Unit 2: Networks (IP)

Sources: L. Cerdà, J. Rexford, ISOC, wikipedia, etc.

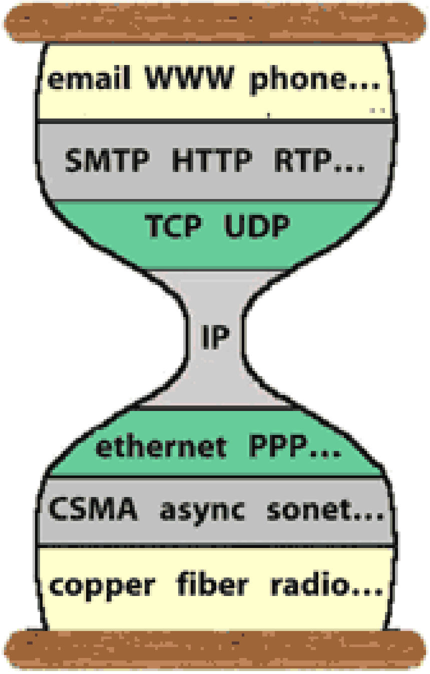
IP Networks

- IP layer service
- IP addresses, subnetting
- Routers: forwarding tables
- Link: address resolution (ARP)
- Data: IP header, fragmentation
- Control (ICMP)
- Host Config (DHCP)
- Routing (RIP, OSPF)
- Private nets: Address Translation (NAT)
- Middleboxes: Security firewalls, virtual private networks (VPN), tunnels

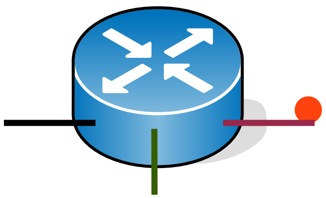
The Internet: an inter-network

- Internet: the most famous interconnection
- Networks can differ, data (Internet Protocol) in common
- "IP over everything, everything over IP"

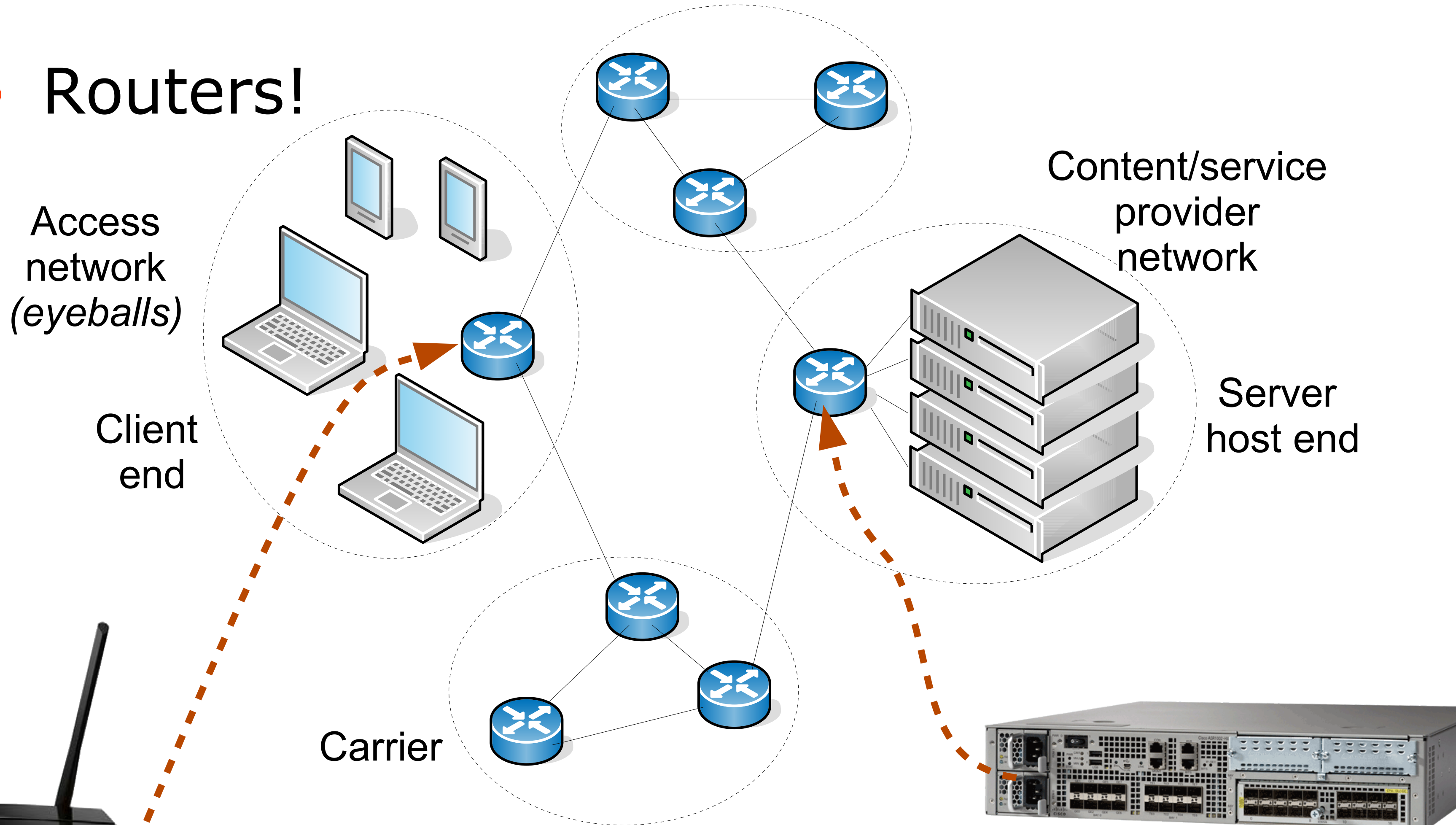
- Routers on multiple networks that pass traffic among them
- Individual networks pass traffic router-router or router-endpoint
- TCP/IP hides details

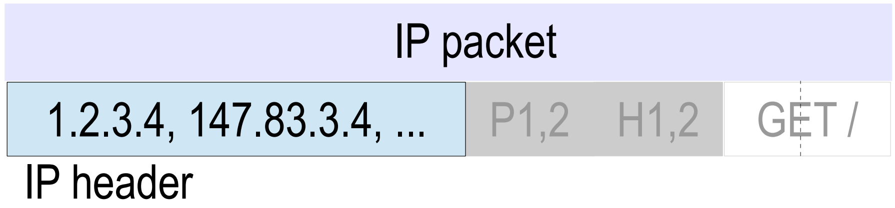


The Internet: an inter-network



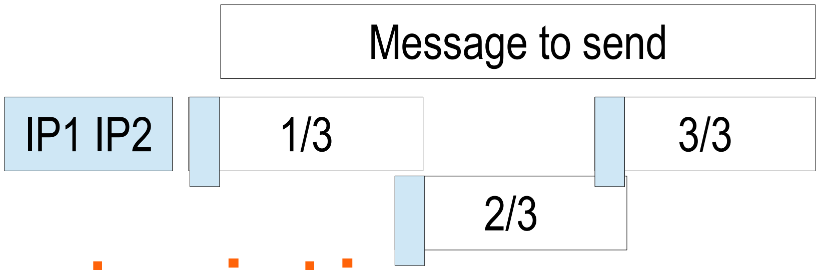
- Routers!





Internet Protocol service

- Task: delivering packets from source host to destination host solely based on the IP addresses in the packet headers.
- Packets encapsulate data, routed
- Connection-less datagram service (Vint Cerf, Bob Kahn, 1974)
- Two versions: IPv4, IPv6

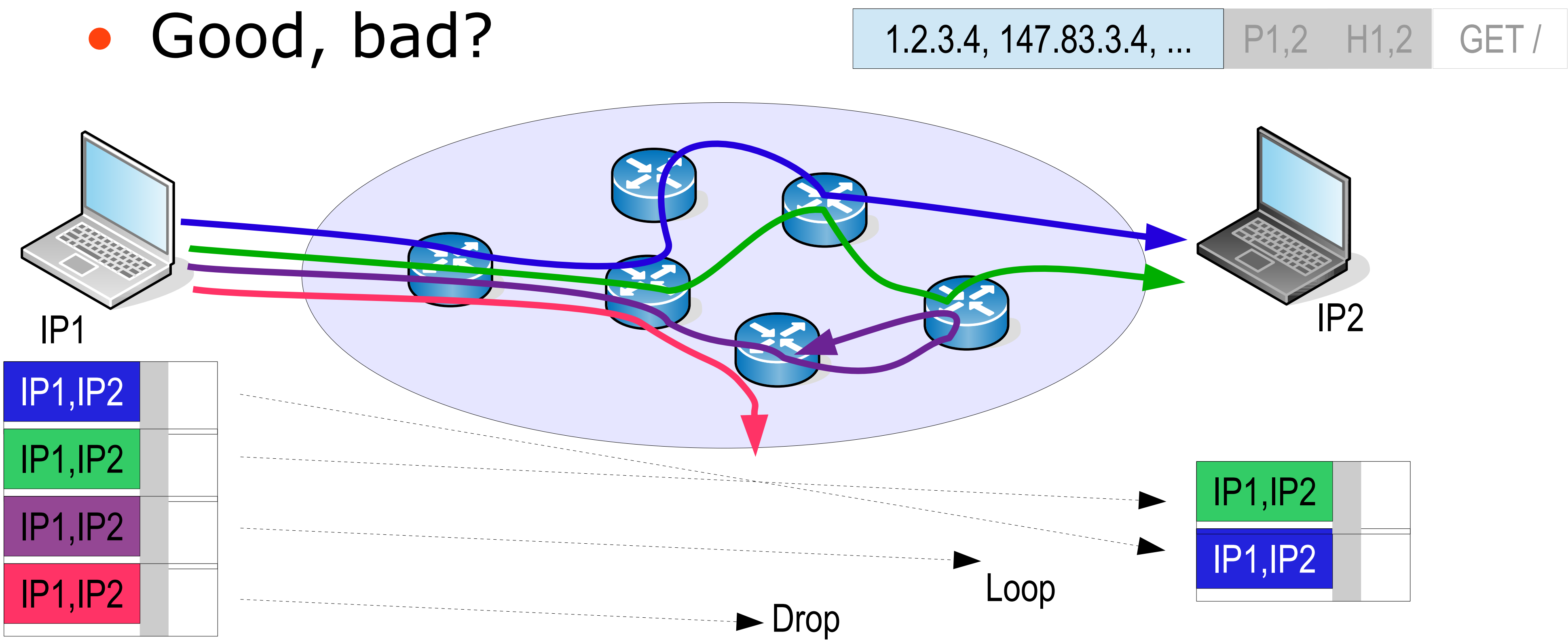


Internet Protocol characteristics

- Characteristics: (dummy core)
  - No connection: Connectionless
  - No memory: Stateless (independent of each other)
  - No guarantee: best effort
- Consequences:
  - Packets can be delivered out-of-order
  - Each packet can take a different path to the destination
  - No error detection or correction in payload
  - No congestion control (beyond “drop”)
- TCP: connection, error correction by retransmission

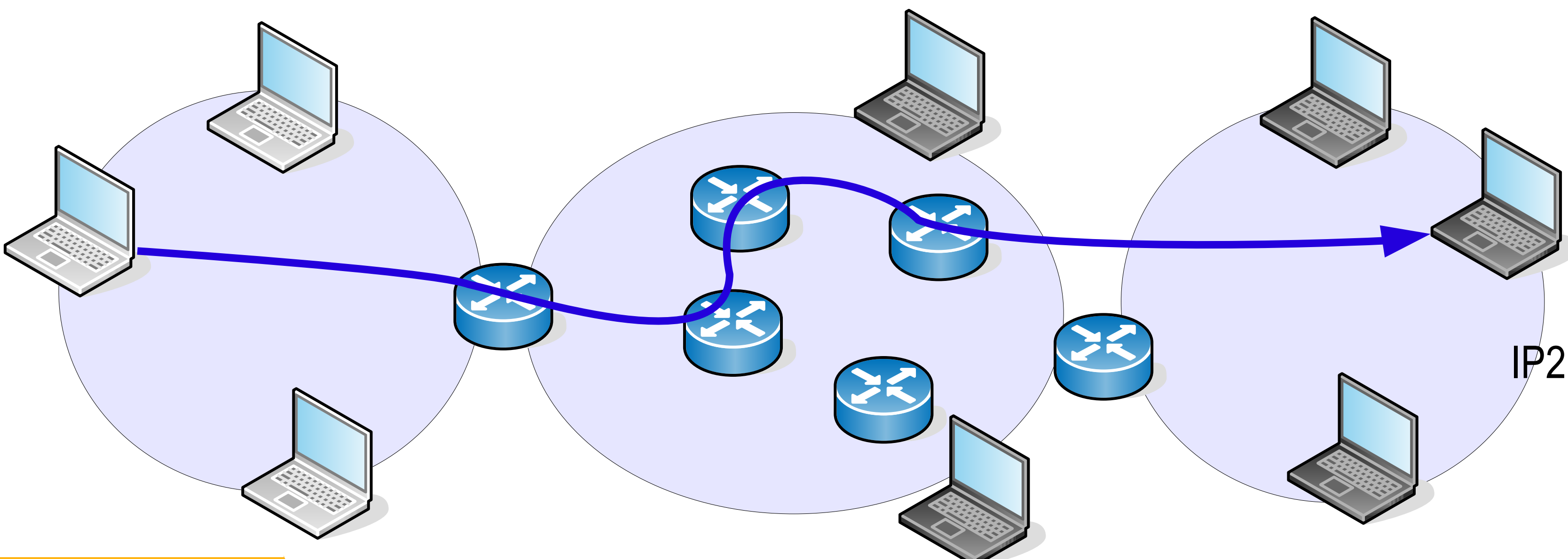
IP traffic: properties

- Each node (host, router) has a unique IP addr
- Each packet is independent, best effort
- Good, bad?



