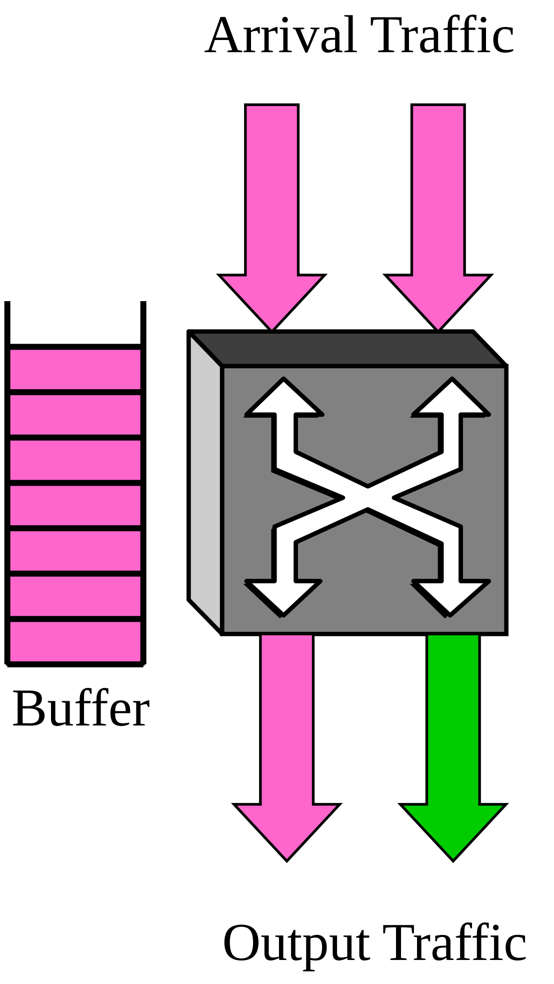
Chapter 3
Packet Switching

Circuit Switching & Packet Switching

- There are two limitations on the directly connected networks:
  - **How many hosts can be attached?**
  - **How large of a geographic area a network can serve?**
- To build networks that can be global in scale
  - To enable communication between hosts that are not directly connected
- **Circuit switching:** used for telephony
- **Packet switching:** used for computer networks
- A packet switch is a device with several inputs and outputs leading to and from the hosts that the switch interconnects
  - Takes packets that arrive on an input
  - Forwards them to the right outputs

Packet Switching

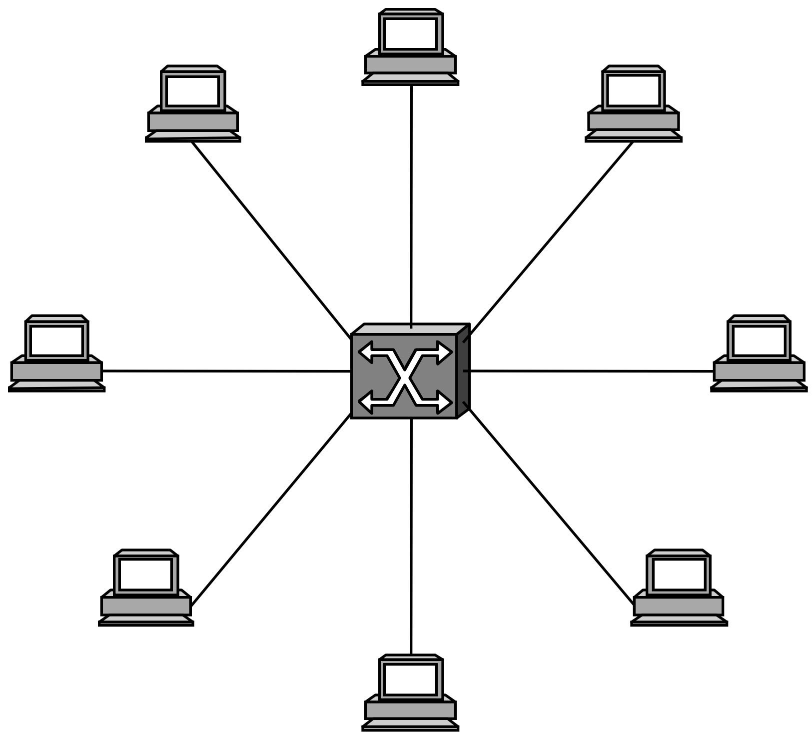
- A key problem for a switch is
  - The bandwidth of its outputs is **finite**
- **Contention:** the arrival rate **exceeds** the capacity of the output
  - The switch **queues packets** until the contention subsides
- If the contention lasts too long
  - The switch will **run out of buffer space** and be forced to **discard packets**
- If packets are discarded **too frequently**
  - The switch is said to be **congested**



Switching and Forwarding

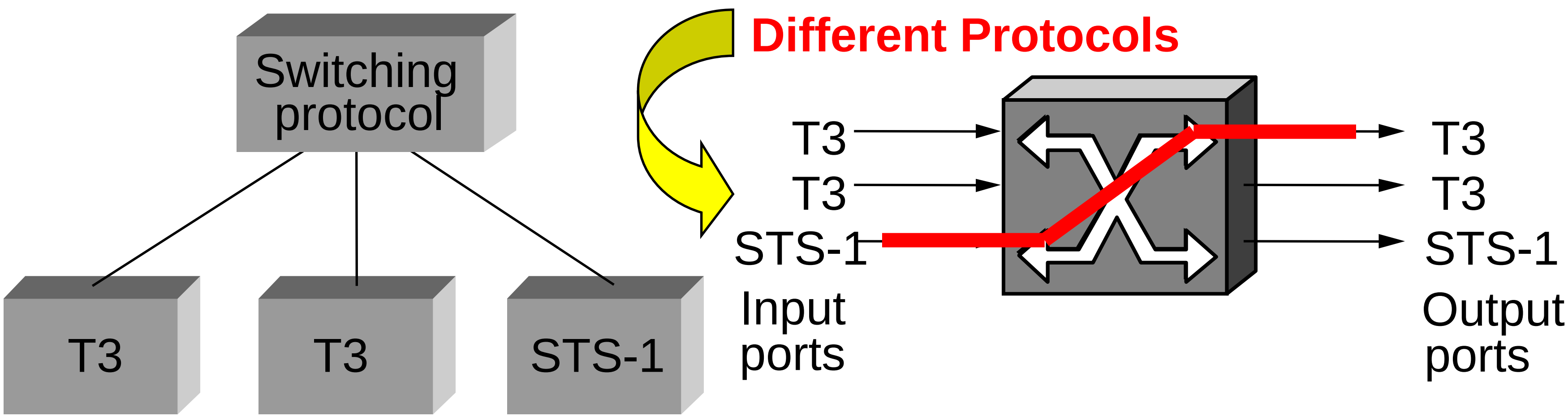
Switching and Forwarding

- A switch is a **multi-input, multi-output** device
  - Transfers packets from an input to one or more outputs
  - **Star topology**



Switching and Forwarding

- A switch is connected to a set of links and, for each links, runs the appropriate **data link protocol**
- Switching and Forwarding is
  - To receive incoming packets on one link and to transmit them on some other link
  - From an **input port** to an **output port**
- Output determination is called **Routing (Network layer)**

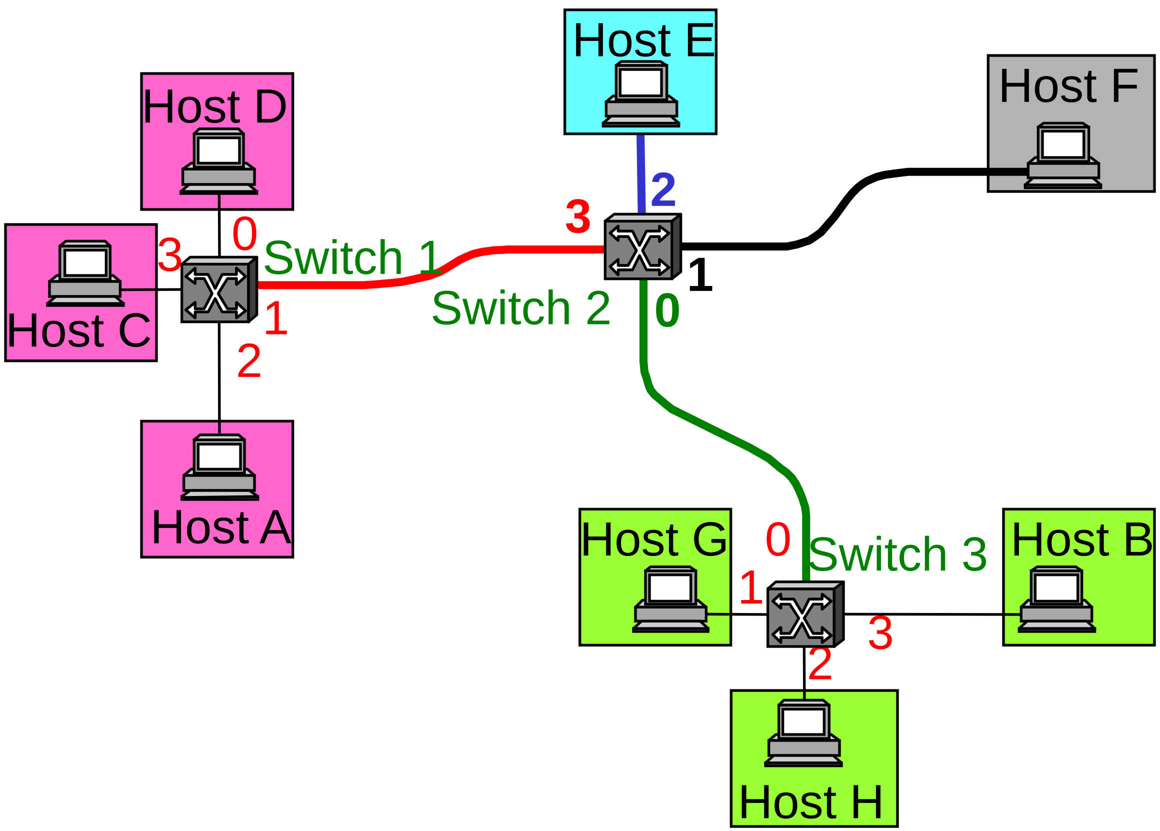


Switching and Forwarding

- **Switching:** looks at the **header** of a packet for an identifier
- Three approaches for switching:
  - **Datagram** or **connectionless** approach
  - **Virtual circuit** (VC) or **connection-oriented** approach
  - **Source routing** (less common)
- **Address:**
  - Nodes are identified by MAC addresses on a network
  - No two nodes on a network have the same address
  - All Ethernet cards are assigned a **globally unique** identifier

Datagram Approach

- Every packet contains the **complete** destination address
- A switch consults a **forwarding table (routing table)** for port determination



