Chapter 3.2
Internetworking

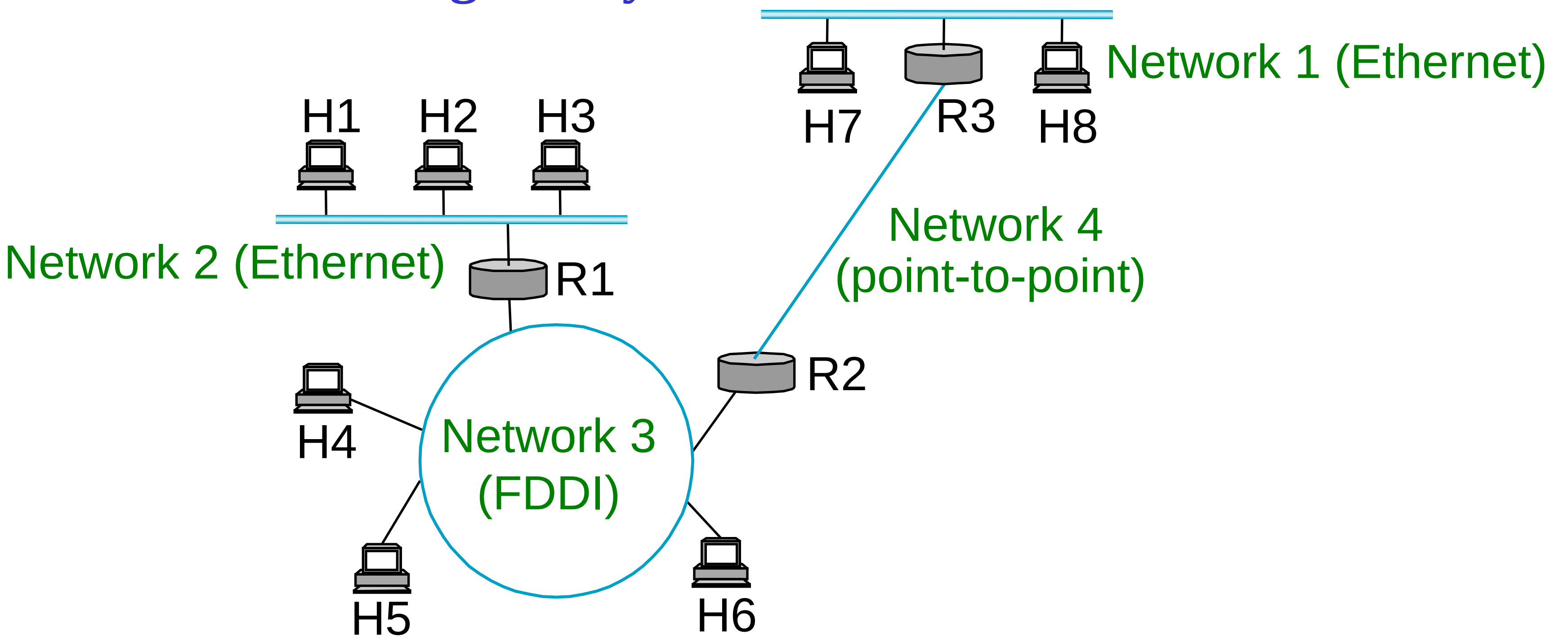
Internetworking

- There are two important problems in network connection:
  - **Heterogeneity:** users on one type of network want to be able to communicate with users on other type of networks
    - To provide a useful and fairly **predictable** **host-to-host service** over different networks
  - **Scale:** to have an efficient and stable operation on the growth of networks
    - **Routing:** to find an efficient path through a network
    - **Addressing:** to provide suitable identifiers for all nodes

Simple Internetworking (Internet Protocol, IP)

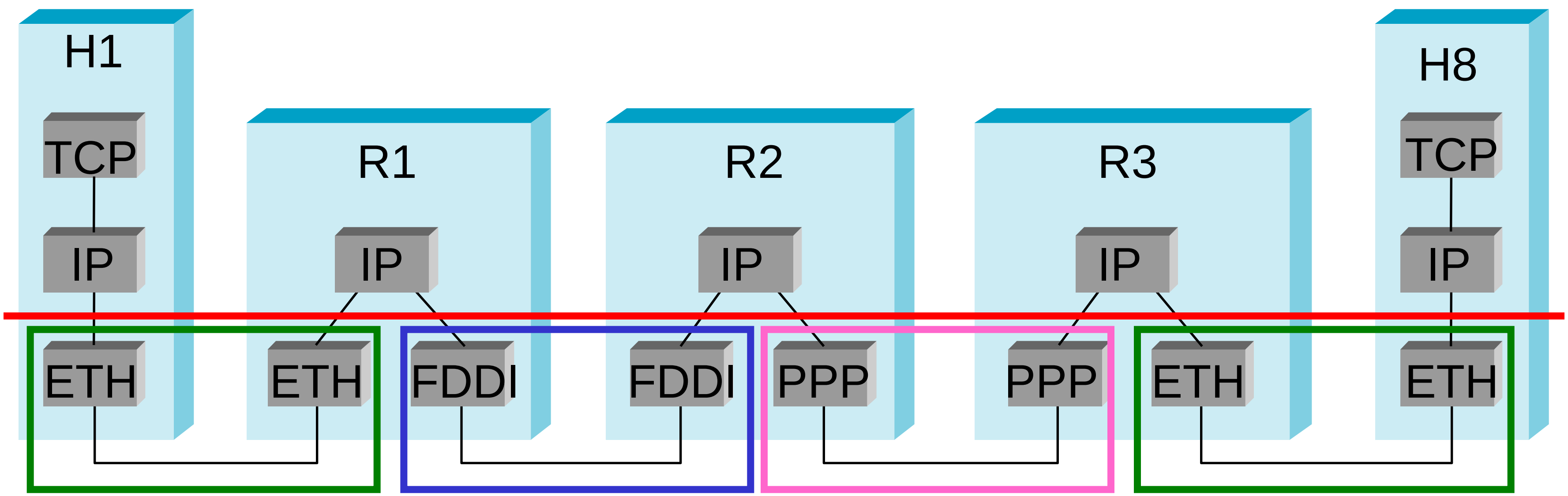
Internetworking

- 4 single-technology networks: Ethernet, FDDI, point-to-point
- **Router:** The nodes that interconnect different networks
  - Also called **gateway**



Internet Protocol (IP)

- The **Internet Protocol** is the key tool used to build **scalable**, **heterogeneous** internetworks
  - Allows all nodes and networks to function as **a single logical internetwork**
- Hosts H1 and H8 are logically connected by the internet



Service Model

- The IP service model can be thought of as having two parts:
  - **An addressing scheme:** provides a way to identify all hosts in the internetwork
  - **A datagram (connectionless) model:** for data delivery
- This service model is called **best effort**
  - Although IP makes every effort to deliver datagrams, **it makes no guarantees**
  - If something goes wrong and the packet fails to reach its intended destination, **the network does nothing**
    - lost, corrupted, misdelivered, …
  - It does not make any attempt to recover from the failure
  - It is sometimes called **an unreliable service**

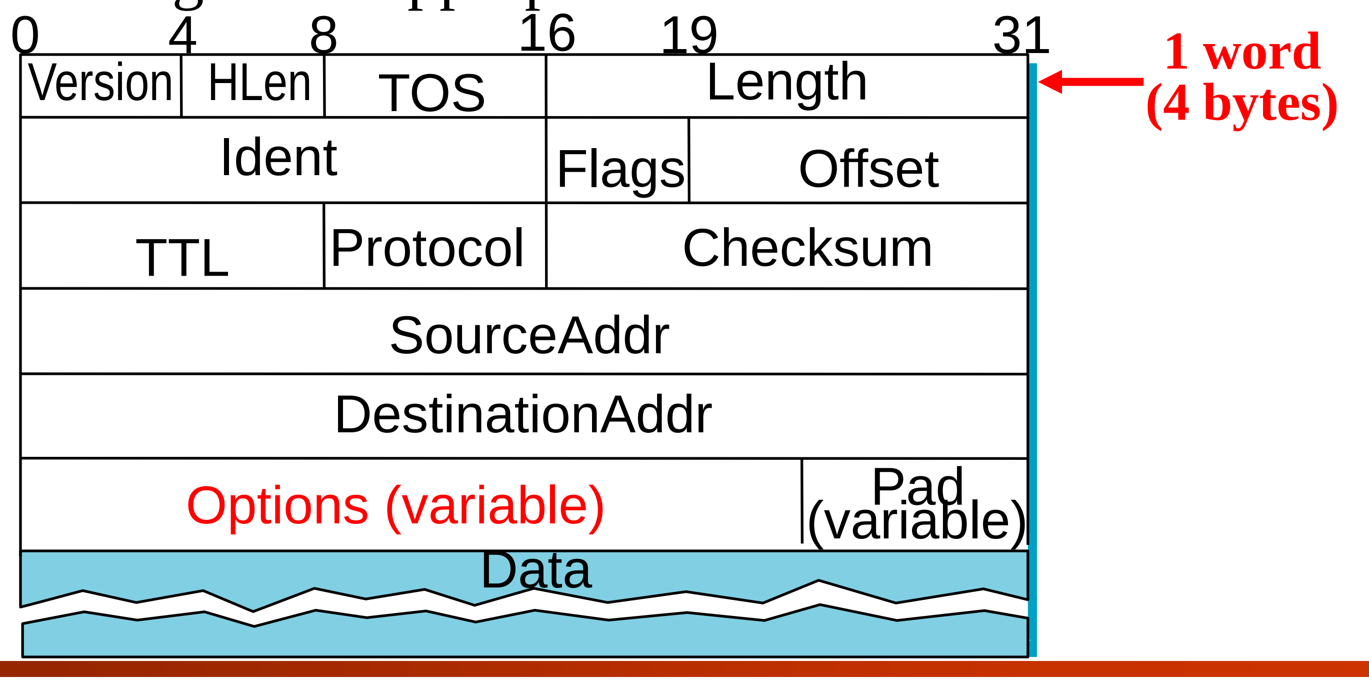
Datagram Delivery

- Keeping the routers as simple as possible was one of the original design goals of IP
  - **Best-effort, connectionless service** is the simplest service for an internetwork
- One of the most important characteristics of IP is
  - It can **"run over anything"**, i.e. any network technologies
- The best-effort service means
  - The packets can **get lost**
  - Sometimes the packets may get delivered **out of order**
  - Sometimes a packet may get delivered **more than once**
  - The higher-level protocols or applications that run above IP need to **be aware of all failure and fix them**

Packet Format

Packet Format

- Version: the version of IP
  - The current version of IP is 4; called **IPv4** ⇨ **IPv6**
  - At the start of the datagram
    - The receiver can process the rest of the packet according to the appropriate format

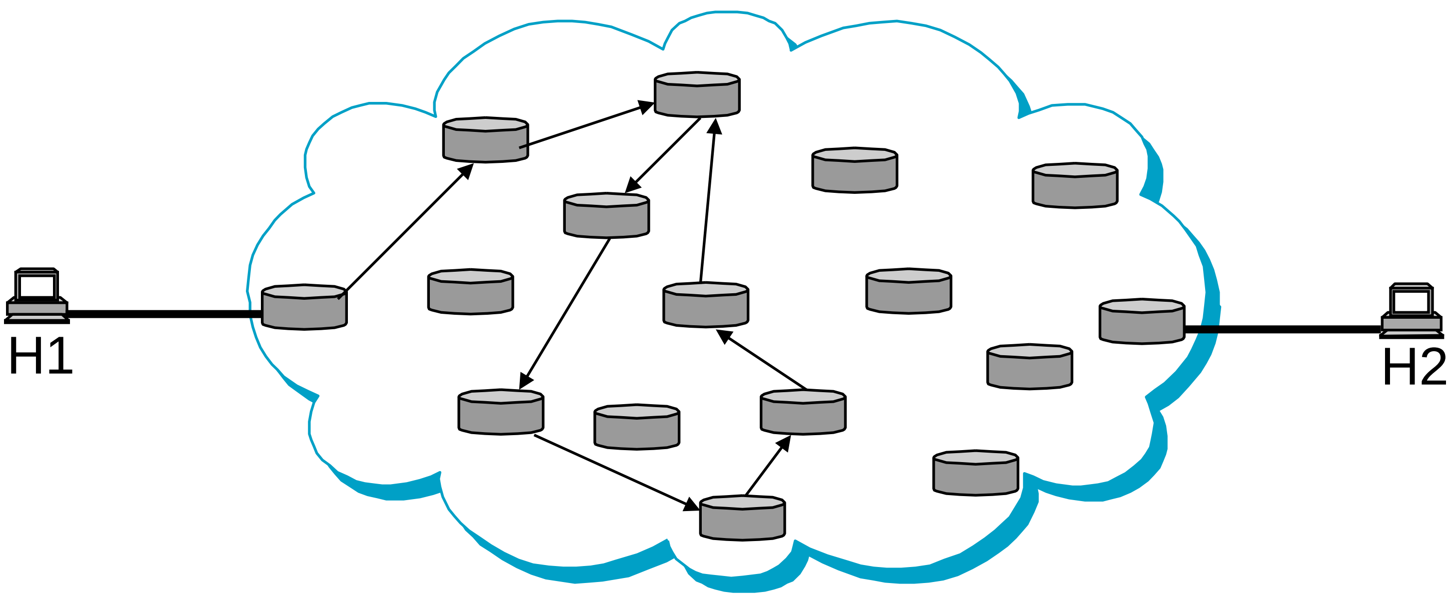


Packet Format

- **HLen:** the length of the header in 32-bit words
  - When there are no option fields, the header is **5 words** (**20 bytes**) long
- **TOS (type of service):** allow packets to be treated differently based on application needs
- **Length:** the length of the datagram, including the header
  - Counted in **bytes**; the maximum size is **65,535 bytes**
- The second word of the header contains information about fragmentation: **Ident, Flags, Offset**
- **TTL (time to live):** TTL **was** set to a specific number of seconds that **the packet would be allowed to live**
  - Now, it became a **hop count**: a good way to catch packets that are struck in routing loops

Packet Format

- The sending host sets the initial value of this field
  - **Set it too high:** packets could circulate rather a lot before getting dropped
  - **Set it too low:** they may not reach their destination
- The value **64** is the current default



Packet Format

- **Protocol:** a **demultiplexing key** that identifies the higher-level protocol: **TCP(6), UDP(17)**
- **Checksum:** calculated by considering the entire **IP header** as a sequence of 16-bit words; **discard** any packet that fails the checksum
- **SourceAddr:** source address
- **DestinationAddr:** destination address

