# Chapter 3 Part 3
# Switching and Bridging

Networking

CS 3470, Section 1

# Forwarding

- A switching device's primary job is to receive incoming packets on one of its links and to transmit them on some other link
  - This function is referred as ***switching and forwarding***
  - According to OSI architecture this is the main function of the network layer

# Forwarding

- How does the switch decide which output port to place each packet on?
  - It looks at the header of the packet for an identifier that it uses to make the decision
  - Two common approaches
    - ***Datagram or Connectionless*** *approach*
    - ***Virtual circuit or Connection-oriented*** *approach*

# Forwarding

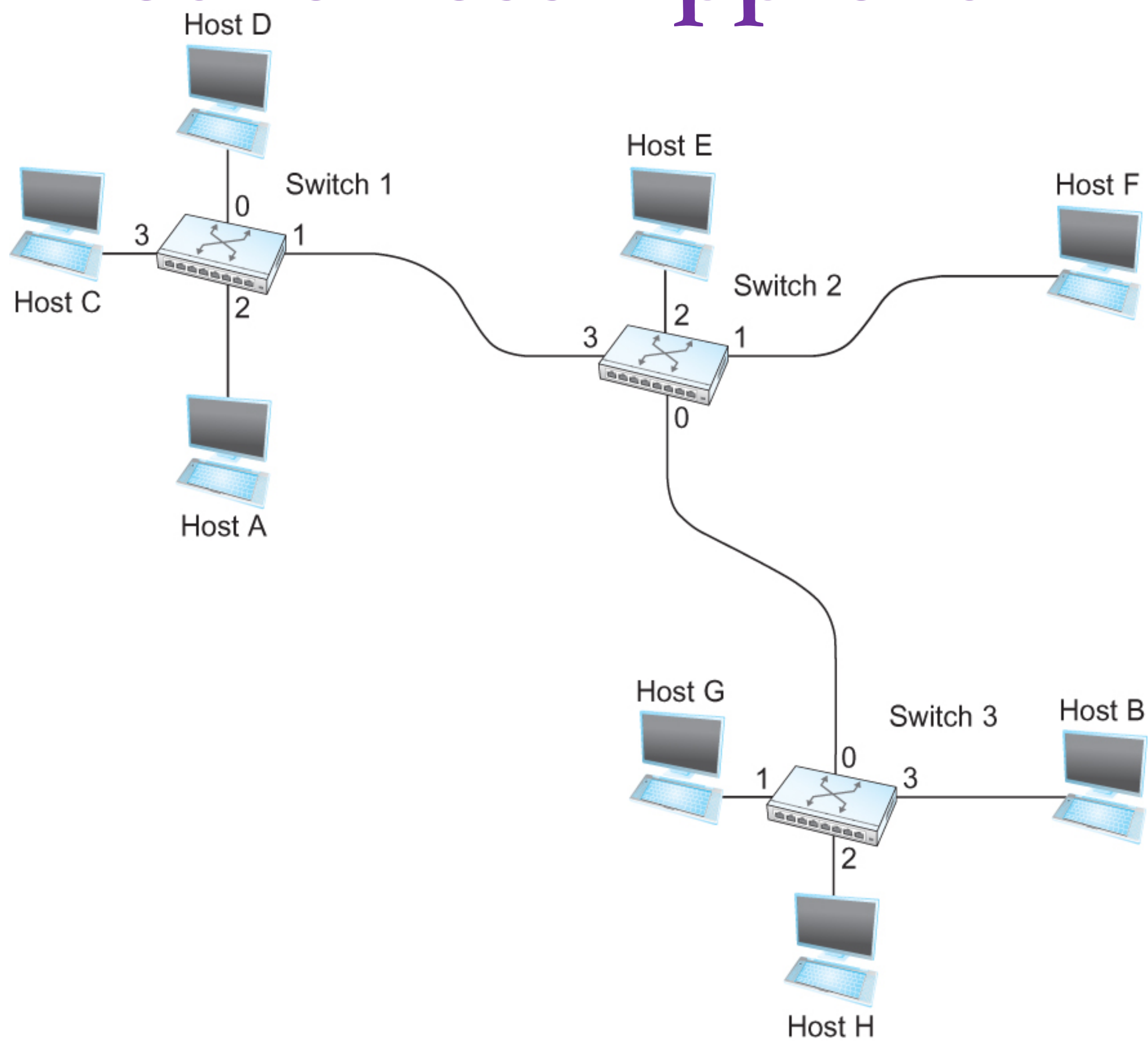
- Assumptions
  - Each host has a globally unique address
  - There is some way to identify the input and output ports of each switch
    - We can use numbers
    - We can use names

# Datagram / Connectionless Approach

- Key Idea
  - Every packet contains enough information to enable any switch to decide how to get it to destination
    - Every packet contains the complete destination address

