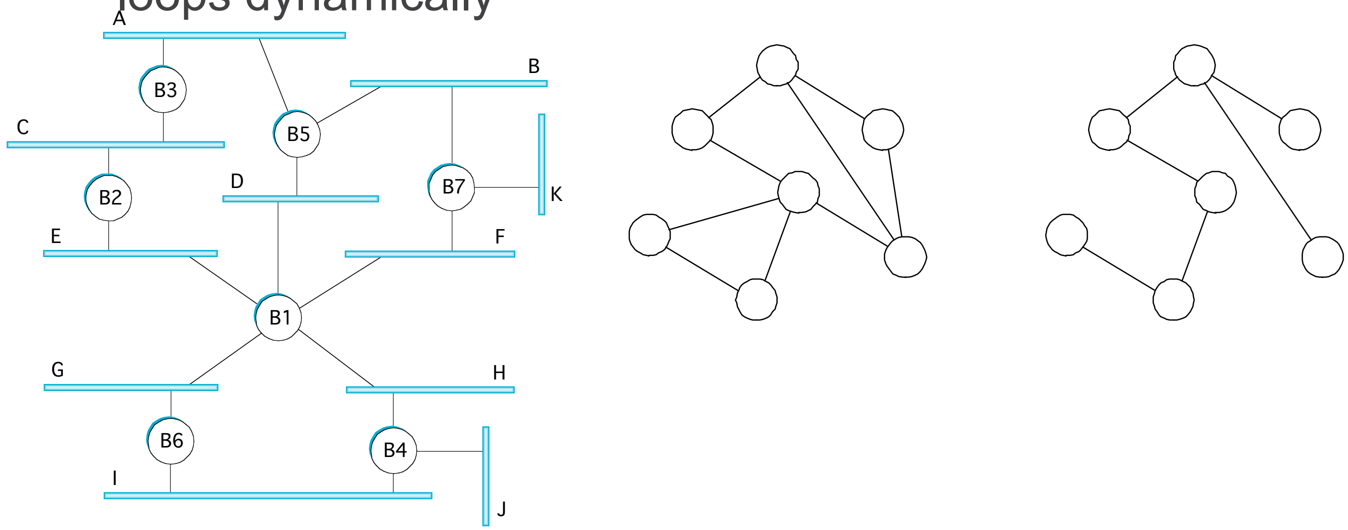


Host	Port
A	1
B	1
C	1
X	2
Y	2
Z	2

- Learn table entries based on source address
- Table is an optimization; need not be complete
- Always forward broadcast frames

Spanning Tree Algorithm

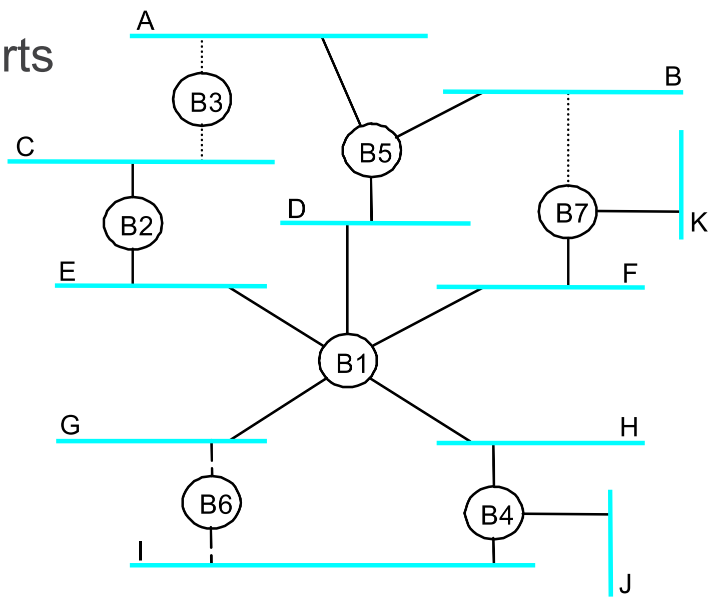
- Problem: loops in cabling can make packets forwarded forever - no mechanism to remove looping frames
  - We can remove loops by maintaining state in the packet, but for layer-2 switching - we are not allowed to change the packet
  - Extra cabling can be good for redundancy if we can remove loops dynamically