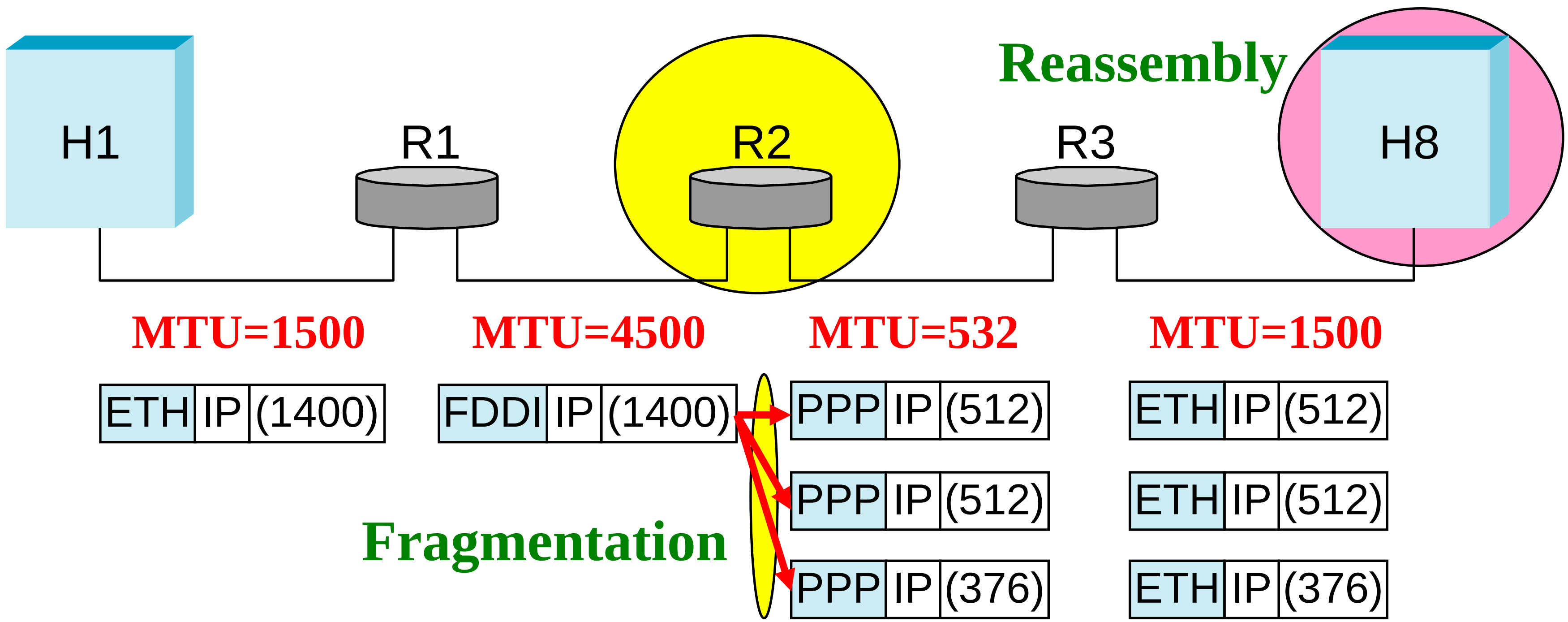
Fragmentation and Reassembly

Fragmentation and Reassembly

- Every network type has a **maximum transmission unit (MTU)**
  - The **largest IP datagram** that it can carry in a frame
  - This value is smaller than the largest packet size **(65,535 B)**
- When a host sends an IP datagram, it can choose any size that it wants: **generally the MTU of the attached network**
- The fragmentation will only be necessary if the path to the destination includes a network **with a smaller MTU**
- To enable these fragments to be reassembled at the receiving host, they all carry **the same identifier** in the **Ident** field
- This identifier is chosen by the sending host
  - **Unique** among all the datagrams that might arrive at the destination over some reasonable time period

Fragmentation and Reassembly

- Assume that Ethernet MTU = 1500; FDDI MTU = 4500; PPP MTU = 532
- 1420-byte datagram: 20-byte IP header plus 1400-byte data
- 532-byte datagram: 20-byte IP header plus 512-byte data



Fragmentation and Reassembly

- **Flags field:** 'XX**1**' there are more fragments to follow
- **Offset field:** the starting of the data in **8-byte** chunks
- The 1st packet: starts with the **1st byte**
- The 2nd packet: starts with the **513th byte**
  - $512 \div 8 =$ **64**
- The 3rd packet: starts with the **1025th byte**
  - $1024 \div 8 =$ **128**

Unfragmented

Start of header				
Ident = x			0	Offset = 0
Rest of header				
1400 data bytes				

Fragmented Packets

Start of header				
Ident = x			1	Offset = 0
Rest of header				
512 data bytes				

Start of header				
Ident = x			1	Offset=64
Rest of header				
512 data bytes				

Start of header				
Ident = x			0	Offset=128
Rest of header				
376 data bytes				

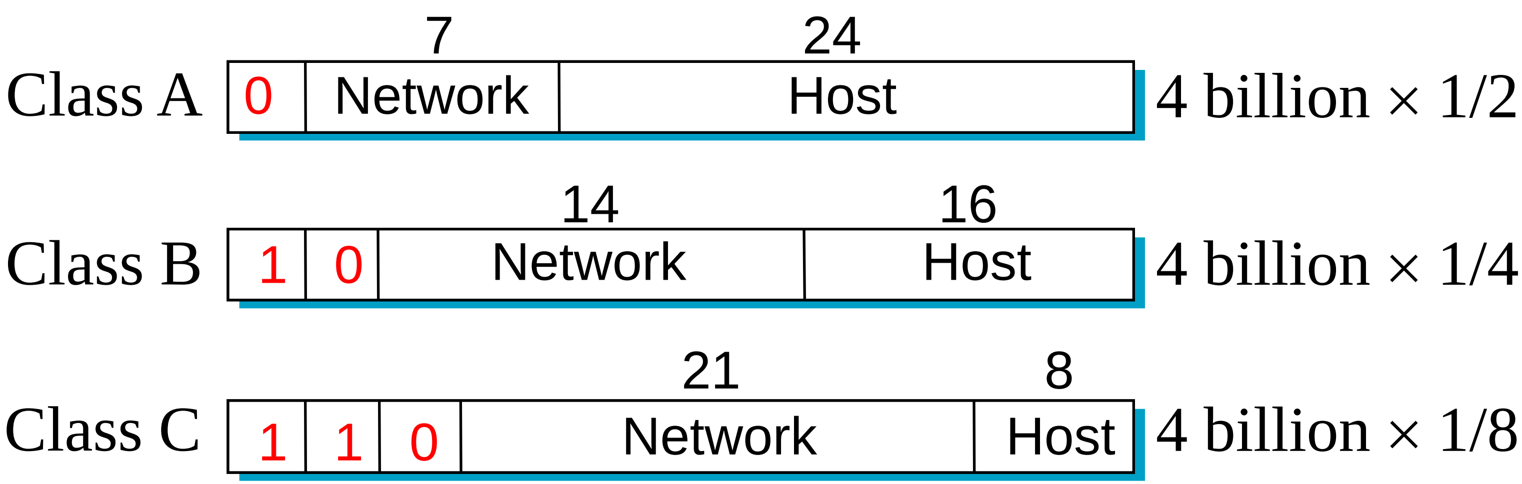
Global Addresses

Global Addresses

- **Global addressing scheme:** no two hosts have the same address, i.e. Global uniqueness (the address belongs to a host)
- Ethernet addresses are **globally unique**
  - They have **no structure** and therefore are not suitable
- **IP addresses (hierarchical):** they are made up of several parts that correspond to some sort of hierarchy in the internetwork
  - Consists of two parts: a **network part** and a **host part**
- The network part identifies **the network** to which the host is attached
- The host part identifies **each host** uniquely on a particular network

Global Addresses

- IP addresses **(32-bits long)** are divided into three different classes: approximately **4 billion** possible IP addresses
  - Each of which defines different-sized network and host parts
- **Class A: 126** class A networks (**0** and **127** are reserved)
  - Each can accommodate up to $2^{24} – 2$ hosts



Global Addresses

- **Class B: 2^{14}** class B networks; up to 65,534 hosts
- **Class C: 2^{21}** class C networks; up to 254 hosts
- The original idea was that the Internet would consist of
  - A small number of wide area networks (Class A)
  - A modest number of site- (campus-) sized networks (Class B)
  - A large number of LANs (Class C)
- IP addresses are written as **four decimal integers** separated by dots
  - Each integer represents the decimal value contained in 1 byte of the address (0 ~ 255)
  - **171.69.210.245**

Datagram Forwarding

Datagram Forwarding in IP

- Every IP datagram contains the IP address of the destination host.
- The “network part” of an IP address uniquely identifies a single physical network that is part of the larger Internet.
- All hosts and routers that share the same network part of their address are connected to the same physical network and thus can communicate with each other by sending frames over that network.
- Every physical network that is part of the Internet has at least one router (the default router) that, by definition, is also connected to at least one other physical network. The router can exchange packets with hosts or routers on either network.

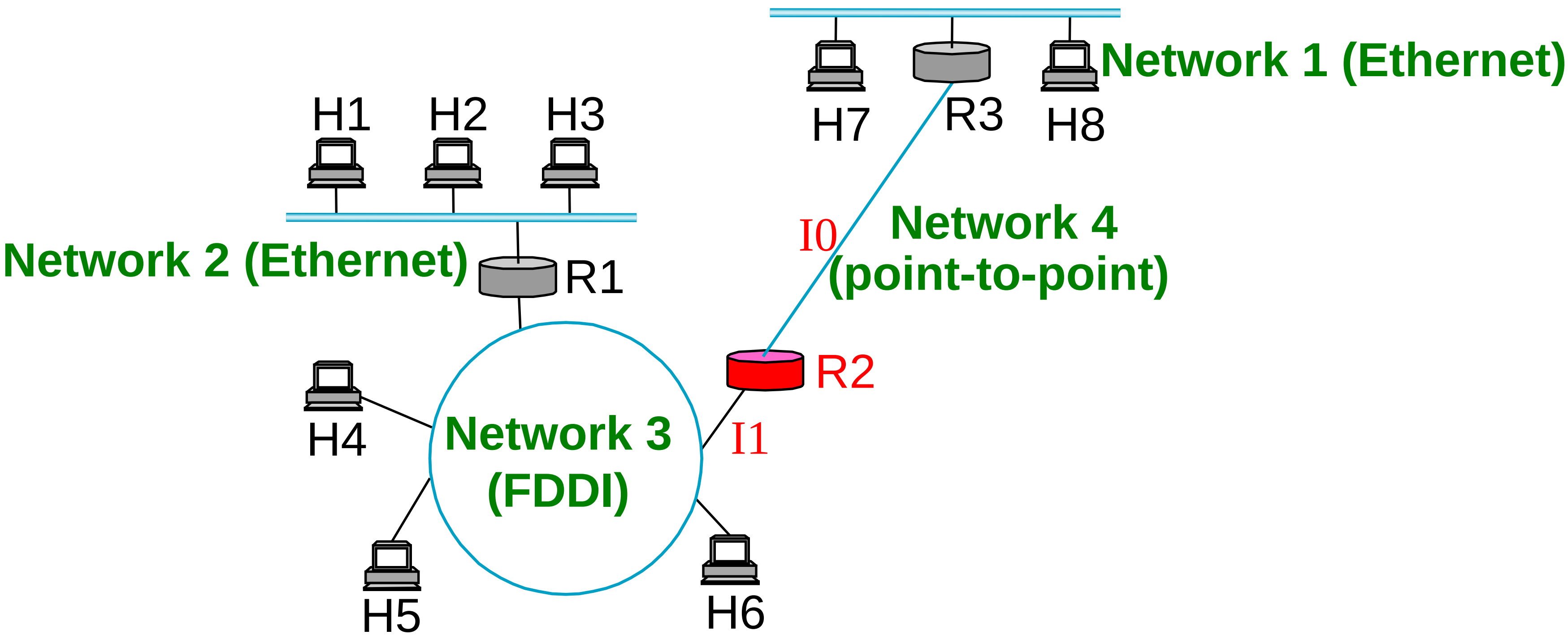
Datagram Forwarding in IP

- The concept of default routers
- Hierarchical aggregation
- Scalabilty

Datagram Forwarding in IP

- Routing Table for **router R2**

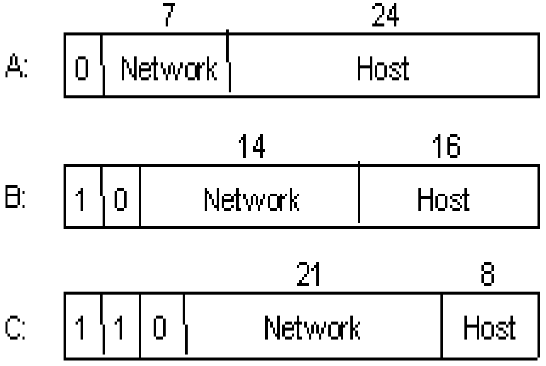
NetworkNum	1	2	3	4
NextHop	R3	R1	Interface 1	Interface 0



Subnetting

How to Make Routing Scale

- Flat versus Hierarchical Addresses
- Inefficient use of Hierarchical Address Space
  – class C with 2 hosts (2/255 = 0.78% efficient)
  – class B with 256 hosts (256/65535 = 0.39% efficient)
- Still Too Many Networks
  – routing tables do not scale
  – route propagation protocols do not scale

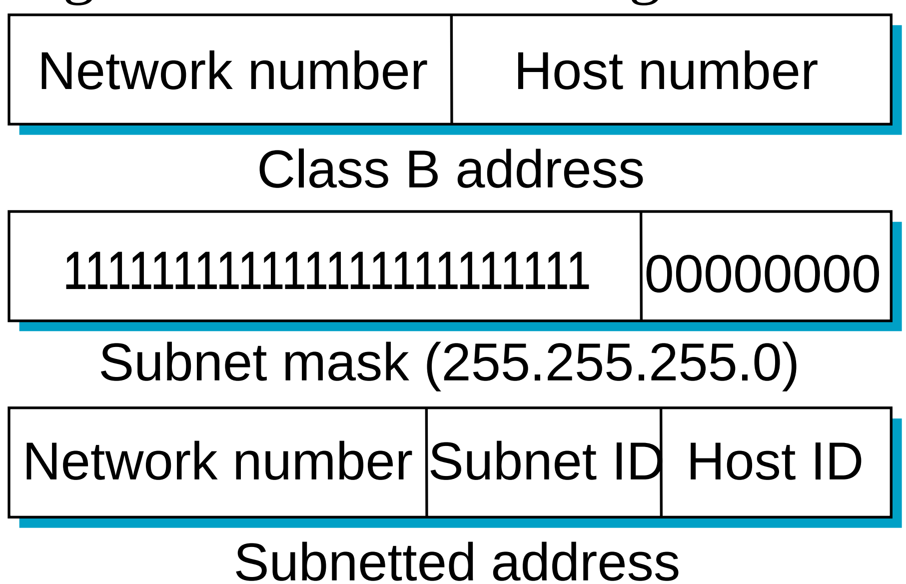


Subnetting

- **Subnetting** provides an simply way to reduce the total number of networks that are assigned
  - Take a single **IP network number** and allocate the IP network address to **several** physical networks – **subnets**
- A single network number is shared among multiple networks
  - All the nodes on each subnet are **configured** with a **subnet mask**
  - The subnet mask introduce a **subnet number**
  - All hosts on the **same physical network** will have the **same subnet number**
  - Introduce another level of hierarchy into the IP address