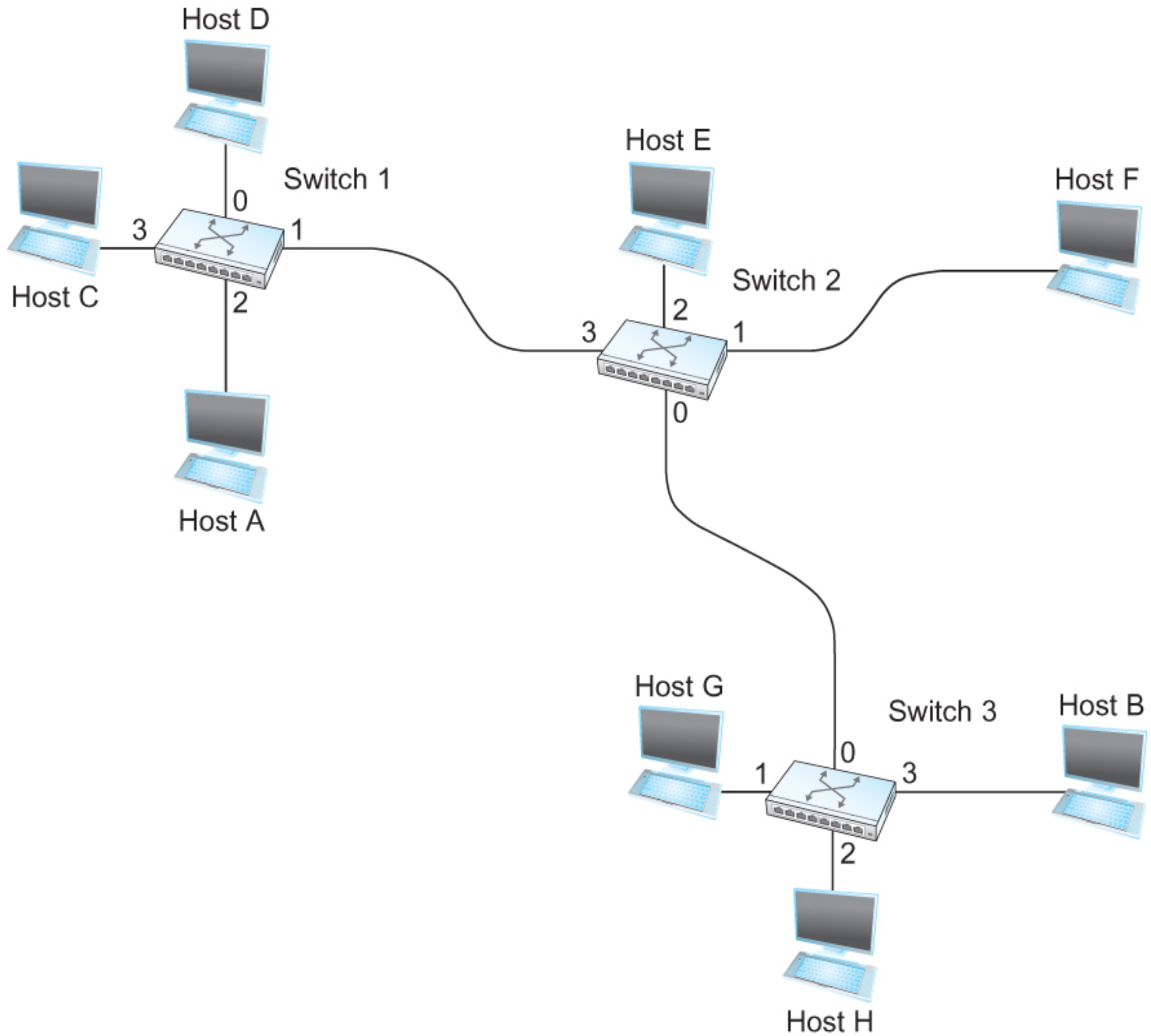
# Datagram / Connectionless Approach

An example network




- To decide how to forward a packet, a switch consults a ***forwarding table*** (sometimes called a ***routing table***)

# Datagram / Connectionless Approach

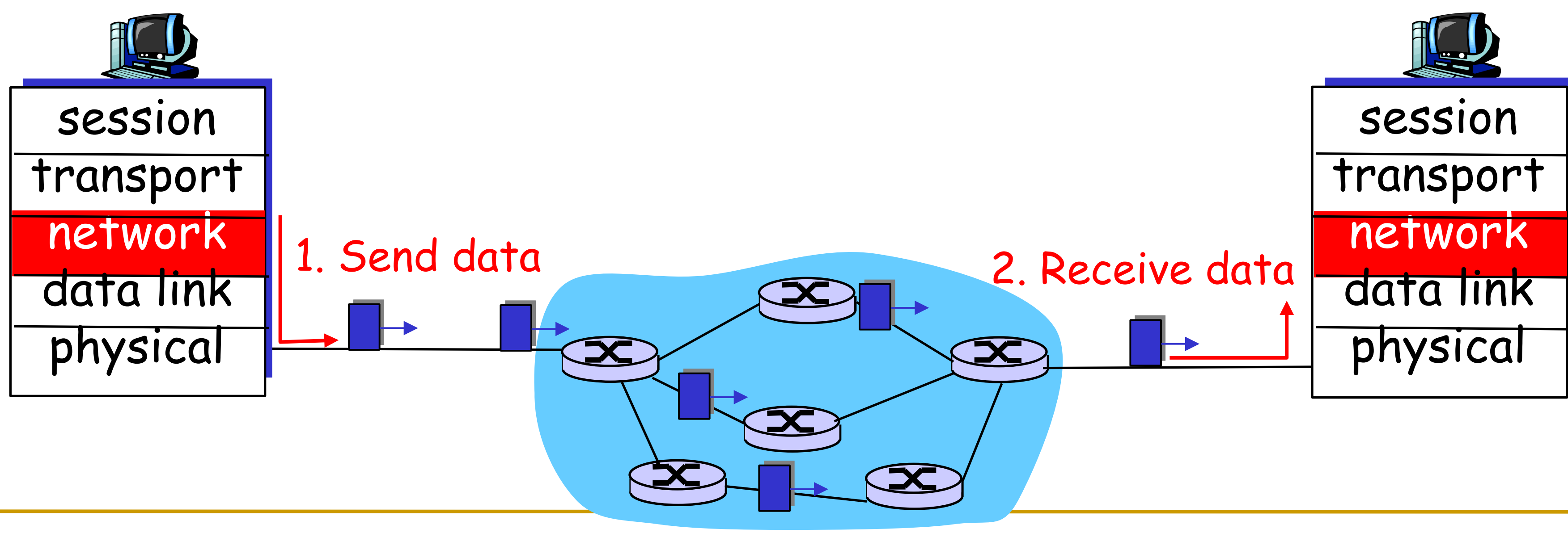



| Destination | Port |
| --- | --- |
| A | 3 |
| B | 0 |
| C | 3 |
| D | 3 |
| E | 2 |
| F | 1 |
| G | 0 |
| H | 0 |

**Forwarding Table for Switch 2**

# Datagram Networks: The Internet Model

- No call setup at network layer
- Routers: no state about end-to-end connections
  - no network-level concept of “connection”
- Packets forwarded using destination host address
  - packets between same source-dest pair may take different paths



# Virtual Circuit Switching

Virtual Circuit Switching

- Widely used technique for packet switching
- Uses the concept of *virtual circuit* (VC)
- Also called a ***connection-oriented model***
- First set up a virtual connection from the source host to the destination host and then send the data

# Virtual Circuit Switching

- Two-stage process
  - Connection setup
  - Teardown
- Each switch contains a VC table
  - ***Virtual Circuit Identifier (VCI)*** for incoming connection (also carried in header of packets)
  - Incoming interface of packet for this VC
  - Outgoing interface where the packet should be sent