- Bridges run a distributed spanning tree algorithm
  - select which bridges actively forward
  - developed by Radia Perlman
  - now IEEE 802.1 specification

Algorithm Overview

- Each bridge has unique id (e.g., B1, B2, B3)
- Select bridge with smallest id as root
  - How to choose root: next slides
- Select bridge on each LAN closest to root as designated bridge (use id to break ties)
  - Each bridge forwards frames over each LAN for which it is the designated bridge
  - Root forwards over all its ports



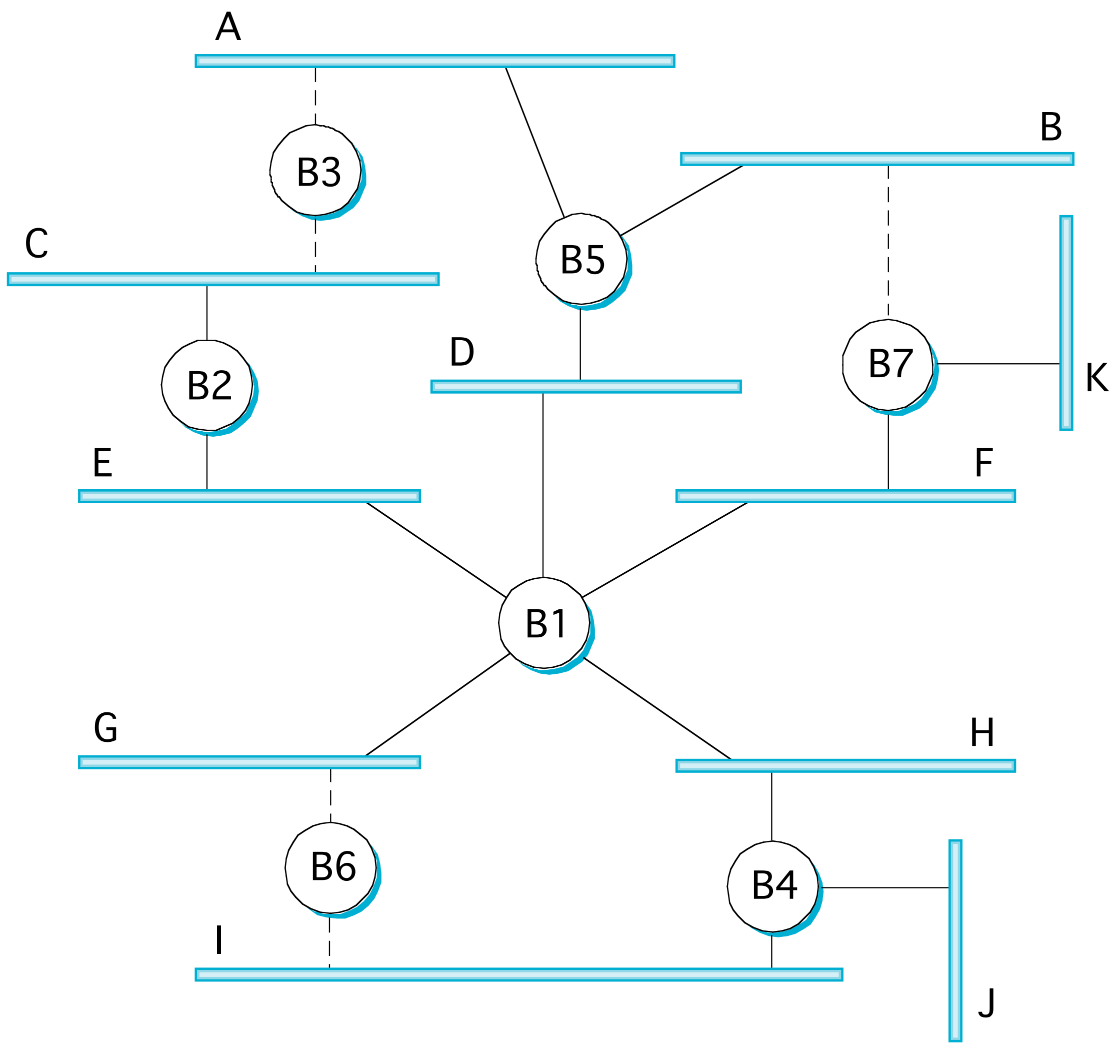
Algorithm Details

- Bridges exchange configuration messages
  - id for bridge sending the message
  - id for what the sending bridge believes to be root bridge
  - distance (hops) from sending bridge to root bridge
- Each bridge records current best configuration message for each port
- Initially, each bridge believes it is the root

Algorithm Detail (cont)

- When a bridge learns that it is not root, stop generating config messages
  - in steady state, only root generates configuration messages
- When a bridge learns that it is not the designated bridge, stop forwarding config messages
  - in steady state, only designated bridges forward config messages
- Root continues to periodically send config messages
- If any bridge does not receive config message after a period of time, it starts generating config messages claiming to be the root

Spanning tree



Spanning tree properties

- Spanning trees avoid loops, they are not designed to find shortest path or "route" against congested paths.
  - All traffic goes towards the root
  - We will develop routers later on in the course which will address these issues

Broadcast and Multicast

- Forward all broadcast/multicast frames
  - current practice
- Learn when no group members downstream
- Accomplished by having each member of group G send a frame to bridge multicast address with G in source field

Tcpdump trace

- tcpdump -p

02:21:52.651816 802.1d config 0000.00:02:2d:71:03:ef.0001 root 0000.00:02:2d:71:03:ef pathcost 0 age 0 max 20 hello 2 fdelay 15

02:21:53.263956 engr-fe21.gw.nd.edu > ALL-SYSTEMS.MCAST.NET: igmp query v2 [tos 0xc0] [ttl 1]