Forwarding table for Switch 2

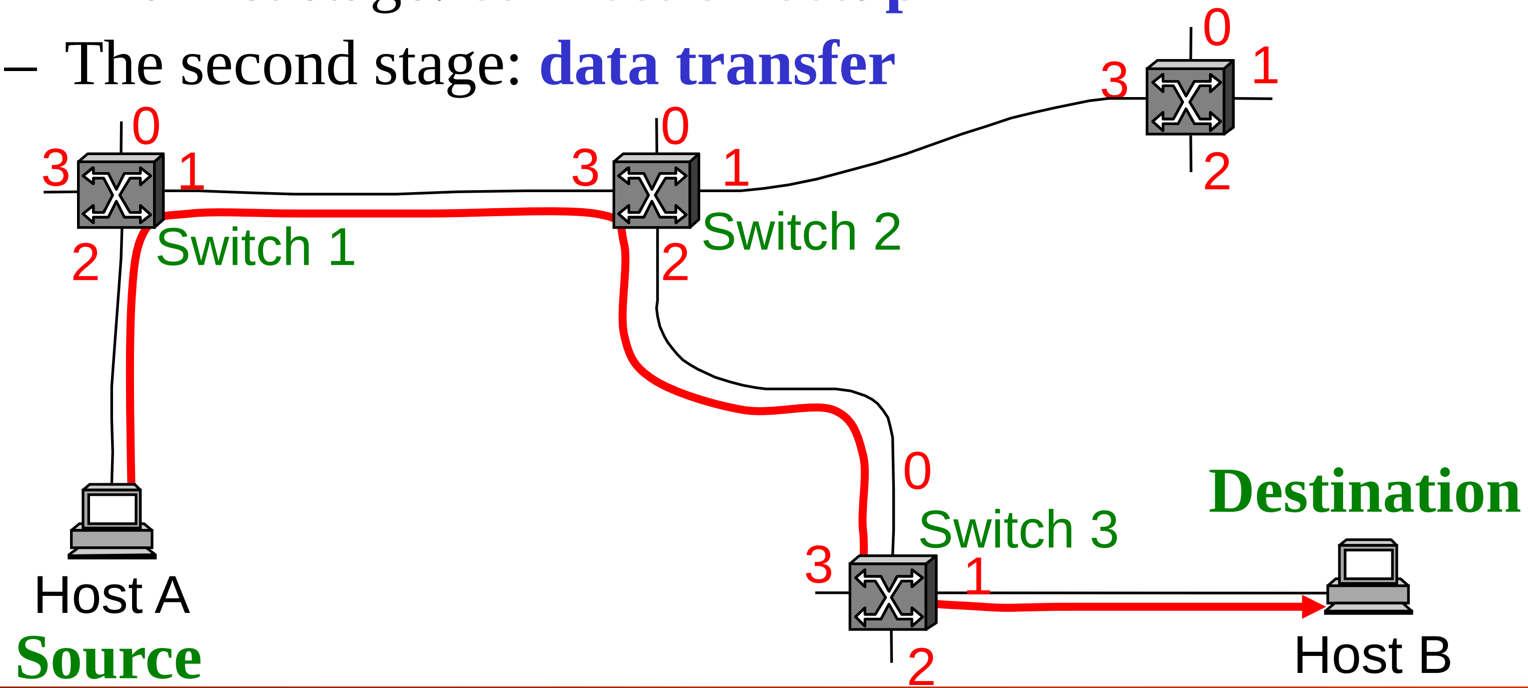
Destination	Port
A	**3**
B	**0**
C	**3**
D	**3**
E	**2**
F	**1**
G	**0**
H	**0**

Datagram Approach

- The table should be configured **statically**: it is hard for large networks with dynamically changing topologies
- Characteristics of connectionless (datagram) networks are
  - A host can send a packet **anywhere at any time**
  - When a host sends a packet, it has no way of knowing
    - **If the network is capable of delivering it or**
    - **If the destination host is up and running**
  - Each packet is forward **independently**
    - Two successive packets may follow **different paths**
  - A switch or link failure might not have any serious effect on communication
    - To find an **alternate route** and **update the table**

Virtual Circuit Approach

- A widely used technique for packet switching
  - Require to **set up a virtual connection** from the source host to the destination host **before** any data is sent
- A two-stage process:
  - The first stage: **connection setup**
  - The second stage: **data transfer**



Virtual Circuit (Connection Setup Phase)

- The connection setup phase:
  - To establish **"connection state"** in **each** of the switches between the source and destination hosts
- A single connection consists of an entry in a **"VC table"** in each switch. Each entry contains:
  - **A virtual circuit identifier (VCI):** uniquely identifies the connection at this switch
  - **An incoming interface** on which packets for this VC arrive at the switch
  - **An outgoing interface** in which packets for this VC leave the switch

Virtual Circuit (Connection Setup Phase)

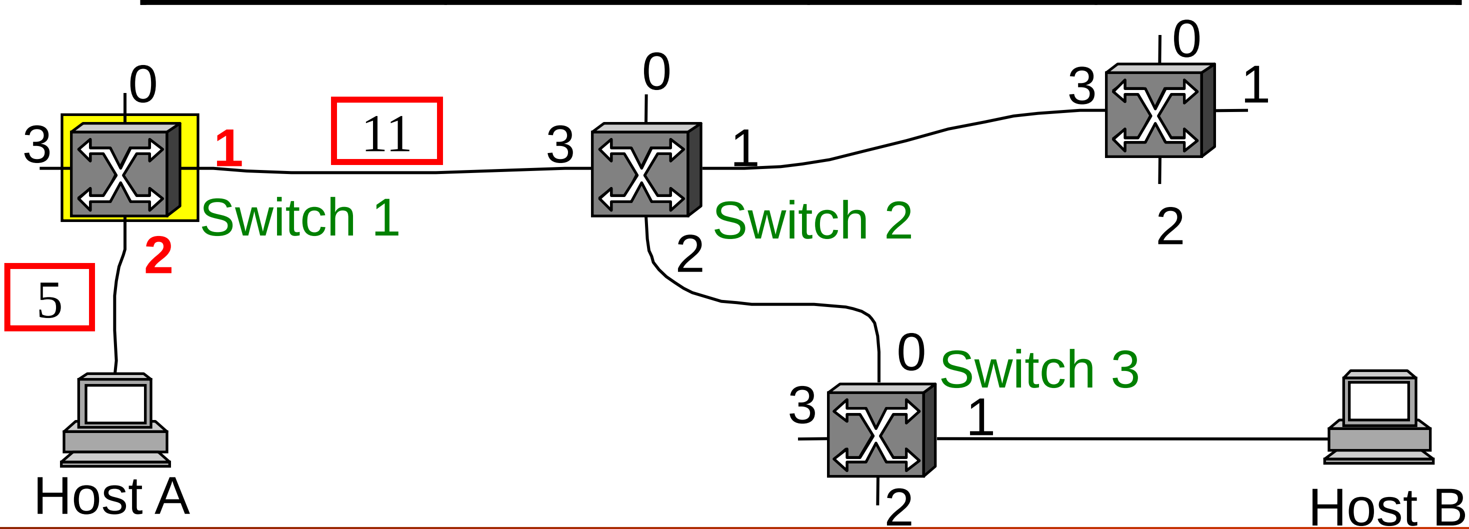
- There are two ways to establish connection state:
  - **Permanent virtual circuit (PVC):** a network **administrator** configures or deletes the state
  - **Switched (signaled) virtual circuit (SVC): a host can send messages** into the network to establish the state (without the involvement of a network administrator)
- If a packet arrives on the designated incoming interface and that packet contains the designated VCI value in its header
  - The packet is sent out the specified **outgoing interface**
  - Inserts the specified **outgoing VCI** value in the packet header
- The VCI is **not** a globally significant identifier for the connection; it has significant only on **a given link**

Virtual Circuit Approach (PVC)

- **PVC:** The administrator picks a VCI value that is currently **unused** on each link for the connection
- For example: Host A ↔ Switch 1: VCI value 5
  Switch 1 ↔ Switch 2: VCI value 11

Switch 1

Incoming Interface	Incoming VCI	Outgoing Interface	Outgoing VCI
2	**5**	**1**	**11**

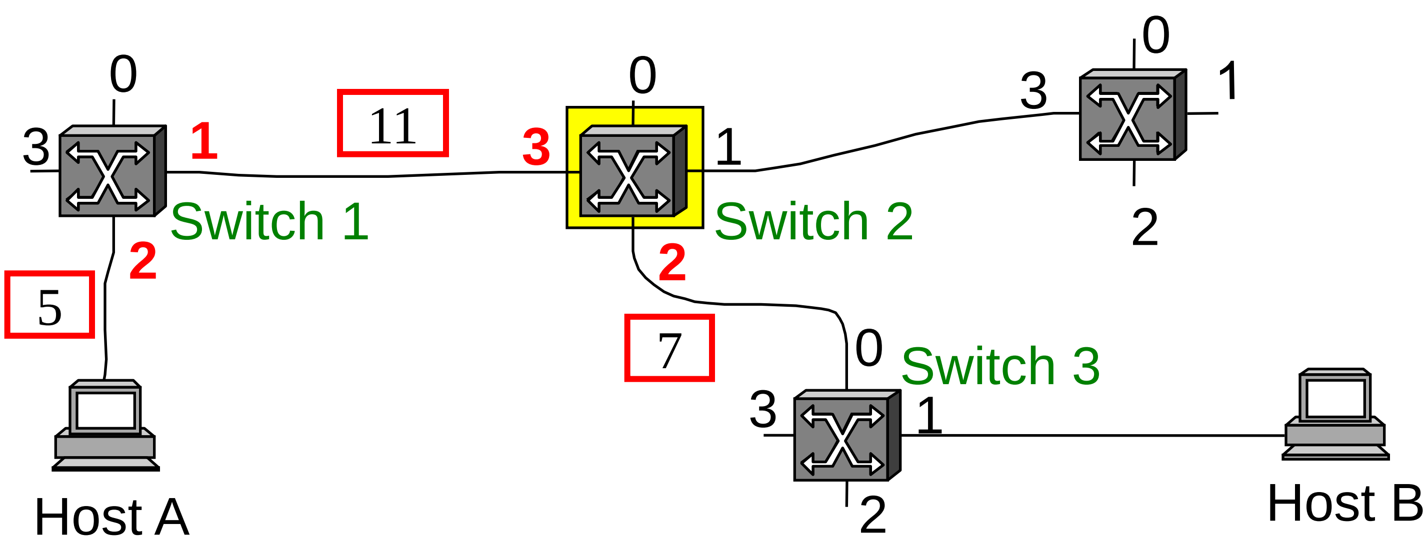


Virtual Circuit Approach (PVC)

- For example: Switch 2 ↔ Switch 3: VCI value 7

Switch 2

Incoming Interface	Incoming VCI	Outgoing Interface	Outgoing VCI
3	**11**	**2**	**7**

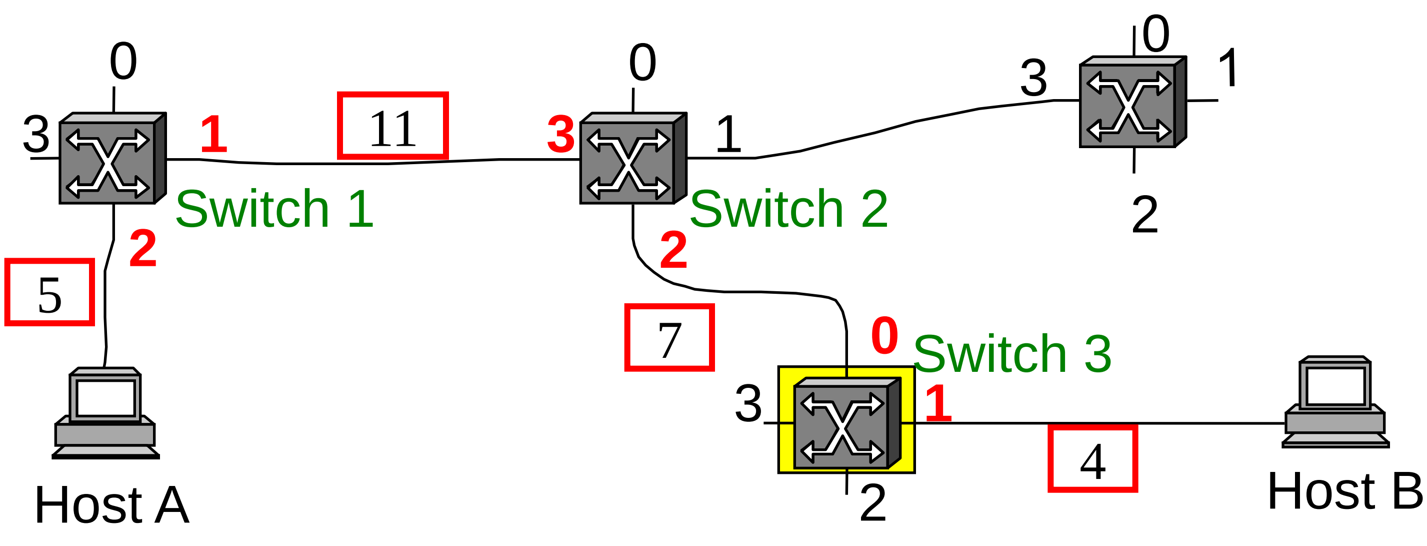


Virtual Circuit Approach (PVC)

