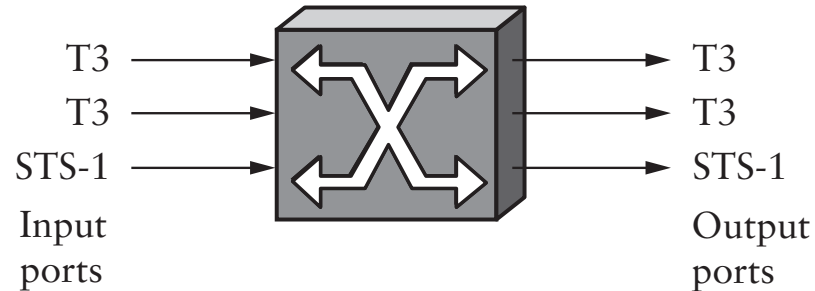


# Switching



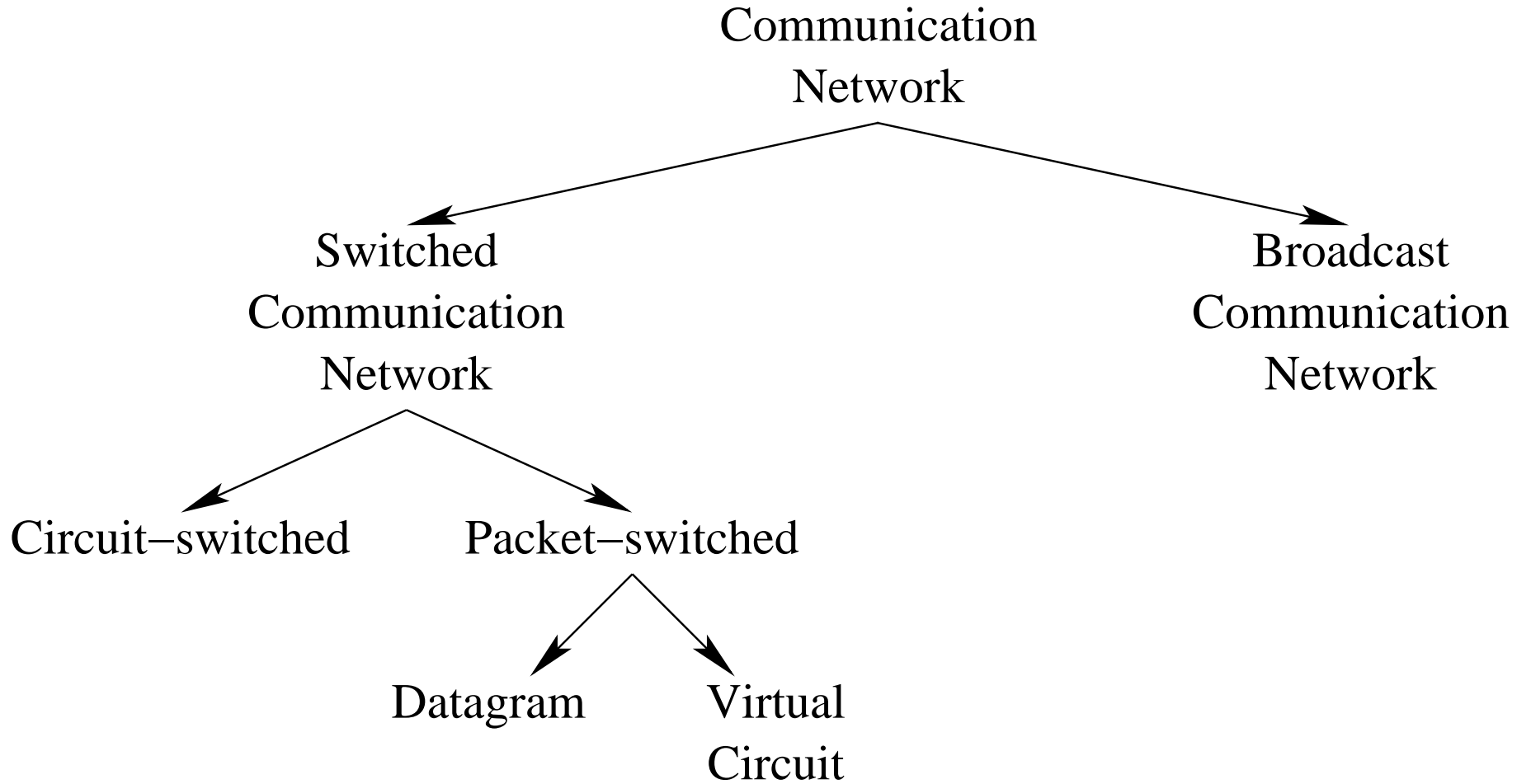
- **Switch**

- Forwards packets from input port to output port
- Port selected based on the packet header