Subnetting

- Share a single class B address among several physical networks
  - Use a subnet mask of **255.255.255.0** (24 bits ‘1’; 8 bits ‘0’)
  - A **network part**, a **subnet part**, and a **host part**
- Subnetting configure a host with both an **IP address** and a **subnet mask**
- All hosts on a given subnet are configured with **the same mask**



Subnetting

- The **bitwise AND** of the IP address and the subnet mask defines the **subnet number** of the host
- 128.96.34.15 AND 255.255.255.128 ⇒ 128.96.**34.0**

  128.96.34.15 ⇒ **10000000.01100000.00100010.00001111**

  255.255.255.128 ⇒ **11111111.11111111.11111111.10000000**

  128.96.34.0 ⇒ **10000000.01100000.00100010.00000000**
- 128.96.34.130 AND 255.255.255.128 ⇒ 128.96.**34.128**

  128.96.34.130 ⇒ **10000000.01100000.00100010.10000010**

  255.255.255.128 ⇒ **11111111.11111111.11111111.10000000**

  128.96.34.128 ⇒ **10000000.01100000.00100010.10000000**

Class B Network Number

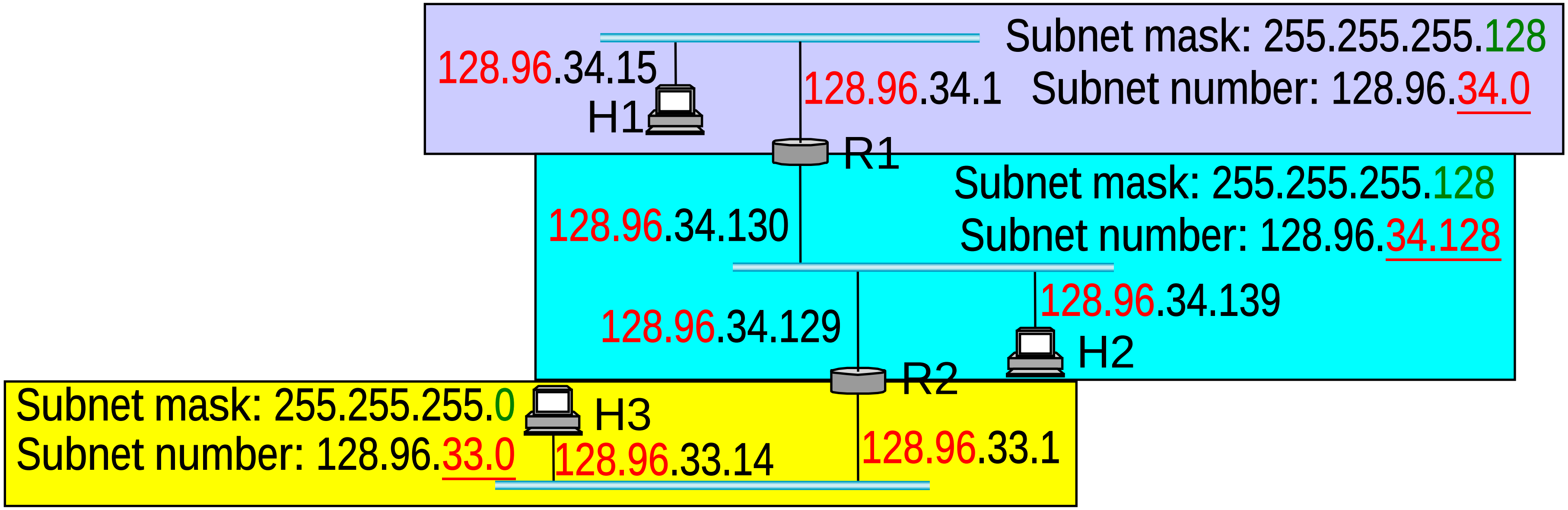
Subnetting

- 128.96.33.14 AND 255.255.255.0 ⇒ 128.96.**33.0**

  128.96.33.14 ⇒ **10000000.01100000.00100001.10001110**

  255.255.255.0 ⇒ **11111111.11111111.11111111.00000000**

  128.96.33.0 ⇒ **10000000.01100000.00100001.00000000**



Subnetting

- When the host wants to send a packet to a certain IP address
  - Perform a bitwise AND between **its own subnet mask** and the **destination IP address** ⇒ check the subnet number
  - If the result **equals** the subnet number of the sending host
    - The packet can be delivered **directly** over the subnet
  - If the results are **not equal**, the packet needs to be sent to a **router** to be forward to another subnet
- A forwarding table should be used to support packet routing

Forwarding table in R1

SubnetNumber	SubnetMask	NextHop
128.96.34.0	255.255.255.128	Interface 0
128.96.34.128	255.255.255.128	Interface 1
128.96.33.0	255.255.255.0	R2

Forwarding Algorithm

```
D = destination IP address
for each entry (SubnetNum, SubnetMask, NextHop)
   D1 = SubnetMask & D
   if D1 = SubnetNum
      if NextHop is an interface
         deliver datagram directly to D
      else
         deliver datagram to NextHop
```

- Use a default router if nothing matches
- Not necessary for all 1s in subnet mask to be contiguous
- Can put multiple subnets on one physical network
- Subnets not visible from the rest of the Internet

Supernetting

- Assign block of contiguous network numbers to nearby networks
  (class c : 192.4.16.x through 192.4.31.x ,the first twenty bits is the same,we have 20 bits' network number)
- Called CIDR: Classless Inter-Domain Routing
- Represent blocks with a single pair
  `<length, value>`
- Restrict block sizes to powers of 2
- Use a bit mask (CIDR mask) to identify block size
- All routers must understand CIDR addressing
- Overlapping prefixes: "longest match"

Address Translation

Address Translation

- IP datagrams contain IP addresses
- The physical interface on the host or router only understands **the particular addressing scheme (link-level address, i.e. MAC address)**
- Need to **translate** the IP address to a **link-level address**
  - Such as a 48-bit Ethernet address
- One simple way is to encode a host's physical address in the host part of its IP address
  - Physical address ⇒ IP address
  - Physical address: 00100001**(33)** 01010001**(81)** → 33. 81
  - The assigned IP address: 128.96.**33.81**
  - Limited to the 16-bit network's physical addresses

Address Translation

- A more general solution is for each host to **maintain a table of address pairs**
  - Table: (IP address, Physical address)
  - Map IP addresses into physical addresses
  - A system administrator may **centrally** manage the mapping table and then copies to each host on the network
- Each host may **dynamically** learn the contents of the table using the network
  - **Address Resolution Protocol (ARP)**

Address Resolution Protocol (ARP)

- If a host wants to send an IP datagram to a host (or router) on the same network
  - It first checks for a mapping in the cache
  - If no mapping is found:
    - **Invoke the ARP** over the network
  - It **broadcasts** an **ARP query** onto the network
    - Containing the **target IP address**
  - Each host checks if the ARP query matches its IP
  - If it does **match**, the host sends back a **response message**
    - Containing its **link-layer address**
  - The originator adds the information to its ARP table

Address Resolution Protocol (ARP)

- The ARP packet format for IP-to-Ethernet address mappings

Ethernet IP

<table>
<tr><th>0</th><th>8</th><th>16</th><th>31</th></tr>
<tr><td colspan="2">Hardware type = 1</td><td colspan="2">ProtocolType = 0x0800</td></tr>
<tr><td>HLen = 48</td><td>PLen = 32</td><td colspan="2">Operation</td></tr>
<tr><td colspan="4">SourceHardwareAddr (bytes 0 – 3)</td></tr>
<tr><td colspan="2">SourceHardwareAddr (bytes 4 – 5)</td><td colspan="2">SourceProtocolAddr (bytes 0 – 1)</td></tr>
<tr><td colspan="2">SourceProtocolAddr (bytes 2 – 3)</td><td colspan="2">TargetHardwareAddr (bytes 0 – 1)</td></tr>
<tr><td colspan="4">TargetHardwareAddr (bytes 2 – 5)</td></tr>
<tr><td colspan="4">TargetProtocolAddr (bytes 0 – 3)</td></tr>
</table>

ATMARP

- Lack of broadcast capability in ATM networks
- ARP server
- A VC to the ARP server is established when an ATM host boots.
- Ask ARP server to provide the corresponding ATM address when sending an IP packet.
- Use ATM signaling to set up a VC to the destined ATM host.

Host Configuration

Host Configuration

- IP address must reflect the structure of the internetwork
  - It is **not possible** for the IP address to be configured into the network adaptor or host **in advance**
- In addition to the IP address, a host needs **some other information** before it can start sending packets
  - For example, **the address of a default router**
- One way is to **manually configure** the IP information needed by a host **(like you computer)**
  - The configuration process is very error-prone
  - Need to ensure that the network number is correct, and that no two hosts using the same IP address

Host Configuration (DHCP)

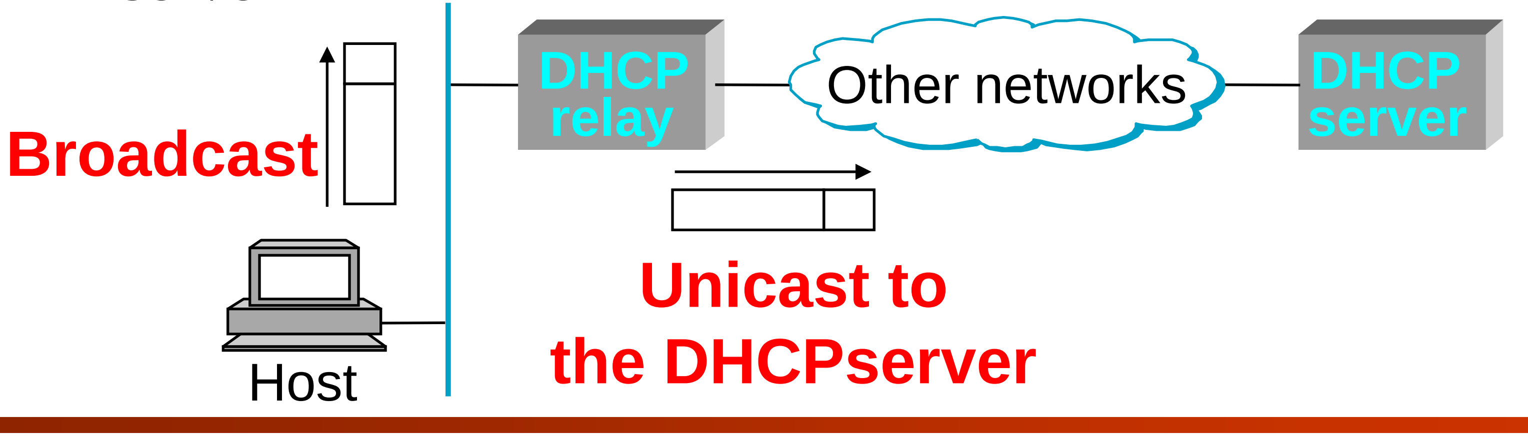
- **Automated configuration:**
  - **Dynamic Host Configuration Protocol (DHCP)**
- DHCP relies on the existence of a **DHCP server**, which
  - Provides **configuration information** to hosts
- The configuration information for each host can be stored in the DHCP server
  - **Pre-configured** host information in a private network
  - Can be **Automatically retrieved** by each host

Host Configuration (DHCP)

- In another scenario, the DHCP server maintains a pool of **available IP addresses**
  - It **dynamically** hands out to hosts **on demand**
  - Hosts **cannot** keep addresses **permanently**; otherwise the server would eventually exhaust its address pool
  - For example, a public network
    - **WLAN**
- DHCP allows addresses to be **'leased'** for some period of time
  - Once the lease time **expires**, the server is free to return that address to its pool

Host Configuration (DHCP)

- To contact a DHCP server, a newly booted or attached host sends a **DHCPDISCOVER** message to a special IP address
  - **255.255.255.255:** an IP broadcast address
- It is not desirable to require one DHCP server on every network: DHCP uses the concept of a **relay agent**
- There is at least **one relay agent** on each network
  - Unicasts the DHCPDISCOVER message to the DHCP server



Host Configuration (DHCP)

- The format of a DHCP message
- The client sends its hardware address (e.g. Ethernet address) to the DHCP server
- The DHCP server replies the **assigned IP address**

Operation	HType	HLen	Hops
Xid			
Secs		Flags	
ciaddr			
yiaddr			
siaddr			
giaddr			
chaddr (16 bytes)			
sname (64 bytes)			
file (128 bytes)			
options			

IP address

Host hardware address

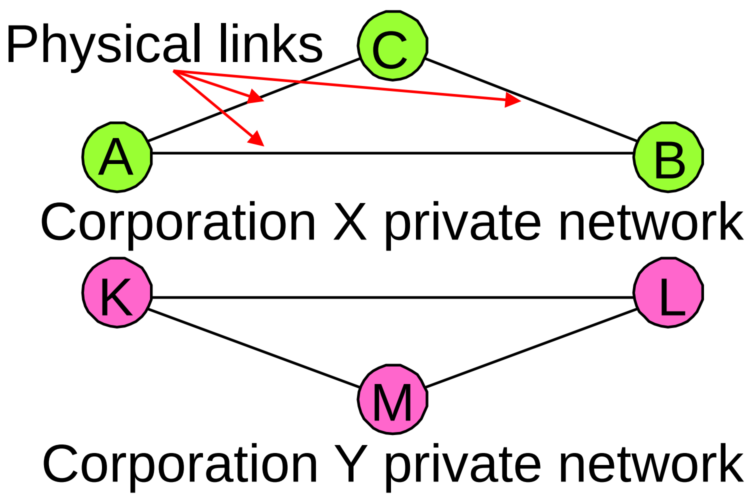
Internet Control Message Protocol (ICMP)

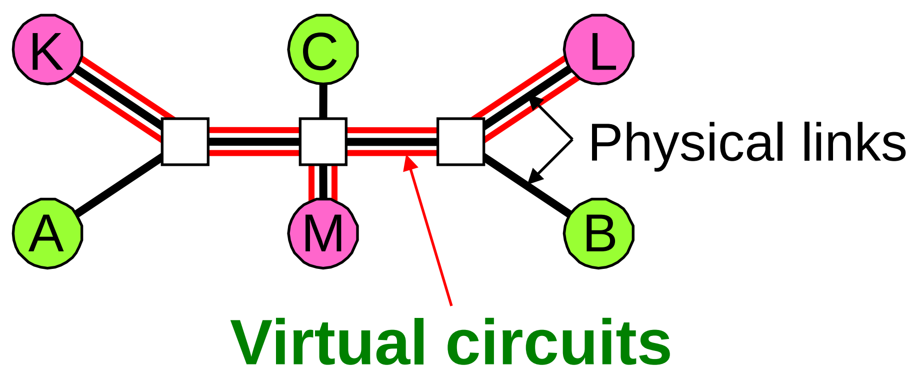
- Echo (ping)
- Redirect (from router to source host)
- Destination unreachable (protocol, port, or host)
- TTL exceeded (so datagrams don’t cycle forever)
- Checksum failed
- Reassembly failed
- Cannot fragment