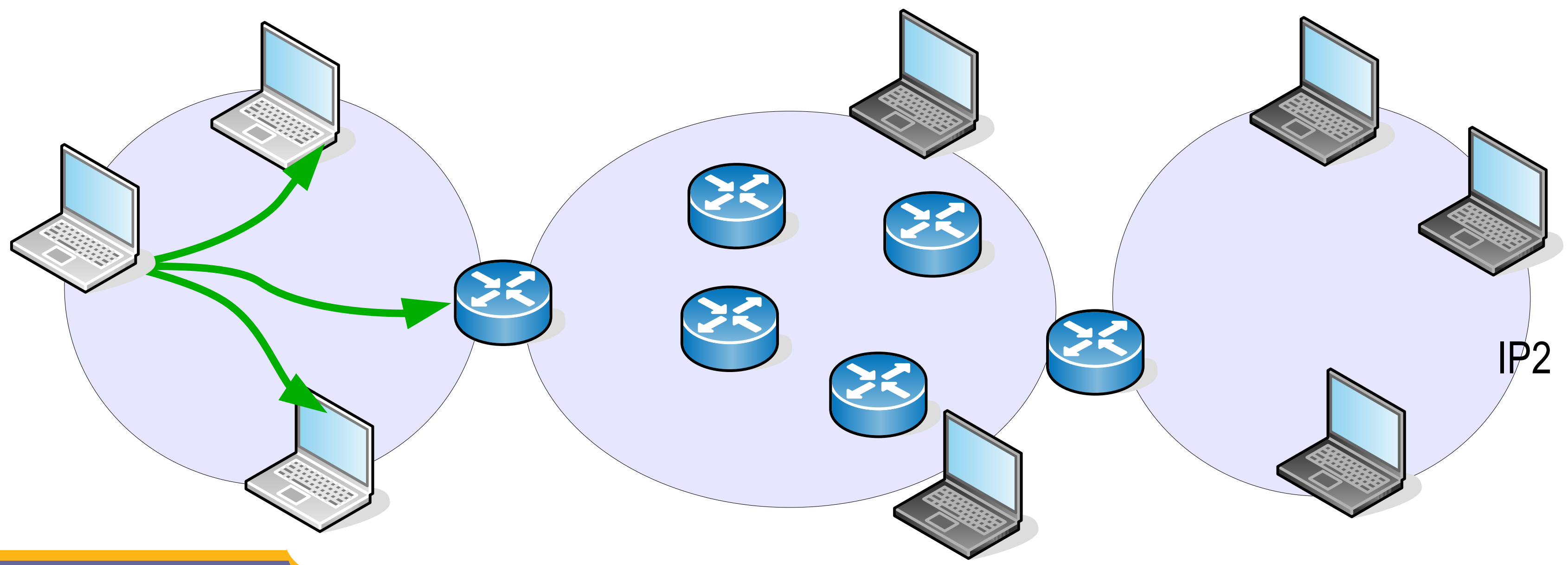
IP traffic: destinations

- Unicast: one-to-one (IP addr)
- Broadcast: one-to-all (in sender's network)
- Multicast: to many, a group (anywhere)
- Anycast: one to one of many



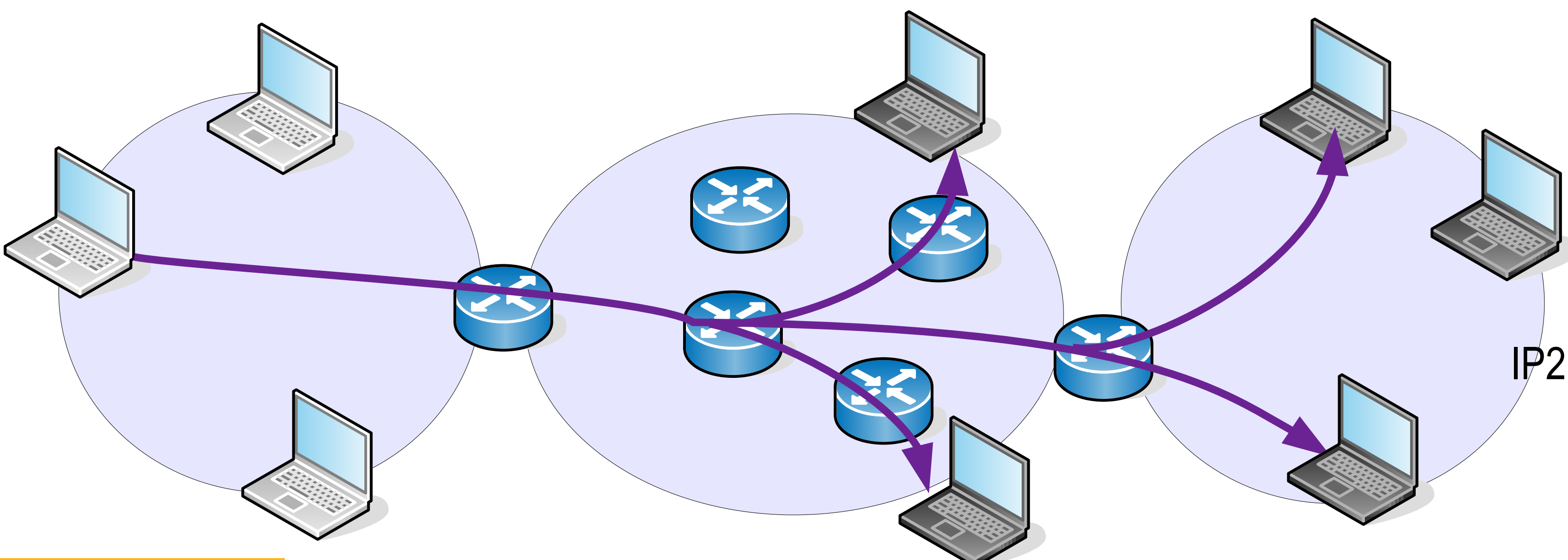
IP traffic: destinations

- Unicast: one-to-one (IP addr)
- Broadcast: one-to-all (in sender's network)
- Multicast: to many, a group (anywhere)
- Anycast: one to one of many

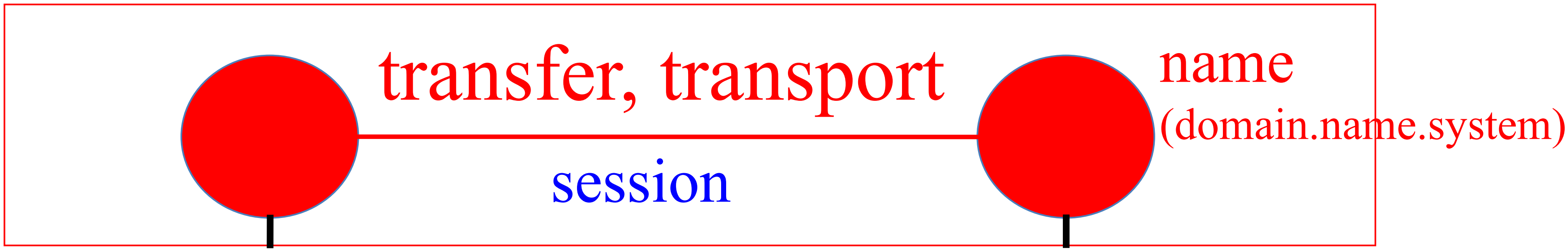


IP traffic: destinations

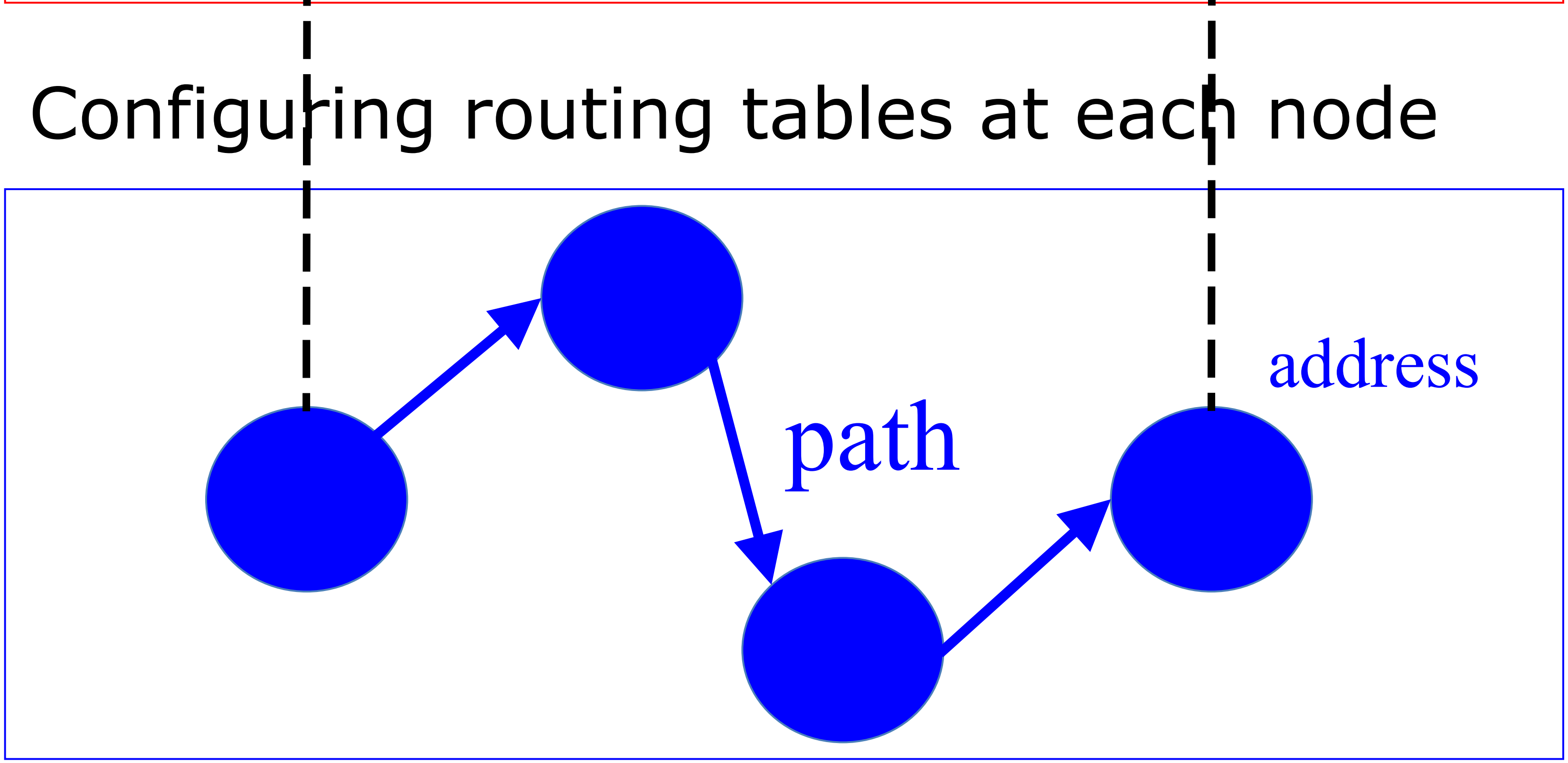
- Unicast: one-to-one (IP addr)
- Broadcast: one-to-all (in sender's network)
- Multicast: to many, a group (anywhere)
- Anycast: one to one of many