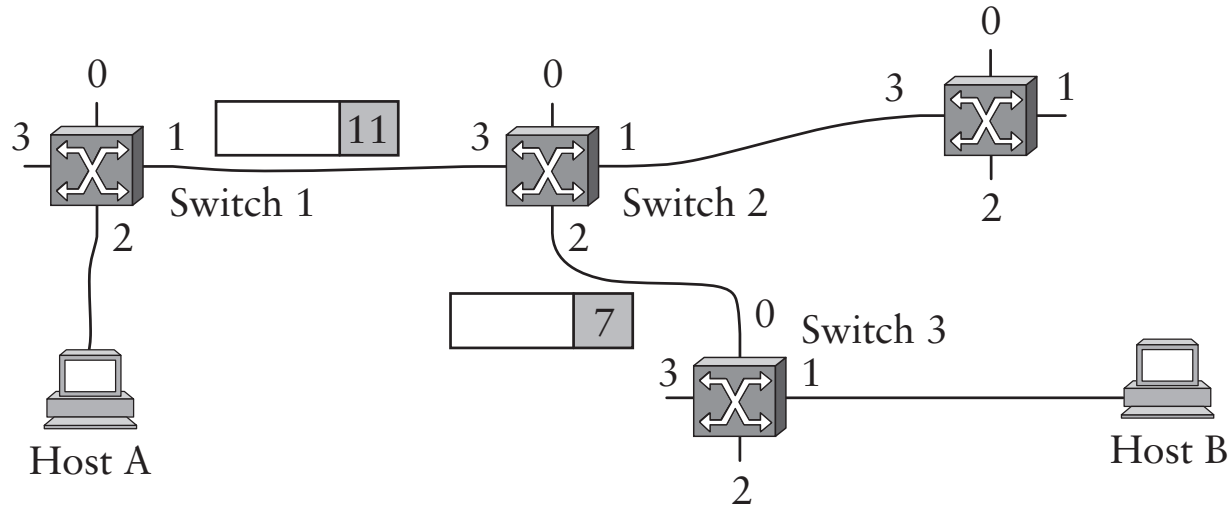
- **Advantages**

- Cover large geographic area (tolerate latency)
- Support large numbers of hosts (scalable bandwidth)

# Big Picture



# Virtual Circuit Switching

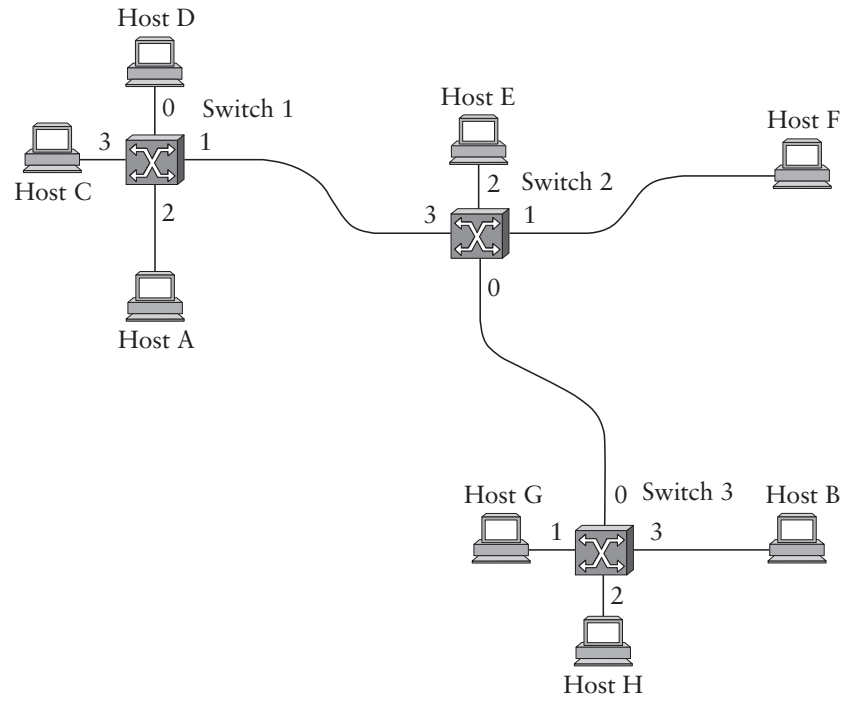


- **Explicit connection setup (and tear-down) phase**
  - Establishes virtual-circuit ID on each link
  - Each switch maintains VC table
- **Subsequent packets follow same circuit**
  - Switch maps  $\langle \text{in-link, in-VCI} \rangle \rightarrow \langle \text{out-link, out-VCI} \rangle$
- **Sometimes called *connection-oriented model***

## Virtual Circuit Model

- **Signaling requires full RTT for connection setup before sending first data packet.**
- **Connection request contains the full address for destination, but data packets contain only a small identifier.**
- **When switch or link fails, the circuit is broken and a new one needs to be established.**
- **Connection setup provides an opportunity to reserve resources (buffers, bandwidth).**

# Datagram switching

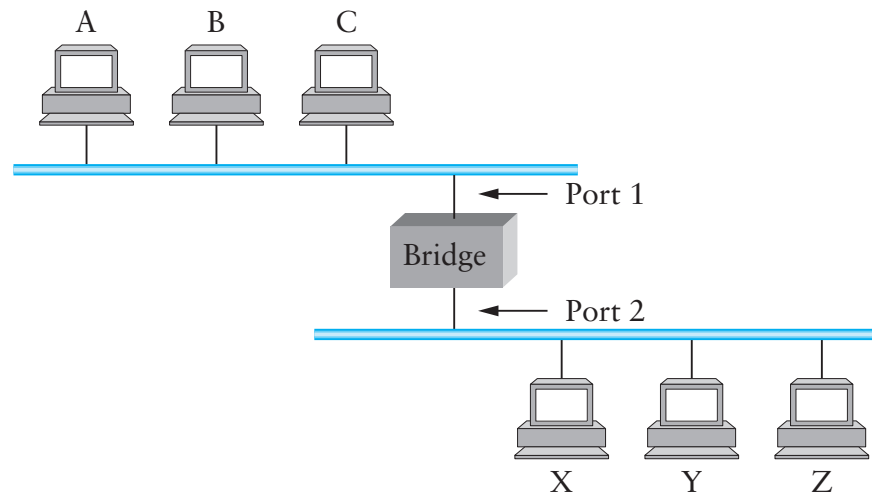


- **No connection setup phase.**
  - Switches have routing table based on node addresses.
- **Each packet forwarded independently.**
- **Sometimes called connectionless model.**