Virtual Networks and Tunnels

Virtual Networks and Tunnels

- **Private network (PN):** the communication is restricted to take place only among the sites
  - It uses the **leased transmission lines**
- **Virtual private network (VPN):** replace the leased transmission lines by some sort of **shared network**
  - It still provides a **logical point-to-point** connection between hosts





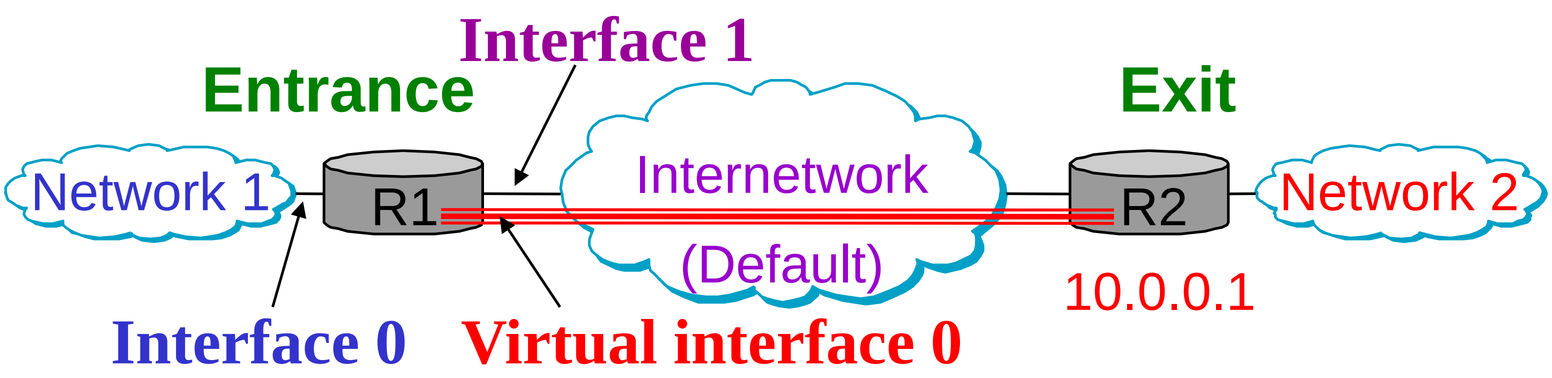
Virtual Networks and Tunnels

- **IP tunnel: avoids the connectivity** between different VPNs sharing the same physical lines
  - A **virtual** point-to-point link between a pair of nodes
    - Is actually separated from other networks
- The virtual link is created within the router at the **entrance** to the tunnel
  - with the IP address of the router at the **far end** of the tunnel
- When the router intends to send a packet over this virtual link
  - It **encapsulates** the packet inside an IP datagram
  - The **destination IP address** is the router at the far end of the tunnel
  - The **source IP address** is the encapsulating router

Virtual Networks and Tunnels

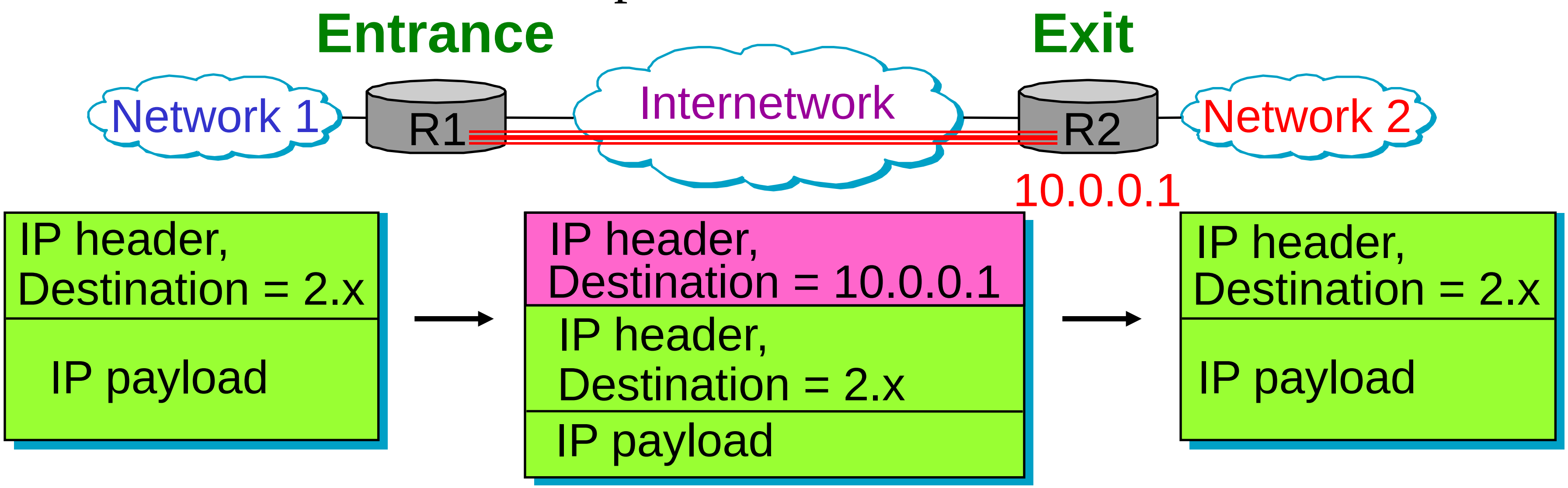
- R1 has two physical interfaces
  – Interface 0: connects to network 1
  – Interface 1: connects to a large internetwork
  – Virtual Interface 0: the interface to the tunnel

NetworkNum	1	2	Default
NextHop	Interface 0	Virtual interface 0	Interface 1



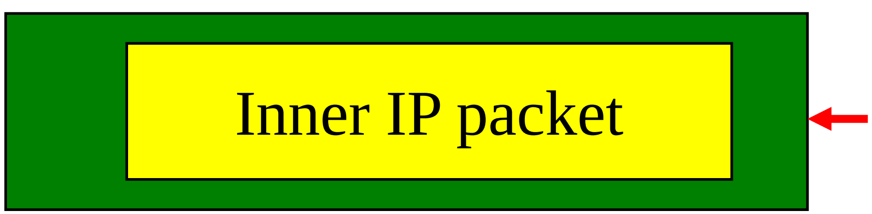
Virtual Networks and Tunnels

- Suppose R1 receives a packet from network 1 that contains an address in network 2
  - This packet should be sent out **virtual interface 0**
- R1 takes the packet and adds an IP header **addressed to R2**
  - IP address of R2: 10.0.0.1
- Then R1 forwards the packet into the **internetwork**



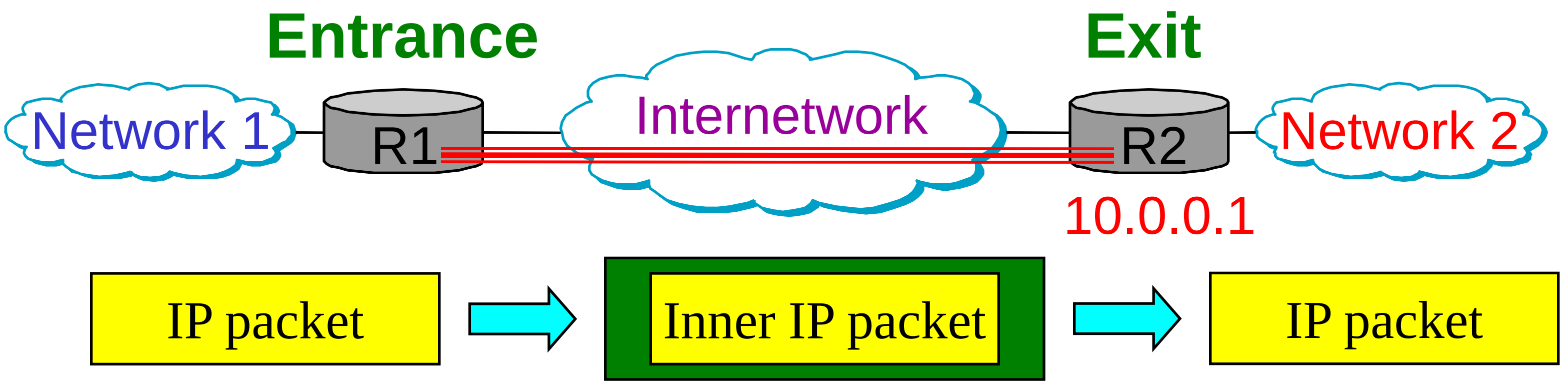
Virtual Networks and Tunnels

- Once the packet leaves R1: it is a normal IP packet destined to R2
  - All routers in the **internetwork** forward this packet until it arrives at R2
- When R2 receives the packet, it finds that **it carries its own address**, so it removes the IP header and looks at the payload
- R2 finds an **inner IP packet** whose destination address is in network 2, and **forwards this packet in network 2**



Packet in the tunnel

Virtual Networks and Tunnels



- Tunneling does have its disadvantageous:
  - It increases the length of packets and **wastes the bandwidth**
  - **More work** than normal forwarding is required at the end of the tunnel

Routing

Routing

- **Forwarding** consists of
  - Taking a packet,
  - Looking at its destination address,
  - Consulting a table, and
  - Sending the packet in a direction determined by that table
- **Routing** is
  - The process by which forwarding tables are built
- Routing depends on complex **distributed** algorithms
  - The Internet is a very large network

Routing

- **Forwarding table:** contains enough information to accomplish the forwarding function
  - The mapping from a **network number** to an **outgoing interface** and some **MAC information**
- **Routing table:** is built up by the routing algorithms as a precursor to build the forwarding table
  - The mapping from **network numbers** to **next hops**

Routing Table

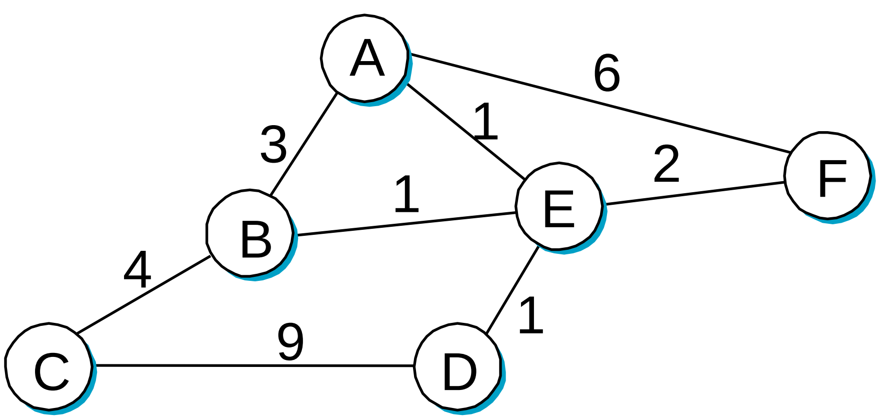
Network Number	Next Hop
10	171.69.245.10

Forwarding Table

Network Number	Interface	MAC Address
10	if0	8:0:2b:e4:b:1:2

Network as a Graph

- The nodes of the graph may be either hosts, switches, routers, or networks
- The **edges** of the graph correspond to the network links
  - Each edge has **a associated cost**
  - Indicates the desirability of sending traffic over that link
- Routing: to find the **lowest-cost path** between any two nodes
  - The cost of a path: the sum of the costs of all the edges on the path



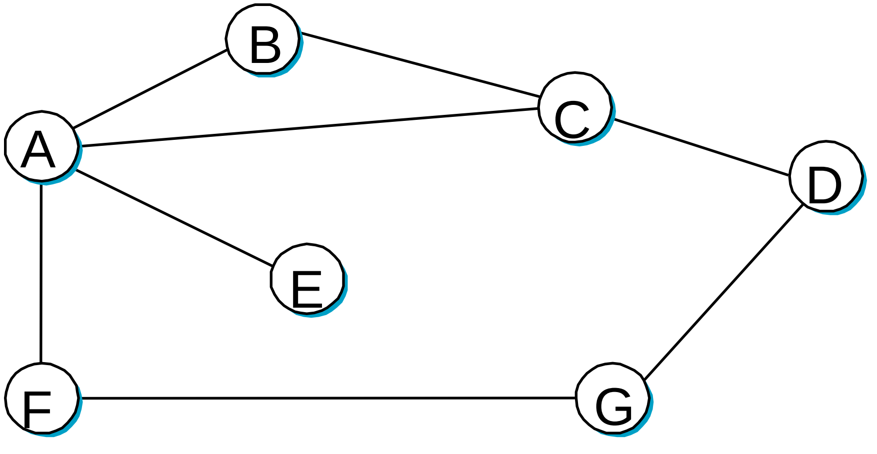
Routing Protocols

- In practical networks, routing is achieved by running **routing protocols** among the nodes
- The protocols provide a **distributed, dynamic** way to solve the problem of finding the lowest-cost path
- The reason of using **distributed** algorithms in routing:
  - It is difficult to make centralized solutions scalable
- The reason of using **dynamic** algorithms in routing:
  - **Link and node failures** and **change of edge costs** are generally present in the network
- Two main classes of routing protocols:
  - **Distance vector**
  - **Link state**

Distance Vector

Distance Vector (Table Building)

- Each node constructs a vector containing the **"distances"** (costs) to all other nodes
  - Distributes that vector to its immediate neighbors
  - A link that is down is assigned an **infinite cost**
- If the cost of each link is set to 1 $\Rightarrow$ a least-cost path is simply the one with the fewest hops



Initial distances (global view)

Information Stored at Node	Distance to Reach Node						
	A	B	C	D	E	F	G
A	0	1	1	∞	1	1	∞
B	1	0	1	∞	∞	∞	∞
C	1	1	0	1	∞	∞	∞
D	∞	∞	1	0	∞	∞	1
E	1	∞	∞	∞	0	∞	∞
F	1	∞	∞	∞	∞	0	1
G	∞	∞	∞	1	∞	1	0

Distance Vector (Table Building)

- Initially, the global view is **not available** at any single host
  - Directly connected nodes: cost 1; all other nodes: cost ∞
- The next step is that every node **sends** a message to its directly connected neighbors containing its **personal list of distances**
- Each node can **update** its routing table with **costs** and **next hops** for all nodes in the network

Initial routing table at node A

Destination	Cost	NextHop
B	1	B
C	1	C
D	∞	–
E	1	E
F	1	F
G	∞	–

Final routing table at node A

Destination	Cost	NextHop
B	1	B
C	1	C
D	2	C
E	1	E
F	1	F
G	2	F

Distance Vector (Table Building)