# Virtual Circuit Switching

- VCIs are not global in the switch network
  - Link local scope – only has significance on given ***link***
  - VCI and interface uniquely define the virtual connection
- Outgoing packets may use a different VCI
- Virtual Circuits can be established before the circuit is needed or on demand

# Virtual Circuit Switching

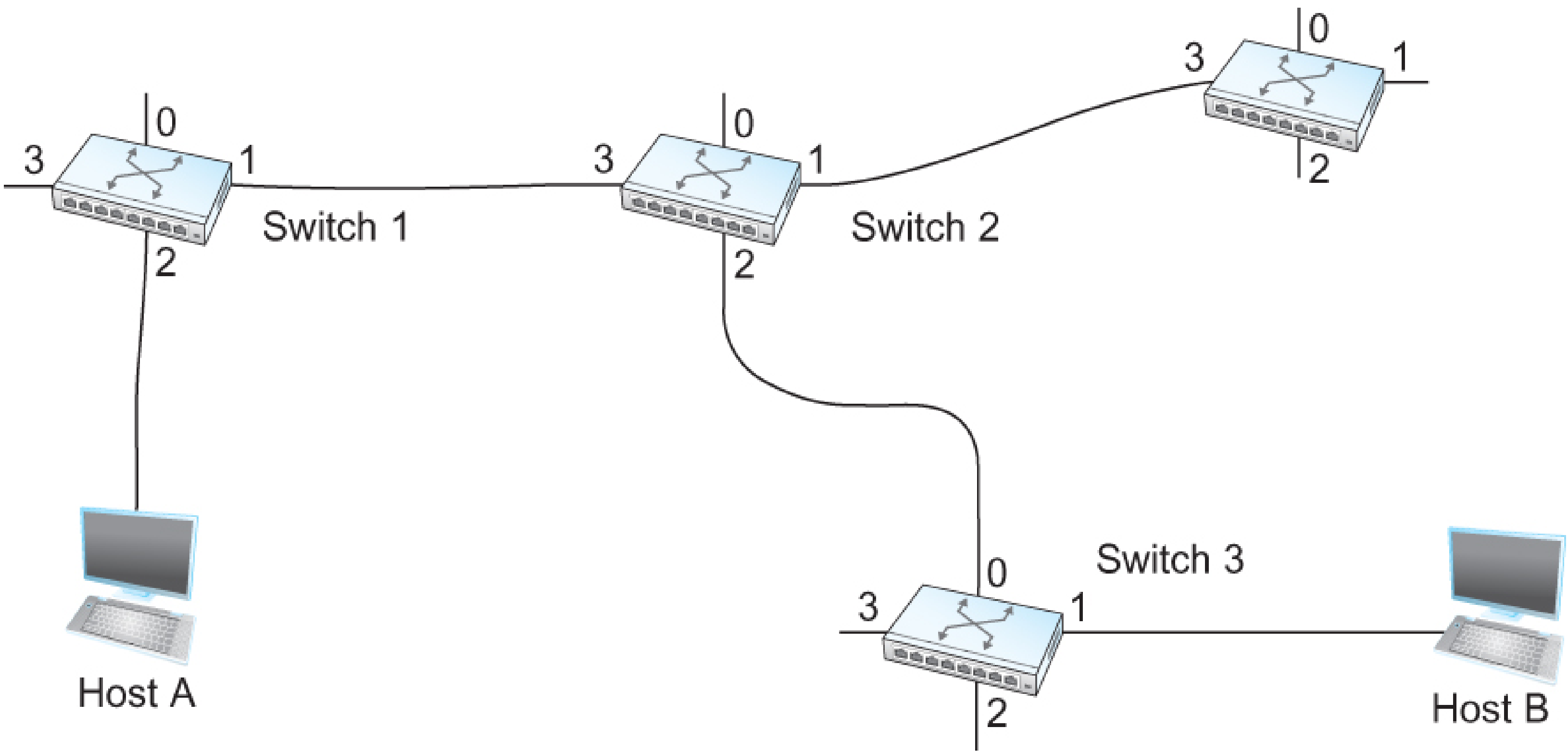
- Two types:
  - Permanent Virtual Circuit (PVC)
    - Network administrator configures the state
  - Switched Virtual Circuit (SVC)
    - Setup and teardown performed by the host requiring the circuit at the time of use

# Virtual circuit summary

- Call setup, teardown for each call before data can flow
- Each packet carries VC identifier (not destination host ID)
- Every router on source-destination path maintains “state” for each passing connection
  - transport-layer connection only involved two end systems
- Link, router resources (bandwidth, buffers) may be allocated to VC
  - to get circuit-like performance

# Virtual Circuit Switching Example

- Manually create a new virtual connection from host A to host B
- First the administrator identifies a path through the network from A to B



# Virtual Circuit Switching Example

- The administrator then picks a VCI value that is currently unused on each link for the connection
  - For our example,
    - Suppose the VCI value 5 is chosen for the link from host A to switch 1
    - 11 is chosen for the link from switch 1 to switch 2
    - 7 is chosen for the link from switch 2 to switch 3