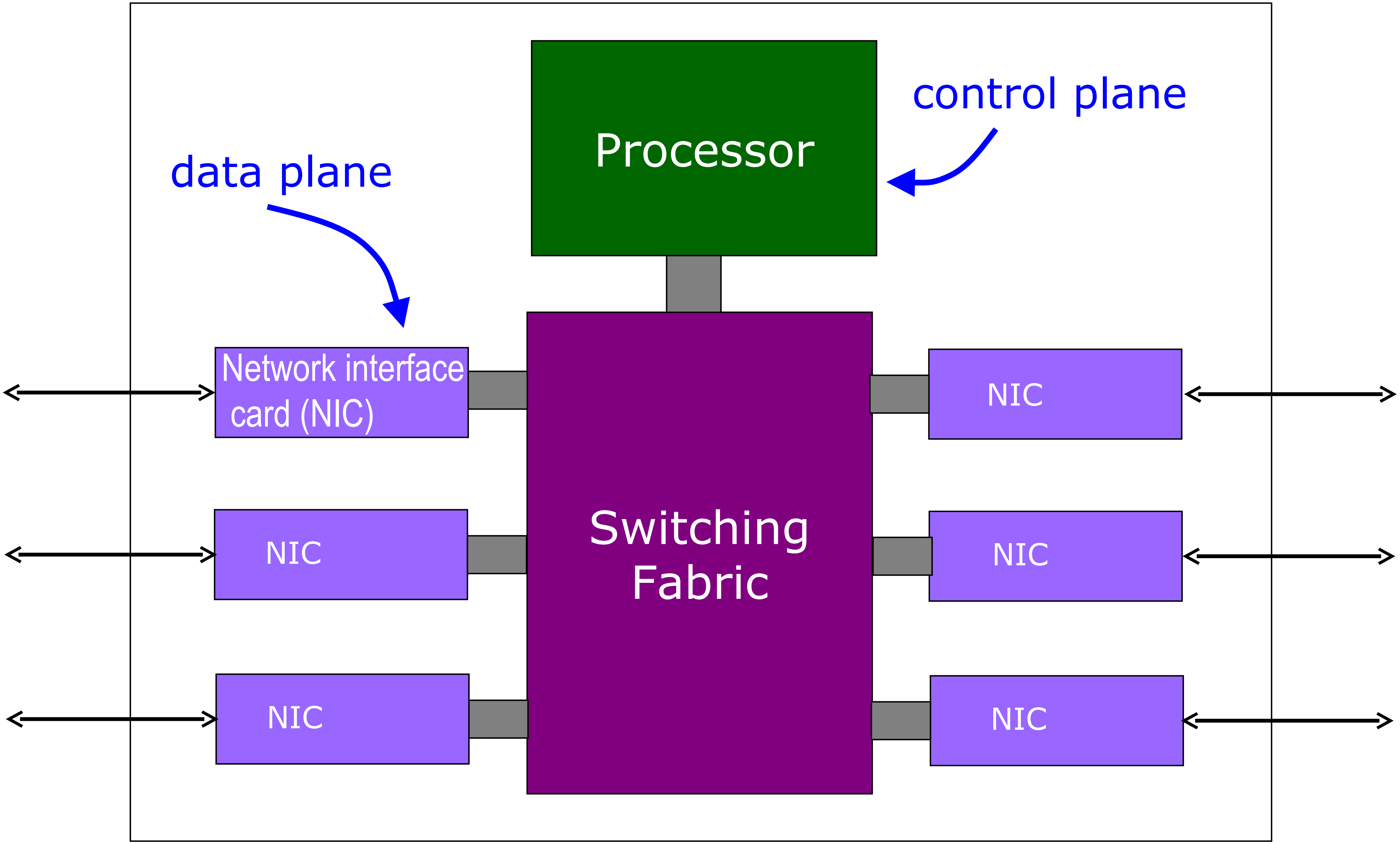
Routing: Mapping end-to-end transfer to Path



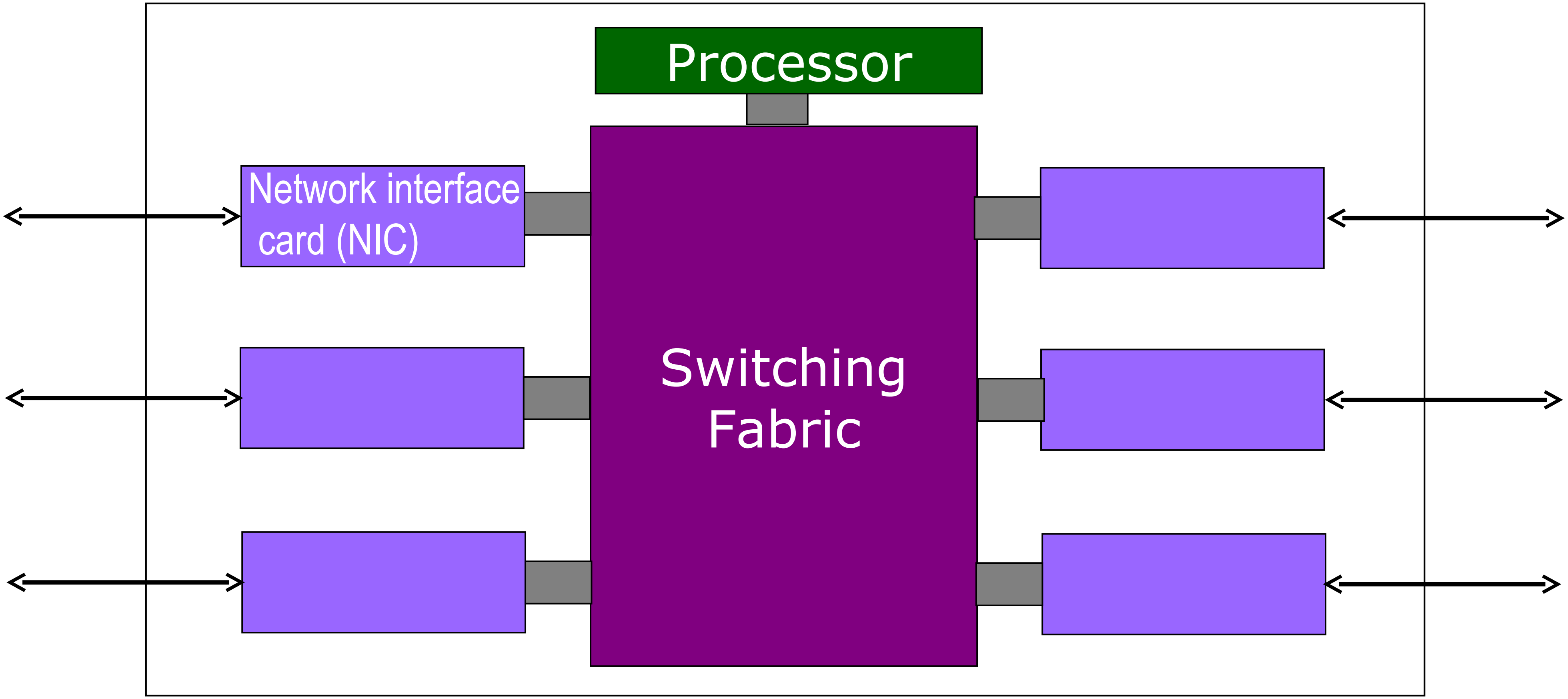
- Configuring routing tables at each node