# Datagram Model

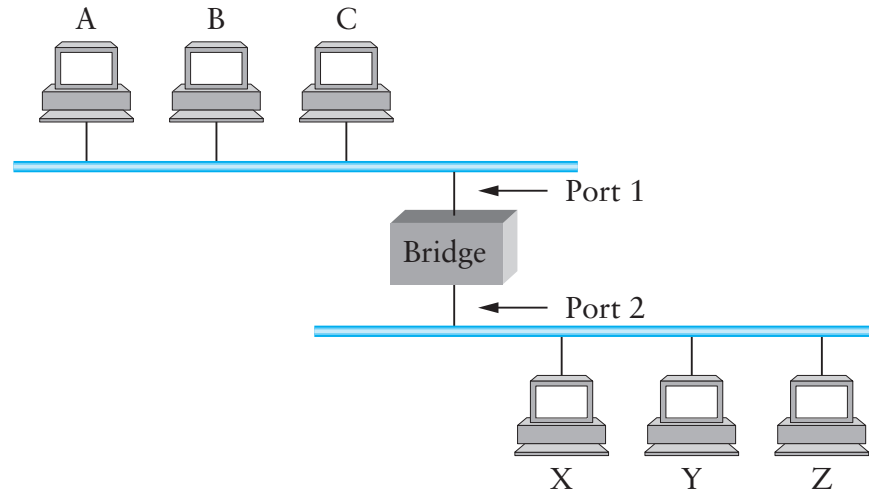
- No RTT delay for connection setup.
- It is possible to route around link and node failures.
- Source cannot know if the network is capable of delivering a packet.
- The overhead per packet is higher than for the connection-oriented model.

# Bridges and extended LANs



- LANs have physical limitations (*e.g.*, 2500m)
- Connect two or more LANs with a *bridge*
  - Operates on MAC address, not IP.
  - No encapsulation required
- Ethernet switch acts like a multi-way bridge

# Learning bridges

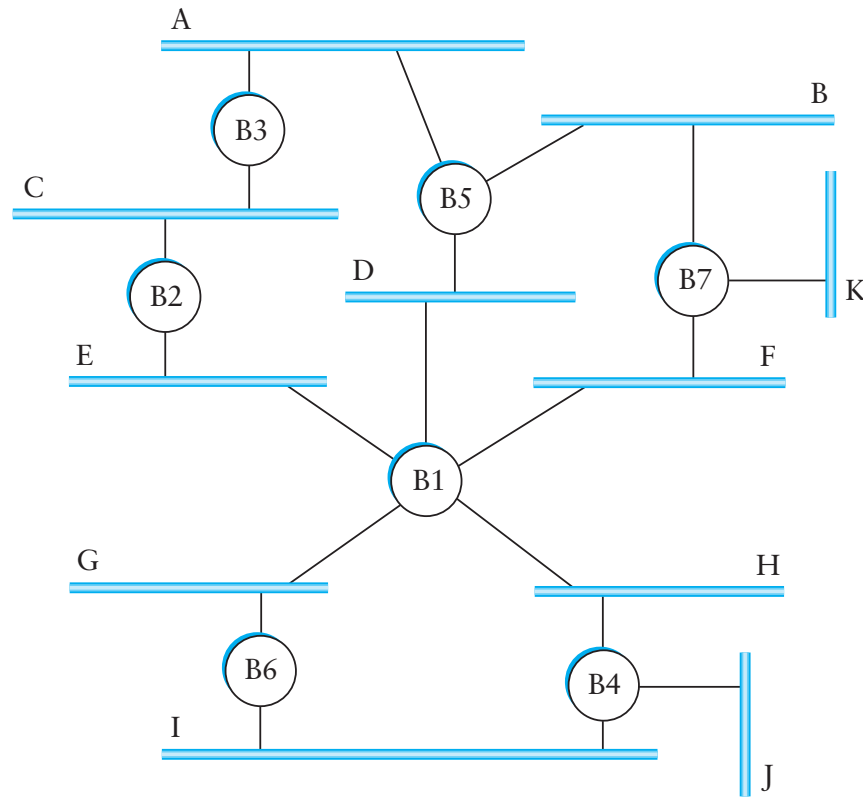


- **Idea: Don't forward packet where it isn't needed.**
  - If you know recipient is not on that port
- **Learn host's location based on source address.**
  - Switch builds a table when it receives packets