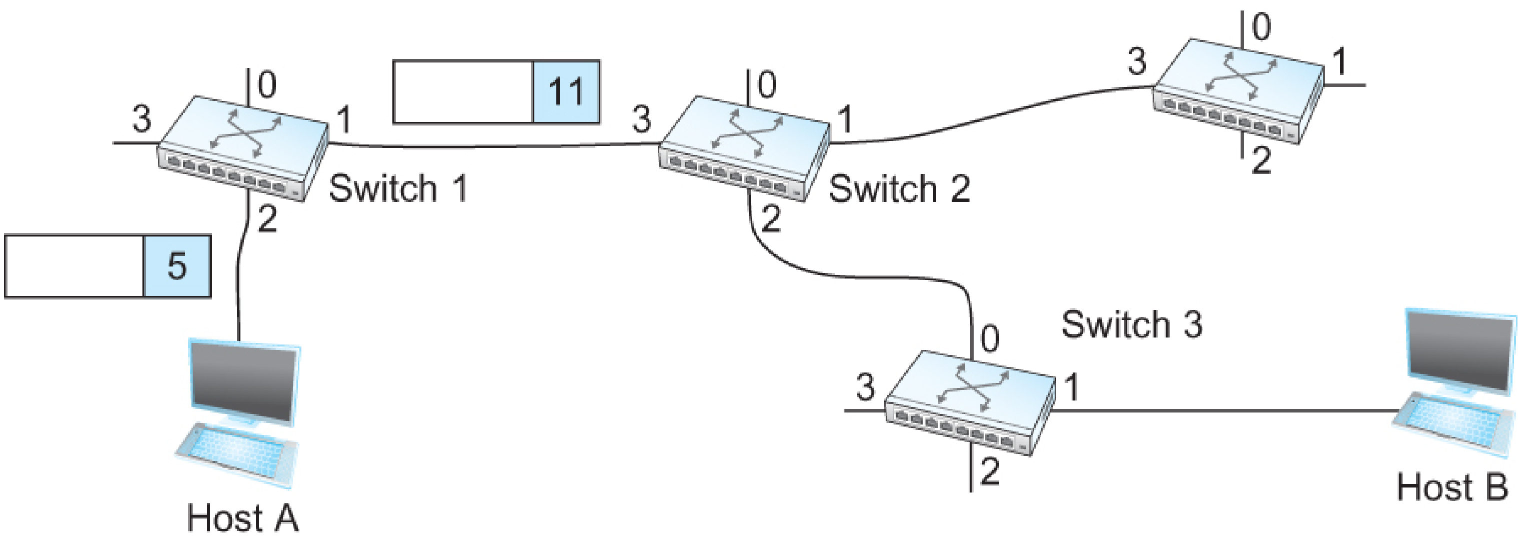
# Virtual Circuit Switching Example

**Switch 1 Table**

| Incoming Interface | Incoming VC | Outgoing Interface | Outgoing VC |
|---|---|---|---|
| 2 | 5 | 1 | 11 |

**Switch 2 Table**

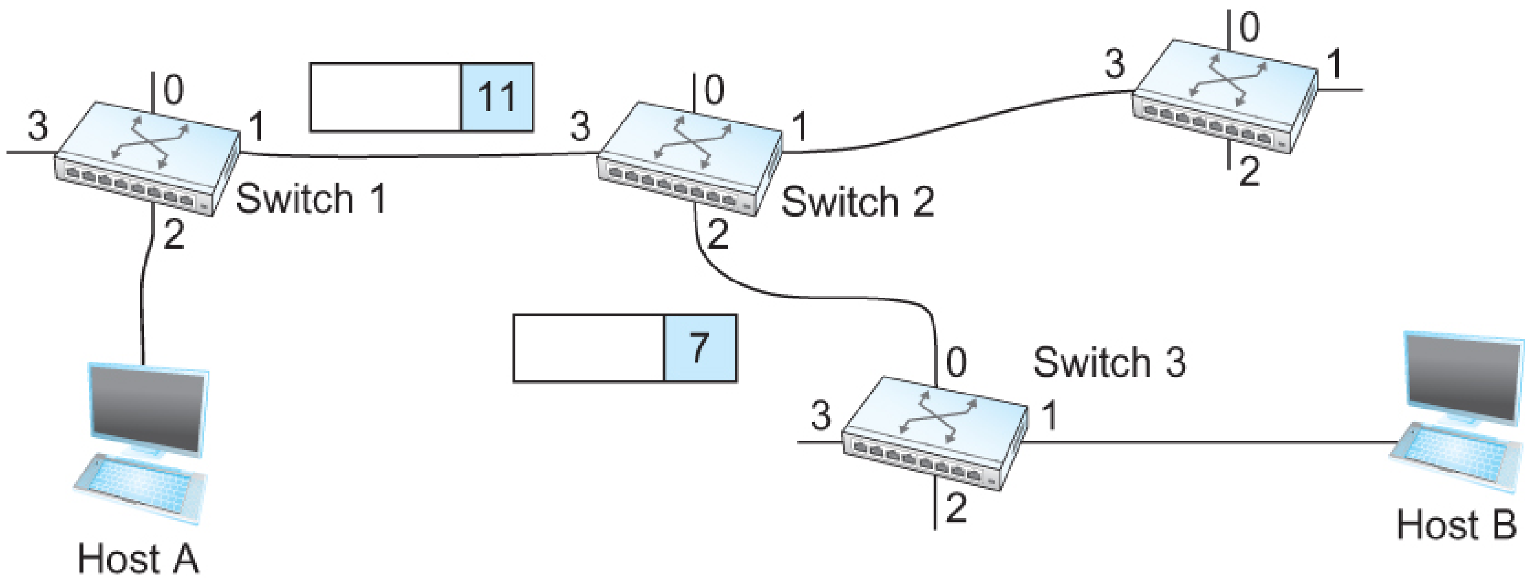
| Incoming Interface | Incoming VC | Outgoing Interface | Outgoing VC |
|---|---|---|---|
| 3 | 11 | 2 | 7 |



# Virtual Circuit Switching Example

**Switch 3 Table**

| Incoming Interface | Incoming VC | Outgoing Interface | Outgoing VC |
|---|---|---|---|
| 0 | 7 | 1 | 4 |