- For example: Switch 3 ↔ Host B: VCI value 4

Switch 3

Incoming Interface	Incoming VCI	Outgoing Interface	Outgoing VCI
0	**7**	**1**	**4**



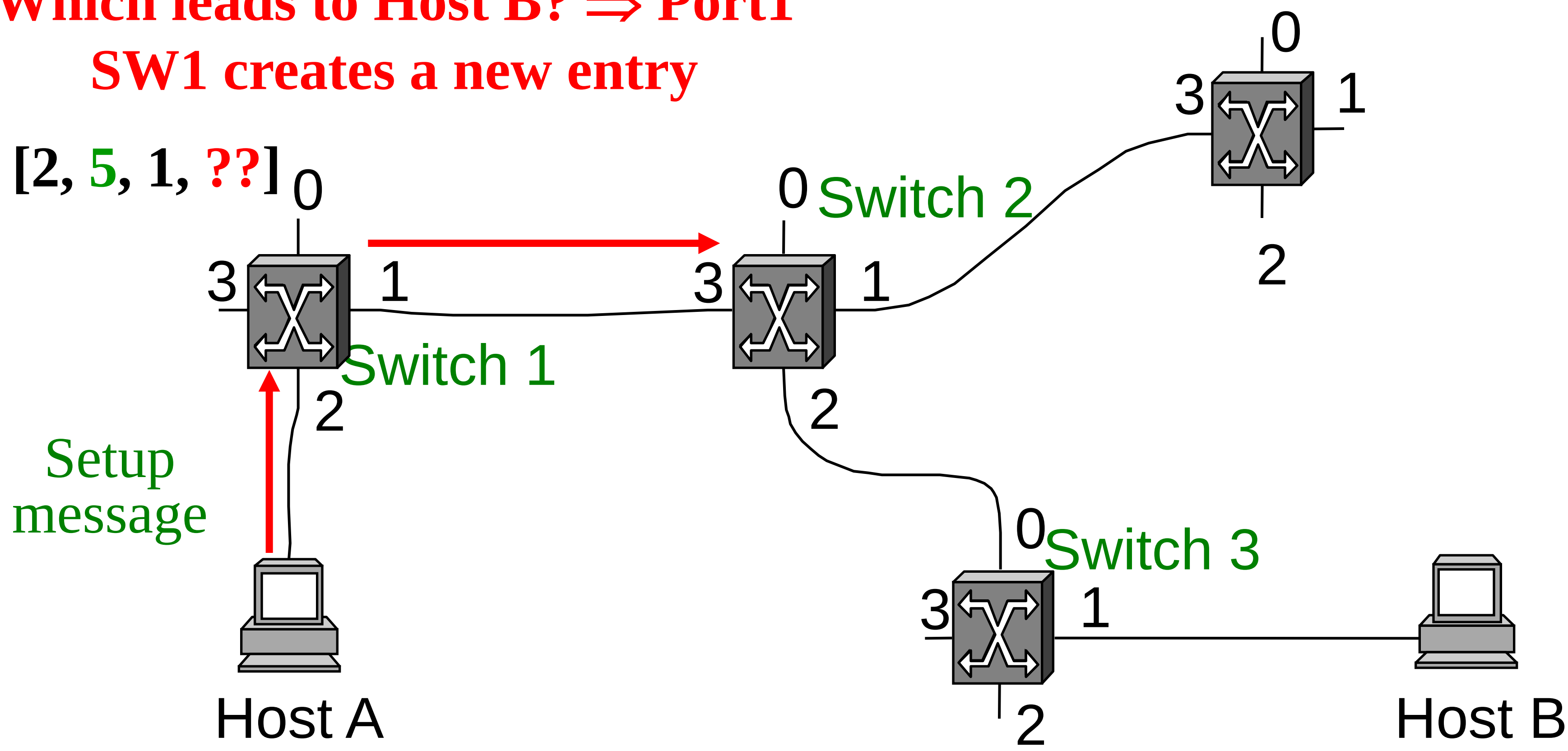
Virtual Circuit Approach (SVC)

- In the case of **PVCs**, signaling is initiated by the **network administrator**
- In the case of **SVCs**, signaling is initiated by one of the **hosts**
- Host A sends a setup message into the network (to SW1)
  - Contains the **complete destination address** of host B
- Each SW has to know which output will lead to host B
  - Sends the setup message to the right output
- When SW1 receives the connection request
  - It sends the setup message to SW2
  - It creates a new entry in its virtual circuit table
  - The task of assigning an **unused** VCI value, e.g. VCI = 5, is performed by the SW

Virtual Circuit Approach (SVC)

Which leads to Host B? ⇒ Port1

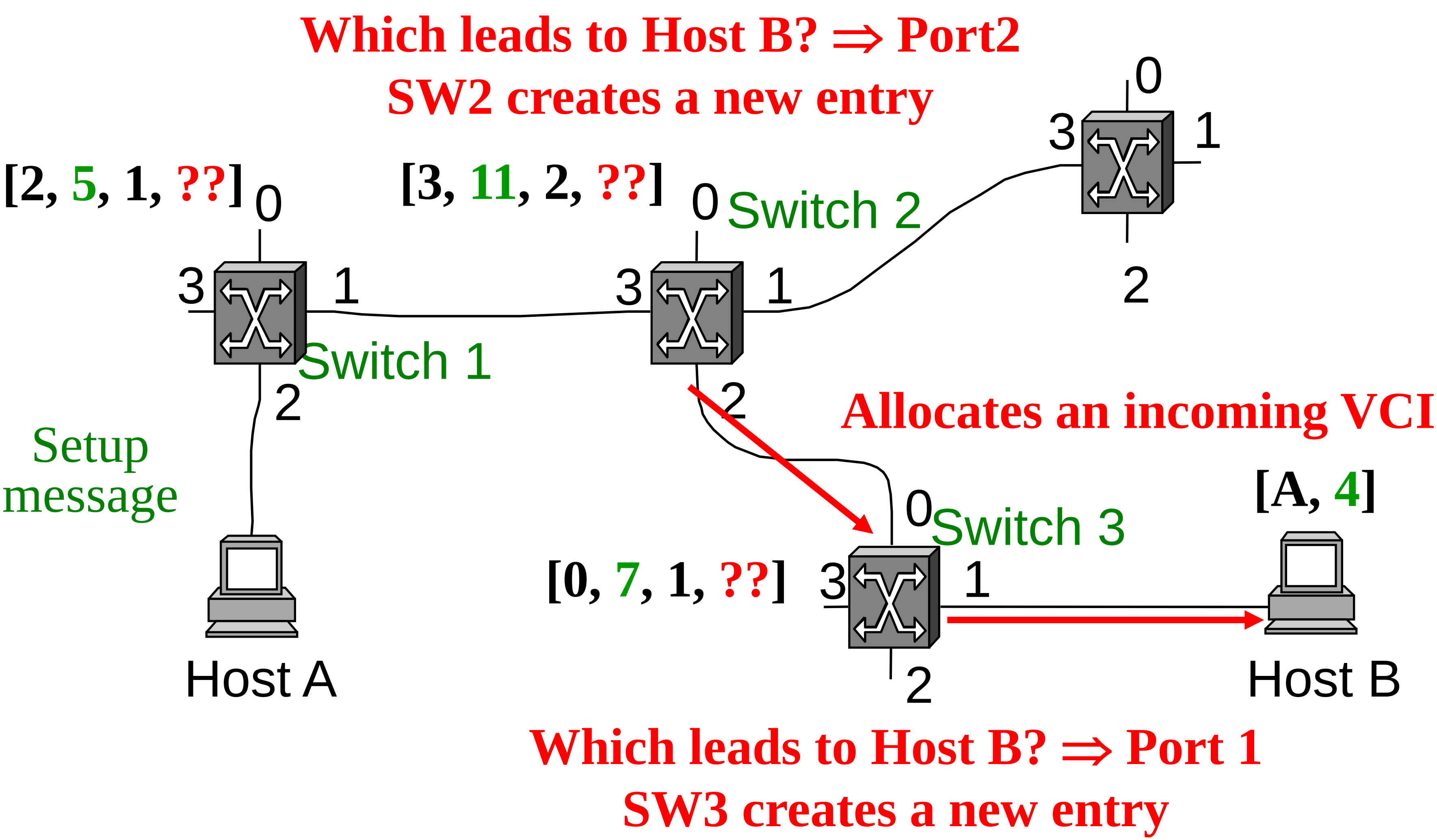
SW1 creates a new entry



Virtual Circuit Approach (SVC)

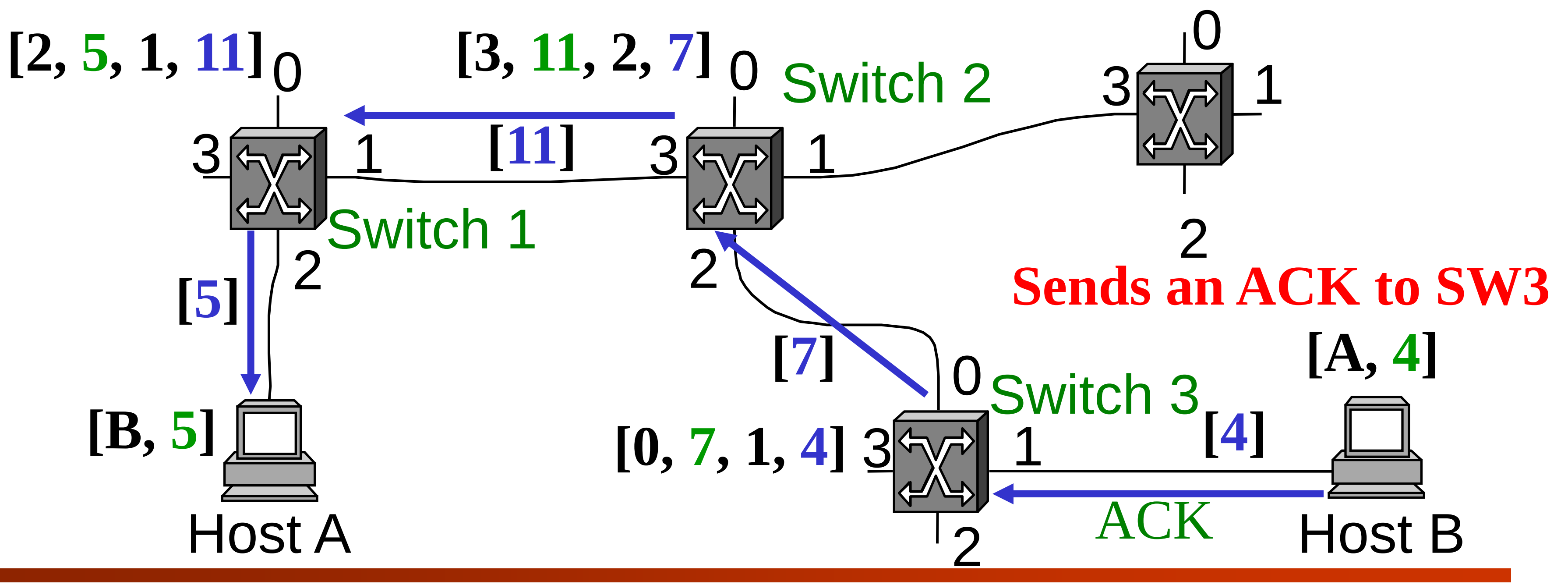
- When SW2 receives the setup message, it performs a similar process, e.g. VCI = 11
- When SW3 receives the setup message, it performs a similar process, e.g. VCI = 7
- Finally, the setup message arrives at host B, and assuming that B is healthy and willing to accept a connection from A
  - It allocates an incoming VCI value, e.g. VCI = 4, used to identify all packets coming from host A