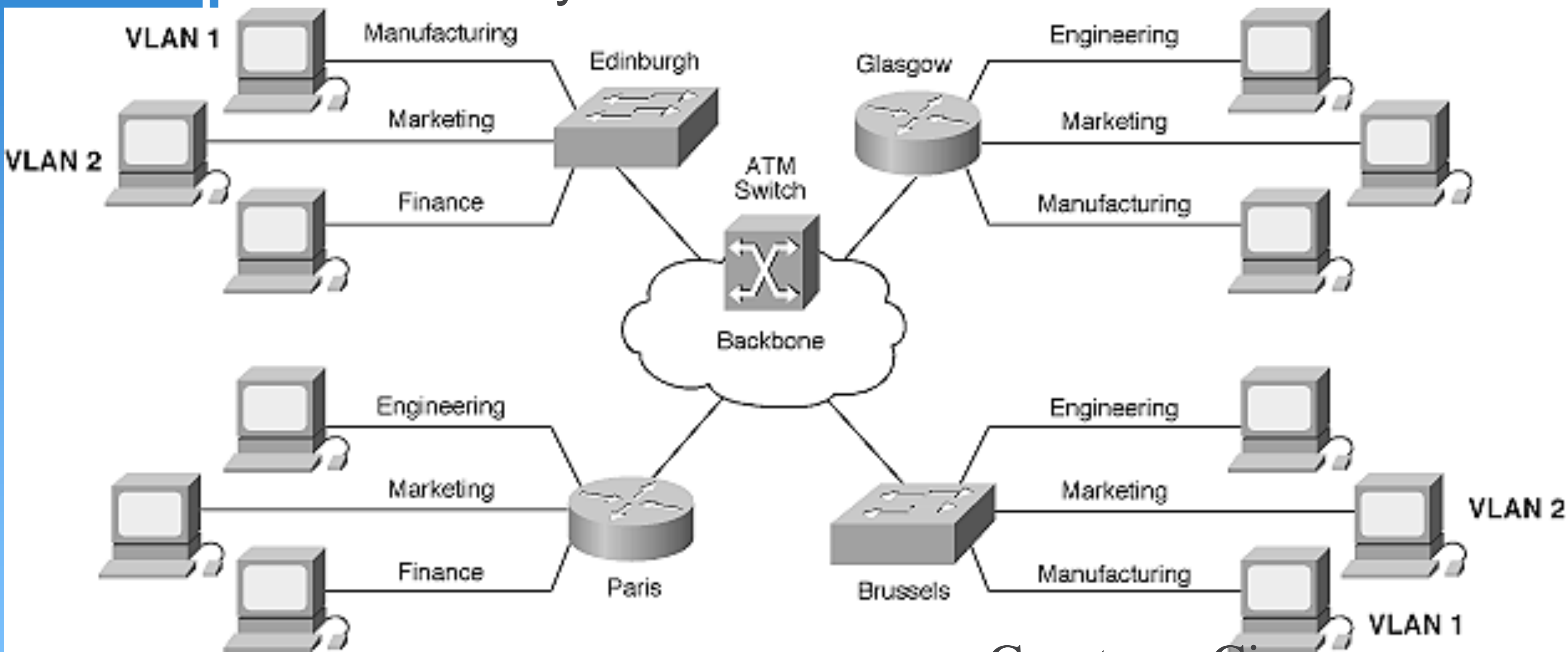
02:25:22.656898 CDP v2, ttl=180s DevID '013183892(hub24-1b.hub.nd.edu)' Addr (1): IPv4 129.74.24.67 PortID '5/10' CAP 0x0e[|cdp]

Limitations of Bridges

- Do not scale
  - spanning tree algorithm does not scale - traffic gets bridged through the root bridge
    - Spanning tree is designed to avoid loops, not traffic balancing: redundant routes are ignored
  - broadcast does not scale
- Do not accommodate heterogeneity