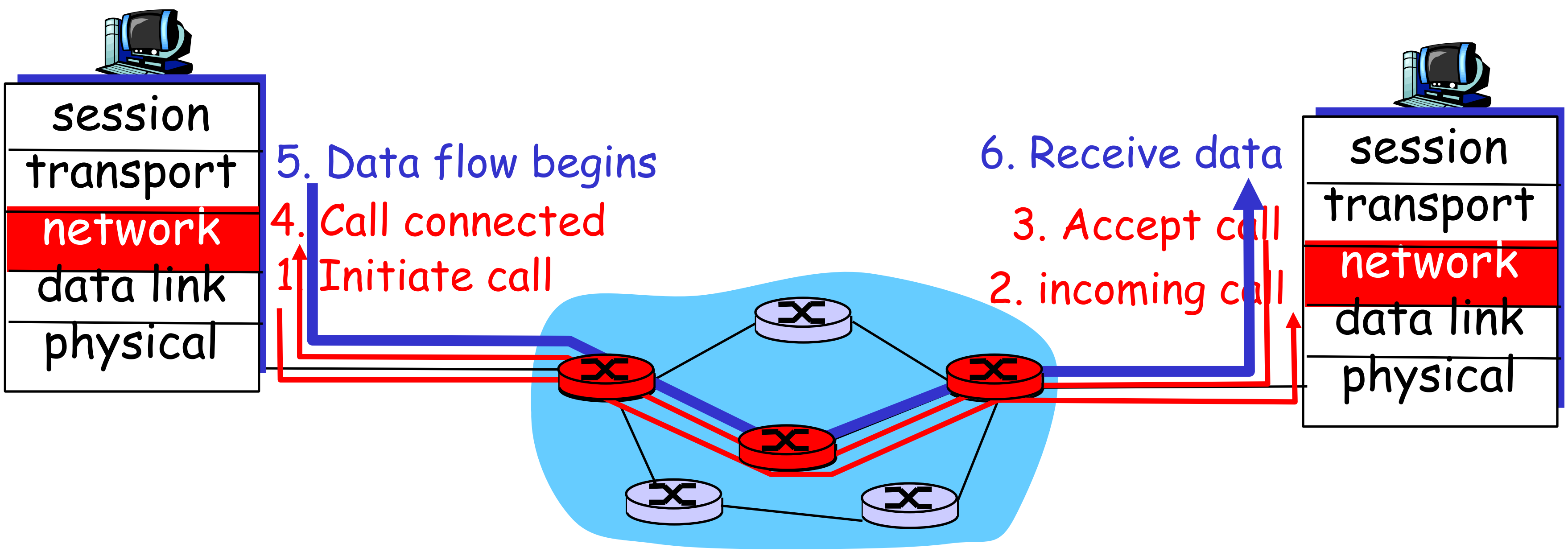

- What is the last VC going from Switch 3 to host B?

# Switched VC's: SVC Example

- Setting up the PVC's (no, not the plumbing pipe, a Permanent Virtual Circuit) in a large network, such as Qwest, can be overwhelming
- Most service providers use signaling of some sort to set up the VCs—even if they are PVCs
- Signaling for the PVC creation can be handled by the hosts or routers.

# Virtual circuits: Signaling Protocols

- Used to setup, maintain  teardown VC
- Used in ATM, frame-relay, X.25
- Not as popular as packet switching in today's Internet



# SVC Process

- Host A sends a setup message to switch 1
  - Contains the complete destination address of host B
  - This message needs to make its way all through the network to host B so that every switch can update its VC table.
- Switch 1 receives the request
  - Updates its VC table
  - Sends it to switch 2
- This continues until the request reaches host B
  - But how??
  - Switches must know enough about network topology

# SVC Process

- We have completed ½ of the process
- Host B now sends an ACK back to A
  - This behaves the same as the original setup request
  - Each switch receives the message
  - Updates the VC tables
  - Forwards the message on
- Every switch now knows the properties of the VC when the message reaches host A.

# SVC Process

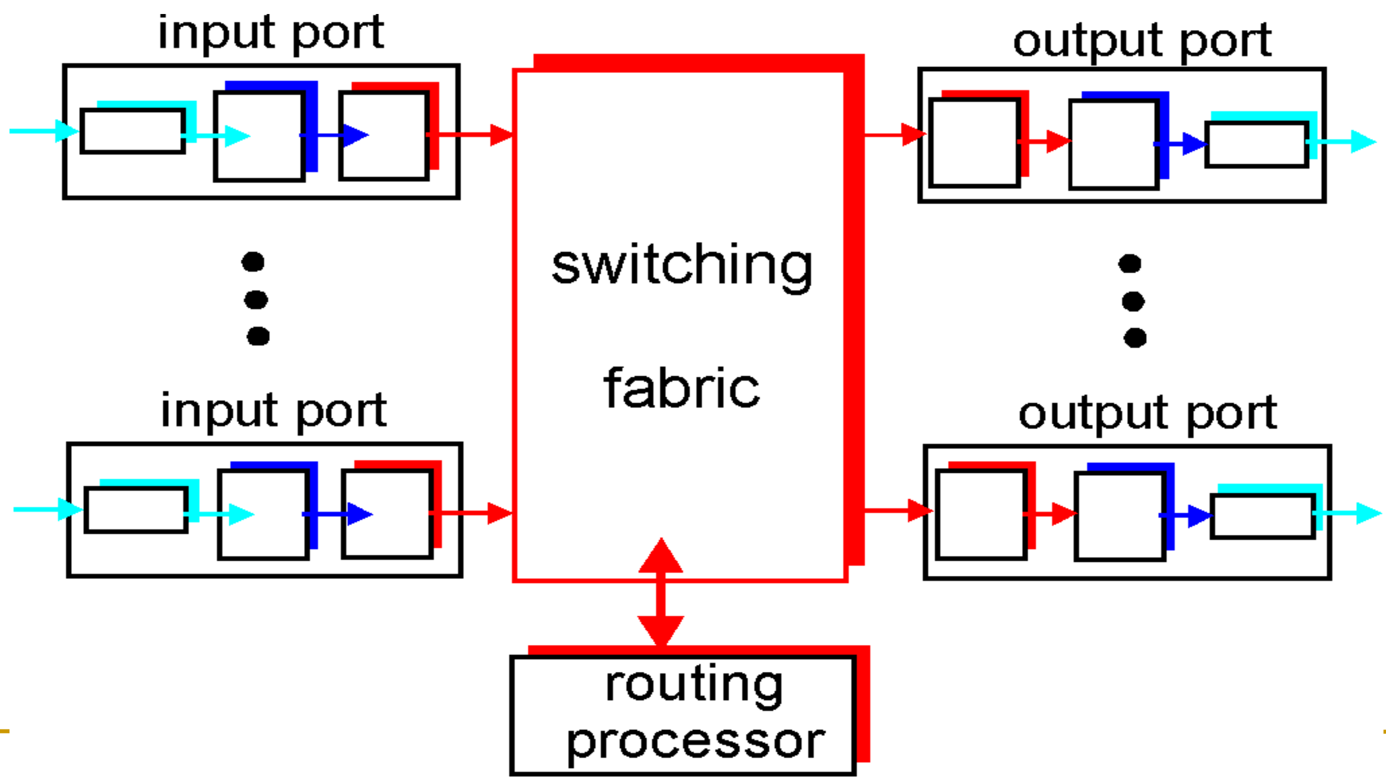
- When A (or B) is done with the connection, it sends a teardown message to the channel, say to switch 1
- Switch 1 forwards the packet to switch 2 and removes the VC entry for host A
- Switch 2 does the same, etc.
- When the teardown message has reached host b, the connection has been removed.