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

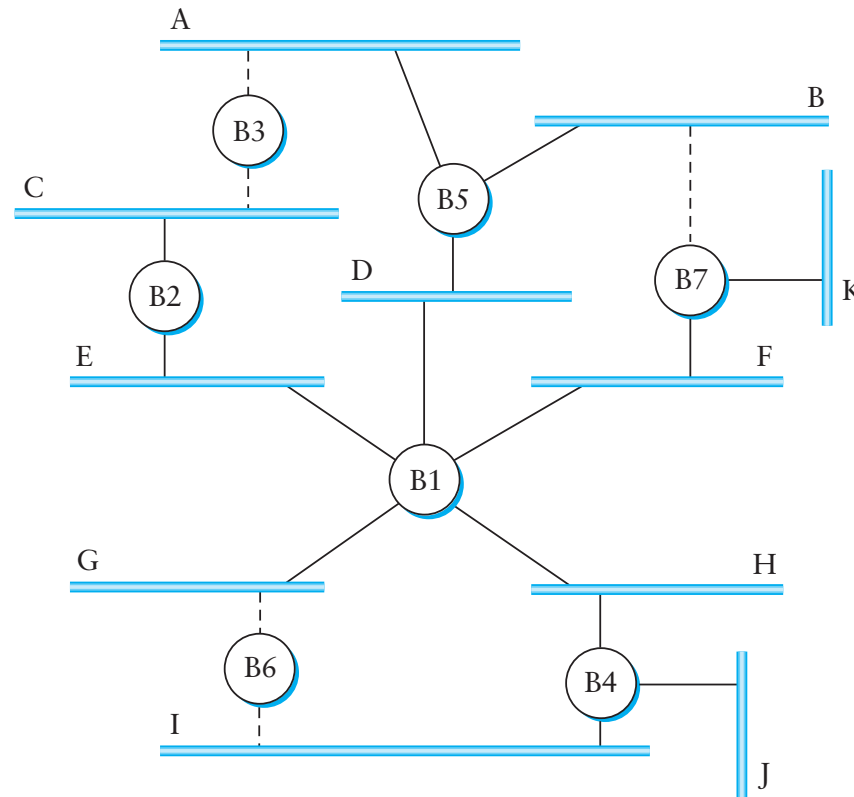
- **Table says when *not* to forward packet.**
  - Does not need to be complete for correct behavior.

# Dealing with loops



- **Problem: People might form loops**
  - Accidentally, or to provide redundancy if link fails
  - Don't want to forward packets infinitely!

# Spanning tree



- **Need to disable ports, to avoid loops in network.**
- **Like creating a spanning tree in a graph.**
  - View switches and networks as nodes, ports as edges

# Distributed spanning tree algorithm

- Every bridge has a unique ID (Ethernet address)
- Let bridge with smallest ID be the **root**
- Each segment has one **designated bridge** responsible for forwarding packets over it.
  - Bridge closest to root is designated bridge.
  - If there is tie, smallest ID is designated bridge.
- **Overview:**
  - Each node assumes it is the root (and thus the dedicated bridge for all its ports)
  - Sends messages, may learn of a better root.
  - Eventually links from nets to designated bridges form a spanning tree.

# Spanning tree protocol

- **Spanning tree messages contain:**
  1. ID of bridge sending message
  2. ID sender believes is the root
  3. Distance (in hops) from sender to root
- **Bridges remember best config message on each port**
- **Send message when you think you are the root**
- **Otherwise, forward messages from best known root**
  - Add one to distance before forwarding
  - Don't forward if you know you aren't dedicated bridge
- **In the end, only root is generating messages**

# Multicast and Broadcast

- **Forward all broadcast/multicast frames**
  - current practice
- **Could learn when no group members downstream**
  - Have each member of group  $G$  periodically send a frame to bridge multicast address with  $G$  in source field

# Limitations of bridges

- **Scaling**

- Spanning tree algorithm does not scale.
- Broadcast does not scale.
- No way to route around congested links.
- Two “close” nodes may be far in the tree.

- **Caution: beware of transparency**

- Some applications could get confused by bridge
- Much more likely to drop packets (with congestion)
- Makes latency between nodes very non-uniform

# ATM

- **Connection-oriented, packet-switched network**
  - Used by telco, often what goes over SONET
- **Based of fixed-size, 53-byte packets called *cells***
  - 5-byte header + 48-byte payload
- **Signaling protocol (Q.2931) can make various Quality of Service (QoS) guarantees**
- **Used for very high speed LAN backbones**
  - Was fastest thing you could get for a while

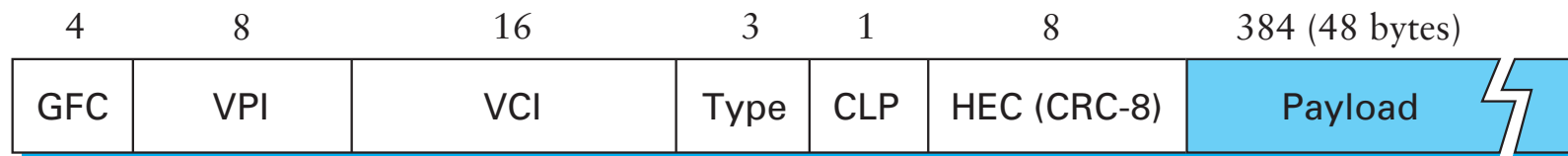
# Why cells?