Virtual Circuit Approach (SVC)



Virtual Circuit Approach (SVC)

- Host B sends an **acknowledgment** of the connection setup to SW3, including the VCI that it chose (i.e. 4)
- SW3 completes the virtual circuit table entry for this connection, and sends the acknowledgment on to SW2
- Finally, the acknowledgment is passed on to host A



Virtual Circuit Approach

- The outgoing VCI value at one switch is the incoming VCI value at the next switch

Switch 1

Incoming Interface	Incoming VCI	Outgoing Interface	Outgoing VCI
2	5	1	11

Switch 2

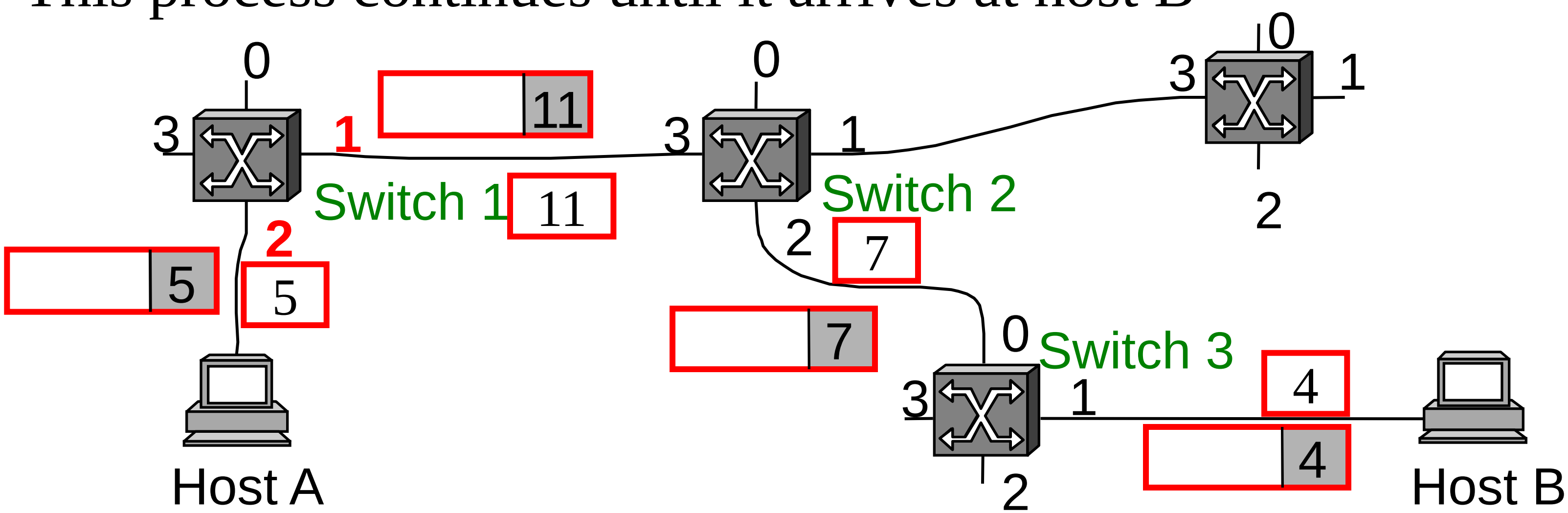
Incoming Interface	Incoming VCI	Outgoing Interface	Outgoing VCI
3	11	2	7

Switch 3

Incoming Interface	Incoming VCI	Outgoing Interface	Outgoing VCI
0	7	1	4

Virtual Circuit Approach

- For a packet that it wants to send to host B
  - Host A puts the **VCI value** 5 in the header and sends to SW1
  - SW1 uses the combination of the **interface** and the **VCI** in the packet header to find the appropriate VC table entry
  - SW1 forwards the packet out of interface 1 and to put the VCI value 11 in the packet header
- This process continues until it arrives at host B



Virtual Circuit Approach

- When host A no longer wants to send data to host B
  - It **tears down** the connection by sending a **teardown message** to SW1
  - The SW **removes** the relevant entry from its table and forwards the message to the other SW in the path
- If host A send a packet with a VCI of 5 to SW1
  - Since the connection does not exist
    - The packet will be **dropped**

Virtual Circuit Approach

- Things about Virtual Circuit Switching:
  - There is at least one **RTT (round-trip time) of delay** before data is sent
    - The transmitter has to wait for the VC being set up
  - The **per-packet overhead** caused by the header is **reduced** relative to the datagram method
    - The **connection request** contains a **global identifier**
    - Each **data packet** contains only a **small identifier**
  - If a switch or a link in a connection **fails**, the connection is broken and needs to be torn down
  - A **routing algorithm** is required for the connection request

Virtual Circuit Approach

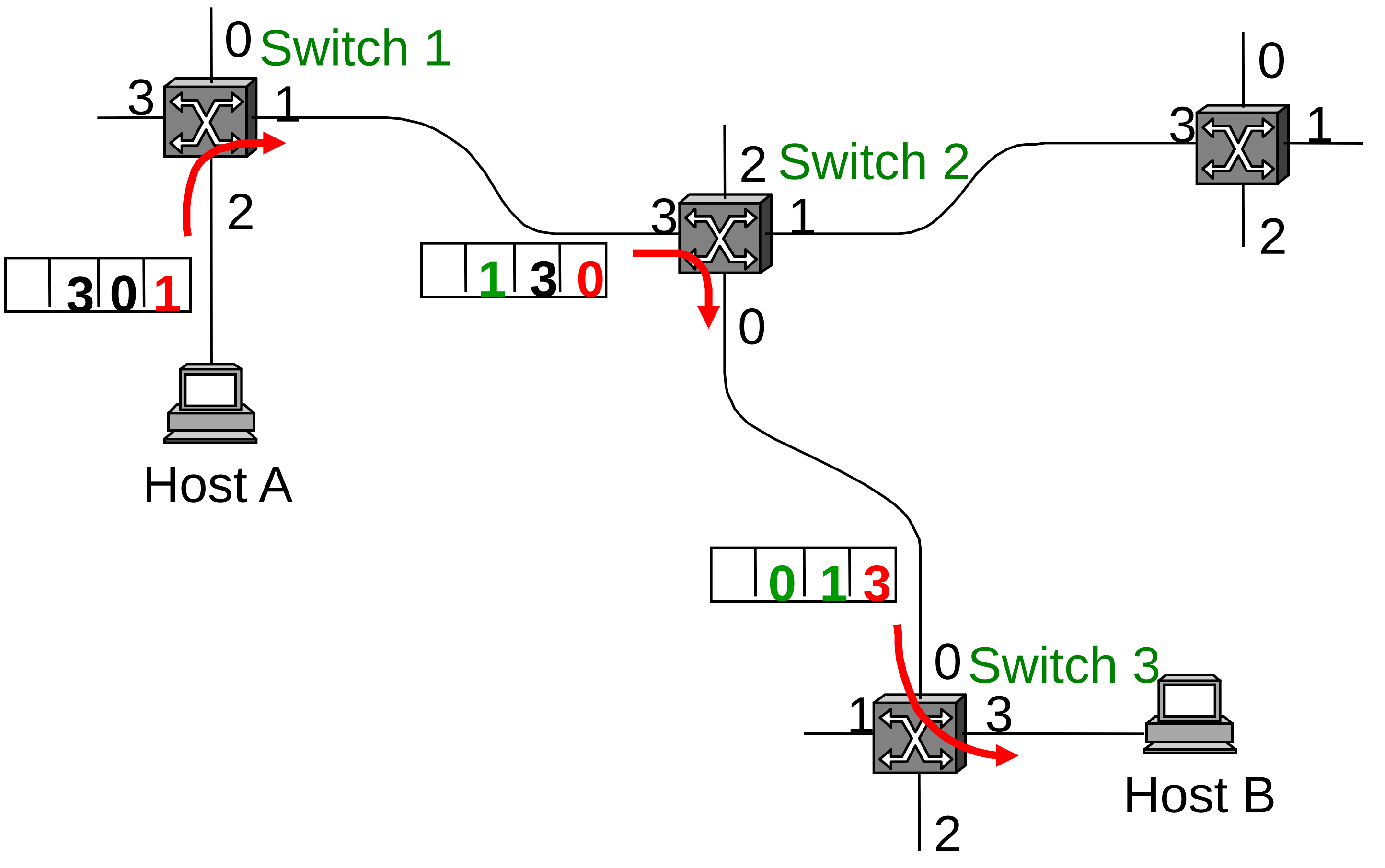
- For VCS, the transmitter knows quite a lot about the network
  - There is a route to the receiver
  - The receiver is willing and able to receive data
  - **Resources** have been allocated to the VC at the time it is **established**
- For example, X.25 network employs the three-part strategy:
  - **Buffers** are allocated to **each VC** when it is initialized
  - The **sliding window protocol** is run between **each pair of nodes** along the VC
  - If not enough buffers are available $\Rightarrow$ the circuit is **rejected**

Source Routing

- All the information, about network topology that is required to switch a packet, is provided by the **source host**
- The **output port of each SW** is placed in the **header** of the packet
  - For each packet that arrives on an input, the SW reads the **port number** in the header and transmits the packet on that output
- The source will put an **ordered list** of SW ports in the packet header
- Each SW will **rotate** the list for each transmission
  - The next SW in the path is always at the front of the list

Source Routing

- Output port list: [1, 0, 3]

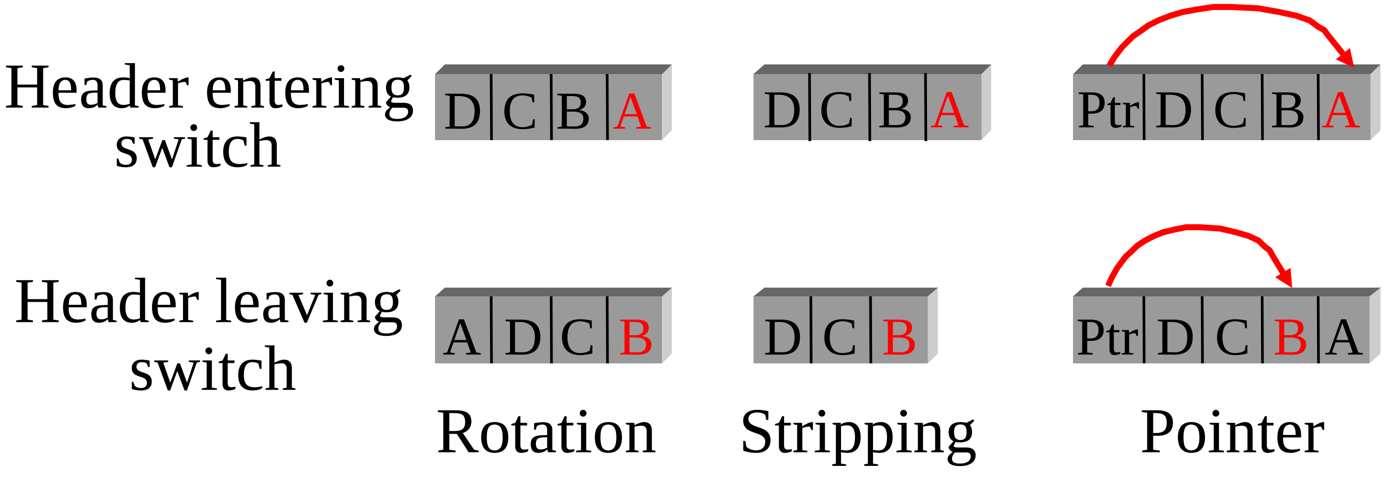


Source Routing

- Things about Source Routing:
  - It assumes that the sending host knows enough information of the right directions **in each SW**
    - Analogous to the problem of building the forwarding tables in a datagram network
  - The size of the packet header **cannot be predicted**
    - The headers are probably of **variable length** with no upper bound (depending on the number of SWs)
  - There are some variations on this approach
    - Rather than rotate the header, each SW could just **strip** the first element used in this SW
    - Another alternative is to have a **pointer** to the current “next port” entry