- Caution: beware of transparency

VLAN (Notre Dame uses these to create departmental LANs)

- Create virtual lans (broadcast domains) without rewiring
- Add a 4 byte VLAN id to each frame



Courtesy: Cisco

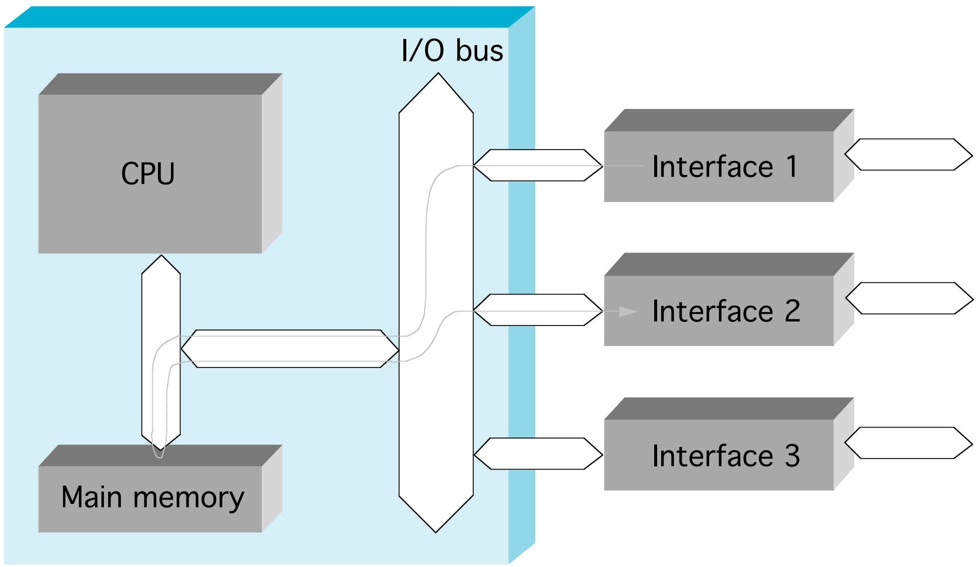
Implementation details

- How do we actually build a switch
  - Need to send data from any input to any output port
  - Its expensive to fully allow this, some paths are shared
  - Packets that are in contention are stored and forwarded