- The process of getting consistent routing information to all the nodes is called **“convergence”**
- Each node only knows its routing table
- Enable all nodes to achieve **a consistent view** of the network without any centralized authority

Final routing table (global view)

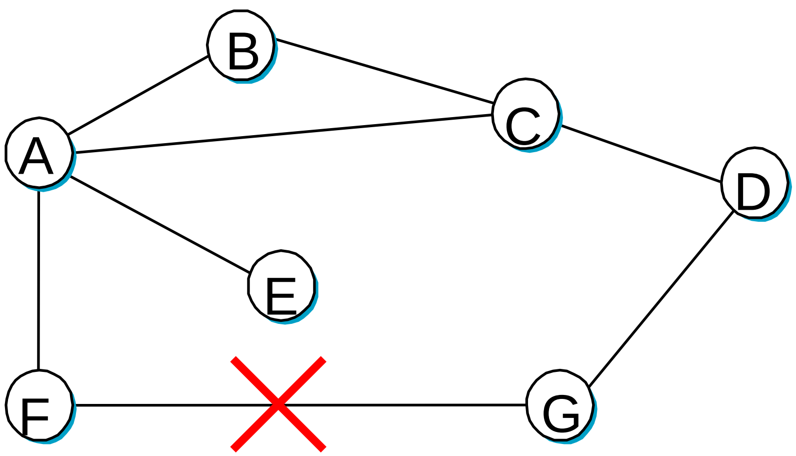
Information Stored at Node	Distance to Reach Node						
	A	B	C	D	E	F	G
A	0	1	1	2	1	1	2
B	1	0	1	2	2	2	3
C	1	1	0	1	2	2	2
D	2	2	1	0	3	2	1
E	1	2	2	3	0	2	3
F	1	2	2	2	2	0	1
G	2	3	2	1	3	1	0

Distance Vector (Table Update)

- There are two circumstances under which a node will send a **routing update** to its neighbors
- **Periodic update:** each node automatically sends an update message, even if nothing has changed (~ sec to ~ min)
  - Let the other nodes know that this node is **still running**
  - Let all nodes keep getting information when their current routes become **un-available**
- **Triggered update:** when a node receives an update from one of its neighbors that causes it to **change one of the routes**
  - Whenever a node's routing table changes, it sends an update to its neighbors
  - This update may lead to a change in the neighbors

Distance Vector (Table Update)

- When a node detects a link failure:
  - the link F–G has failed
- F sets the distance to G to infinity and passes to A
- A sets the distance to G to infinity

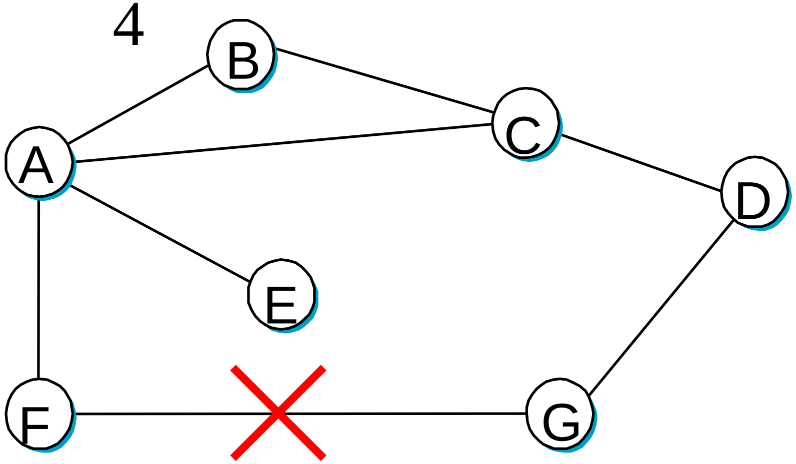


Information Stored at Node	Distance to Reach Node						
	A	B	C	D	E	F	G
A	0	1	1	2	1	1	2
B	1	0	1	2	2	2	3
C	1	1	0	1	2	2	2
D	2	2	1	0	3	2	1
E	1	2	2	3	0	2	3
F	1	2	2	2	2	0	∞
G	2	3	2	1	3	∞	0

Information Stored at Node	Distance to Reach Node						
	A	B	C	D	E	F	G
A	0	1	1	2	1	1	∞
B	1	0	1	2	2	2	3
C	1	1	0	1	2	2	2
D	2	2	1	0	3	2	1
E	1	2	2	3	0	2	3
F	[illegible]	2	2	2	2	[illegible]	∞
G	∞	3	2	1	3	∞	0

Distance Vector (Table Update)

- According to the next update from C
- A learns that C has a 2-hop path to G
- A sets the distance to G to 3 and passes to F
- F sets the distance to G to 4



Information Stored at Node	Distance to Reach Node						
	A	B	C	D	E	F	G
A	0	1	1	2	1	1	**3**
B	1	0	1	2	2	2	3
C	1	1	0	1	2	2	2
D	2	2	1	0	3	2	1
E	1	2	2	3	0	2	3
F	1	2	2	2	2	0	∞
G	**3**	3	2	1	3	∞	0

Information Stored at Node	Distance to Reach Node						
	A	B	C	D	E	F	G
A	0	1	1	2	1	1	**3**
B	1	0	1	2	2	2	3
C	1	1	0	1	2	2	2
D	2	2	1	0	3	2	1
E	1	2	2	3	0	2	3
F	[illegible]	2	2	2	2	0	**4**
G	**3**	3	2	1	3	**4**	0

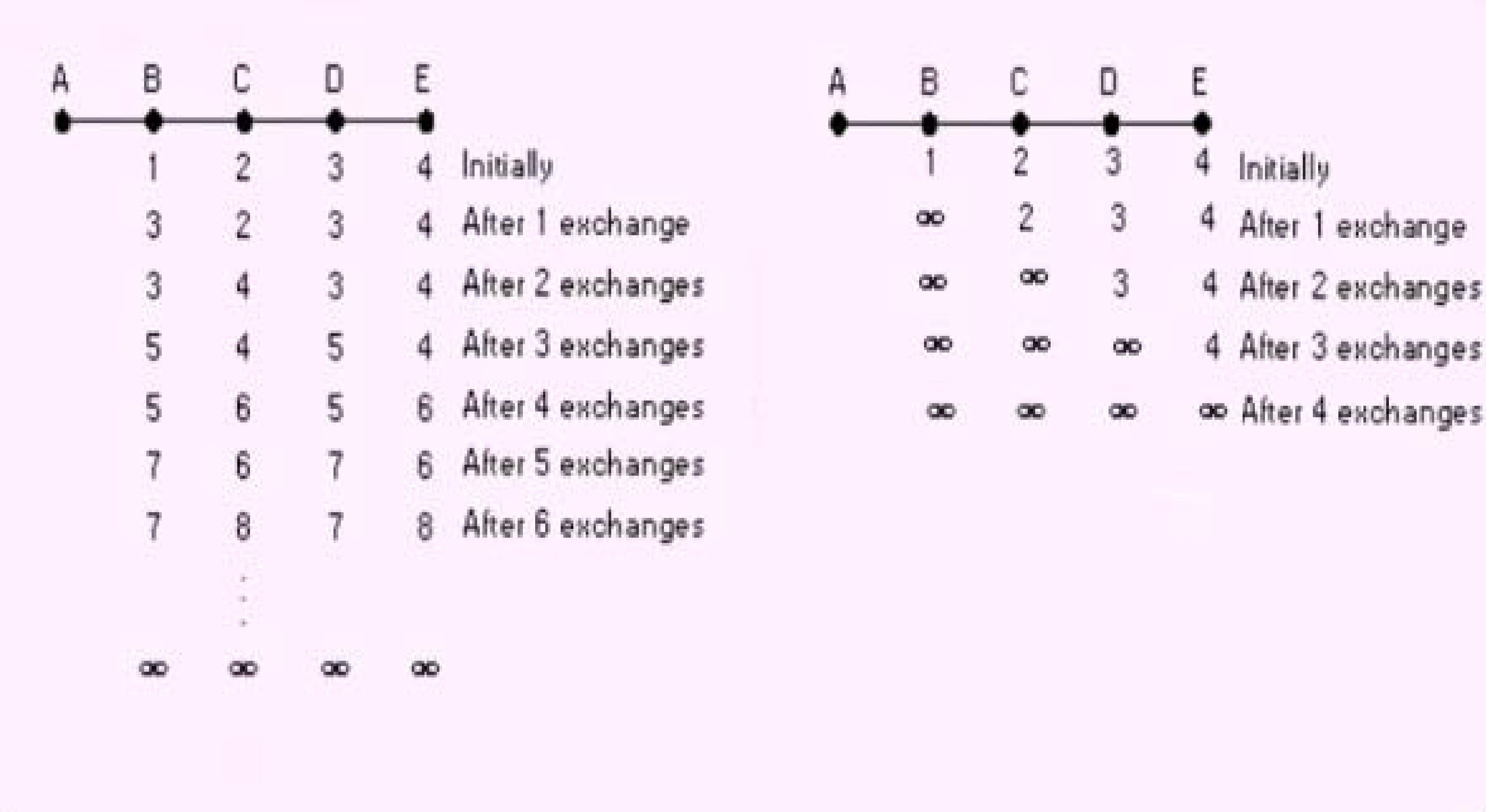
Routing Loops

- Example 1
  - F detects that link to G has failed
  - F sets distance to G to infinity and sends update t o A
  - A sets distance to G to infinity since it uses F to reach G
  - A receives periodic update from C with 2-hop path to G
  - A sets distance to G to 3 and sends update to F
  - F decides it can reach G in 4 hops via A
- Example 2
  - link from A to E fails
  - A advertises distance of infinity to E
  - B and C advertise a distance of 2 to E
  - B decides it can reach E in 3 hops; advertises this to A
  - A decides it can reach E in 4 hops; advertises this to C
  - C decides that it can reach E in 5 hops…

Loop-Breaking Heuristics

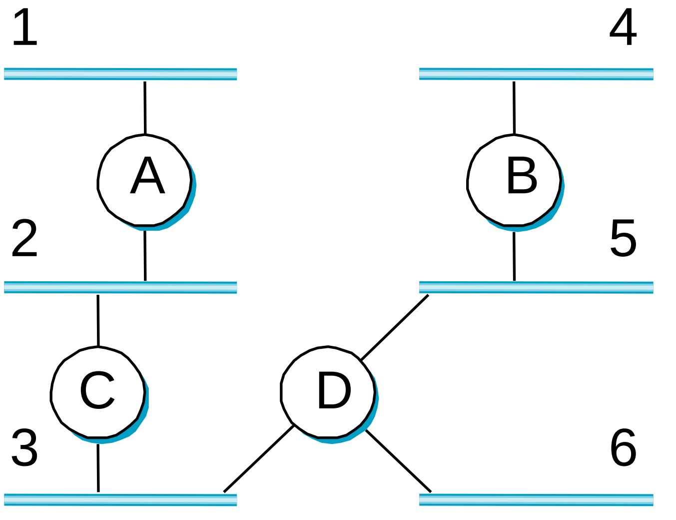
- Set infinity to 16
- Split horizon
- Split horizon with poison reverse

The Count-to-Infinity Problem (cont.)



Distance Vector (RIP)

- **RIP: Routing Information Protocol**
- For routers to learn how to forward packets to various network
  - Routers advertise the cost of reaching networks
- For example, router C:
  - It can reach networks 2 and 3 at a cost 0
  - It can reach networks 5 and 6 at a cost 1
  - It can reach networks 4 at a cost 2



Distance Vector (RIP)

- **RIP packet format:** containing **<network-address, distance>** pairs
- Routers send their **advertisements** every **30 seconds**

0 8 16 31

Command	Version	Must be zero
Family of net 1		Address of net 1
Address of net 1		
Distance to net 1		
Family of net 2		Address of net 2
Address of net 2		
Distance to net 2		

Link State

Link State

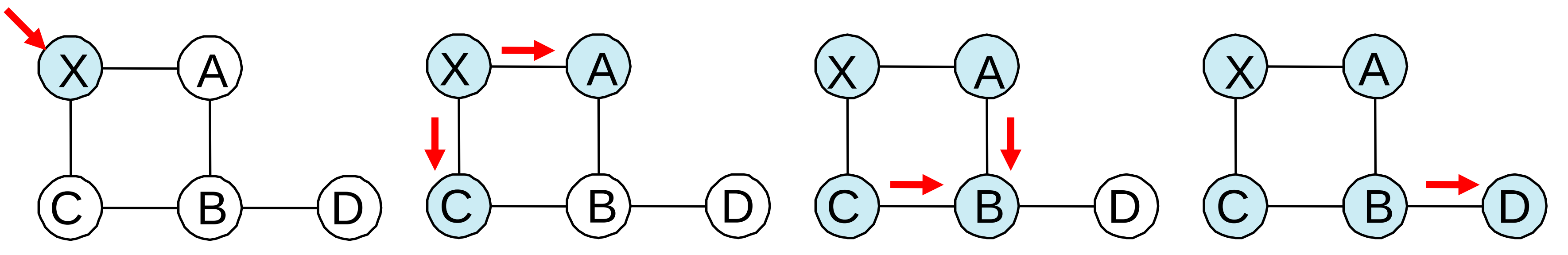
- Each node is assumed to be capable of finding out the **state of the link** to its neighbors and the cost of each link
  - To find the **least-cost path** to any destination
- **Link state routing protocols** rely on two mechanisms:
  - Reliable **dissemination** of link-state information
  - The **calculation** of routes from the sum of all the accumulated link-state knowledge

Link State (Reliable Flooding)

- **Reliable Flooding:** make sure that all the nodes get **a copy** of the link-state information from all the other nodes
  - A node sends its **link-state information** out on all of its directly connected links
- A **link-state packet (LSP):** an update packet containing
  - The ID of the node
  - A list of directly connected neighbors of that node, with the cost of the link to each node
  - A sequence number
  - A time to live for this packet
- The first two items are needed to enable **route calculation**
- The last two are used to make sure the process being **reliable**

Link State (Reliable Flooding)