Source Routing

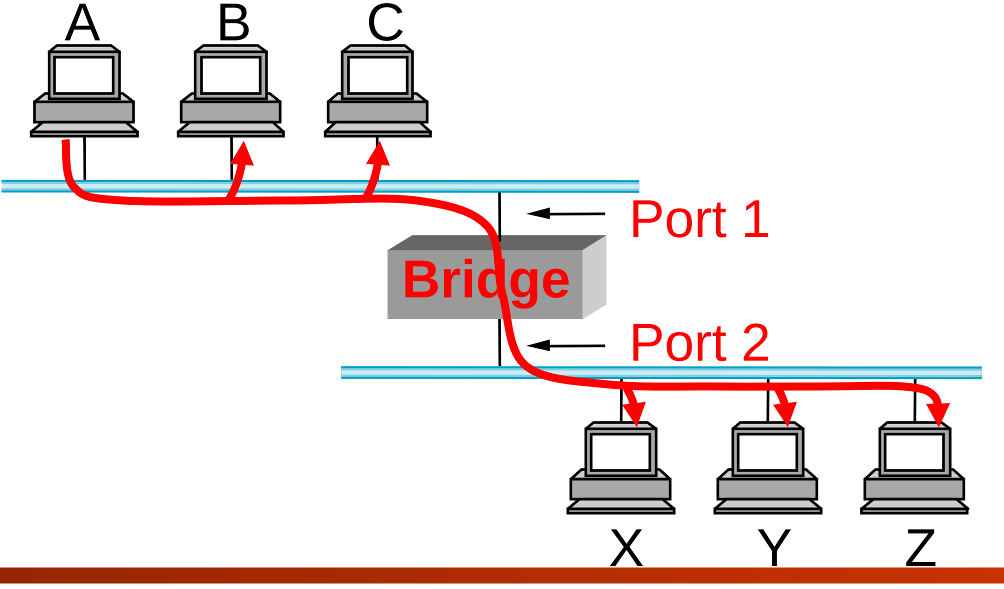
- **Rotation** has an advantage over stripping:
  - Host B gets a copy of the complete header → **the way back to host A**



Bridges and LAN Switches

Bridge

- **Bridge:** a node put between two Ethernets to **interconnect** these two networks
- **Extended LAN:** a collection of LANs connected by one or more bridges
  - Bridges simply accept LAN frames on their inputs and forward them out on **all other outputs**

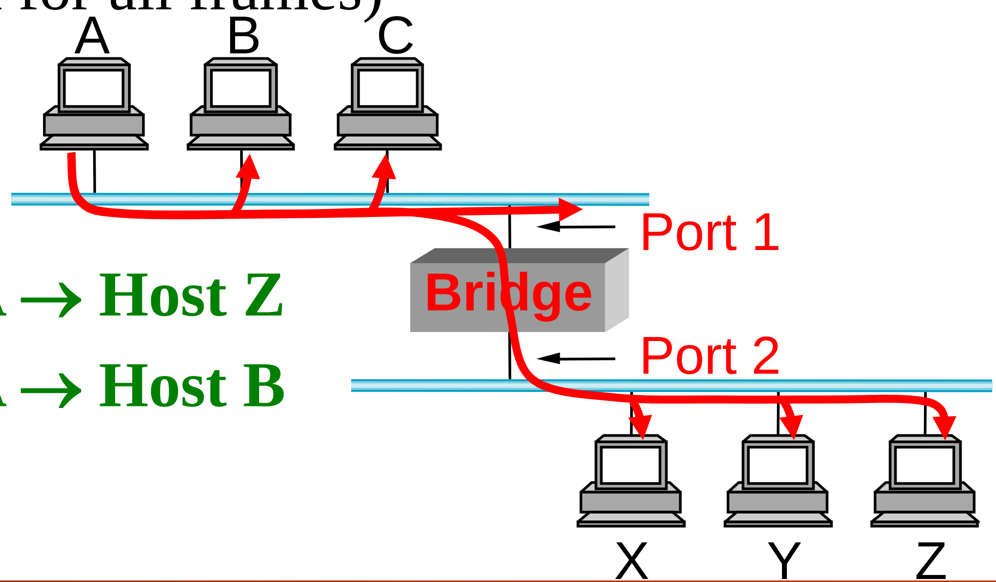


Learning Bridges

- The bridge need not forward all frames that it receives
  - For example: the frames from host A to host B
- A bridge should learn on which port the various hosts reside?
  - One option: **Manually download** a table into the bridge
  - The **datagram model** should be applied (global identifier is required for all frames)

Host A → Host Z

Host A → Host B



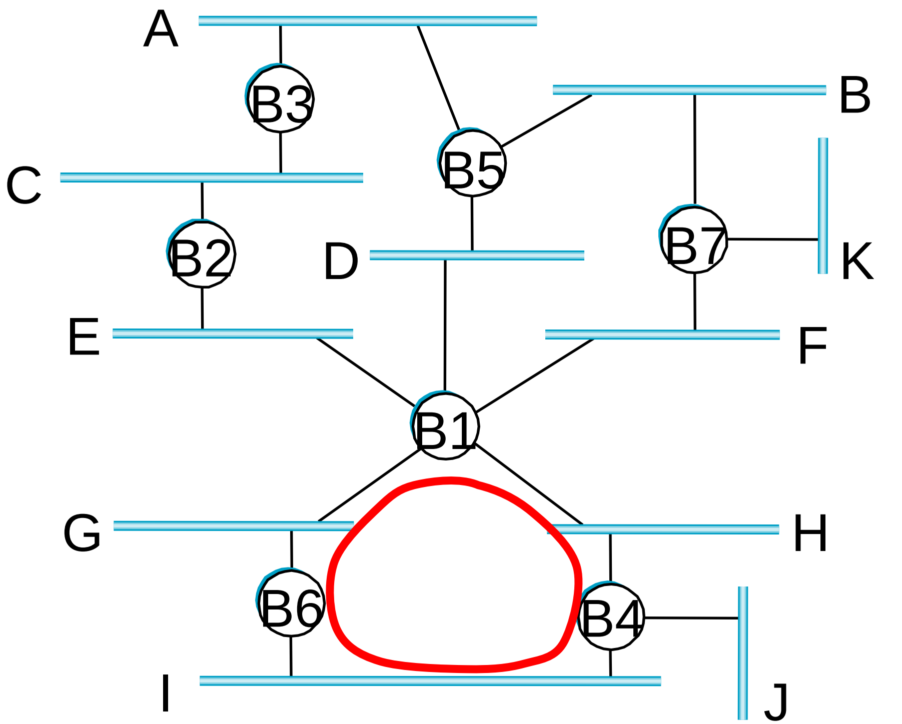
Learning Bridges

- Another option: a bridge learn this information **automatically**
  - Bridges inspect the **source address** in all received frames
- When host A sends a frame to a host on either side
  - The bridge receives this frame on port 1 and records the fact that host A resides on the side of port 1
  - **Then the bridge can build a table**
- A **timeout** is associated with each entry
  - In order to protect against the situation that a host is moved from one network to another
  - The bridge **discards the entry** after **timeout** occurs

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

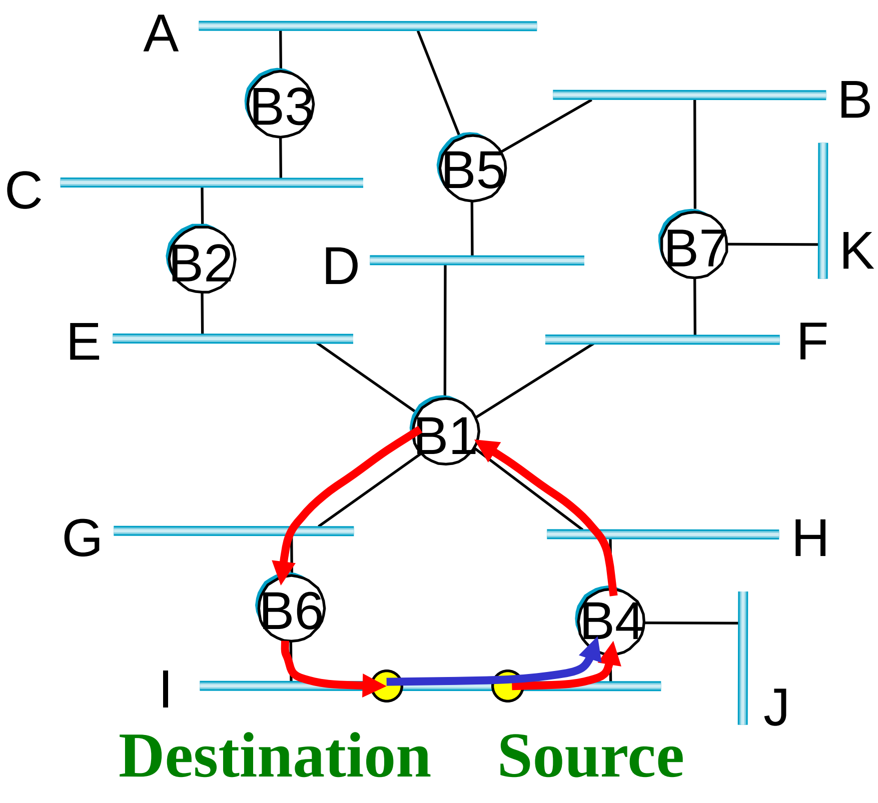
Spanning Tree Algorithm

- The network may be managed by more than one administrator
- No **single** person knows the entire configuration of the network → a **loop** might be added without anyone knowing
- Loops may be built into the network on purpose (**redundancy**)



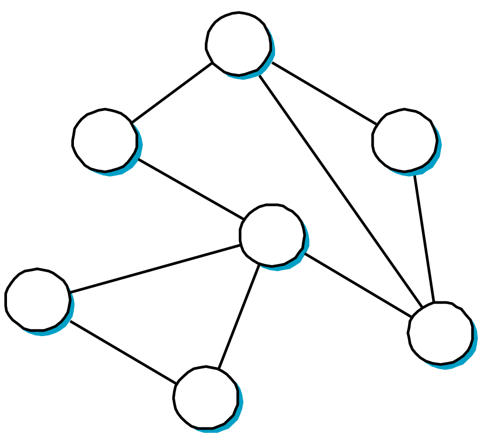
Spanning Tree Algorithm

- If the extended LAN has a loop in it
  - Frames potentially loop through the extended LAN **forever**