  - Control logic inspects every packet
  - Processing power needed depends on packet size

Workstation-Based

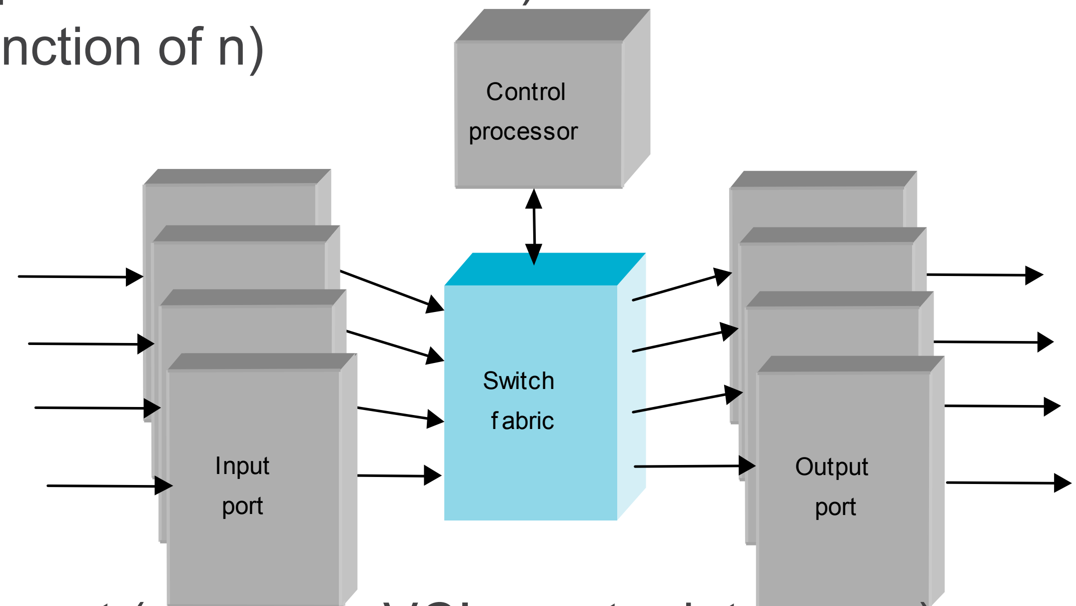
- Aggregate bandwidth
  - 1/2 of the I/O bus bandwidth
  - capacity shared among all hosts connected to switch
  - example: 1Gbps bus can support 5 x 100Mbps ports (in theory)



- 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

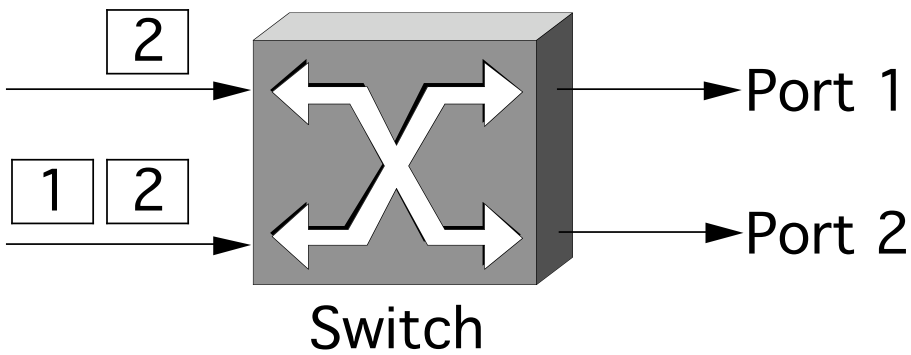
- Design Goals
  - throughput (depends on traffic model)
  - scalability (a function of n)