- **Easier to implement in hardware**
  - Implementing functionality in hardware is faster
  - Easier to build hardware for fixed packet lengths
- **Easier to have parallelism in implementation**
  - Simple building block to process packet in fixed time
  - Process parallel packets & complete simultaneously
- **Queue management is much easier**
  - Can switch between flows at very fine granularity
  - Especially important for controlling delay & jitter
  - Never get stuck waiting to finish sending long packet
- **Small delay from store & forward**
  - Consider voice samples @ 1 byte / 125  $\mu$ s

## Why 53 bytes?

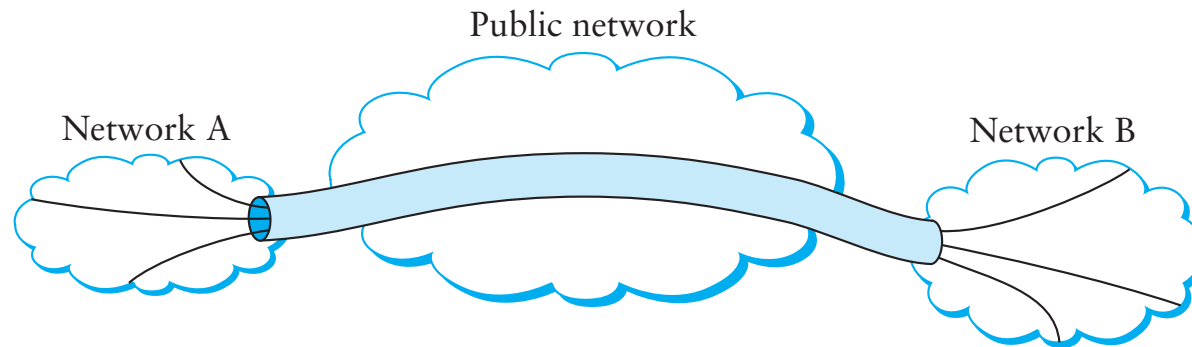
- **Packets have 48 bytes payload + 5 byte header**
- **The issue: echo cancellation needed if long delay**
  - Echo cancellation equipment is expensive
  - Unnecessary with 32B packetization delay + RTT across France
  - But U.S. needs equipment anyway, so wanted more efficient 64B
- **The result: Compromise at 48 bytes**
  - Less efficient *and* requires echo cancellation
  - Moral of the story: Design by committee sucks

# ATM cell format



- **GFC** – Generic flow control (mostly unused)
- **VPI/VCI** – Virtual path/circuit identifier
- **CLP** – Cell loss priority
- **HEC** – Header error correction (header only)

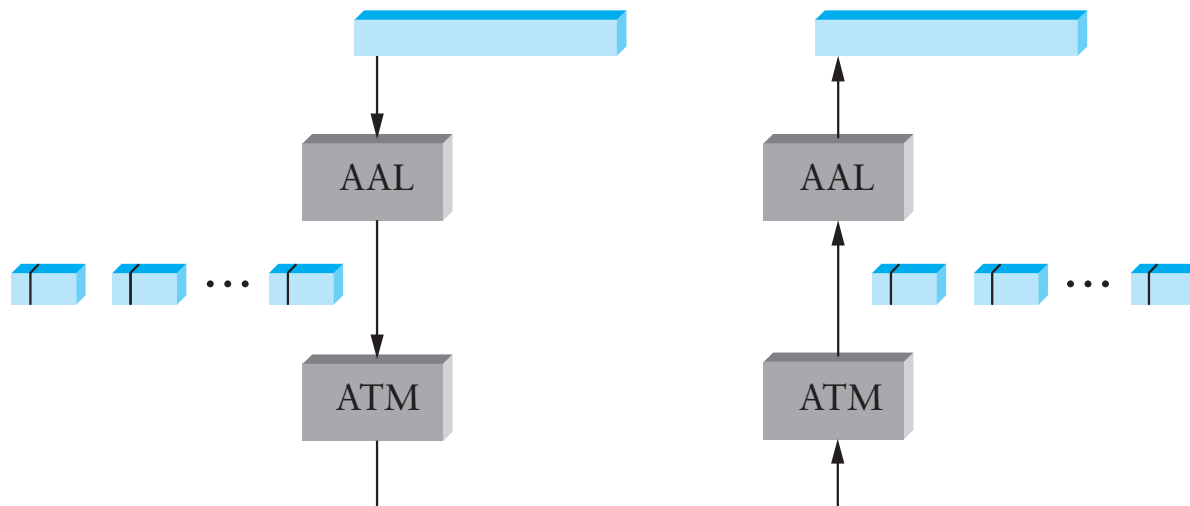
# VPI vs. VCI



- **Public networks care only about VPI**
- **Leaves VCI for individual sites**
  - Another level of demultiplexing
  - Many machines on net A talk to many machines on net B, with only one VPI.
  - Limits state in the public network

# Segmentation and reassembly

- Recall *encapsulation* from first lecture
  - One layer (say IP) sends packet to lower layer (say Ethernet)
  - Ethernet prepends its header, IP header now Ethernet payload
- Doesn't work so well with 48-byte ATM cell payloads!
- Solution: **AAL** – *ATM adaptation layer*
  - Segments larger packets into cells, reassembles later