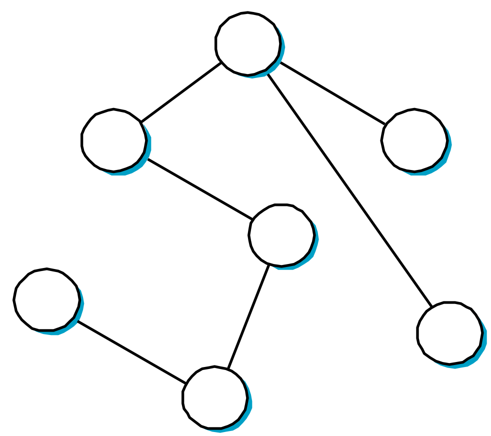


Spanning Tree Algorithm

- Bridges must be able to correctly handle loops
- The bridges must run a **distributed** **spanning tree algorithm**
  - A protocol used by a set of bridges to agree upon a particular extended LAN
- A spanning tree is a subgraph of the originally graph that covers (spans) all the vertices, but **contains no cycles**



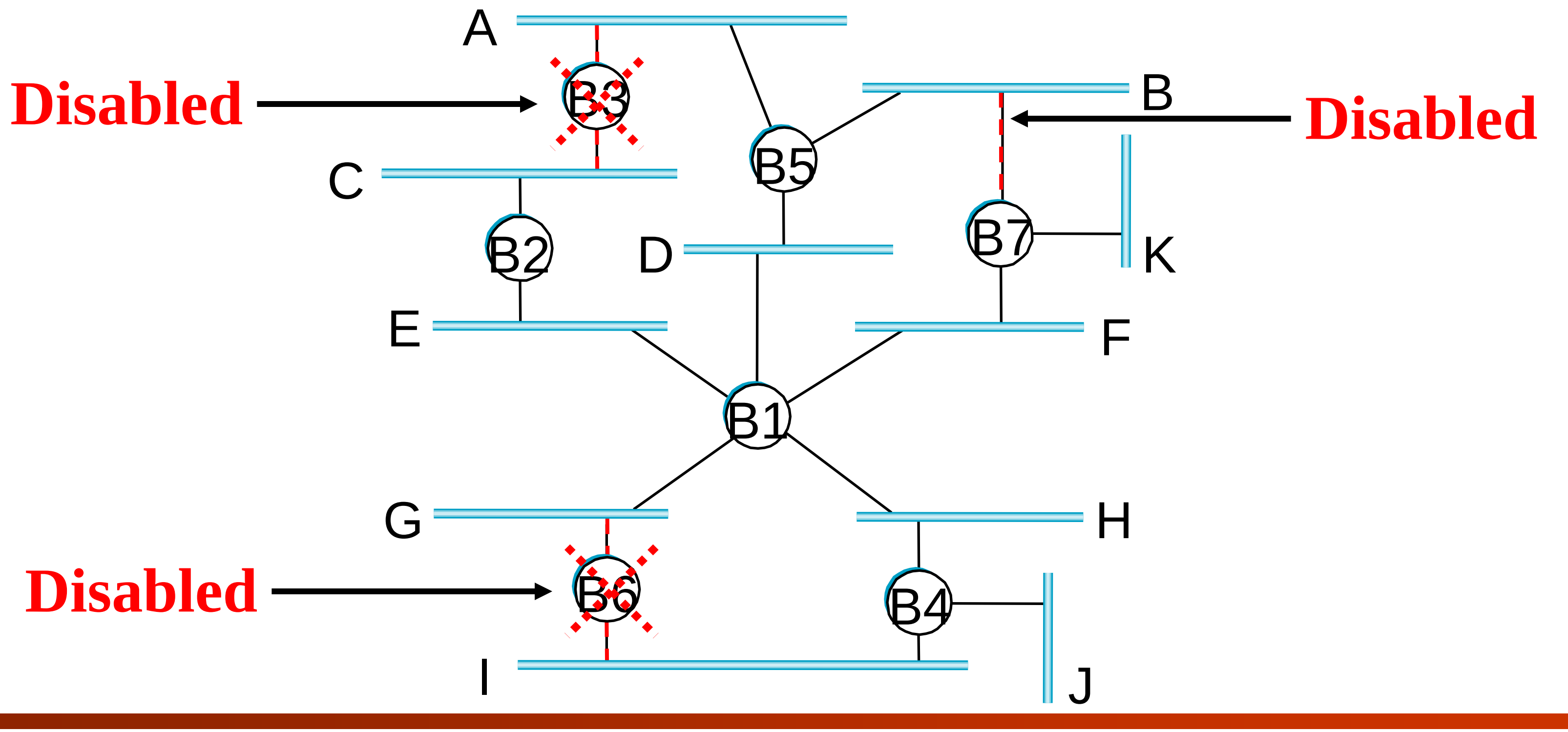
Cyclic graph



Spanning tree

Spanning Tree Algorithm

- In practice, this means that each bridge decides the ports over which **it is and is not willing to forward frames**
- The algorithm is **dynamic**: should some bridges fail, the bridges **reconfigure** themselves into a new spanning tree



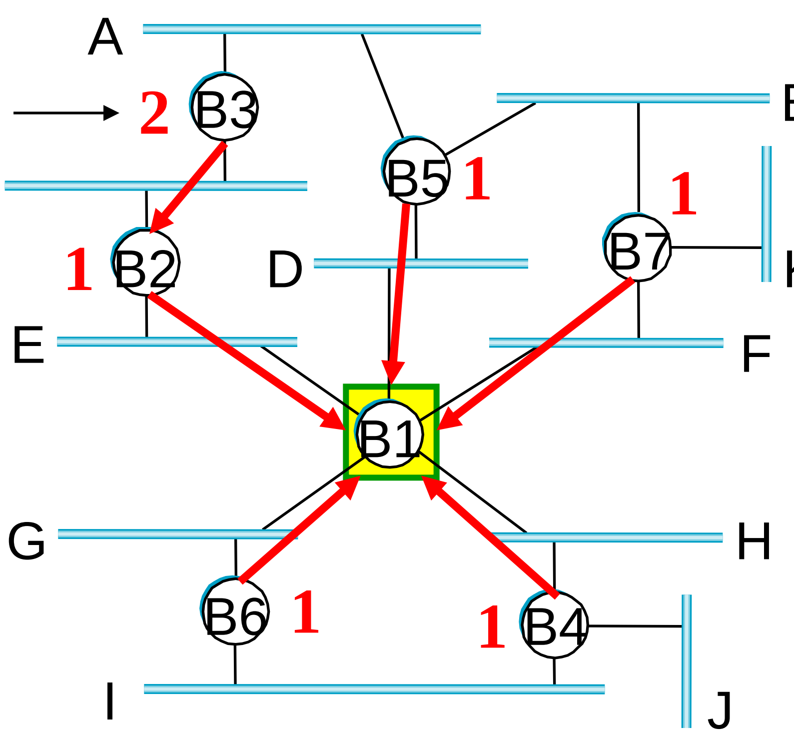
Spanning Tree Algorithm

- The main purpose: for bridges to select the ports over which they will forward frames
- Each bridge has a **unique identifier**
- First elects the bridge with the **smallest ID** as the **root** of the spanning tree
  - The root bridge **always** forwards frames out **over all ports**
- Each bridge computes the **shortest path** to the root and notes which of its ports is on this path
- Each LAN select **a single designated bridge** that will be responsible for forwarding frames toward the root bridge
  - The one that is **closest to** the root
  - If **ties** occur, the one with **smallest ID** is selected

Spanning Tree Algorithm

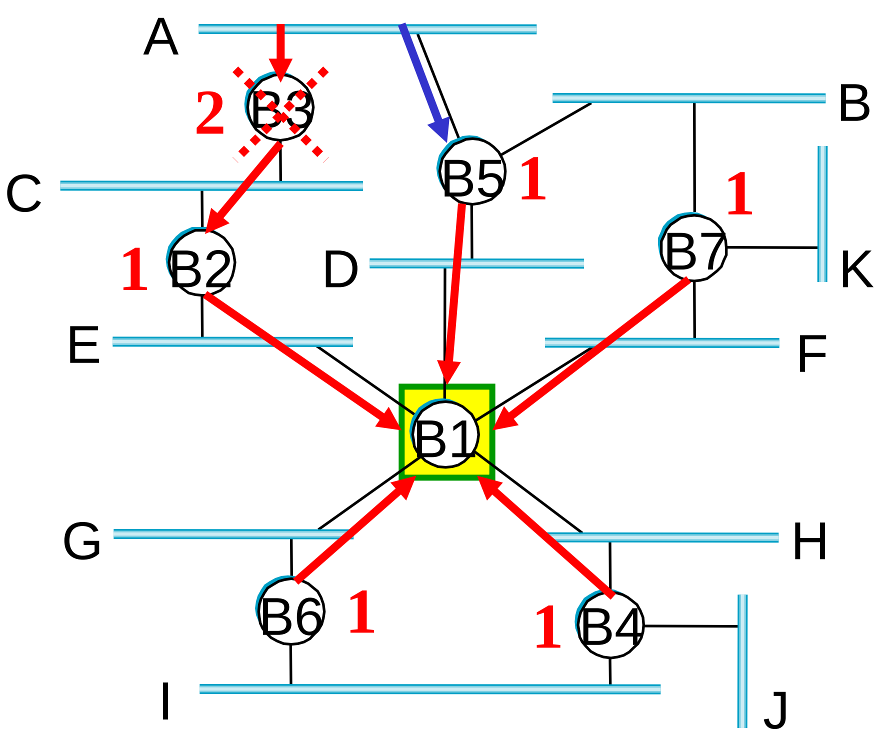
- B1 is the root bridge
- For all bridges, find the **shortest path** to the root

Number of hops →



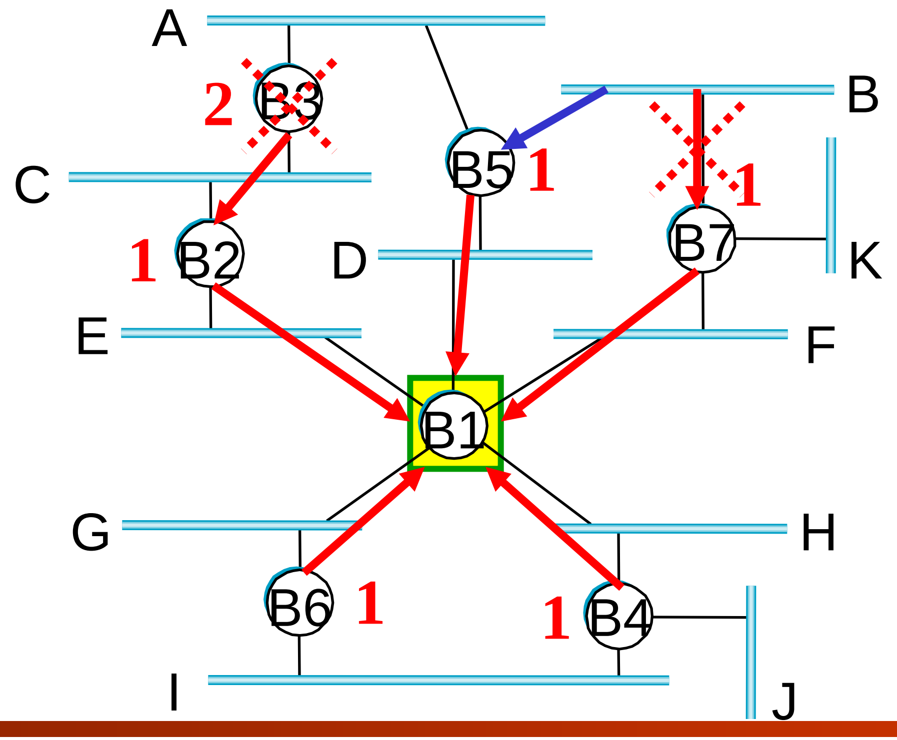
Spanning Tree Algorithm

- For LAN A:
  - Two bridges are available: B3 and B5
  - The designated bridge is **B5**