- Ports
  - circuit management (e.g., map VCIs, route datagrams)
  - buffering (input and/or output)
- Fabric
  - as simple as possible
  - sometimes do buffering (internal)

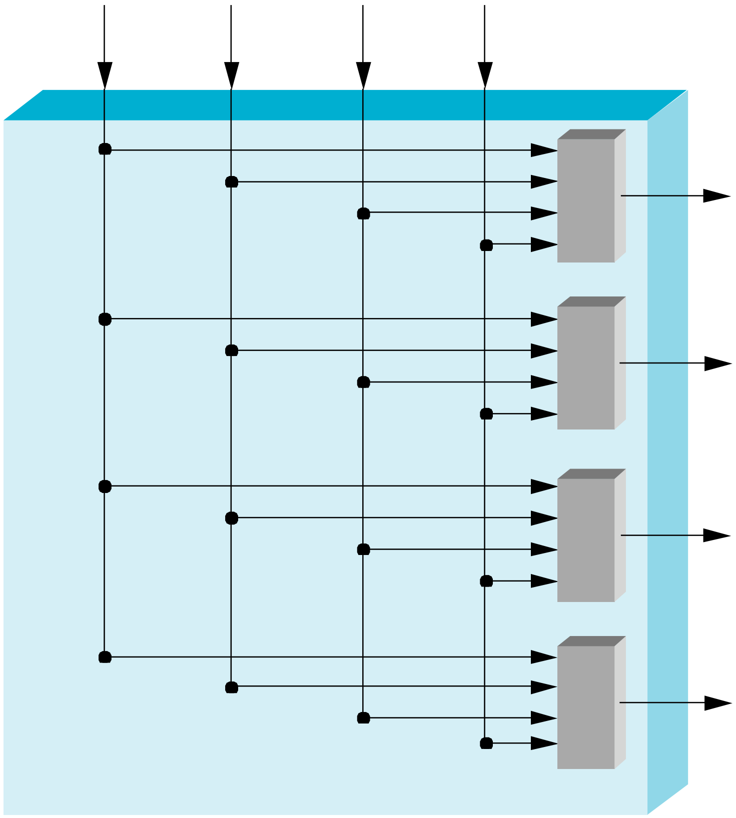
Buffering

- 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 - 1 is blocked even though there is no contention for port1