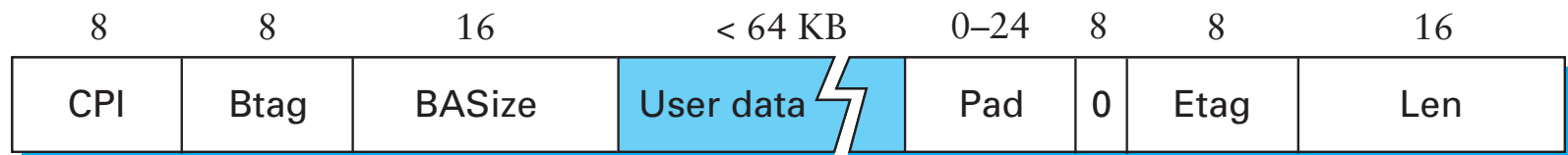
## Several different AAL layers

- Different formats for different needs
- **AAL 1/2** – for apps like voice needing guarantees
- **AAL 3** – connection-oriented packet services (X.25)
- **AAL 4** – datagram services (like IP)
  - Distinction between 3 & 4 turns out not to matter
- **AAL 5** – saner, simpler replacement for AAL 3/4

## AAL 3/4 CS-PDU

- **CS-PDU** – *convergent sublayer protocol data unit*

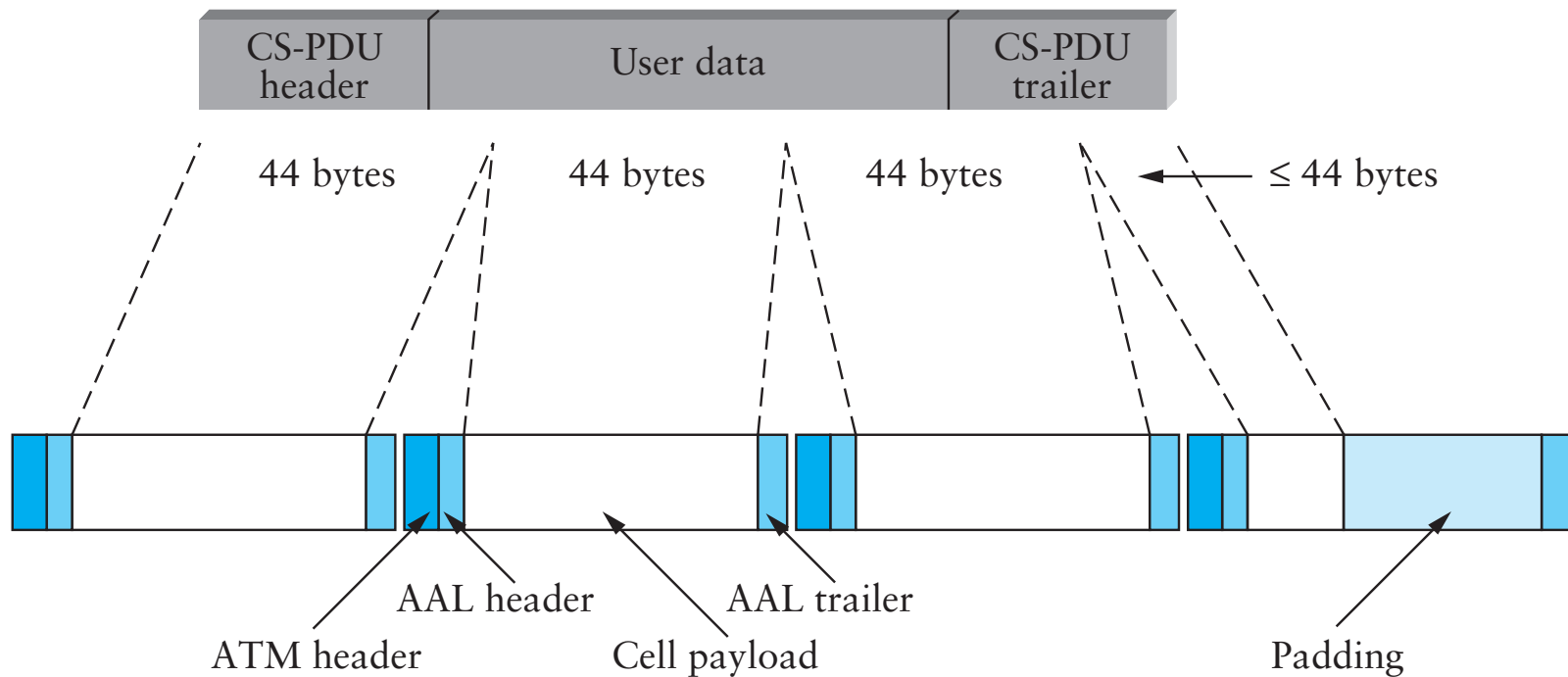
- Encapsulates higher-level protocol packets, e.g., IP



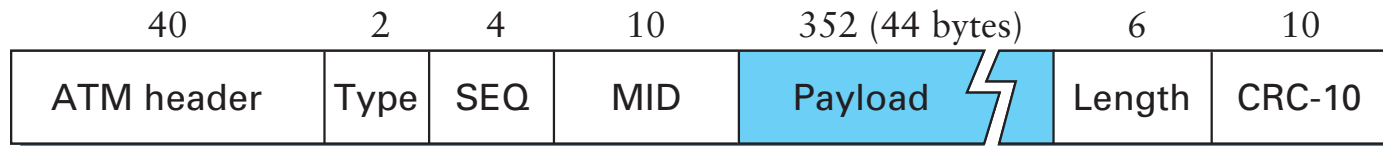
- **CPI** – common part indicator (version) always 0
- **Btag/Etag** – must match (avoids framing errors)
- **BAsize** – buffer allocation size is hint, not length of packet