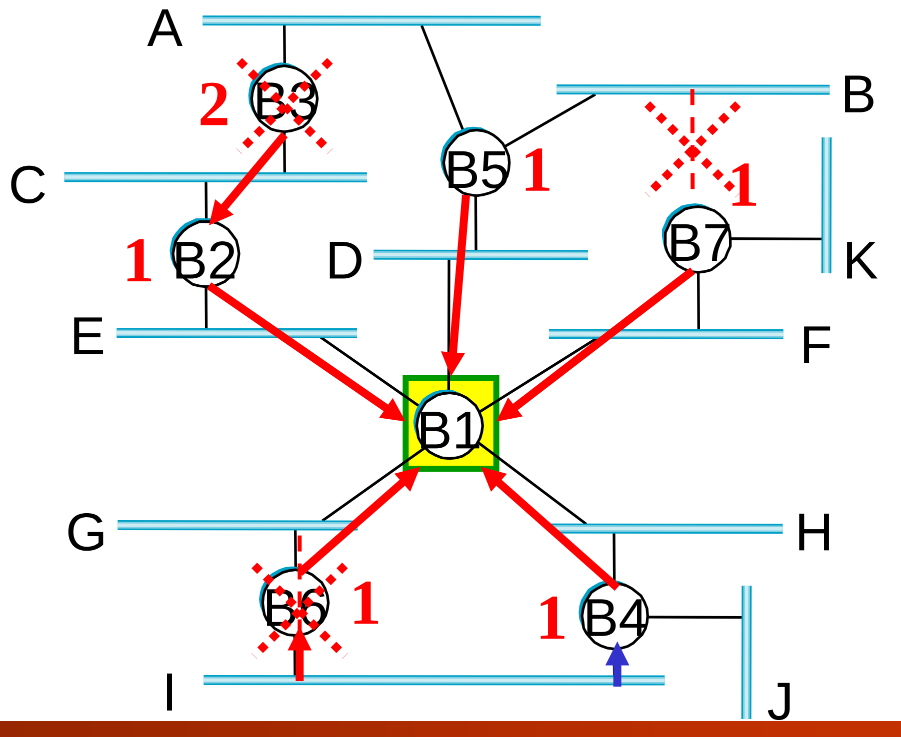
Spanning Tree Algorithm

- For LAN B:
  - Two bridges are available: B5 and B7
  - The distances are Tie
  - The designated bridge is **B5**



Spanning Tree Algorithm

- For LAN I:
  - Two bridges are available: B4 and B6
  - The distances are Tie
  - The designated bridge is **B4**



Spanning Tree Algorithm

- The bridges have to exchange **configuration messages**
  - To decide whether or not they are the **root** or a **designated bridge**
- The information contained in the configuration messages is
  - The **ID** for the bridge that is sending the message
  - The **ID** for what the sending bridge **believes** to be **the root bridge**
  - The **distance**, measured in **hops**, from the sending bridge to the root bridge
- Each bridge records the current **"best"** configuration message it has seen on each of its ports

Spanning Tree Algorithm

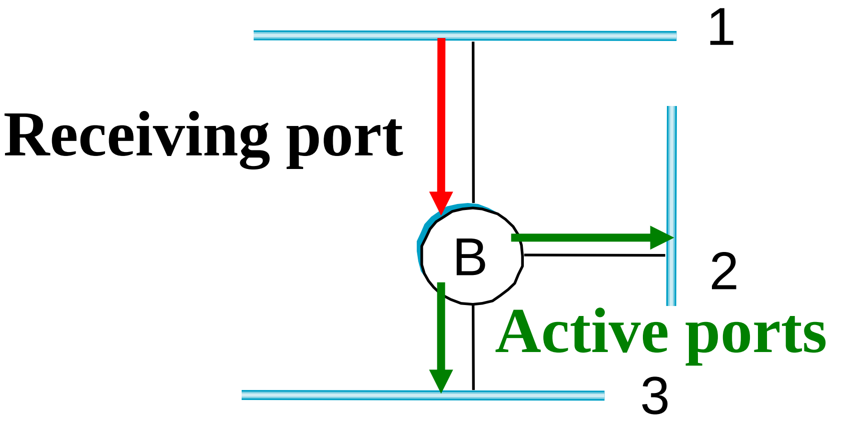
- Initially, each bridge thinks it is the root, and sends a configuration message identifying itself as the root
- Upon receiving a configuration message over a particular port
  - The bridge checks that if the new message is **better** than the current recorded **best configuration message**
- The new configuration message is considered **"better"** if
  - It identifies a root with **a smaller ID**, or
  - It identifies a root with an equal ID but with **a shorter distance**, or
  - The root ID and distance are equal, but **the sending bridge has a smaller ID**

Spanning Tree Algorithm

- If the new message is **better**
  - The bridge discards the old information and saves the new information
  - The bridge adds 1 to the distance-to-root field
- When a bridge receives a configuration message indicating that it is not the root bridge (i.e. a message with a smaller ID)
  - **Stops** to generate the configuration message **on its own**
  - Forwards configuration messages from other bridges
- When the system stabilizes, only the **root bridge** is still generating configuration messages

Broadcast and Multicast

- **Broadcast:**
  - Each bridge forwards a frame with a destination broadcast address out on each active port other than the one on which the frame was received
- **Multicast:**
  - Implemented in **exactly the same way**
  - Each host decides whether or not to accept the message

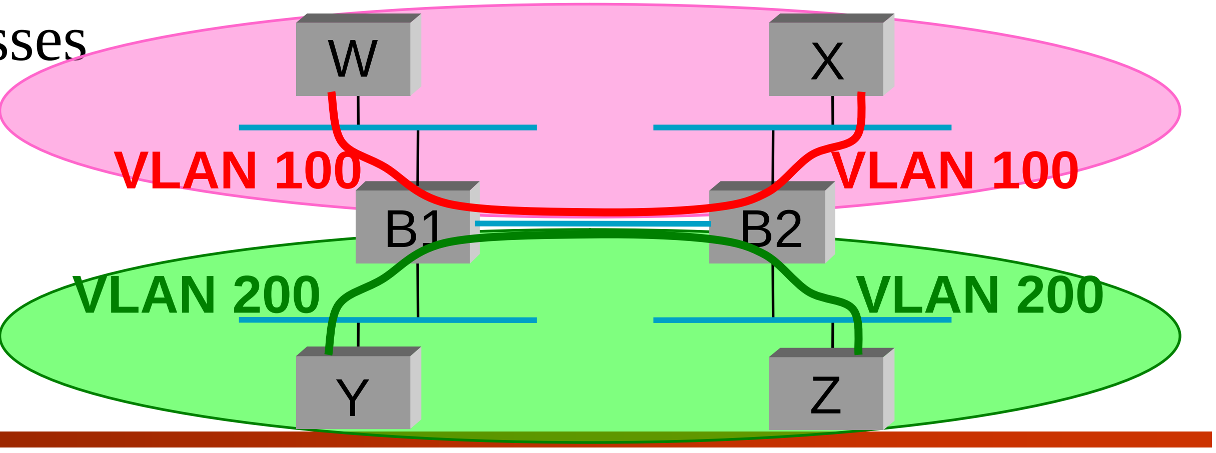


Limitations of Bridges (Scale)

- **Scale Limitation:** it is not realistic to connect more than a few (tens of) LANs by means of bridges
  - The spanning tree algorithm **scales linearly** (bad efficiency)
  - Bridges forward all broadcast frames (**heavy traffic load**)
- **Reality:**
  - The broadcast message should be seen only for all hosts **within a limited setting** (e.g. a department or a company)
  - All the hosts in a larger environment (e.g. a university) will not want to be bothered by each other's broadcast messages
- Broadcast does not scale $\Rightarrow$ **extended LANs do not scale**

Virtual LANs

- VLANs allow a single extended LAN to be partitioned into several seemingly (not practically) separate LANs
  - Each virtual LAN is assigned an identifier
- Packets can only travel from one segment to another **if both segments have the same identifier**
  - This limits the number of segments that will receive any given broadcast packet
- It can change the logical topology without moving wires or changing addresses



Limitations of Bridges (Heterogeneity)

- **Heterogeneity Limitation:** Bridges are fairly limited in the kinds of networks they can interconnect
  - Bridges make use of the network’s frame header
  - Bridges can support only networks that have exactly **the same address format**
  - Bridges connect Ethernets to Ethernets, 802.5 to 802.5, and Ethernets to 802.5 rings ⇒ **48-bit address format**
  - Bridges do not generalize to other kinds of networks, such as ATM

Limitations of Bridges

- **Advantage of bridges:** allow multiple LANs to be transparently connected
  - Networks can be connected without the end hosts having to run any additional protocols
- **Disadvantage of bridges:** this transparency can be dangerous
  - The application and transport protocol running on a host may be programmed **under the assumption of running on a single LAN**
  - If a bridge becomes **congested**, it may have to drop frames
    - It is rare that a single Ethernet drops a frame
  - The **latency** between any pair of hosts on an extended LAN becomes both **larger** and more **highly variable**

Cell Switching (ATM)

ATM

- **ATM: Asynchronous Transfer Mode**
- ATM is a **connection-oriented**, **packet-switched** technology
  - It uses **virtual circuits**
- The **QoS capabilities** of ATM are one of its greatest strengths
- The ATM packets are of **fixed length**
  - **53 bytes** – **5 bytes** of header followed by **48 bytes** of payload
  - These fixed-length packets are named **"cells"** ⇒ cell switching

Variable Packet Length

- All the packet-switching technologies have used **variable-length packets**
  - The variable-length is constrained within some bounds
- If a host only has **1 byte** to send
  - Puts the data in a **minimum-sized packet**
  - Minimizes the extra **padding**
- If a host has a **large file** to send
  - Breaks it up into **maximum-sized packets**
  - Drives down the ratio of header to data bytes $\Rightarrow$ increases bandwidth efficiency
  - Minimizes the total number of packets sent
- **Cells**, in contrast, are both **fixed in length** and **small in size**

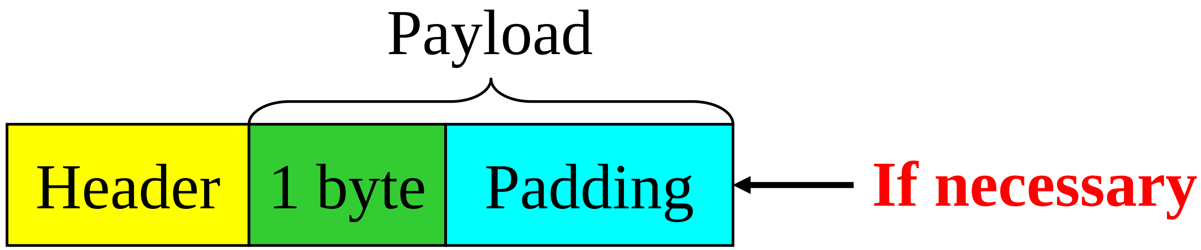
Why fixed-length cells?