# SVC Notes

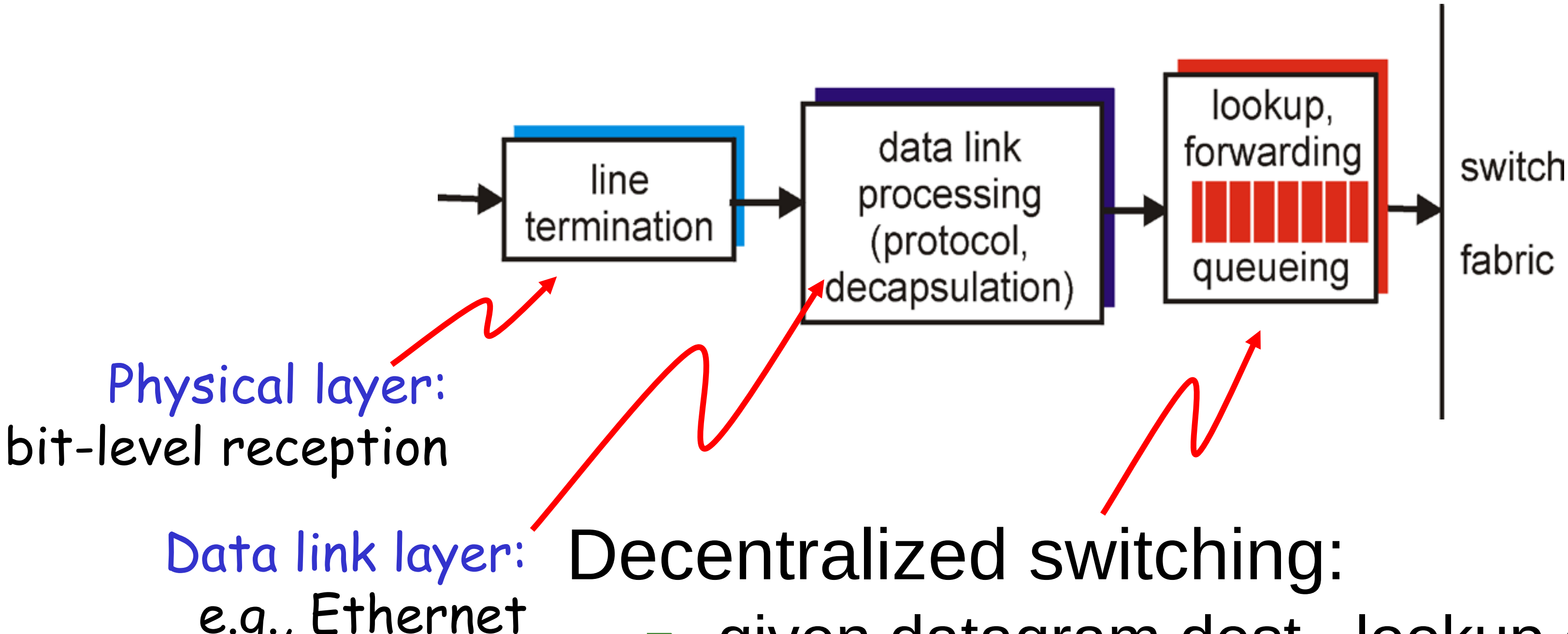
- Takes 1 full RTT to set up the path
- Buffers are allocated in the switches as the connection is set up.
- Advanced sliding windows keep the remote nodes behaving
- Circuit setup requests are rejected if a node does not have enough buffers
- Hop-by-hop flow control.

# Router Architecture Overview

- Two key router functions:
  - run routing algorithms/protocol (RIP, OSPF, BGP)
  - switching datagrams from incoming to outgoing link



# Input Port Functions

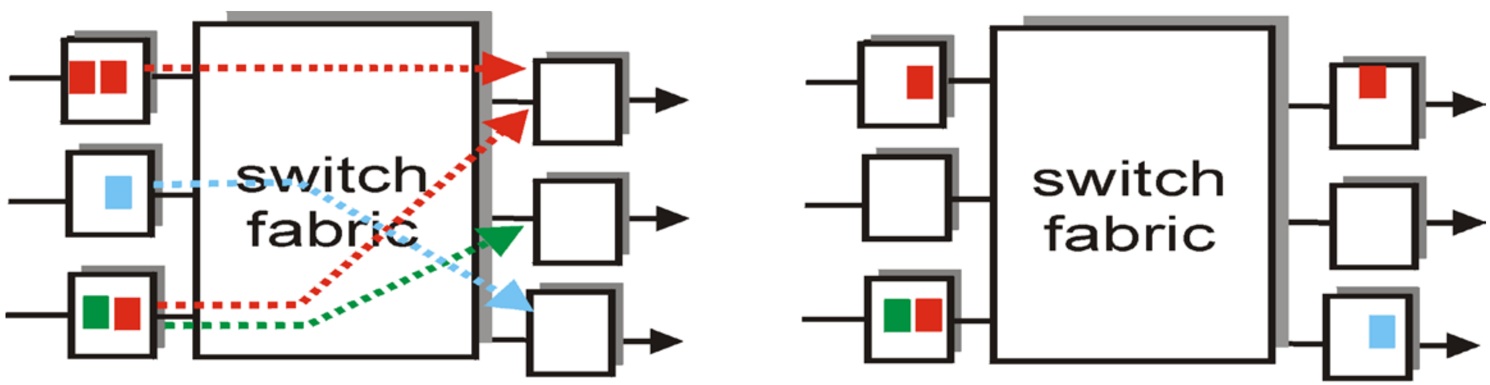



Decentralized switching:

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

# Input Port Queuing

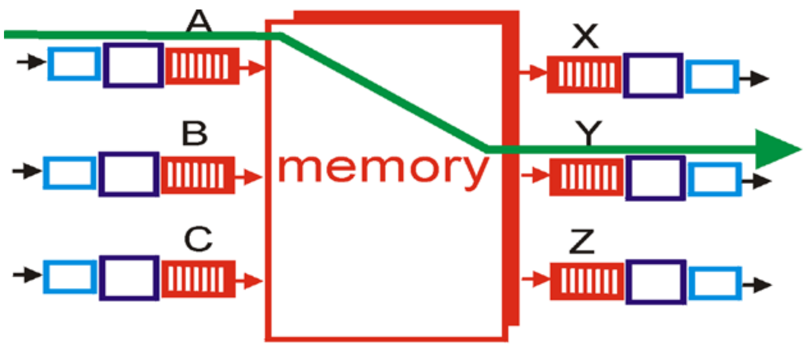
- Fabric slower that input ports combined -> queueing may occur at input queues
- Head-of-the-Line (HOL) blocking: queued datagram at front of queue prevents others in queue from moving forward
- ***queueing delay and loss due to input buffer overflow!***