# Packing AAL 3/4 CS-PDUs into cells

- Break CS-PDU into cells, each with AAL header/trailer



## AAL 3/4 header/trailer format

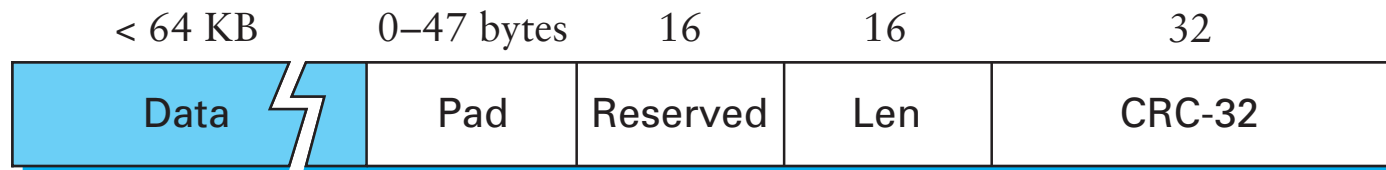


- **Type** – first/last/middle cell, or single-cell CS-PDU
- **MID** – *multiplexing identifier*
  - Allows multiple CS-PDUs to be interleaved on one VC
- **SEQ** Sequence number, detects loss/reordering
- **CRC** Detects bit errors in cell

# AAL 5

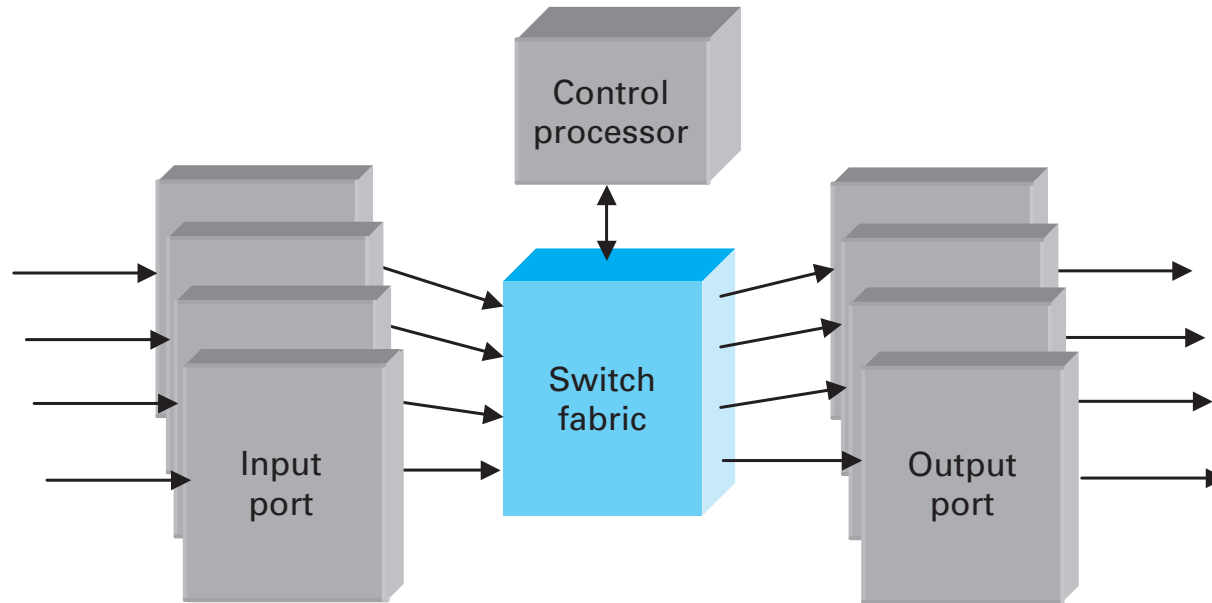
- **AAL 3/4 uses up too many bits of payload**
  - Really just want to know when end of frame is...
  - Idea: Why not just co-opt one bit in 5-byte ATM header?

- **AAL 5 CS-PDU:**



- Broken into 48 byte chunks and placed in cells
  - Stolen header bit indicates end of frame
- **What's missing compared to AAL 3/4 cells?**
  - No MID (means you can't interleave within VC)
  - No per-cell SEQ, CRC – but stronger CRC in CS-PDU will catch dropped/reordered packets