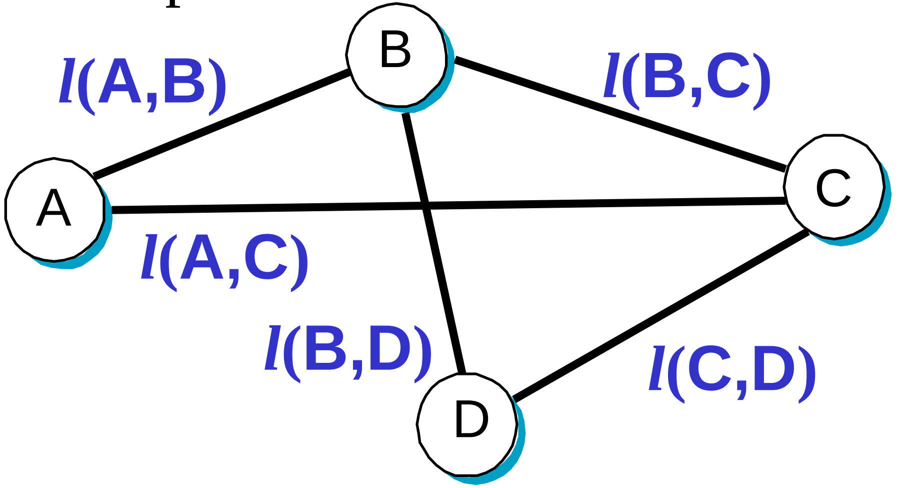
- The transmission of LSPs between adjacent routers is made reliable using **acknowledgments and retransmissions**
- Consider a node X that receives a copy of an LSP from node Y
  - If X does not have stored a copy of an LSP from Y, it **stores the LSP**
  - If X already has a copy, it **compares** the sequence numbers
    - If the new LSP has a **larger** sequence number **(more recent)**, that LSP is stored to replace the old one



Link State (Rout Calculation)

- The calculation is based on a well-known algorithm from graph theory – **Dijkstra's shortest-path algorithm**
- ***N*** denotes the set of nodes in the graph
- ***l*(*i*, *j*)** denotes the **nonnegative cost** associated with the edge between nodes $i, j \in N$
- ***M*** denotes the set of nodes incorporated so far
- ***C*(*n*)** denotes the cost of the path from ***s*** to each node ***n***

N = **{A, B, C, D}**



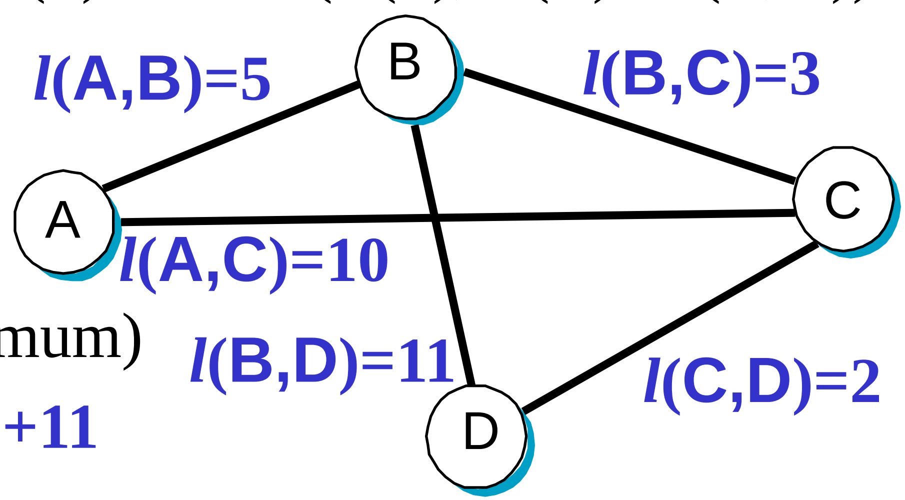
Link State (Rout Calculation)

- **Dijkstra's shortest-path algorithm**
  - $M = \{s\}$
  - For each n in $N - \{s\}$ $\quad C(n) = l(s, n)$
  - While ($N \neq M$) $\quad M = M \cup \{w\}$ such that $C(w)$ is the minimum for all w in $(N - M)$
  - For each n in $(N - M)$ $\quad C(n) = \text{MIN}\,(C(n), C(w) + l(w, n))$

$N = \{A, B, C, D\}$

$M = \{A\}$ $\quad n = B, C, D$

$C(B) = 5$, $C(C) = 10$, $C(D) = \infty$



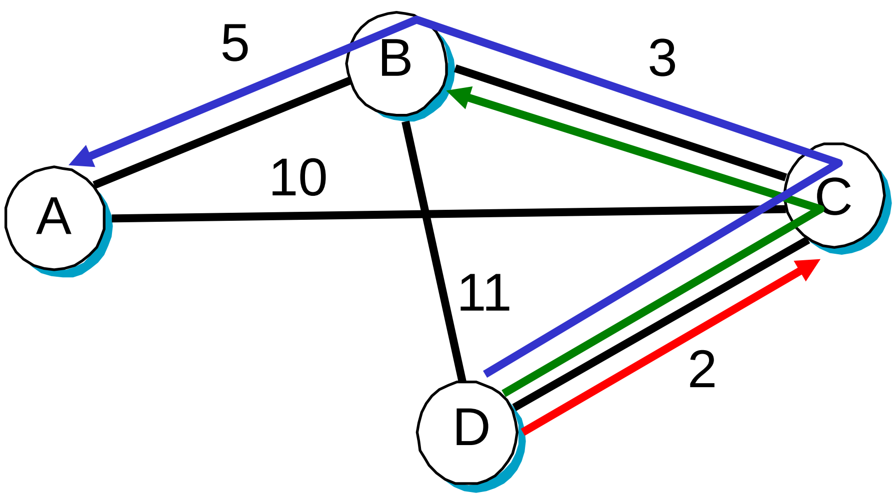
$M = \{A\} \cup \{B\}$ ($C(B)$ is minimum)

$C(C) = 10$ or $5+3$, $C(D) = \infty$ or $5+11$

$C(C) = 8$, $C(D) = 16$

Link State (Rout Calculation)

- Example: the entries **(Destination, Cost, NextHop)**
- For node D



Step	Confirmed	Tentative
1	(D, 0, -)	
2	(D, 0, -)	(B, 11, B) **(C, 2, C)**
3	(D, 0, -) (C, 2, C)	(B, 11, B)
4	(D, 0, -) (C, 2, C)	**(B, 5, C)** (A, 12, C)
5	(D, 0, -) (C, 2, C) (B, 5, C)	(A, 12, C)
6	(D, 0, -) (C, 2, C) (B, 5, C)	**(A, 10, C)**
7	(D, 0, -) (C, 2, C) (B, 5, C) (A, 10, C)	

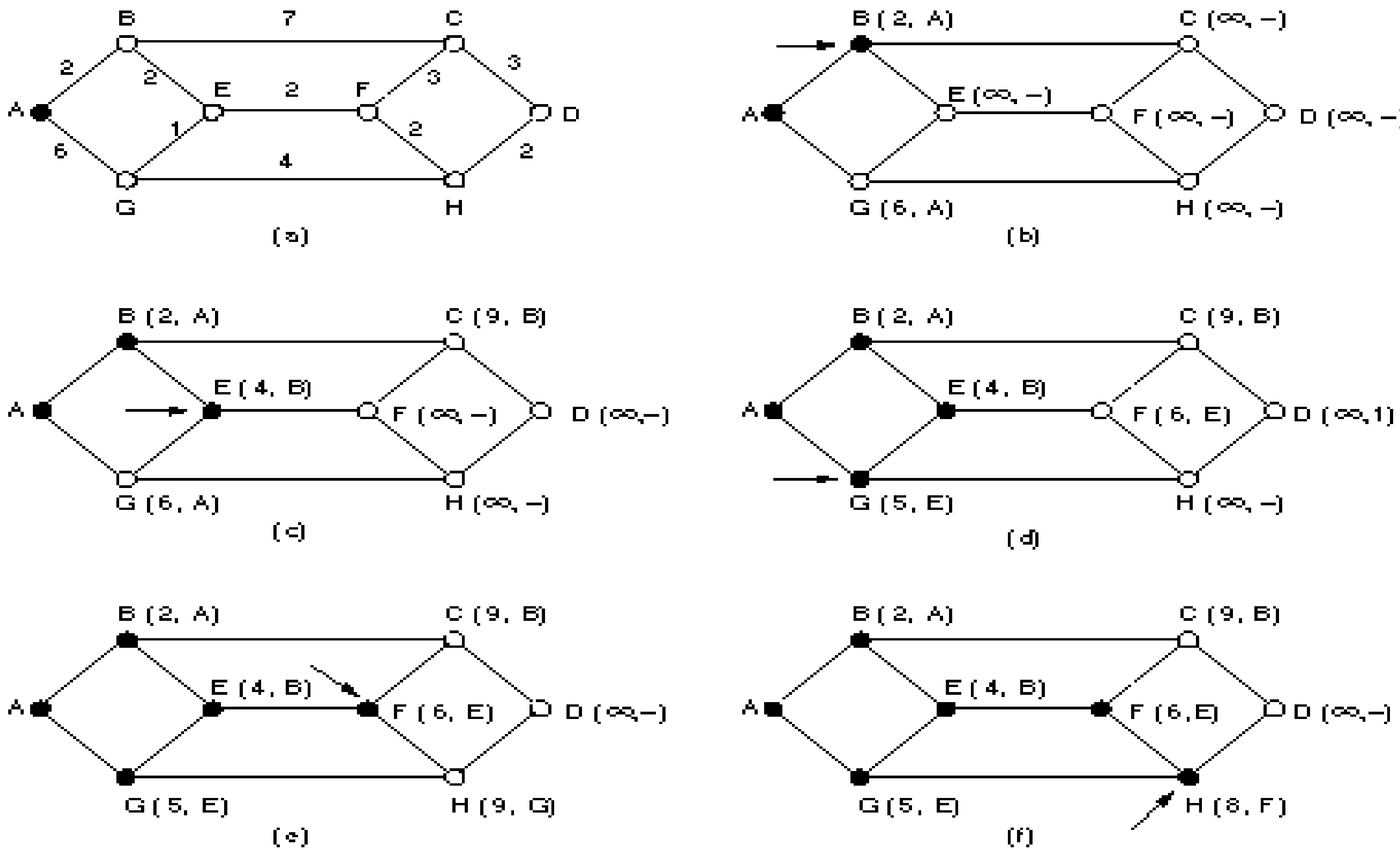


Fig. 5-6. The first five steps used in computing the shortest path from A to D. The arrows indicate the working node.

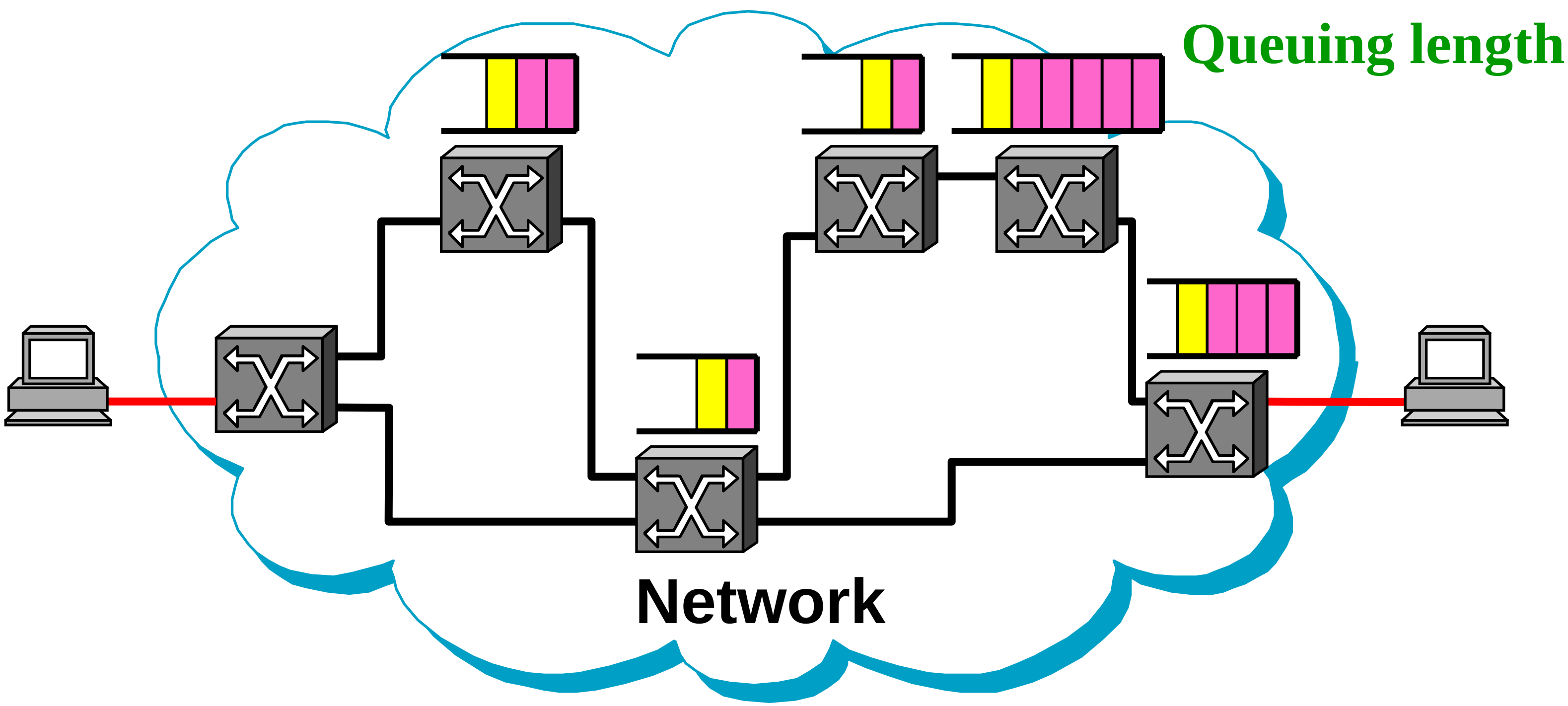
Link Metrics

Metrics

- The link **costs (metrics)** may simply assumed to be **one**
  - The least-cost route $\Rightarrow$ the one with the **fewest hops**
- Such an approach has several **drawbacks**:
  - It does not distinguish between links on a **latency** basis
    - 250-ms latency (satellite) VS 1-ms latency (terrestrial)
  - It does not distinguish between links on a **capacity** basis
    - 9.6-kbps link VS 45-Mbps link
  - It does not distinguish between links based on their **current load**
    - Over-loaded link VS light-loaded link
- **A hard problem: try to capture the complex and dynamic characteristics of a link in a single scalar cost**

Metrics

- The original ARPANET routing metric measured the **number of packets** that were queued waiting to be transmitted on a link
  - **Did not work well**, since it forwards packets toward the **shortest queue** rather than toward the destination



Metrics

- The second version ARPANET: took both **link bandwidth** and **latency** (**delay** rather than queue length) into consideration
  - This approach still had a lot of problems
  - Under **light load**: it worked reasonably well
  - Under **heavy load**:
    - A congested link would advertise a very high cost

      $\Rightarrow$ All traffic moves off this link
    - A idle link would advertise a low cost