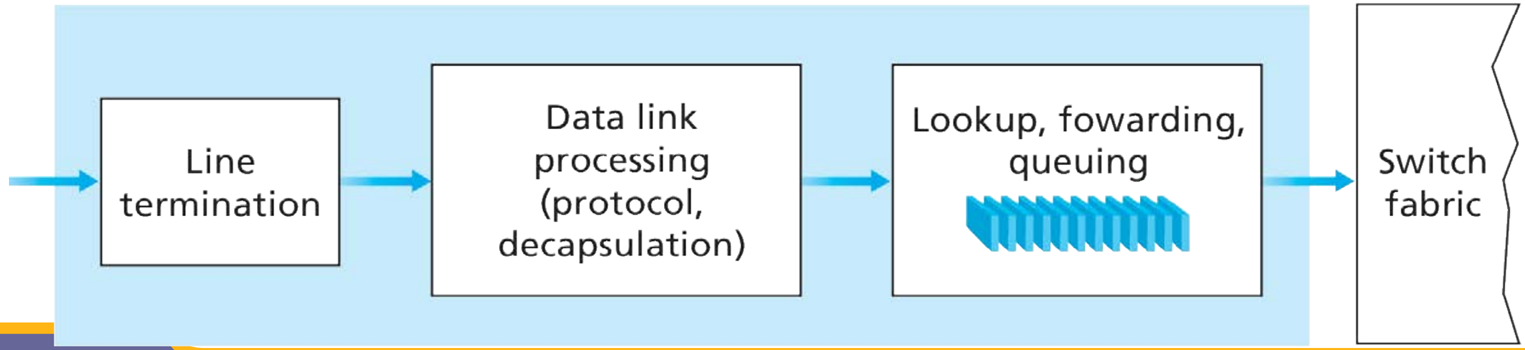


Data and Control Planes

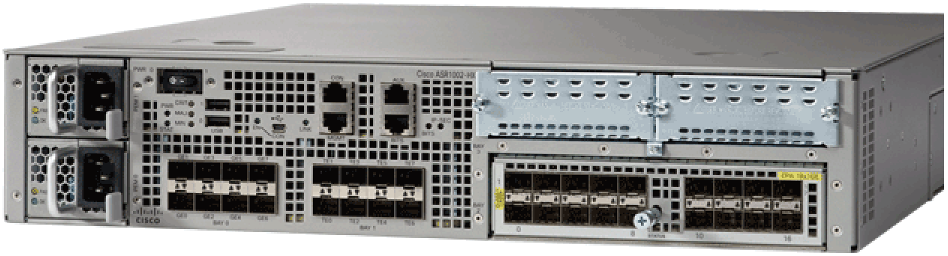


Data and Control Planes

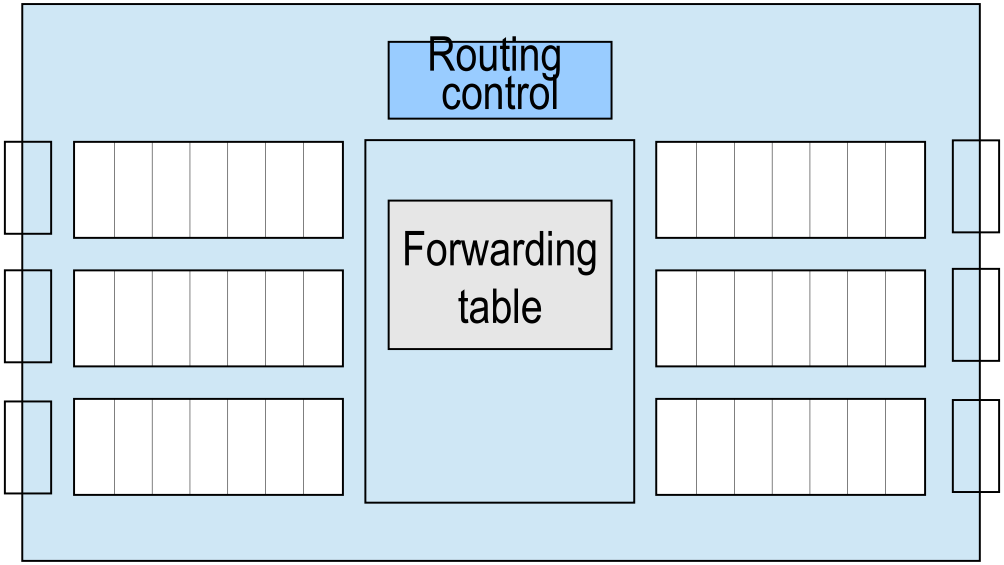




An ideal router

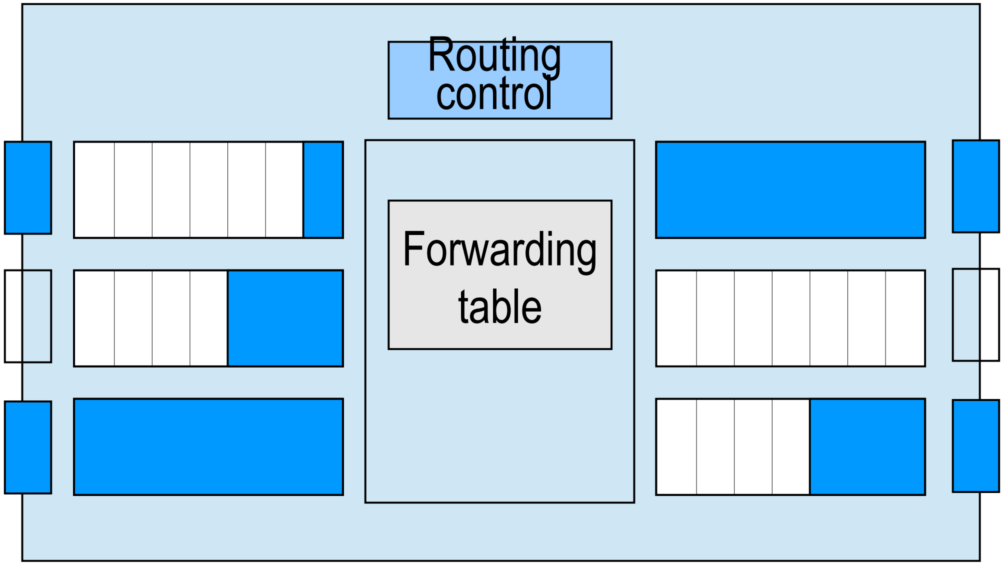


- Structure, forwarding, routing, buffer queues



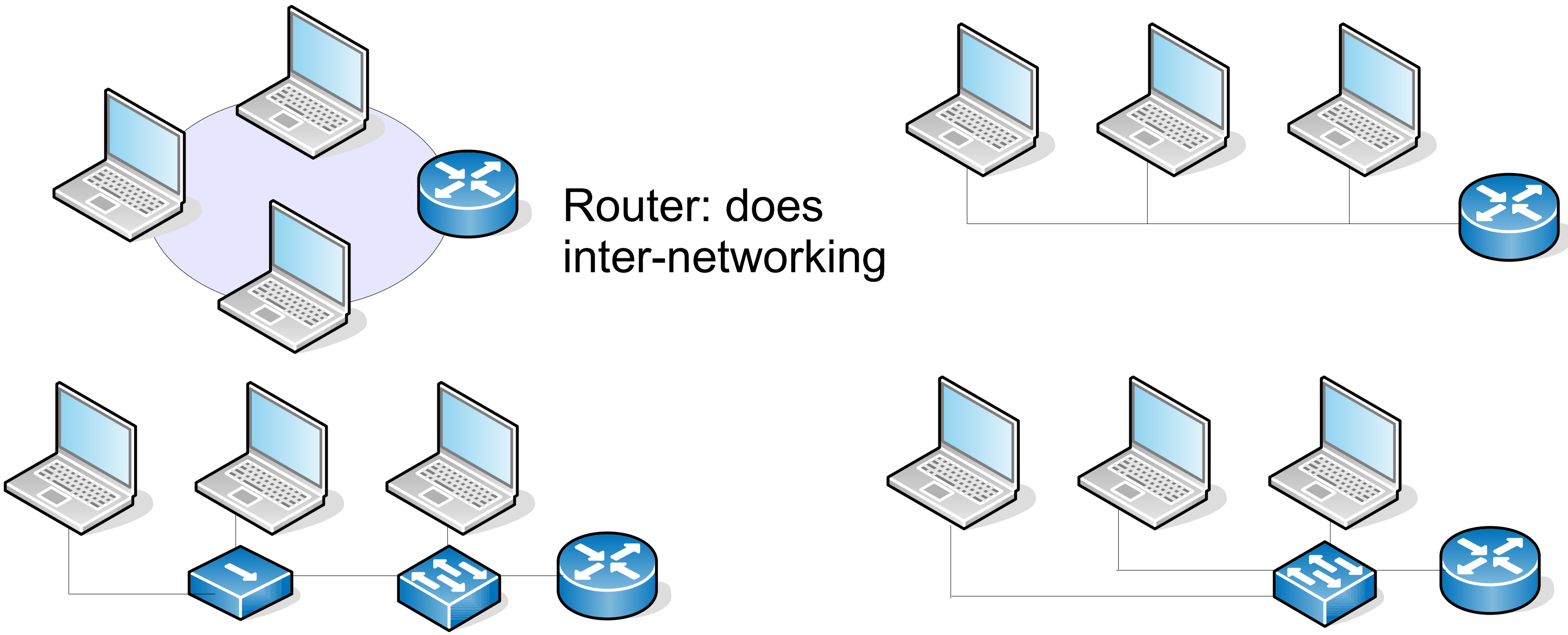
An ideal router with traffic

- Traffic load → delay,
- Buffer overflow: loss



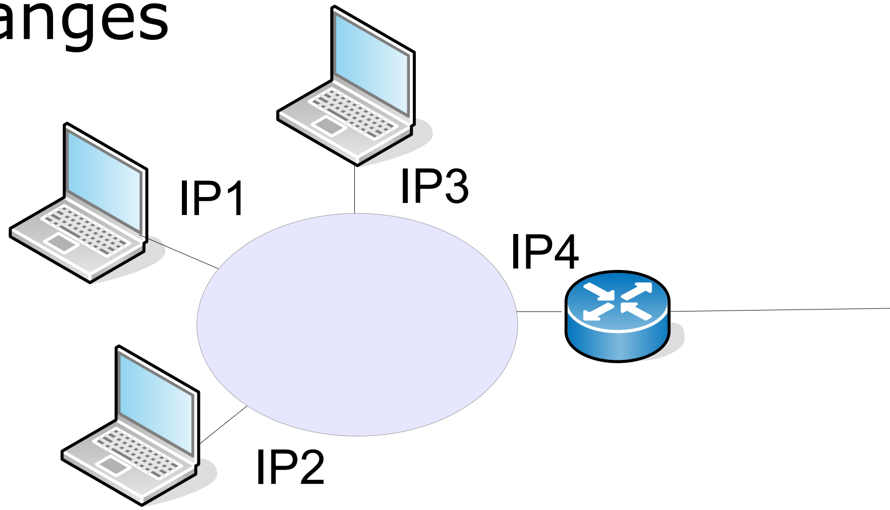
An ideal IP network

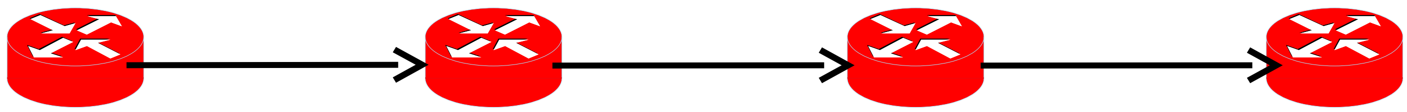
- A group of hosts and routers interconnected
  - By layer 1 (hub) or layer 2 (switch) devices
  - Each network shares a common IP range



IP addressing: layer 3

- An IP address:
  - An interface connected to a network
  - Layer 3 interfaces (host, router)
  - NOT to layer 1, 2 interfaces/devices :hub, switch
  - All IP addresses different, but groups: network addresses or ranges





Addressing, Routing, Forwarding

- Addressing:
  - Identification of hosts, routers, and networks
- Routing: control plane
  - Computing paths the packets will follow
  - Routers talking amongst themselves
  - Creating the forwarding tables
- Forwarding: data plane
  - Directing a data packet to an outgoing link
  - Using the forwarding tables

Layer 3 - IPv4 datagram

<table>
<tr><td>Version</td><td>IHL</td><td colspan="2">Type of Service</td><td colspan="4">Total Length</td></tr>
<tr><td colspan="4">Identification</td><td>Flags</td><td colspan="3">Fragment Offset</td></tr>
<tr><td colspan="2">Time to Live</td><td colspan="2">Protocol</td><td colspan="4">Header Checksum</td></tr>
<tr><td colspan="8">Source Address (32-bit IPv4 address)</td></tr>
<tr><td colspan="8">Destination Address (32-bit IPv4 address)</td></tr>
<tr><td colspan="6">Options</td><td colspan="2">Padding</td></tr>
<tr><td colspan="8">Data (contains layer 4 segment)</td></tr>
</table>

- Version = 4
  If no options, IHL = 5
  Source and Destination are 32-bit IPv4 addresses.
- Protocol = 6 means data portion contains a TCP segment. Protocol = 17 means UDP.

Purpose of an IPv4 address

- Unique Identification of:
  - Source
    - So the recipient knows where the message is from
    - Sometimes used for security or policy-based filtering of data
  - Destination
    - So the networks know where to send the data
- Network Independent Format
  - IP over anything

Purpose of an IP Address

- Identifies a machine’s connection to a network
- Physically moving a machine from one network to another requires changing the IP address
- *Unique*; assigned in a hierarchical fashion:
  - IANA (Internet Assigned Number Authority)
  - IANA to Regional Internet Registries (RIRs): AfriNIC, ARIN, RIPE, APNIC, LACNIC
  - RIR to ISPs and large organisations
  - ISP or company IT department to end users
- IPv4 uses unique 32-bit addresses
- IPv6 used similar concepts but 128-bit addresses

Basic Structure of an IPv4 Address

- 32 bit number (4 octet number):
  (e.g. 133.27.162.125)
- Decimal Representation:

133	27	162	125

- Binary Representation:

10000101	00011011	10100010	01111101

- Hexadecimal Representation:

85	1B	A2	7D

Addressing in Internetworks

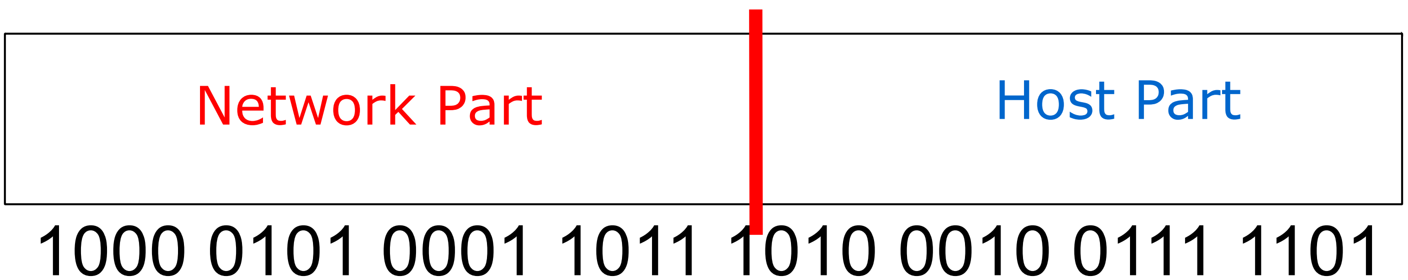
- The problem we have
  - More than one physical network
  - Different Locations
  - Larger number of hosts
  - Need a way of numbering them all
- We use a structured numbering system
  - Hosts that are connected to the same physical network have “similar” IP addresses
  - Often more then one level of structure; e.g. physical networks in the same organisation use “similar” IP addresses
  - Ex: 147.83.2.1, 147.83.2.2, 147.83.2.3 ...

Network part and Host part

- Remember IPv4 address is 32 bits
- Divide it into a “network part” and “host part”
  - “network part” of the address identifies which network in the internetwork (e.g. the Internet)
  - “host part” identifies host on that network
  - Hosts or routers connected to the same link-layer network will have IP addresses with the same network part, but different host part.
  - Host part contains enough bits to address all hosts on the subnet; e.g. 8 bits allows 256 addresses
  - Ex: 147.83.2.1…147.83.2.3: 147.83.2.0/30+2

Dividing an address

- Hierarchical Division in IP Address:
  - Network Part (or Prefix) – high order bits (left)
    - describes which physical network
  - Host Part – low order bits (right)
    - describes which host on that network



  - Boundary can be anywhere
    - choose the boundary according to number of hosts
    - very often NOT a multiple of 8 bits

Network Masks

- “Network Masks” help define which bits are used to describe the Network Part and which for the Host Part
- Different Representations:
  - decimal dot notation: 255.255.224.0
  - binary: 11111111 11111111 11100000 00000000
  - hexadecimal: 0xFFFFE000
  - number of network bits: /19
    - count the 1's in the binary representation
- Above examples all mean the same: 19 bits for the Network Part and 13 bits for the Host Part
- Ex: 147.83.2.1…147.83.2.3 (00..11):
  – 147.83.2.0/ 30 mask bits
  – 147.83.2.0/ 255.255.255.252 subnet mask or 0.0.0.3 subnet wildcard

Example Prefixes

- 137.158.128.0/17 (netmask 255.255.128.0)

  1111 1111 1111 1111 1 000 0000 0000 0000

  1000 1001 | 1001 1110 | 1 000 0000 | 0000 0000

- 198.134.0.0/16 (netmask 255.255.0.0)

  1111 1111 1111 1111 0000 0000 0000 0000

  1100 0110 | 1000 0110 | 0000 0000 | 0000 0000

- 205.37.193.128/26 (netmask 255.255.255.192)

  1111 1111 1111 1111 1111 1111 11 00 0000

  1100 1101 | 0010 0101 | 1100 0001 | 10 00 0000

Special Addresses

- All 0’s in host part: represents <u>this</u> network
  - e.g. 193.0.0.0/24
  - e.g. 138.37.64.0/18
  - e.g. 196.200.223.96/28
- All 1’s in host part: <u>broadcast</u> this network
  - e.g. 193.0.0.255 (prefix 193.0.0.0/24)
  - e.g. 138.37.127.255 (prefix 138.37.64.0/18)
  - e.g. 196.200.223.111 (prefix 196.200.223.96/28)
- 127.0.0.0/8: <u>loopback</u>, internal addr (127.0.0.1)
- 0.0.0.0: Various special purposes (anything else)

Address ranges

- 1.2.3.0/32 = one specific IP address
- 1.2.3.0/30 = 00..11, addr: .1, .2
- 1.2.3.0/29 = 000..111, addr: 1..6
- 1.2.3.0/28 = 0000..1111, addr: 1..14
- 1.2.3.0/27 = 00000..11111, addr: 1..30
- 1.2.3.0/26 = 000000..111111, addr: 1..62
- 1.2.3.0/25 = 0000000..1111111, addr: 1..126
- 1.2.3.0/24 = 00000000..11111111, addr: 1..254

128 64 32 16 8 4 2 1

0000 0000 0000 0000
0000 0000 0000 0001
0000 0000 0000 0010
0000 0000 0000 0011
0000 0000 0000 0100
0000 0000 0000 0101
0000 0000 0000 0110
0000 0000 0000 0111
0000 0000 0000 1000
0000 0000 0000 1001
0000 0000 0000 1010
0000 0000 0000 1011
0000 0000 0000 1100
0000 0000 0000 1101
0000 0000 0000 1110
0000 0000 0000 1111
0000 0000 0001 0000
0000 0000 1111 1111

Exercise

- Verify that the previous examples are all broadcast addresses:
  - 193.0.0.255 (prefix 193.0.0.0/24)
  - 138.37.127.255 (prefix 138.37.64.0/18)
  - 196.200.223.111 (prefix 196.200.223.96/28)
- Do this by finding the boundary between network part and host part, and checking that the host part (if written in binary) contains all 1's.

Maximum number of hosts per network

- The number of bits in the host part determines the maximum number of hosts (2^h)
- The all-zeros and all-ones addresses are reserved, can't be used for actual hosts (-2)
- E.g. a subnet mask of 255.255.255.0 or /24 means 24 network bits, 8 host bits (24+8=32)
  - 28 - 2 = 254 possible hosts
- Similarly a subnet mask of 255.255.255.224 or /27 means 27 network bits, 5 host bits (27+5=32)
  - 25 - 2 = 30 possible hosts

More Address Exercises

- Assuming there are 9 routers on the classroom backbone network:
  - what is the **minimum number of host bits** needed to address each router with a unique IP address?
  - with that many host bits, how many **network bits**?
  - what is the corresponding **prefix length** in “slash” notation?
  - what is the corresponding **netmask** (in decimal)?
  - with that netmask, what is the **maximum number of hosts**?