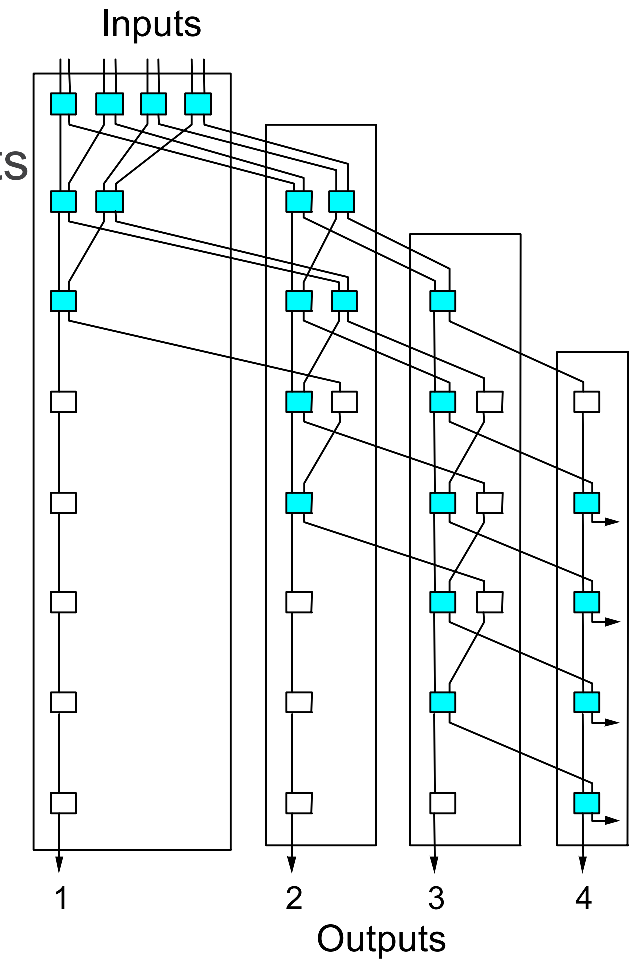
Crossbar Switches

- Each port = total switch throughput



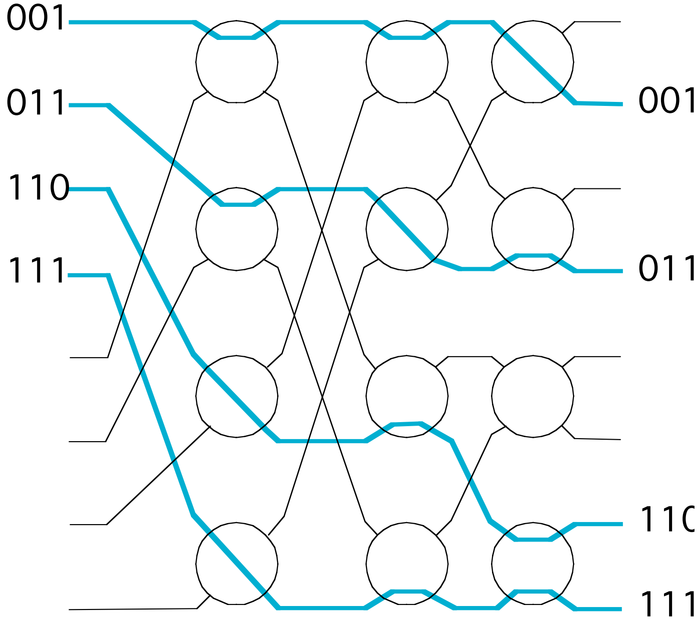
Knockout Switch

- Example crossbar
- Concentrator
  - select l of n packets
- Complexity: n^2



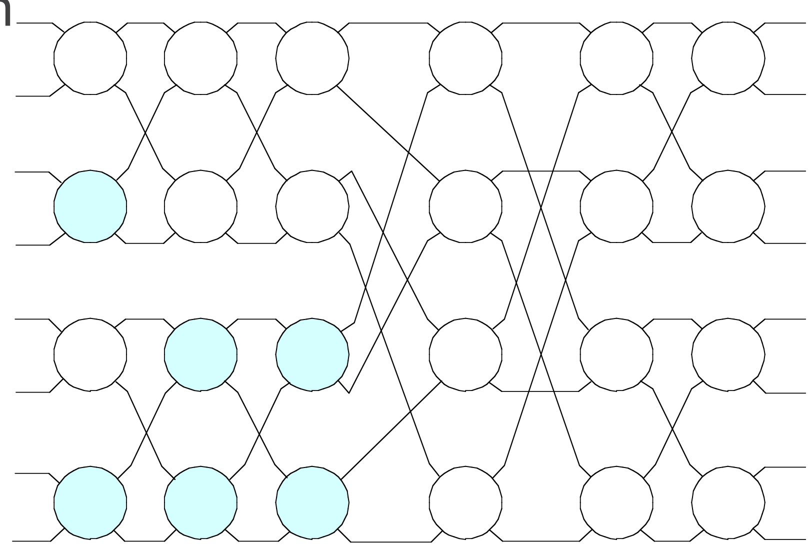
Self-Routing Fabrics

- Banyan Network
  - constructed from simple 2 x 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 log22 n