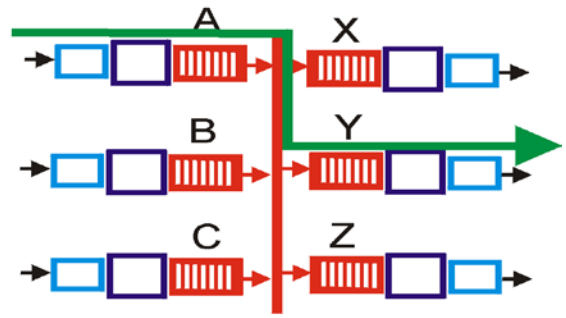

output port contention at time t - only one red packet can be transferred

green packet experiences HOL blocking

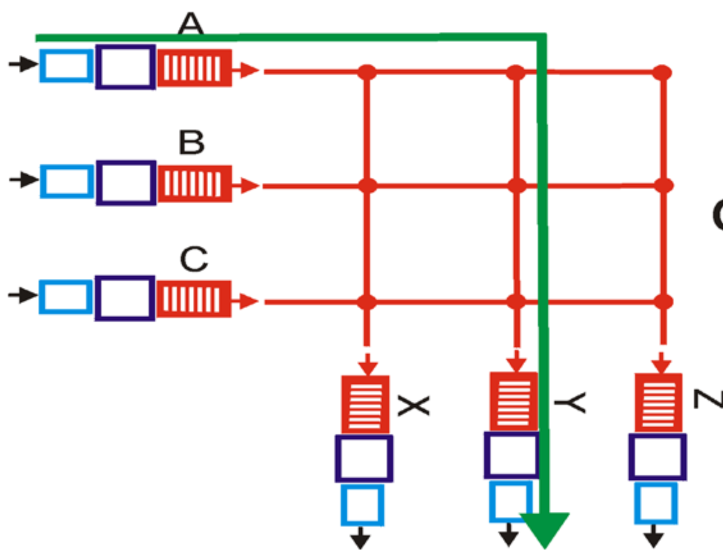
# Three Types of Switching Fabrics




memory



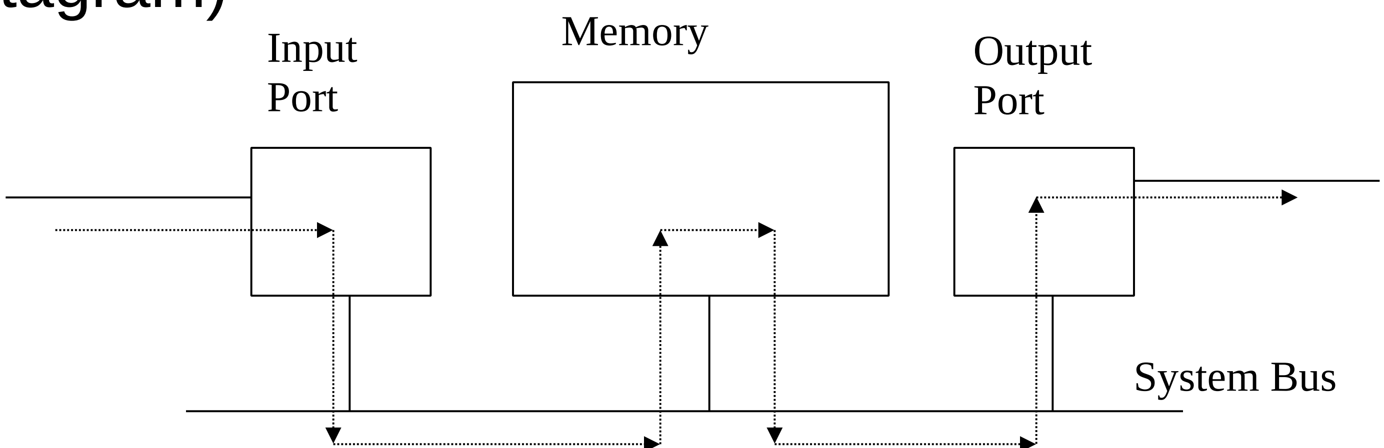

bus




crossbar

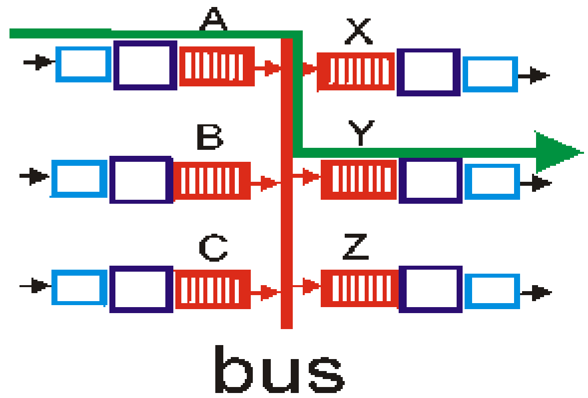
# Switching Via Memory

- First generation routers:
  - packet copied by system's (single) CPU
  - speed limited by memory bandwidth (2 bus crossings per datagram)




- Modern Routers
  - Input port processor performs lookup, copy into memory
  - Cisco Catalyst 8500