More levels of address hierarchy

- Extend concept of “network part” and “host part”:
  - arbitrary number of levels of hierarchy
  - blocks don’t all need to be the same size
  - but each block size must be a power of 2
- Very large blocks allocated to RIRs (e.g. /8)
  - Divided into smaller blocks for ISPs (e.g. /17)
  - Divided into smaller blocks for businesses (e.g. /22)
  - Divided into smaller blocks for local networks (e.g. /26)
  - Each host gets a host address
- What if addresses overlap??

Ancient History: Classful Addressing

- Nowadays, we always explicitly say where the boundary between network and host part is
  - using slash notation or netmask notation
- Old systems used restrictive rules (obsolete)
  - Called “Class A”, “Class B”, “Class C” networks
  - Boundary between network part and host part was implied by the class
- Nowadays (since 1994), no restriction
  - Called “classless” addressing, “classless” routing

Ancient History: Sizes of classful nets

- Different classes were used to represent different sizes of network (small, medium, large)

- Class A networks (large): implied /8
  - 8 bits network part, 24 bits host part
- Class B networks (medium): implied /16
  - 16 bits network part, 16 bits host part
- Class C networks (small): implied /24
  - 24 bits network part, 8 bits host part

Ancient History:
What class is my address?

- Just look at the address to tell what class it is.
  - Class **A**: 0.0.0.0 to 127.255.255.255
    - binary 0nnnnnnnhhhhhhhhhhhhhhhhhhhhhhhh
  - Class **B**: 128.0.0.0 to 191.255.255.255
    - binary 10nnnnnnnnnnnnnnhhhhhhhhhhhhhhhh
  - Class **C**: 192.0.0.0 to 223.255.255.255
    - binary 110nnnnnnnnnnnnnnnnnnnnnhhhhhhhh

  - Class **D**: (multicast) 224.0.0.0 to 239.255.255.255
    - binary 1110xxxxxxxxxxxxxxxxxxxxxxxxxxxx
  - Class **E**: (reserved) 240.0.0.0 to 255.255.255.255

Example: UPC

- Has 147.83.0.0/16 assigned by RIPE
- Part of AS13041 Consorci de Serveis Universitaris de Catalunya (csuc.cat)
- Class B:
- Subnet Mask: 255.255.0.0
- IP Range: 147.83.0.1 - 147.83.255.254
- Max Hosts / Subnet : 65534
- BitMap: nnnnnnnn.nnnnnnnn.hhhhhhhh.hhhhhhhh
- BinMap: <u>10</u>010011.01010011.00000000.00000000

Classless addressing

- Class A, Class B, Class C terminology and restrictions are now of historical interest only
  - Obsolete in 1994
- Internet routing and address management today is classless
- CIDR = Classless Inter-Domain Routing
  - routing does not assume that class A, B, C implies prefix length /8, /16, /24
- VLSM = Variable-Length Subnet Masks
  - routing does not assume that all subnets are the same size

Classless addressing example

- An ISP gets a large block of addresses
  - e.g., a /16 prefix, or 65536 separate addresses
- Allocate smaller blocks to customers
  - e.g., a /22 prefix (1024 addresses) to one customer, and a /28 prefix (16 addresses) to another customer (and some space left over for other customers)
- An organisation that gets a /22 prefix from their ISP divides it into smaller blocks
  - e.g. a /26 prefix (64 addresses) for one department, and a /27 prefix (32 addresses) for another department (and some space left over for other internal networks)

Classless addressing exercise

- Consider the address block 147.83.162.0/23
- Allocate 5 separate /29 blocks, one /27 block, and one /25 block
- What are the IP addresses of each block allocated above?
  - in prefix length notation
  - netmasks in decimal
  - IP address ranges
- What blocks are still available (not yet allocated)?
- How big is the largest available block?

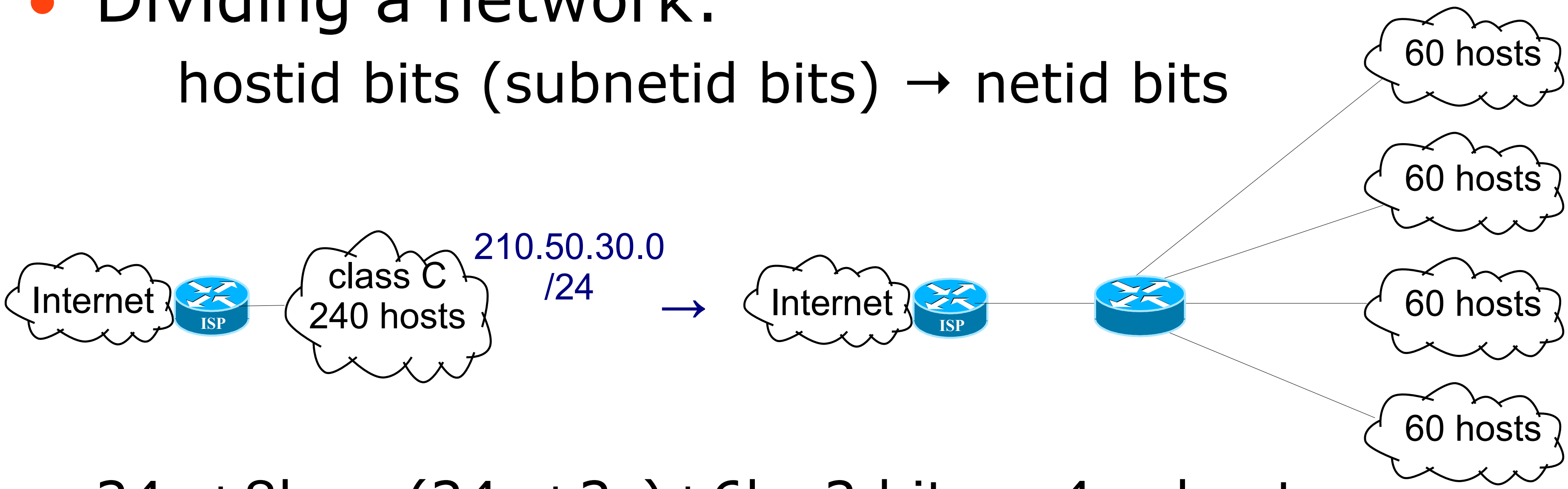
Private addresses

- For nodes with non public addresses
- Not assigned to any RIR (not unique, reused):
  - 1 class A network: 10.0.0.0
  - 16 class B networks: 172.16.0.0 ~ 172.31.0.0
  - 256 class C networks:
    192.168.0.0 ~ 192.168.255.0

Subnetting

- Dividing a network:

  hostid bits (subnetid bits) → netid bits



- 24n+8h → (24n+2s)+6h: 2 bits = 4 subnets
- Mask: /24 → /26, 255.255.255.192

Subnetting example

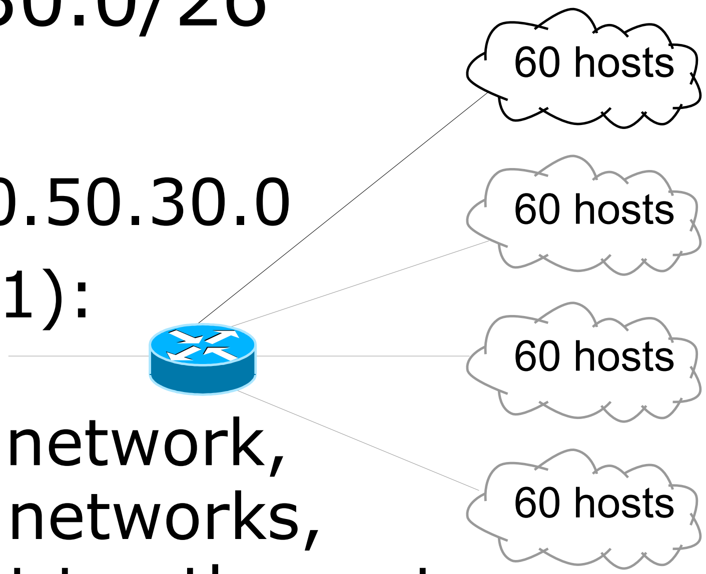
- Subnet range 210.50.30.0/24 in 4 subnets

$B = 210.50.30$

subnet	subnetid	IP net. addr.	range	broadcast	available
S1	00	B.0/26	B.0 $\sim$ B.63	B.63	$2^6 - 2 = 62$
S2	01	B.64/26	B.64 $\sim$ B.127	B.127	$2^6 - 2 = 62$
S3	10	B.128/26	B.128 $\sim$ B.191	B.191	$2^6 - 2 = 62$
S4	11	B.192/26	B.192 $\sim$ B.255	B.255	$2^6 - 2 = 62$

Subnetting

- One sub-network: 210.50.30.0/26
- Addresses:
  - The network (subnetID): 210.50.30.0
  - Broadcast (all hostID bits to 1):
    210.50.30.63
    To all interfaces in the same network,
    does not go beyond to other networks,
    cannot send broadcast packet to other nets.
  - Unicast addresses:
    210.50.30.1, 210.50.30.2, … 210.50.30.62



Variable Length Subnet Mask (VLSM)

- Subnets of different sizes:
  - Example, subnetting a /24

$$
\begin{array}{lll}
\underline{\mathbf{0}000} & & \\
\mathbf{1}000 \} \rightarrow & \underline{\mathbf{10}00} & \\
 & \mathbf{11}00 \} \rightarrow & \underline{\mathbf{1100}} \\
 & & \underline{\mathbf{1101}} \\
 & & \underline{\mathbf{1110}} \\
 & & \underline{\mathbf{1111}}
\end{array}
$$

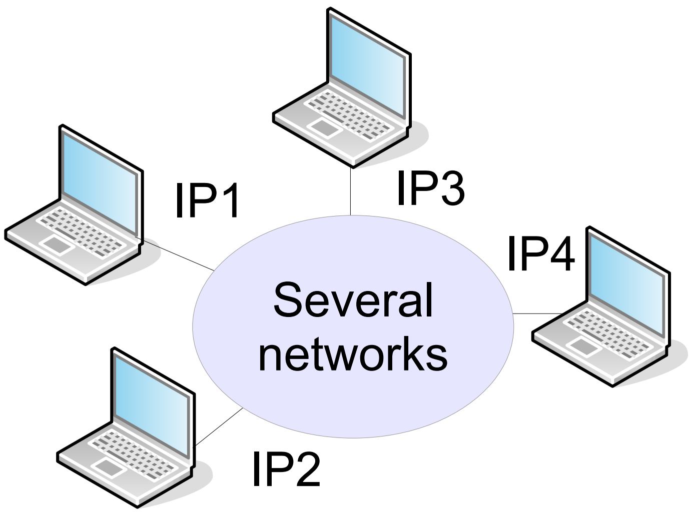
subnet	subnetid	IP net. addr.	range	broadcast	available
S1	0	B.0/25	B.0 ∼ B.127	B.127	$2^7 - 2 = 126$
S2	10	B.128/26	B.128 ∼ B.191	B.191	$2^6 - 2 = 62$
S3	1100	B.192/28	B.192 ∼ B.207	B.207	$2^4 - 2 = 14$
S4	1101	B.208/28	B.208 ∼ B.223	B.223	$2^4 - 2 = 14$
S5	1110	B.224/28	B.224 ∼ B.239	B.239	$2^4 - 2 = 14$
S6	1111	B.240/28	B.240 ∼ B.255	B.255	$2^4 - 2 = 14$

Subnet operations

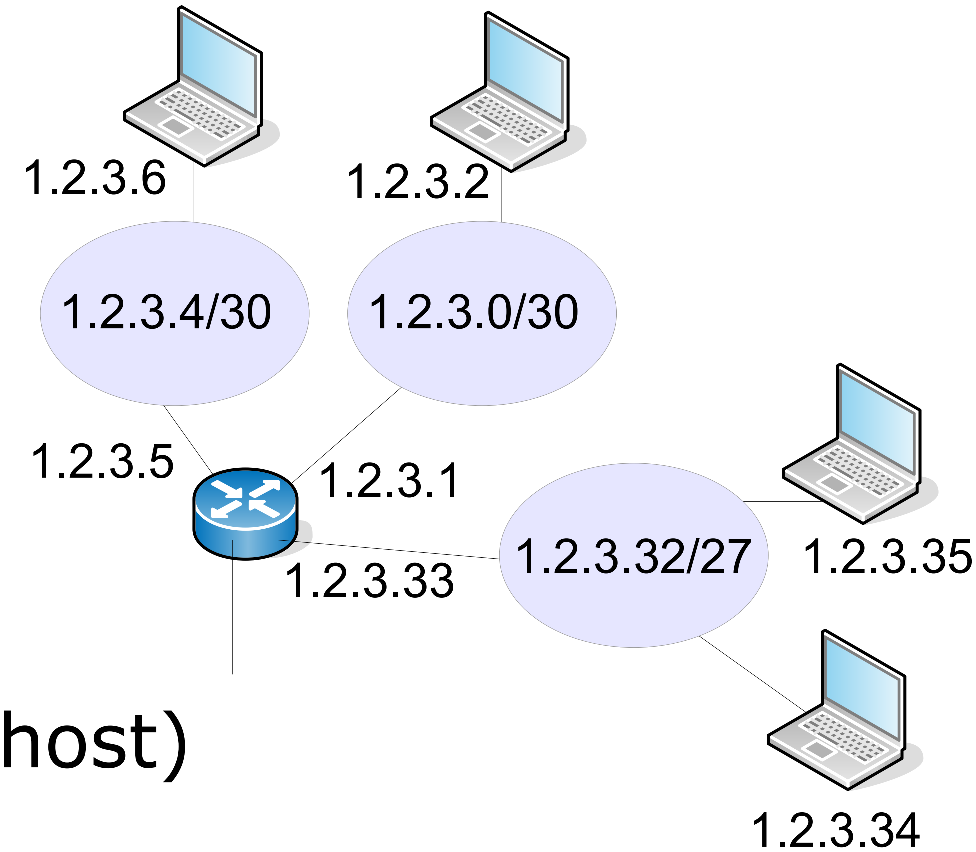
- Subnetting: splitting though subnet bits
- Aggregation:
  - 200.1.10.0/24 + 200.1.11.0/24 → 200.1.10.0/23
  - Summarization: group of subnets summarized to classful range
  - Must be compact, otherwise imprecise as 0.0.0.0/0

Subnets

- ~~IP1 $\in$ 1.2.3.4/24,~~
- IP1 $\in$ 1.2.3.4/30,
- IP2 $\in$ 1.2.3.32/27
- IP3 $\in$ 1.2.3.0/30
- IP4 $\in$ 1.2.3.34/32
- Network configuration?