- High speed switching
- Easy and simple
- Parallelization
- Cut-through effect (smaller queueing time for store-and-forward networks)

Variable Packet Length

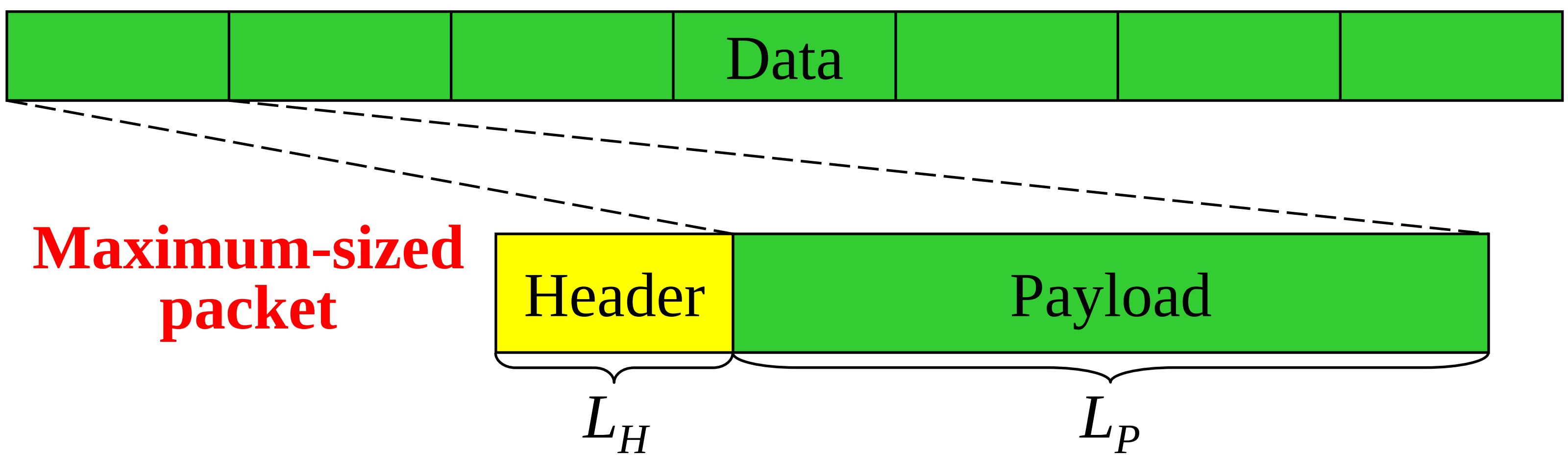
- If a host only has **1 byte** to send

minimum-sized packet



- If a host only has a **large file** to send

Maximum-sized packet



Efficiency: $L_P / (L_H + L_P)$

Cells Size

- The reasons of using fixed-length packets are
  - **Easy implementation** of hardware switches
    - When the length of each packet is fixed and known, the job of processing packets is simpler
  - **Enable parallelism**, improves the **scalability** of switch designs
    - Cell switching eases the task of building hardware
    - If all packets are the same length, we can have lots of **switching elements** all doing the same thing **in parallel**

Cells Size

- The delay variation is important for some applications
- If the link speed is 100 Mbps and the packet-length is **4KB**
  - Transmission time: $4096 \times 8/100$ = **327.68 μs**
  - A **high-priority** packet may need to wait for 327.68 μs
- If the packet-length is **53 byte**
  - Transmission time: $53 \times 8/100$ = **4.24 μs** (much smaller)
- If the packet-length is **too short**
  - The amount of header information is fixed
  - The bandwidth efficiency **goes down**
- If the packet-length is **too big**
  - Need to pad transmitted data to fill a complete cell
  - Data: 1 byte; Payload size: 48 bytes $\Rightarrow$ Padding: 47 bytes

Cells Size

- Voice services use 64 kbps PCM (Pulse-Code Modulation)
  - 8-bit samples taken at 8 KHz sampling rate
    - 8-bit: 256 levels are used
    - 8 KHz: voice signal bandwidth is 4 Khz
  - 1 byte is sampled for every 125 μs
- If a cells are **1000 bytes long**, it take 125 ms to collect a full cell before the start of transmission
  - **Long latency** is noticeable to a human listener
  - Long latency ⇒ Echo ⇒ can be eliminated by an **echo canceller**

Latency ≥ transmit time + propagation delay

- Use a large cell size or a small cell size?

Cells Size

- US telephone companies were pushing for a **64-byte** cell size
  - US is a **large enough country** (large propagation distance ⇒ **Long latency**), the echo cancellers are needed anyway
  - Larger cell size will improve the header-to-payload ratio
- Europeans were advocating **32-byte** cells
  - Their countries are a **small enough size**
  - No echo cancellers are needed if the latency is small enough, i.e. the cell size is small enough
- **Compromise:** Averaging the cell size
  - (64 + 32)/2 = **48 bytes**

Cell Format

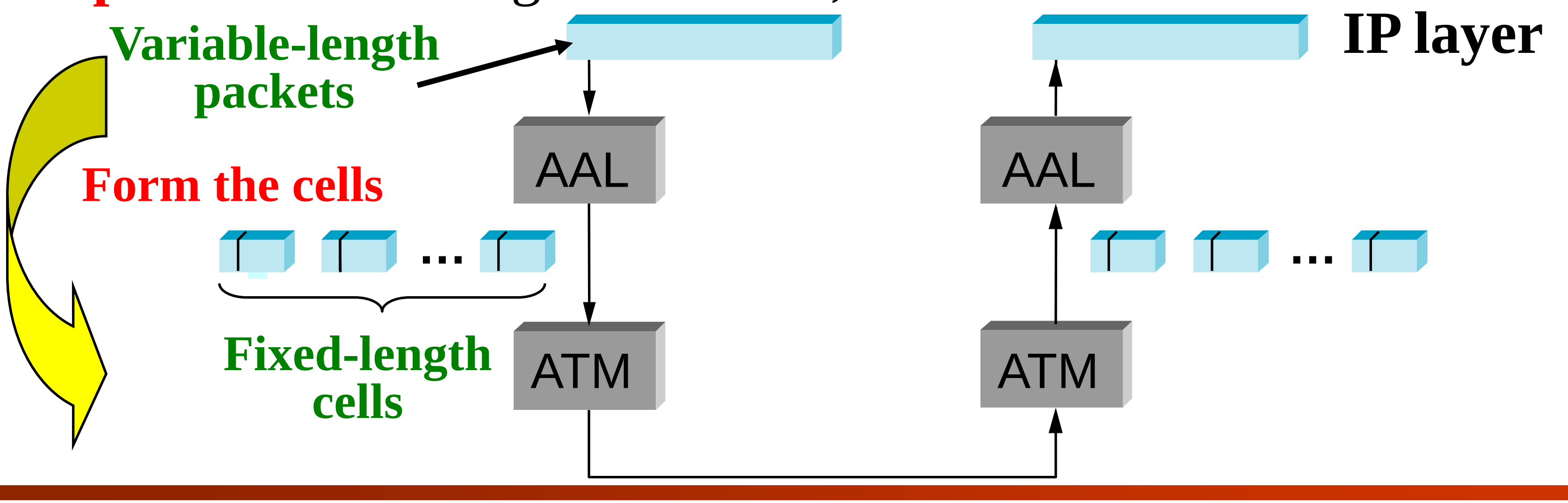
4	8	16	3	1	8	384 (48 bytes)
GFC	VPI	VCI	Type	CLP	HEC (CRC-8)	Payload

UNI cell format

- Two types of cell format:
  - UNI (user-network interface) format: customer-operator
  - NNI (network-network interface) format: pair of operators
- NNI format replace the GFC field with 4 extra bits of **VPI**
- **GFC (generic flow control):** have local significance at a site
- **8-bit VPI (virtual path identifier)** and **16-bit VCI (virtual circuit identifier):** 24-bit used to identify a virtual connection
- **Type:** When the first bit is **set** ("1") or **clear** ("0"), the cell relates to **management functions** or **user data**, respectively.
- **CLP (cell loss priority):** set this bit to indicate cells that should be dropped **preferentially** in the event of overload ← **QoS**

Segmentation and Reassembly (SAR)

- **Segmentation (Fragmentation):** fragments the high-level message into low-level packets at the source
- **Reassembly:** reassembles the fragments back together at the destination
- A protocol layer **(ATM Adaptation Layer, AAL)** was added between ATM and the **variable-length packet protocols** that might use ATM, such as IP



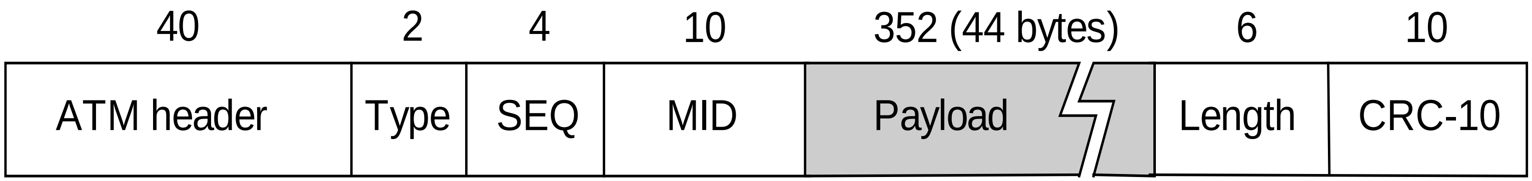
AAL 3/4

- Convergence Sublayer Protocol Data Unit (CS-PDU)

8	8	16	< 64 KB	0– 24	8	8	16
CPI	Btag	BASize	User data	Pad	0	Etag	Len

  - CPI: commerce part indicator (version field)
  - Btag/Etag:beginning and ending tag
  - BAsize: hint on amount of buffer space to allocate
  - Length: size of whole PDU

Cell Format



- Type
  - BOM: beginning of message
  - COM: continuation of message
  - EOM end of message
- SEQ: sequence of number
- MID: message id
- Length: number of bytes of PDU in this cell

AAL5

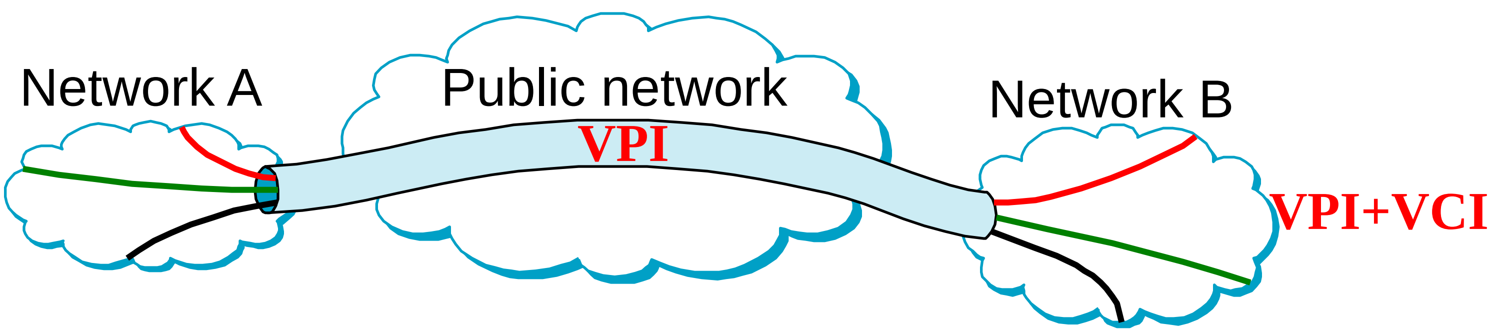
- CS-PDU Format

< 64 KB	0– 47 bytes	16	16	32
Data	Pad	Reserved	Len	CRC-32

  - pad so trailer always falls at end of ATM cell
  - Length: size of PDU (data only)
  - CRC-32 (detects missing or misordered cells)
- Cell Format
  - end-of-PDU bit in Type field of ATM header

Virtual Paths

- The switch in the public network would use the **VPI** to make **forwarding decisions**
  - A virtual circuit network with **8-bit** circuit identifiers
- The **16-bit VCI** is of no interest to these public switches
  - **Not used for switching in the public network**
- The virtual path acts like a fat pipe that contains a bundle of virtual circuits



Physical Layers for ATM

- ATM can run over several different physical media and physical-layer protocols
  - ATM-over SONET
  - Wireless ATM

ATM in LAN

- ATM is a **switching technology**, whereas Ethernet and 802.5 are **shared-media technologies**