# Generic switch architecture

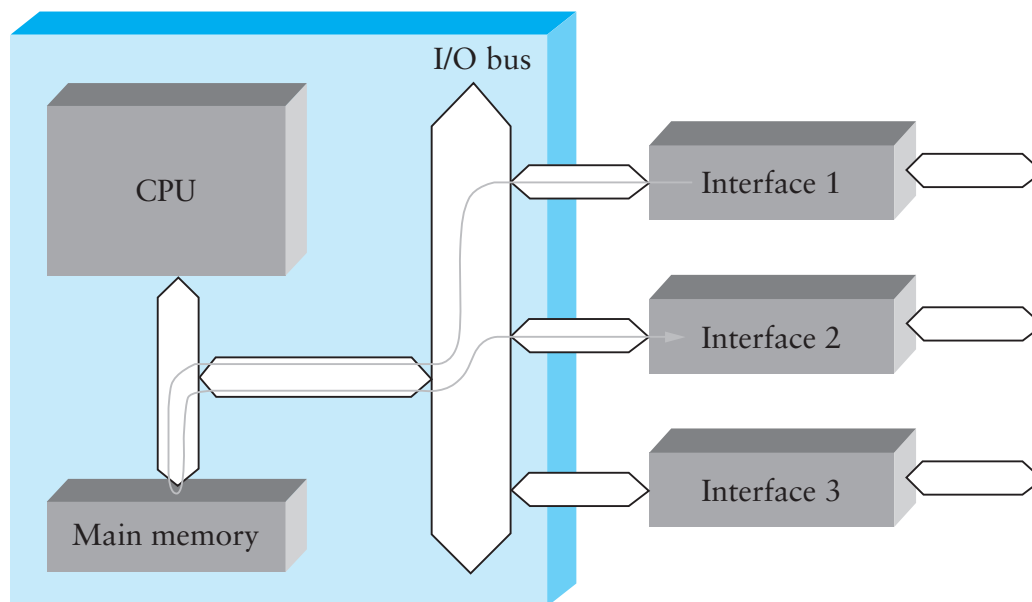


- **Goal: deliver packets from input to output ports**
- **Three potential performance concerns:**
  - Throughput in terms of bytes/time
  - Throughput in terms of packets/time
  - Latency

# Cut through vs. store and forward

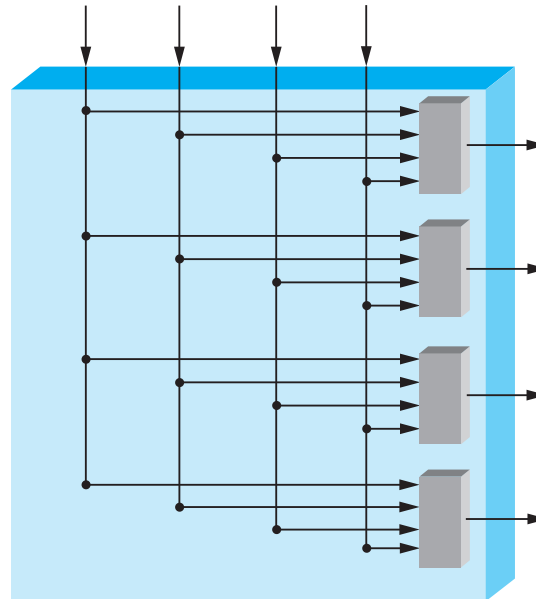
- **Two approaches to forwarding a packet**
  - Receive a full packet, then send it on output port
  - Start retransmitting as soon as you know output port, before you have even received the full packet (*cut-through*)
- **Cut-through routing can greatly decrease latency**
- **Disadvantage: Can't always send useful packet**
  - If packet corrupted, won't check CRC till after you started transmitting
  - If Ethernet collision, may have to send runt packet on output link, wasting bandwidth.

# Shared bus switch



- **Shared bus – like your PC**
  - NIC DMAs packet to memory over I/O bus
  - CPU examines header, sends to destination NIC over bus
  - I/O bus is serious bottleneck
  - For small packets, CPU may be limited, too
- **Shared memory – similar, has memory bottleneck**

# Crossbar switch

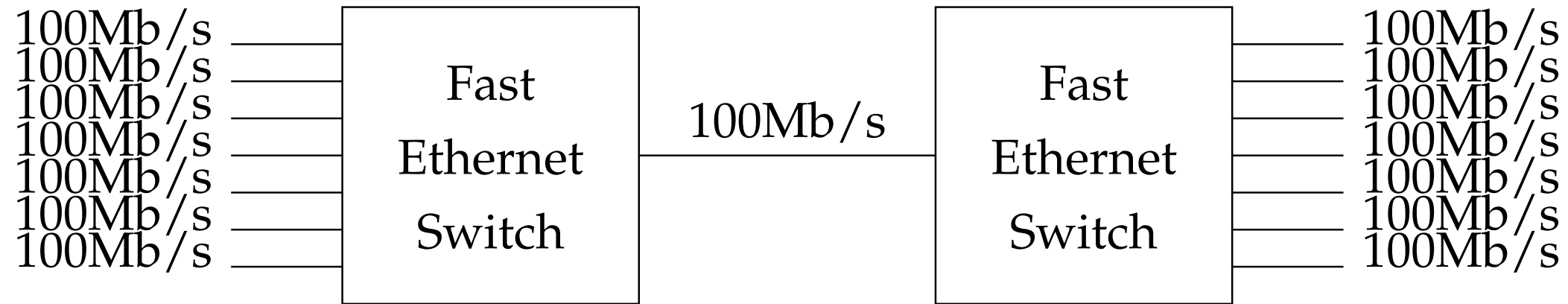


- **Can connect any input to any output**
  - Trivially allows any input→output permutation
  - More than one input to same output requires trickery

# Bisection bandwidth

- **Can speak of the bandwidth between sets of ports**
  - Bandwidth is maximum achievable aggregate bandwidth between the two sets
- **Bisection** bandwidth is important property of network
  - Lowest possible bandwidth between equal-sized sets of ports
  - Or almost equal-sized if odd number of ports
- **A network with bad bisection bandwidth may offer poor behavior**
  - Even if no conflict between input and output link utilization, may have internal bottlenecks reducing throughput.

## Example: Poor bisection bandwidth



- **Connect two Ethernet switches with Ethernet**
  - Suppose all clients on left, and all servers on right. . .
  - Aggregate bandwidth between all clients and servers only 100Mbit/s

## Coming up

- **Tomorrow, 5pm: Snowcast due**
- **Mon: Apple infosession**
- **Tue: Homework 1 out**
- **Thu: GTECH infosession**