Subnets



- IP1 ∈ 1.2.3.4/30,
- IP2 ∈ 1.2.3.32/27
- IP3 ∈ 1.2.3.0/30
- IP4 ∈ 1.2.3.34/32 (host)
- Networks:
  - 1.2.3.4, 1.2.3.32, 1.2.3.0
  - 1.2.3.4..7, 1.2.3.32..63, 1.2.3.0..3
  - 1.2.3.0/30 + 1.2.3.4/30 → 1.2.3.0/29 ? (3 bits)

Forwarding

The need for Packet Forwarding

- Many small networks can be interconnected to make a larger internetwork
- A device on one network cannot send a packet directly to a device on another network
- The packet has to be forwarded from one network to another, through intermediate nodes, until it reaches its destination
- The intermediate nodes are called “routers”

An IP Router

- **A device with more than one link-layer interface**
- Different IP addresses (from different subnets) on different interfaces
- Receives packets on one interface, and forwards them (usually out of another interface) to get them one hop closer to their destination
- Maintains forwarding tables

IP Router - action for each packet

- Packet is received on one interface
- Checks whether the destination address is the router itself – if so, pass it to higher layers
- Decrement TTL (time to live), and discard packet if it reaches zero
- Look up the destination IP address in the forwarding table
- Destination could be on a **directly** attached link, or **indirect**, through another router

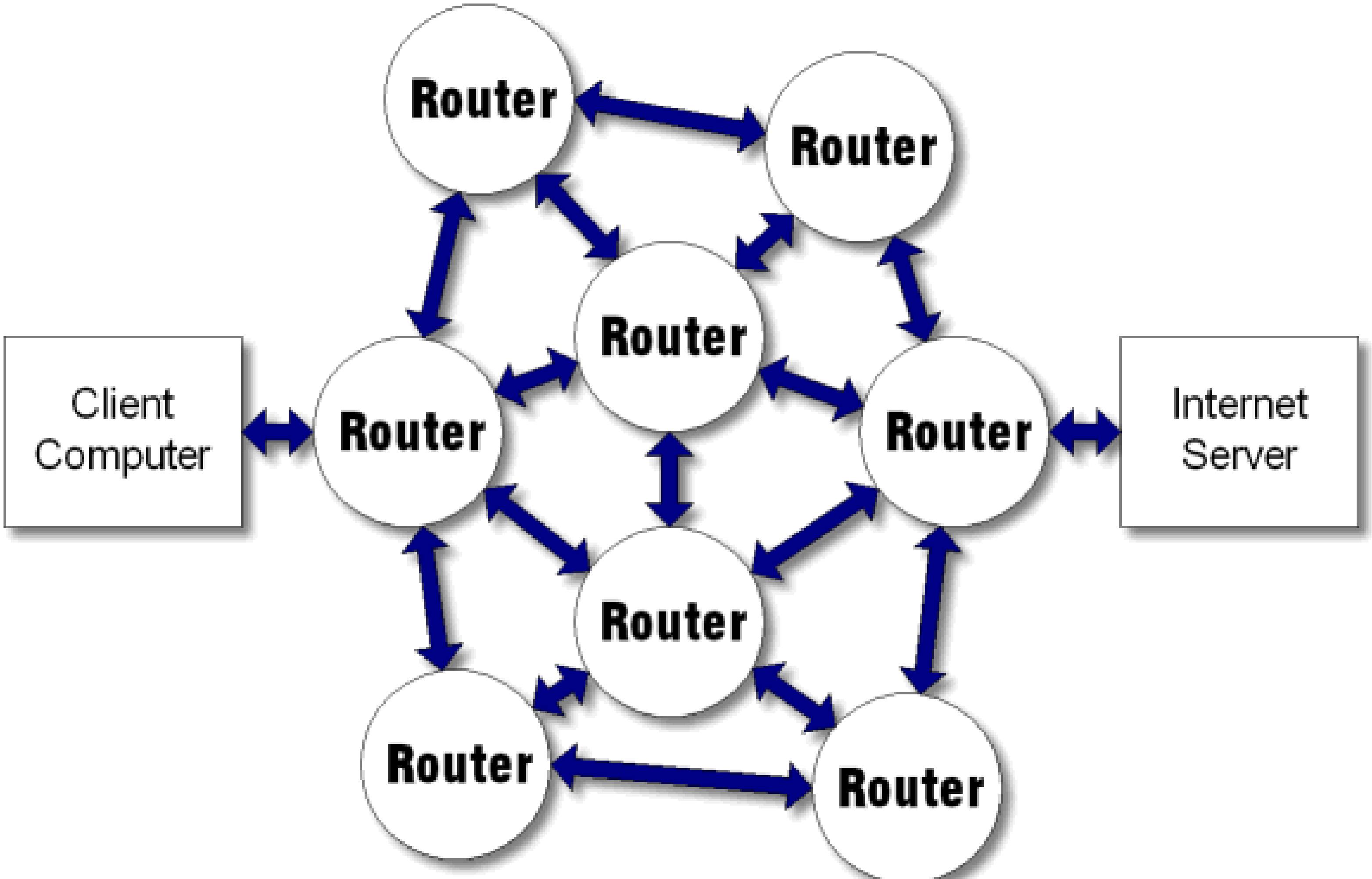
Forwarding vs. Routing

- **Forwarding**: the process of moving packets from input to output
  - The forwarding table
  - Information in the packet
- **Routing**: process by which the forwarding table is built and maintained
  - One or more routing protocols
  - Procedures (algorithms) to convert routing info to forwarding table.
- (Much more later …)

Forwarding is hop by hop

- Each router tries to get the packet one hop **closer** to the destination
- Each router makes an **independent** decision, based on its own forwarding table
- Different routers have **different** forwarding tables and make different decisions
  - If all is well, decisions will be consistent
- Routers talk routing protocols to each other, to help **update** routing and forwarding tables

Hop by Hop Forwarding



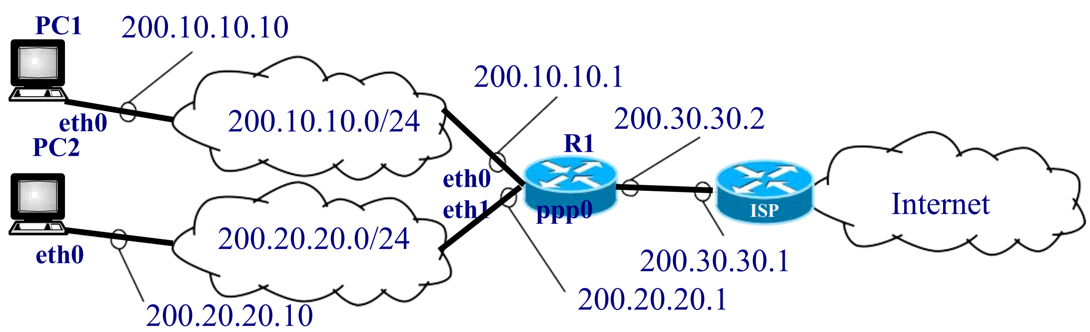
Router Functions

- Determine optimum routing paths through a network
  - Lowest delay
  - Highest reliability
- Move packets through the network
  - Examines destination address in packet
  - Makes a decision on which port to forward the packet through
  - Decision is based on the Routing Table
- Interconnected Routers exchange routing tables in order to maintain a clear picture of the network
- In a large network, the routing table updates can consume a lot of bandwidth
  - a protocol for route updates is required

Forwarding table structure

- We don't list every IP number on the Internet - the table would be huge
- Instead, the forwarding table contains prefixes (network numbers)
  - "If the first /n bits matches this entry, send the datagram that way"
- If more than one prefix matches, the longest prefix wins (more specific route)
- 0.0.0.0/0 is "default route" - matches anything, but only if no other prefix matches

Routing Table – Unix Example



known destinations

how to reach the destinations

PC1 routing table:

Destination	Genmask	Gateway	Iface
200.10.10.0	255.255.255.0	0.0.0.0	eth0
0.0.0.0	0.0.0.0	200.10.10.1	eth0

R1 routing table:

Destination	Genmask	Gateway	Iface
200.10.10.0	255.255.255.0	0.0.0.0	eth0
200.20.20.0	255.255.255.0	0.0.0.0	eth1
0.0.0.0	0.0.0.0	200.30.30.1	ppp0

Routing Table – Datagram Delivery Algorithm

- 1. Check if itself is destination:

```
if(Datagram Destination == address of any of the interfaces)
    send datagram to upper layers
```

- 2. Lookup the routing table:

```
for each routing table entry (Longest Prefix Match first)
        if((Datagram Dest IP addr & mask) == Dest table entry)
                return (gateway, interface)
```

- 3. Forward the datagram

```
if(direct routing) {
        send the datagram to the Datagram Destination IP address
} else { /* indirect routing */
        send datagram to gateway IP address
}
```

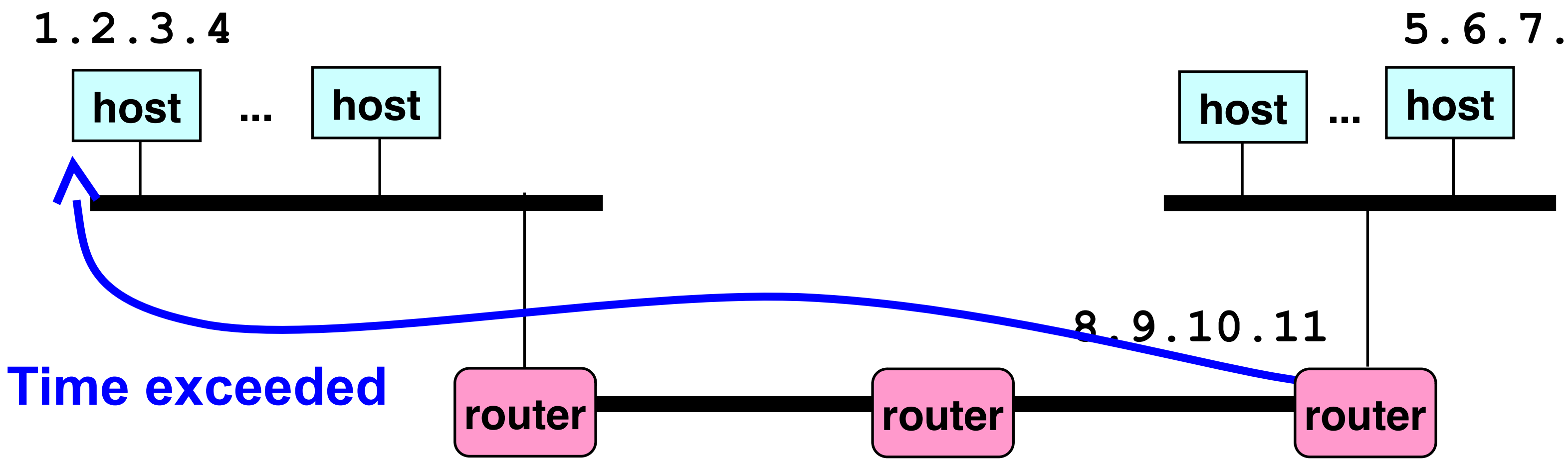
Network control

Internet Control Message Protocol

- ICMP runs on top of IP
  - In parallel to TCP and UDP
  - Though still viewed as an integral part of IP
- Diagnostics
  - Triggered when an IP packet encounters a problem
    - E.g., time exceeded or destination unreachable
  - ICMP packet sent back to the source IP address
    - Includes the error information (e.g., type and code)
    - … and an excerpt of the original data packet for identification
  - Source host receives the ICMP packet
    - And inspects the except of the packet (e.g., protocol and ports)
    - … to identify which socket should receive the error