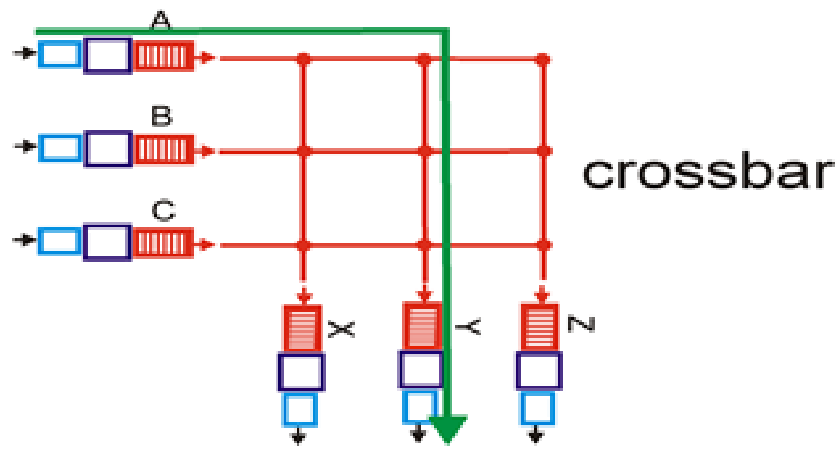
# Switching Via a Bus




- 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)

# Switching via Crossbar

- Matrix of pathways that can be configured to connect any input port to any output port



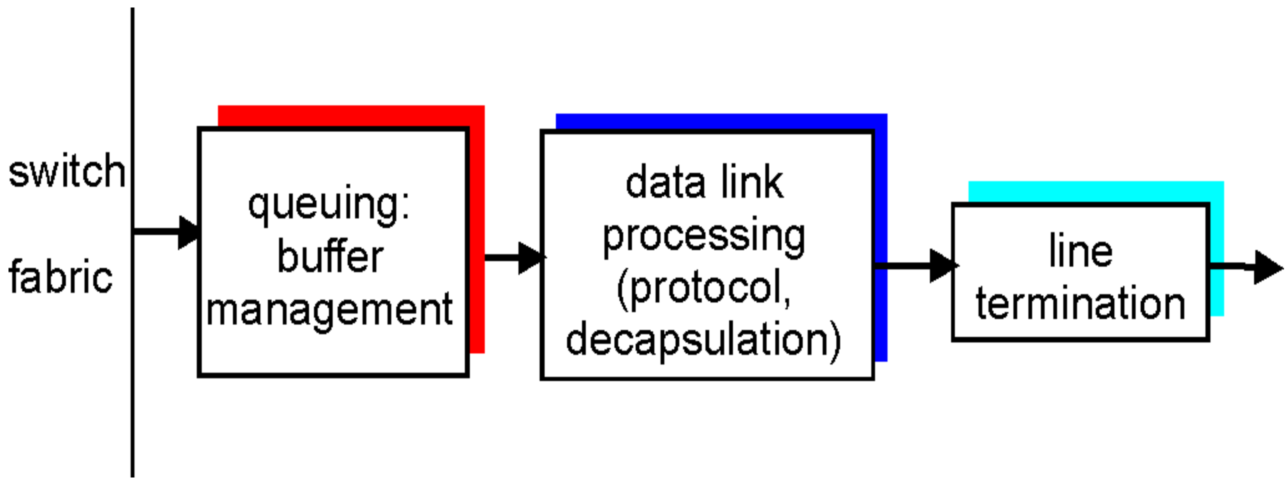

- Biggest problem is that they require output port to accept packets from all inputs at once
  - Implying each port has memory bandwidth equal to total switch throughput
- In reality, more complex designs

# Self-Routing Design

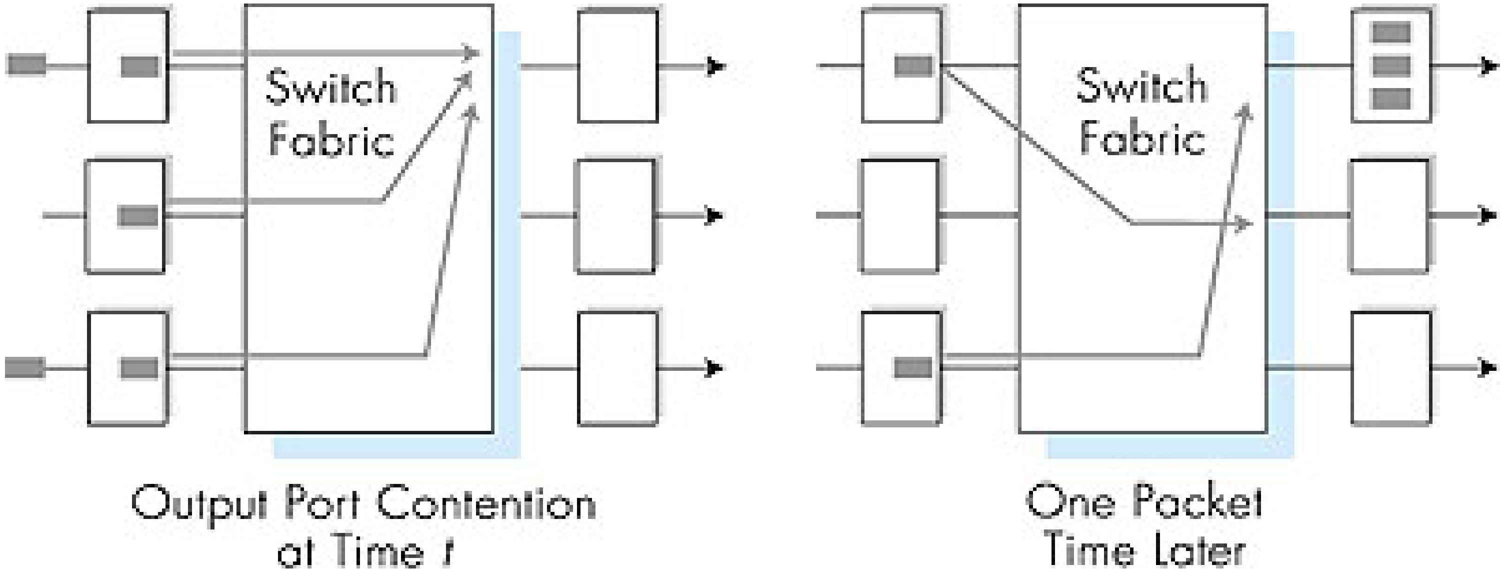
- One of most scalable approaches of fabric design
- Rely on information in packet header to direct each packet to correct output
  - "Self-routing header" is appended to packet by input port
  - Extra header is removed after packet leaves the switch
- Book example: Banyan networks
- Cisco 12000: switches Gbps through the interconnection network

# Output Ports

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



# Output port queueing




- Buffering when arrival rate via switch exceeds output line speed
- ***Queuing (delay) and loss due to output port buffer overflow!***