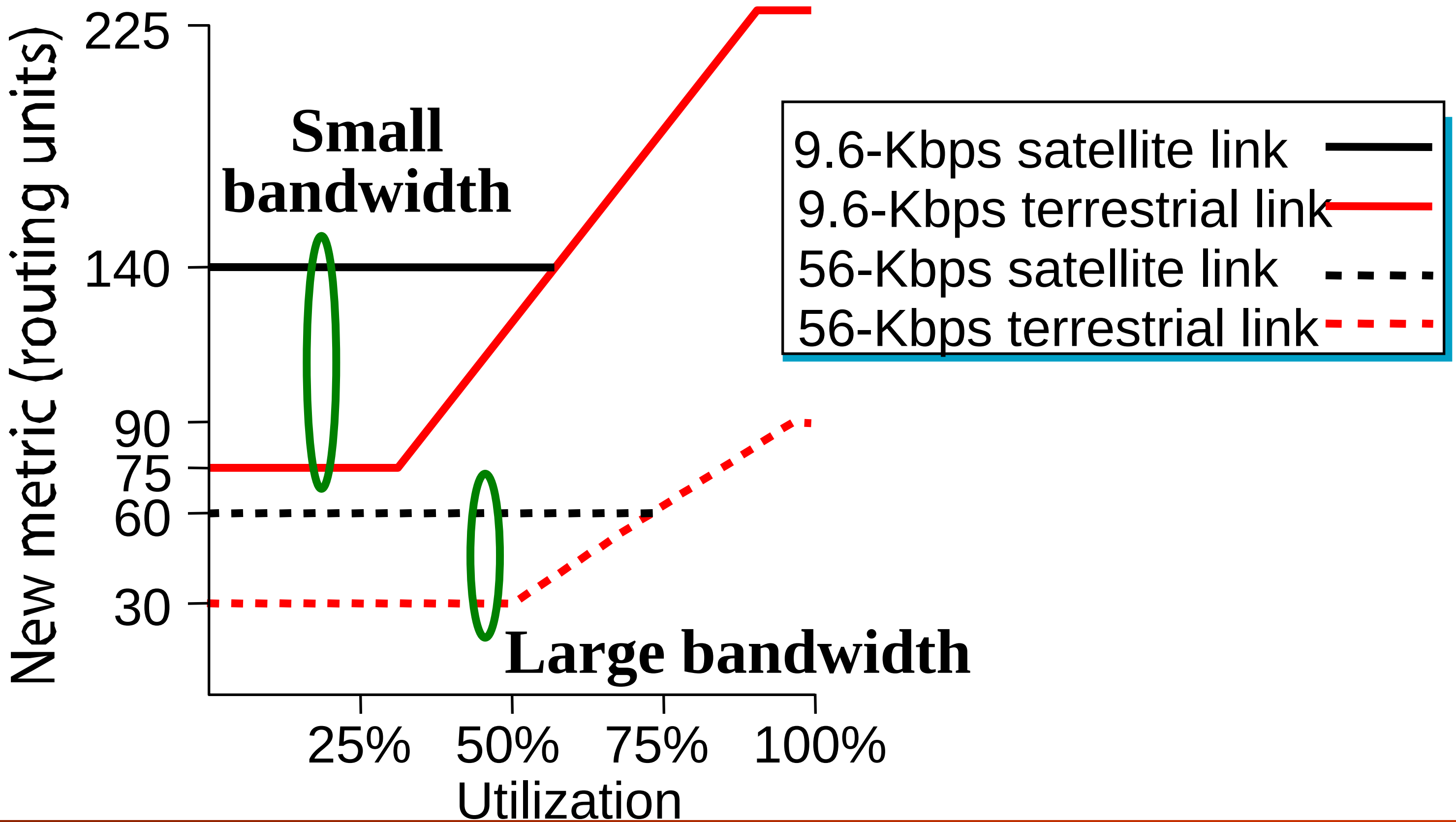
      $\Rightarrow$ Attracting all traffic
  - The range of the link values was **too large**: induce wrong decision

Metrics

- The third approach "revised ARPANET": addressed those above-mentioned problems
  - **Compress** the dynamic range of the metric
  - Account for the **link type**
  - **Smooth the variation** of the metric with time
- The **smoothing** was achieved by several mechanisms:
  - The delay measurement was transformed to a **link utilization**
    - **Averaged with the last reported utilization** to suppress sudden changes
  - There was a **hard limit** on how much the metric could change **in one cycle**

Metrics

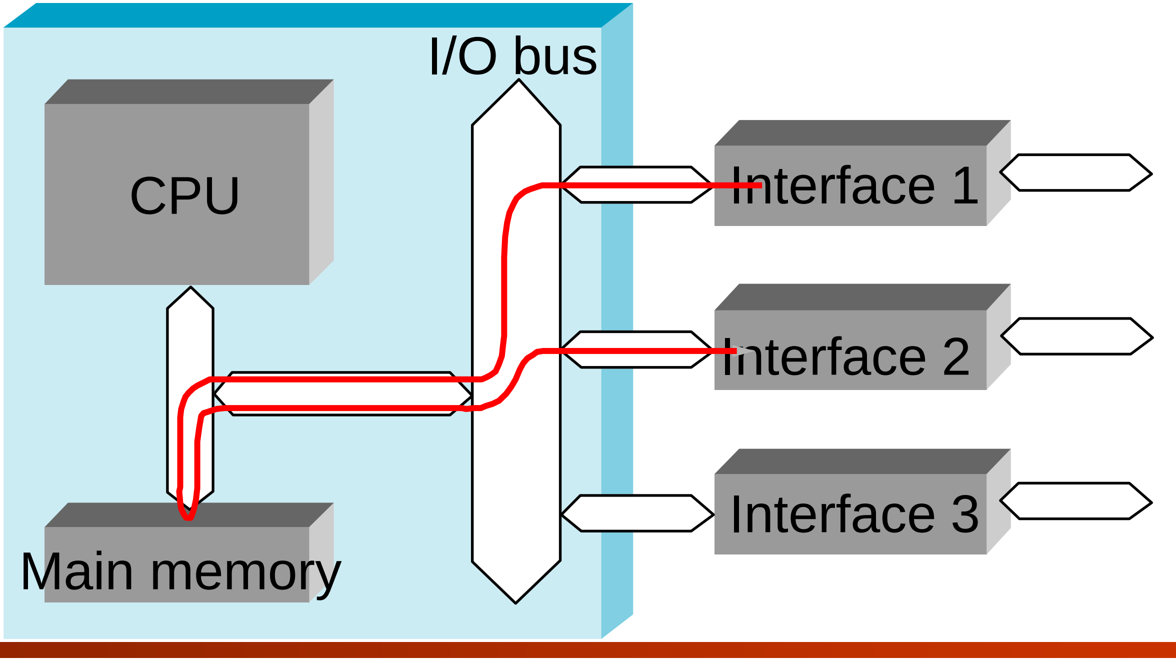
- The compression of the dynamic range was achieved by:
  - Feed the **measured utilization**, the **link type**, and the **link speed** into a function



Implementation and Performance

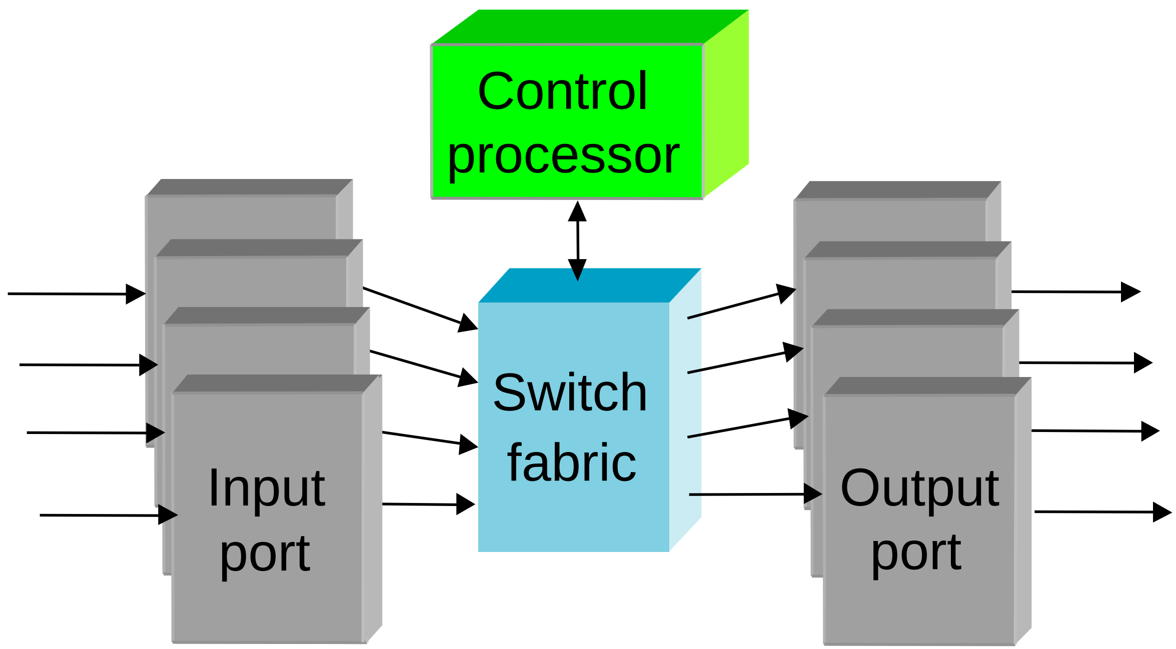
Switch

- A switch: a workstation with three network interfaces
- Each packet across the I/O bus **twice** (written and read)
- The **upper bound** on aggregate throughput is
  - Either **half the main memory bandwidth** or **half the I/O bus bandwidth**



Switch

- A switch consists of:
  - A number of **input ports** and **output ports**
  - A **switch fabric**
  - At least one **control processor**



Switch

- The **ports:** deal with the communication with the **outside world**, e.g. fiber optic receiver, lasers, buffers, …
- When a packet is handed from an input port to the fabric, the port has figured out **where** the packet needs to go
  - Using the **virtual circuit mapping tables** or
  - Using the **Ethernet addresses and output ports mapping tables**
- The **fabric** has a very simple and well-defined job: **delivers a packet to the right output port**
- **Self-routing fabric:** fabrics that switch packets by looking only at the information **in the packet**
  - No external control is required

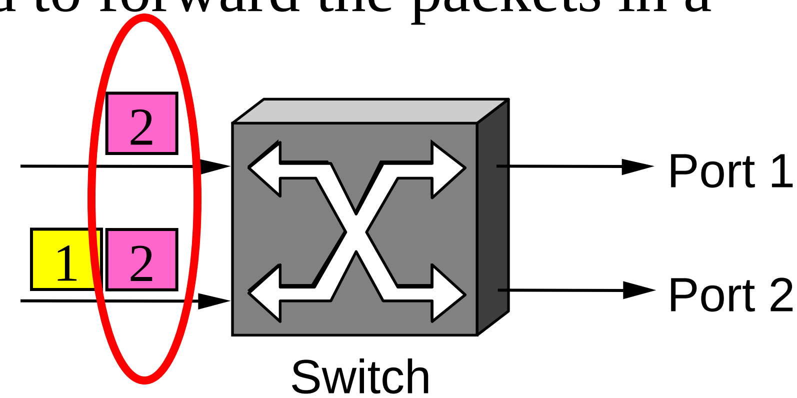
Ports

- Performance bottlenecks: **Packet header analysis** & **Buffering**
- **Packet header analysis:** when the average packet size is very small: for example, an OC-48 link with 64-byte packets

  $\Rightarrow 2.48 \times 10^9 \div (64\times 8) = 4.83 \times 10^6$ pps (packets per second)

  $\Rightarrow$ The input port has only **200 ns** to process for each packet
- **Buffering:** as packets arrive at the switch, they are placed in the input buffer. The switch then tried to forward the packets in a **FIFO** manner
  - If several different input ports are destined for the same output port at the same time

    $\Rightarrow$ **only one** can be forwarded



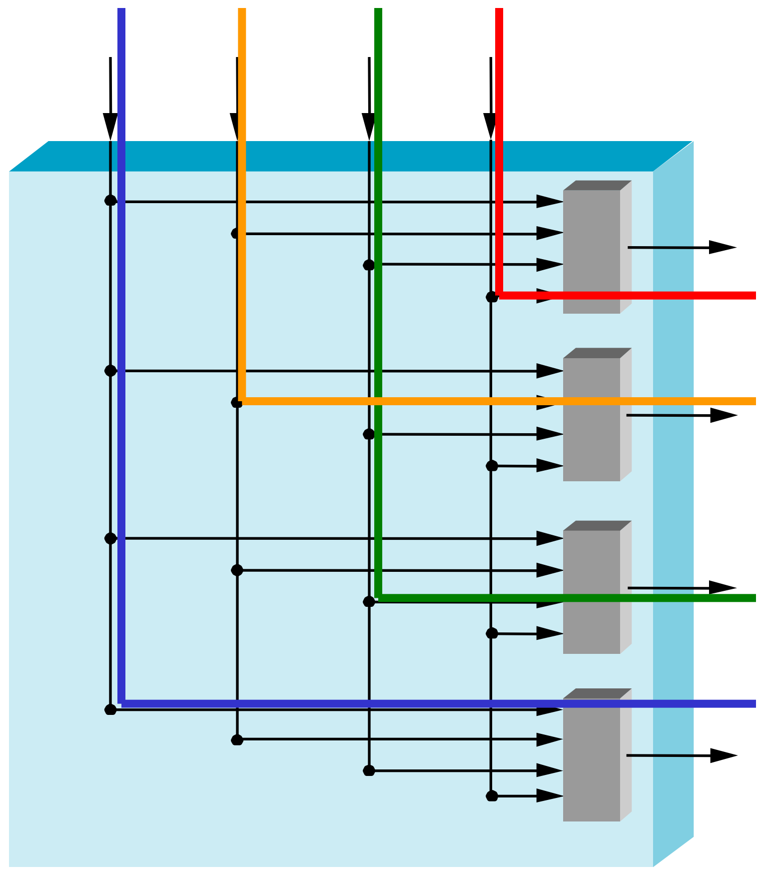
Head-of-line blocking

Fabrics

- A high-performance fabric can move one packet from **each of its *n* input ports** to one of the output ports **at the same time**
- A sample of fabric types includes:
  - **Shared-bus:** like a conventional workstation used as a switch
    - The **bus bandwidth** determines the switch throughput
  - **Shared-memory:** packets are written into a memory location by an input port and then read from memory by the output ports
    - The **memory bandwidth** determines the switch throughput