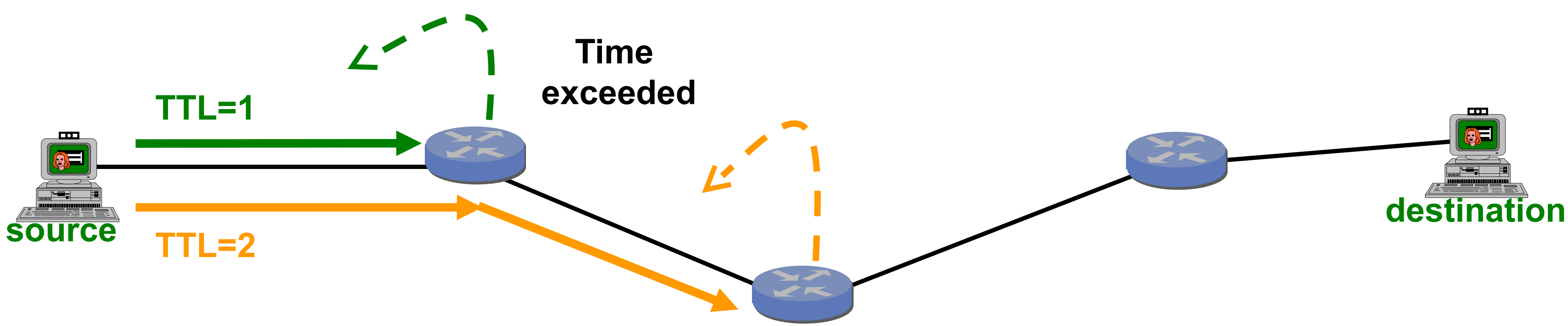
Example: Time Exceeded

- Host sends an IP packet
  - Each router decrements the time-to-live field
- If time-to-live field reaches 0
  - Router generates an ICMP message
  - Sends a “time exceeded” message back to the source



Traceroute: Exploiting "Time Exceeded"

- Time-To-Live field in IP packet header
  - Source sends a packet with a TTL of *n*
  - Each router along the path decrements the TTL
  - "TTL exceeded" sent when TTL reaches *0*
- Traceroute tool exploits this TTL behavior



Send packets with TTL=1, 2, … and record source of "time exceeded" message

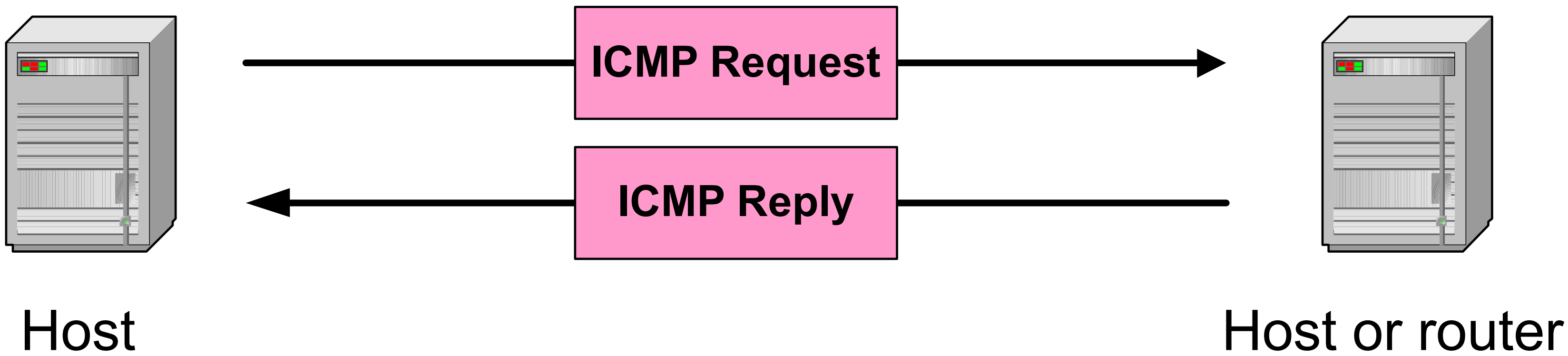
Control functions

- Host unreachable
- Port, Protocol unreachable
- Network unreachable
- Redirect
- IP packet does not fit in link packet: fragmentation
- Congestion: source quench

Ping: Echo and Reply

- ICMP includes a simple "echo" function
  - Sending node sends an ICMP "echo" message
  - Receiving node sends an ICMP "echo reply"
- Ping tool
  - Tests the connectivity with a remote host
  - … by sending regularly spaced echo commands
  - … and measuring the delay until receiving the reply
- Pinging a host
  - "ping www.upc.edu" or "ping 147.83.2.135"
  - Used to test if a machine is reachable and alive
  - (However, some nodes have ICMP disabled… ☹)

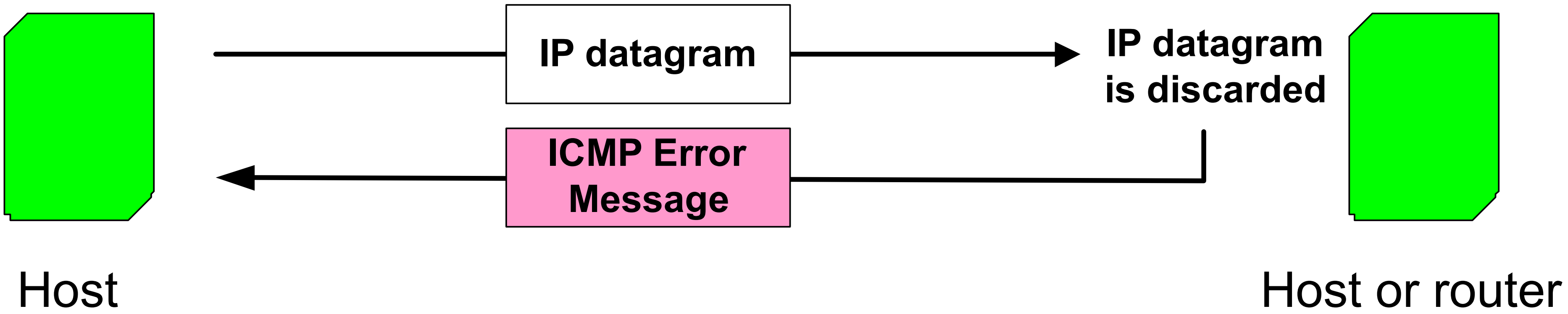
ICMP Query message



ICMP query:

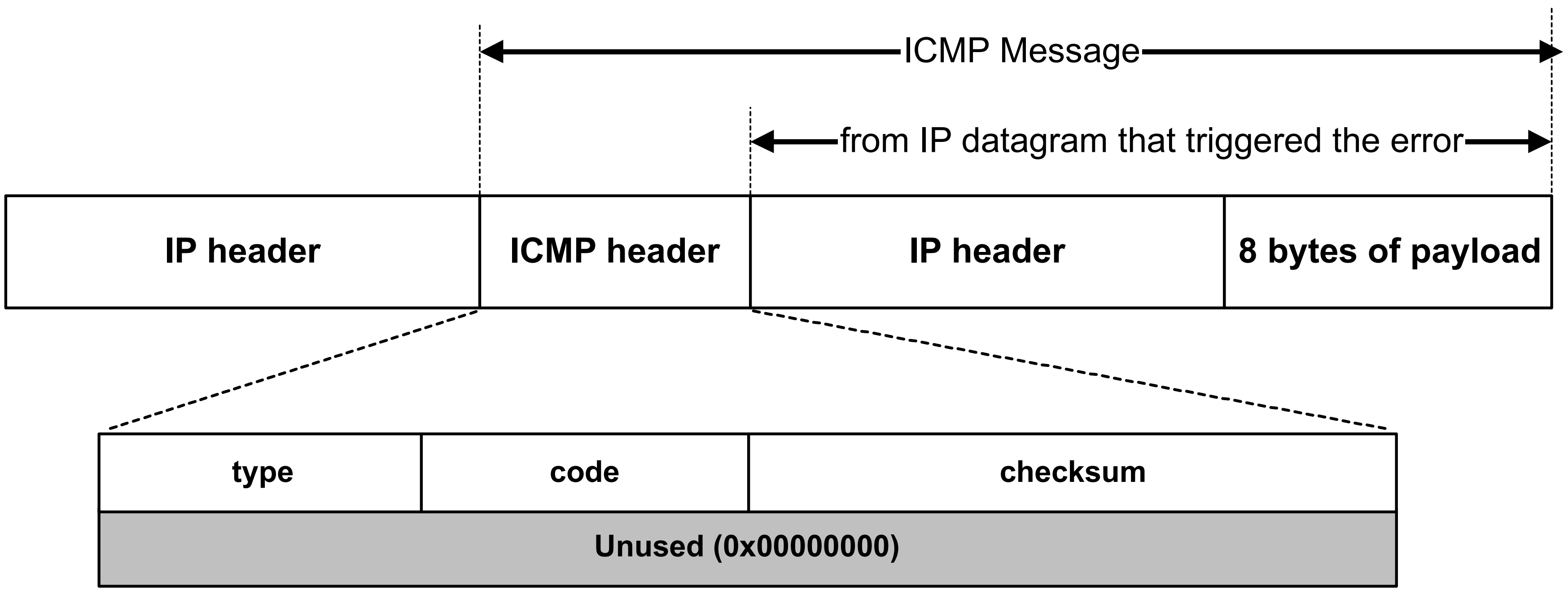
- Request sent by host to a router or host
- Reply sent back to querying host

ICMP Error message



- **ICMP error messages report error conditions**
- **Typically sent when a datagram is discarded**
- **Error message is often passed from ICMP to the application program**

ICMP Error message



- ICMP error messages include the complete IP header and the first 8 bytes of the payload (typically: UDP, TCP)

Frequent ICMP Error message

Type	Code	Description	
3	0–15	Destination unreachable	Notification that an IP datagram could not be forwarded and was dropped. The code field contains an explanation.
5	0–3	Redirect	Informs about an alternative route for the datagram and should result in a routing table update. The code field explains the reason for the route change.
11	0, 1	Time exceeded	Sent when the TTL field has reached zero (Code 0) or when there is a timeout for the reassembly of segments (Code 1)
12	0, 1	Parameter problem	Sent when the IP header is invalid (Code 0) or when an IP header option is missing (Code 1)

Some subtypes of the “Destination Unreachable”

Code	Description	Reason for Sending
0	Network Unreachable	No routing table entry is available for the destination network.
1	Host Unreachable	Destination host should be directly reachable, but does not respond to ARP Requests.
2	Protocol Unreachable	The protocol in the protocol field of the IP header is not supported at the destination.
3	Port Unreachable	The transport protocol at the destination host cannot pass the datagram to an application.
4	Fragmentation Needed and DF Bit Set	IP datagram must be fragmented, but the DF bit in the IP header is set.

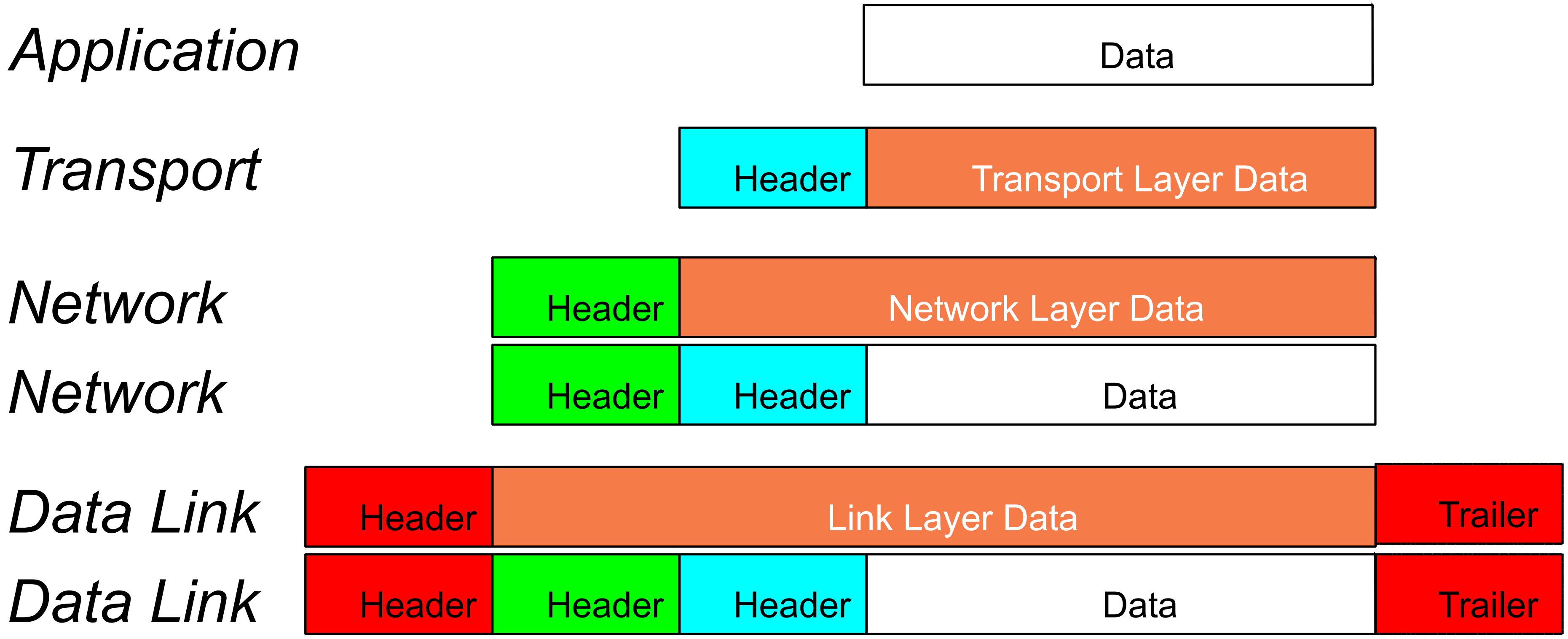
Network mappings

Network mappings

- Discovery:
  - Finding a neighbour
  - Finding a path (routing)
- Mapping (resolving) a host
  - Down at media access (MAC addresses)
  - Up at application names (DNS host names)
- Mapping IP packet (fragmentation)
- Mapping addresses (net addr translation)
- Bootstrapping:
  - Host learns about its environment (config)