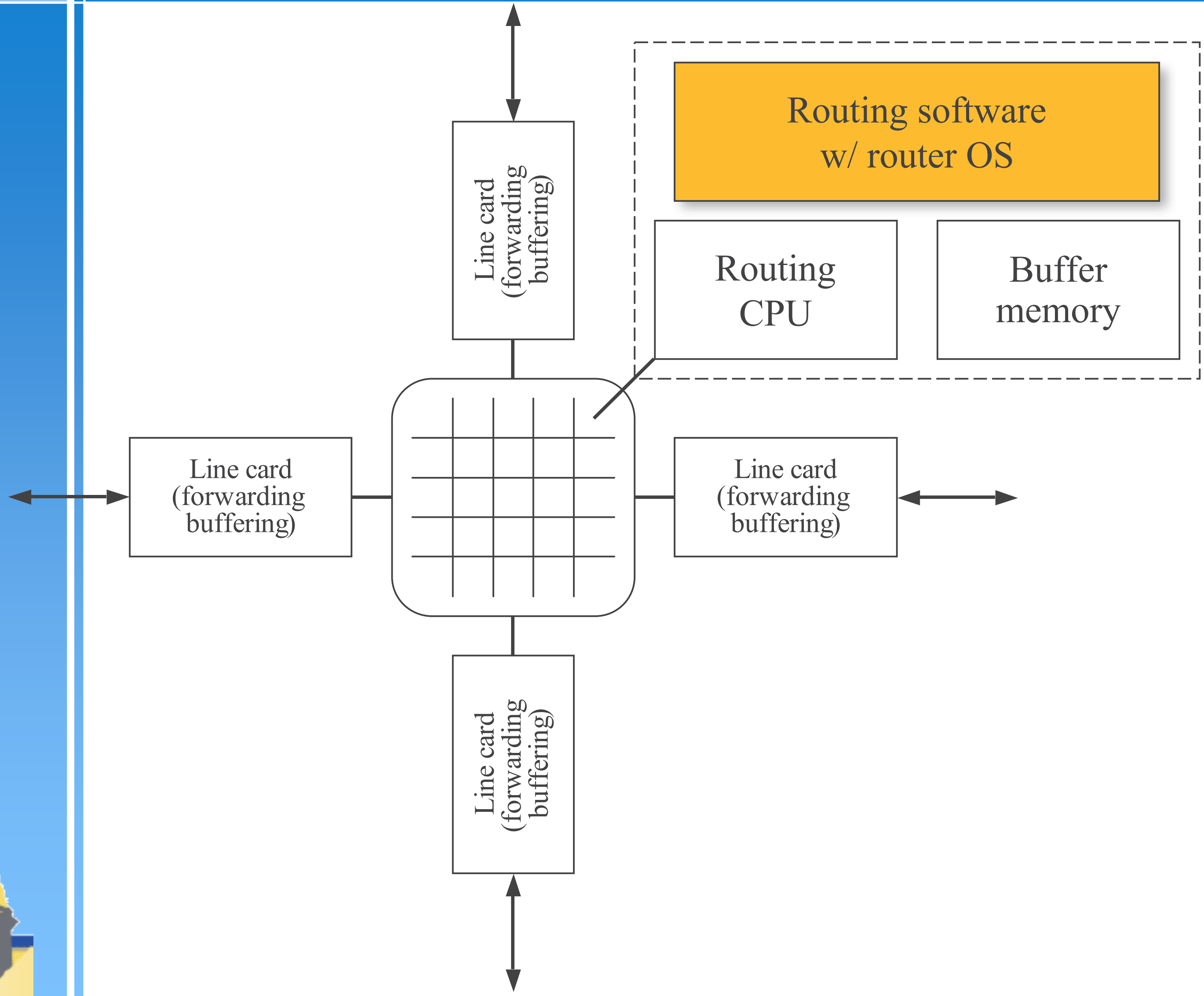


- Common Design: Batcher-Banyan Switch

High-Speed IP Router

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

High-Speed Router



Alternative Design