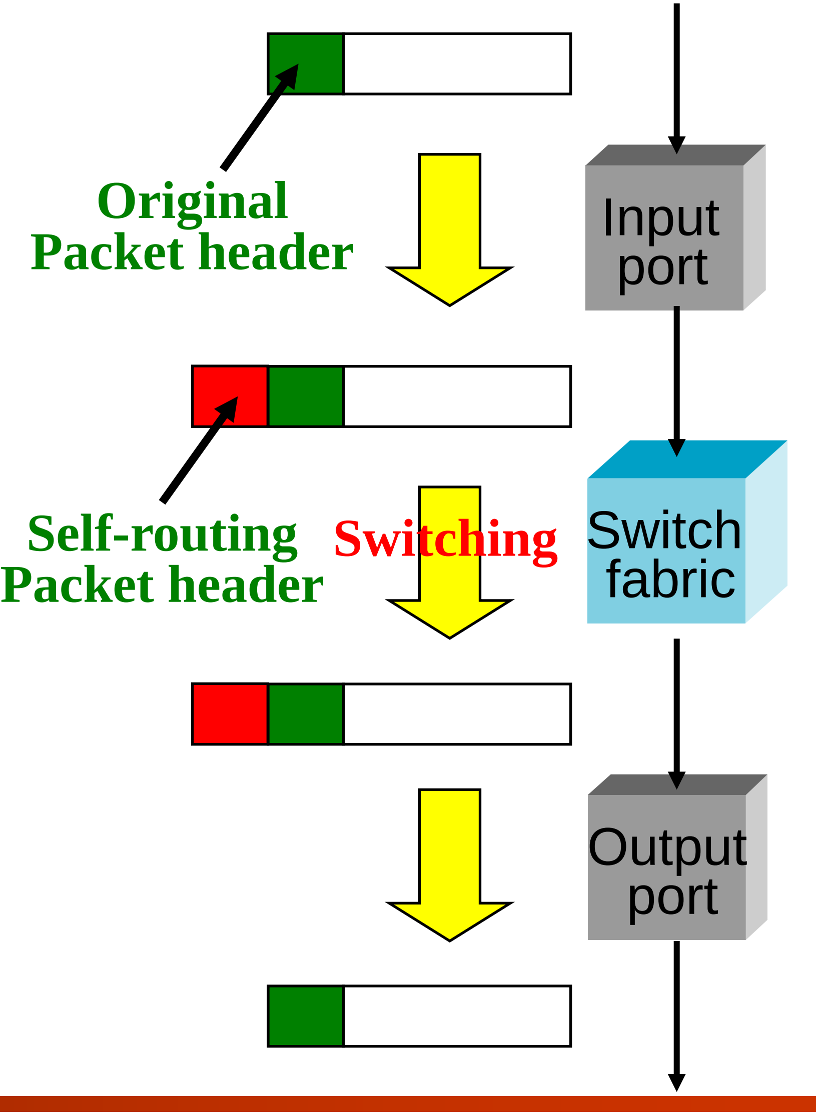
Fabrics

– **Crossbar:** a crossbar switch is **a matrix of pathways** that can be configured to connect any input port to any output port



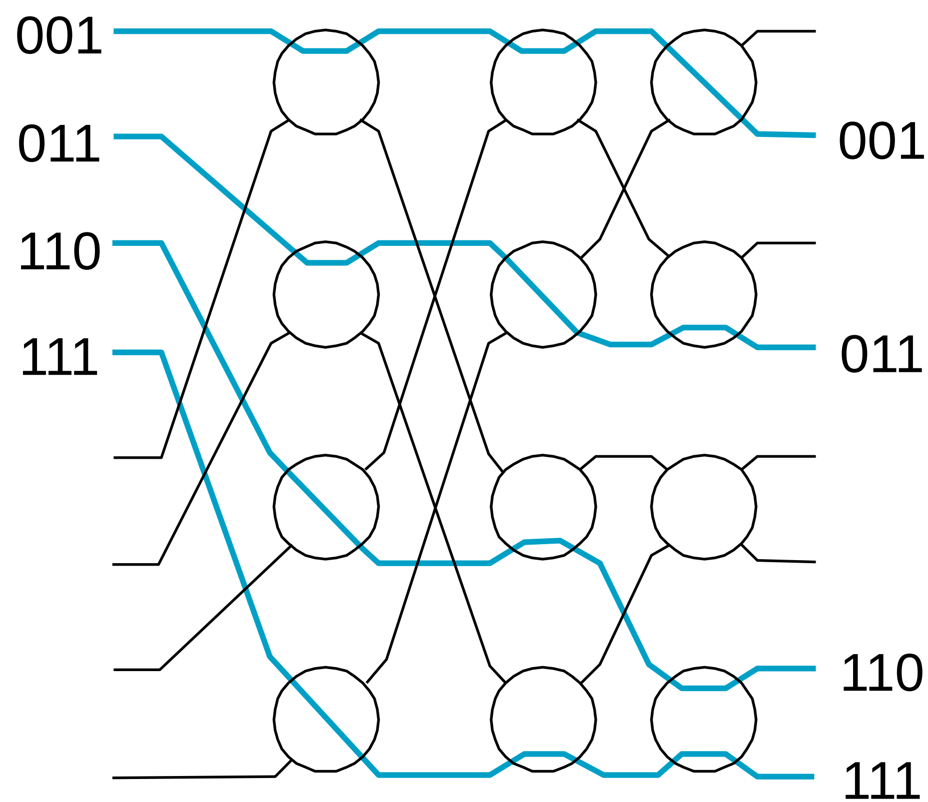
Fabrics

- **Self-routing:** a special **"self-routing header"** is appended to the packet by the **input port**
- This extra header is **removed** before the packet leaves the switch
- Often built from large numbers of simple **2×2** switching elements interconnected in regular patterns, such as the **banyan switching fabric**



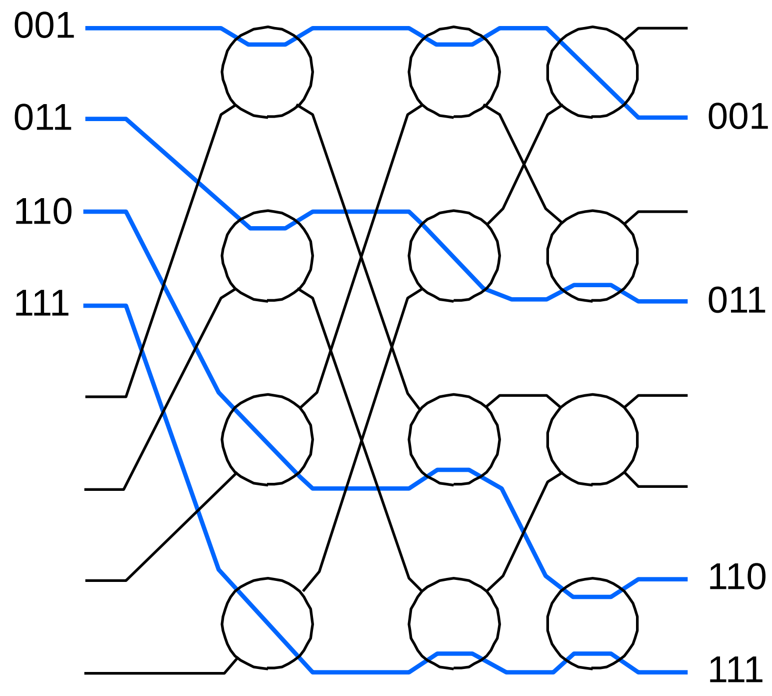
Self-Routing Fabrics

- Each element looks at **1 bit** in each self-routing header
  - If it is **0**: route packets toward the **upper output**
  - If it is **1**: route packets toward the **lower output**
- If two packets arrive at a banyan element at the same time and both have the bit set to the same value
  - A **collision** will occur (internal blocking)



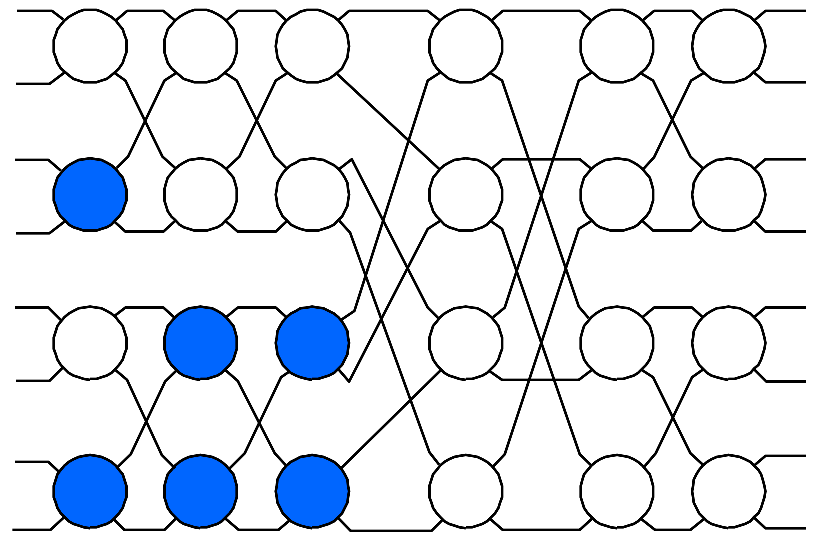
Self-Routing Fabrics

- 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)

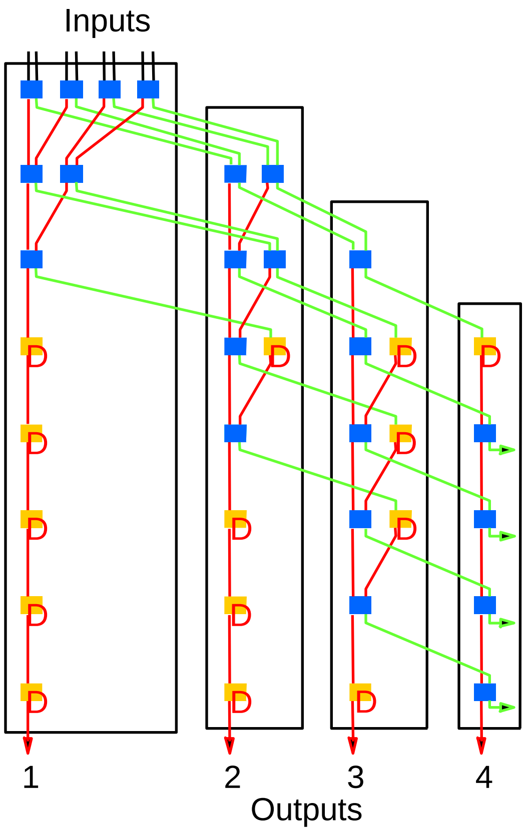
- Batcher Network
  - switching elements sort two numbers
    - some elements sort into ascending (clear)
    - some elements sort into descending (shaded)
  - elements arranged to implement merge sort
  - complexity: $n \log^2_2 n$



- Common Design: Batcher-Banyan Switch

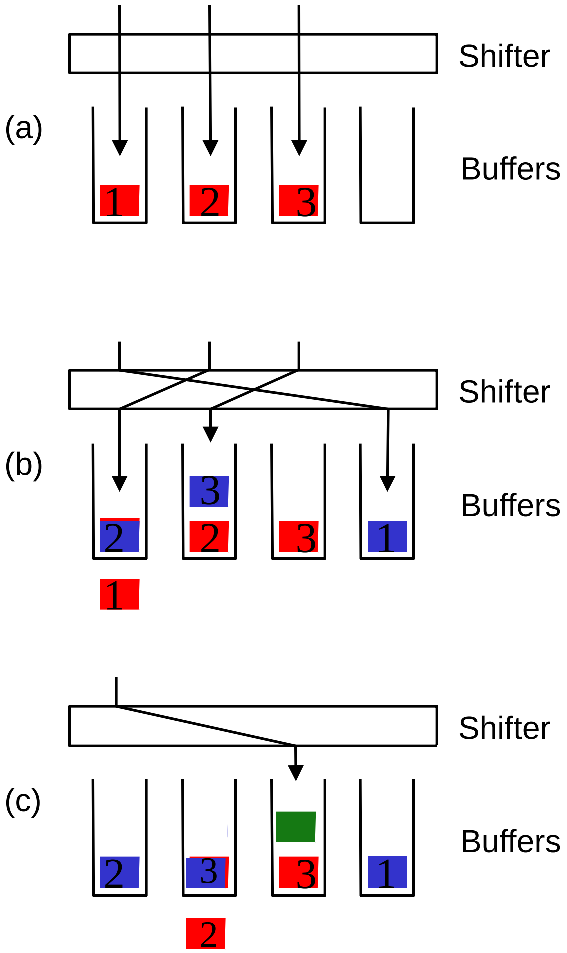
Knockout Switch

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



Knockout Switch (cont)

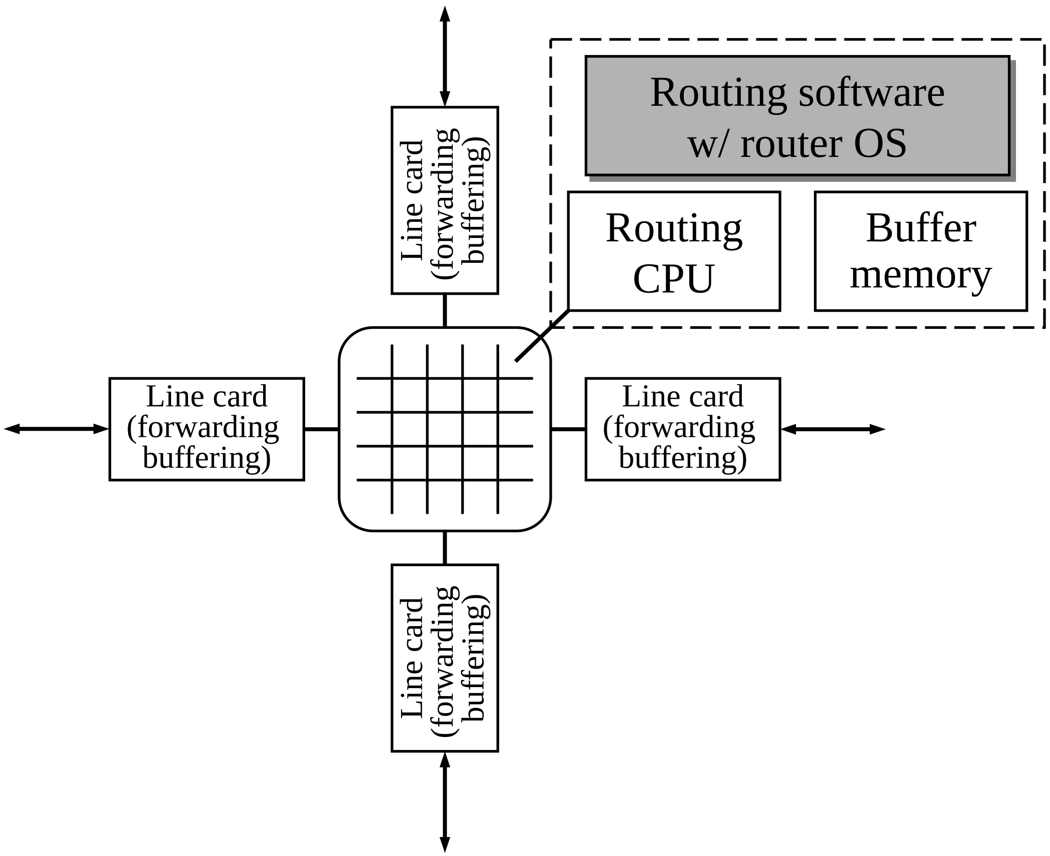
- Output Buffer



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