Finding neighbours

- Encapsulation reminder:
  Lower layers add headers (and sometimes trailers) to data from higher layers



Ethernet briefing

- Ethernet is a broadcast medium
- Structure of Ethernet frame:

Preamble	Dest	Source	Type	Data	CRC

- Entire IP packet makes data part of Ethernet frame
- Data part: 1492, 1500, 1501-9198 (Jumbo frames)
- Delivery mechanism (CSMA/CD)
  - back off and try again when collision is detected

Ethernet/IP Address Resolution

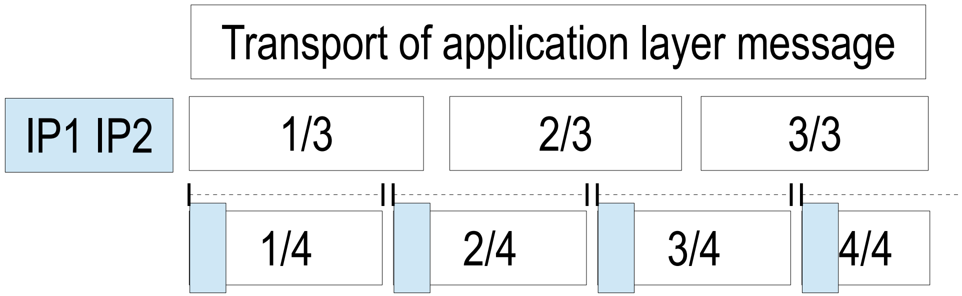
- Internet Address
  - Unique worldwide (excepting private nets)
  - Independent of Physical Network technology
- Ethernet Address
  - Unique worldwide (excepting errors)
  - Ethernet Only
- Need to map from higher layer to lower (i.e. IP to Ethernet, using ARP)

Layer 3 - IPv4 packet

<table>
<tr><td>Version</td><td>IHL</td><td colspan="2">Differentiated Services</td><td colspan="4">Total Length</td></tr>
<tr><td colspan="4">Identification</td><td>Flags</td><td colspan="3">Fragment Offset</td></tr>
<tr><td colspan="2">Time to Live</td><td colspan="2">Protocol</td><td colspan="4">Header Checksum</td></tr>
<tr><td colspan="8">Source Address (32-bit IPv4 address)</td></tr>
<tr><td colspan="8">Destination Address (32-bit IPv4 address)</td></tr>
<tr><td colspan="6">Options</td><td colspan="2">Padding</td></tr>
<tr><td colspan="8">Data (contains layer 4 segment)</td></tr>
</table>

- Version = 4
  If no options, IHL = 5
  Source and Destination are
  32-bit IPv4 addresses

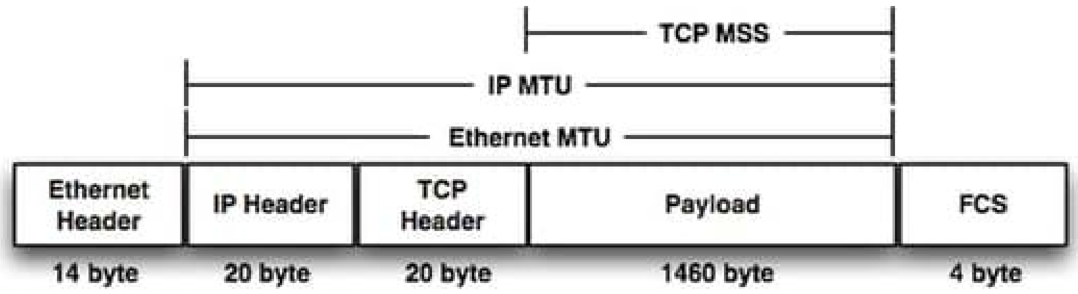
- Protocol = 6 means data portion contains a TCP segment. Protocol = 17 means UDP.



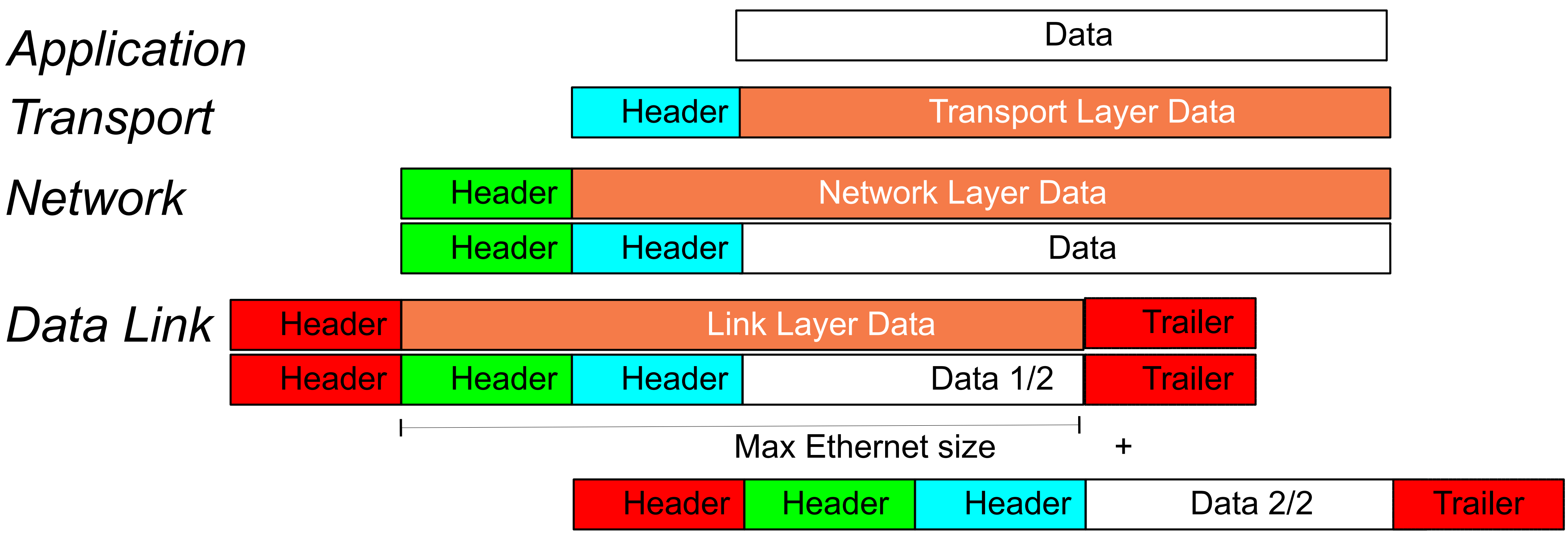
Fragmentation

- Supports dividing a large IP packet into fragments
- … in case a link cannot handle a large IP packet
  – Packet identifier (16b): datagram,
  – Flags (3b): Don't fragment, More fragments,
  – Fragment offset (13b): position in original datagram, measured in units of eight-byte blocks
- Maximum Transfer Unit (MTU)
  – Around 1500 on Ethernet
  – How to discover on a path? Try sizes,DF=1 + ICMP error

Fragmentation

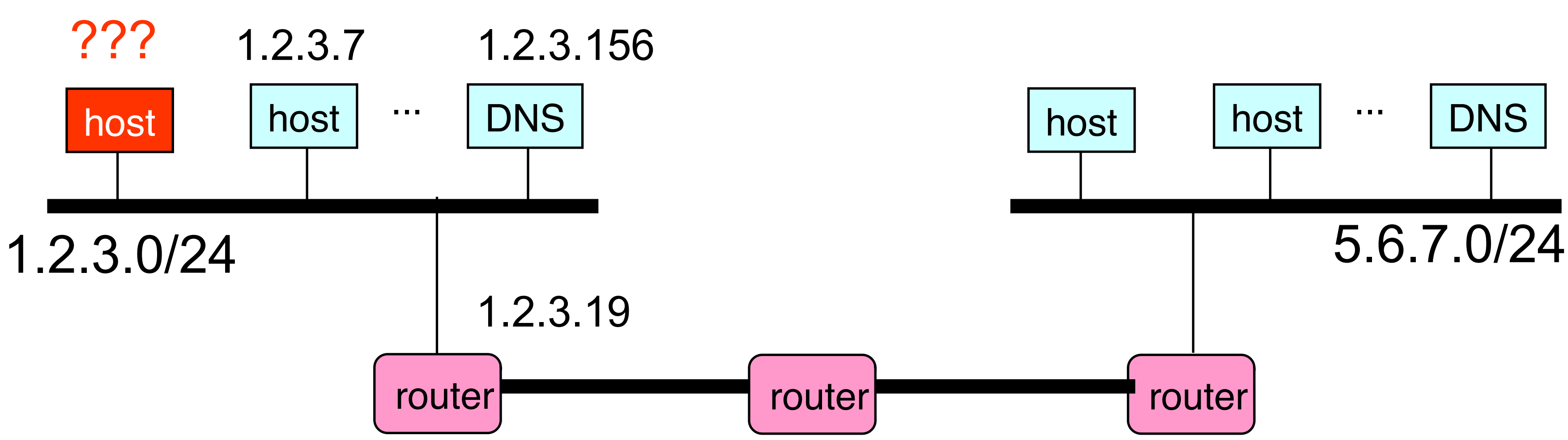


- Once fragmented, reassembled at destination



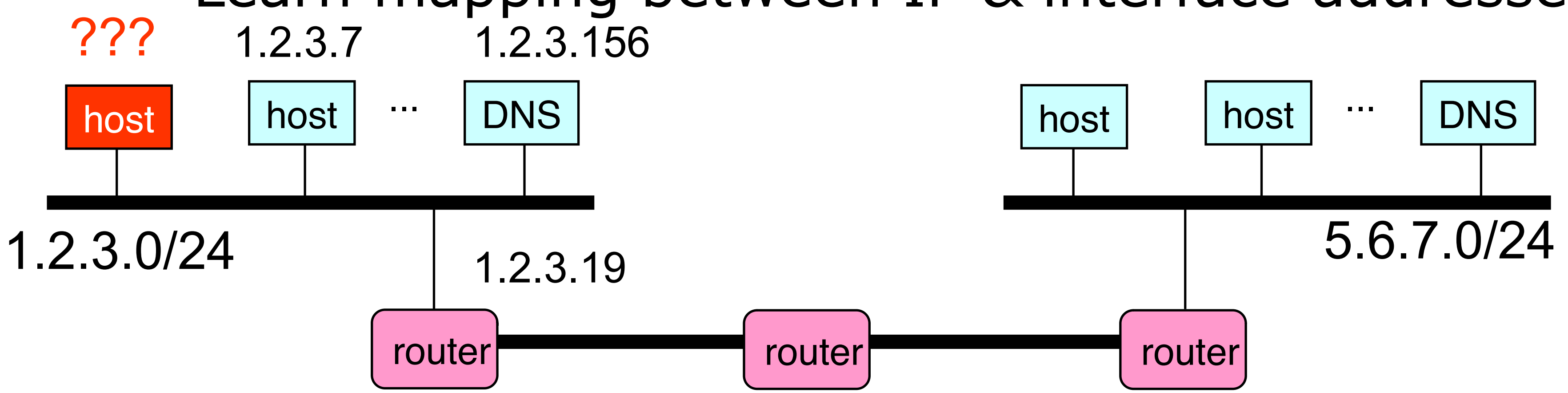
How To Bootstrap an End Host?

- What local DNS server to use?
- What IP address the host should use?
- How to send packets to remote destinations?



Avoiding Manual Configuration

- Dynamic Host Configuration Protocol (DHCP)
  - End host learns how to send packets
  - Learn IP address, DNS servers, and gateway
- Address Resolution Protocol (ARP)
  - Others learn how to send packets to the end host
  - Learn mapping between IP & interface addresses

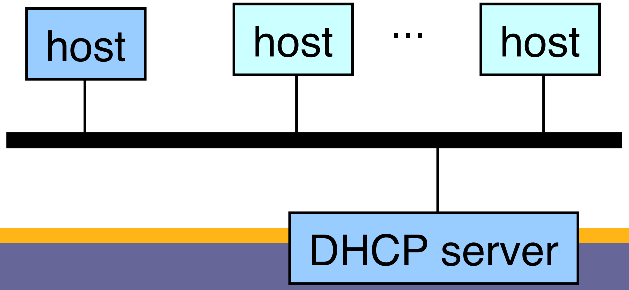


Key Ideas in Both Protocols

- **Broadcasting**: when in doubt, shout!
  - Broadcast query to all hosts in local-area-network
- **Caching**: remember the past for a while
  - Store the information you learn to reduce overhead
  - Remember your address & other host’s addresses
- **Soft state**: … but eventually forget the past
  - Associate a time-to-live field with the information
  - … and either refresh or discard the information

Bootstrapping Problem

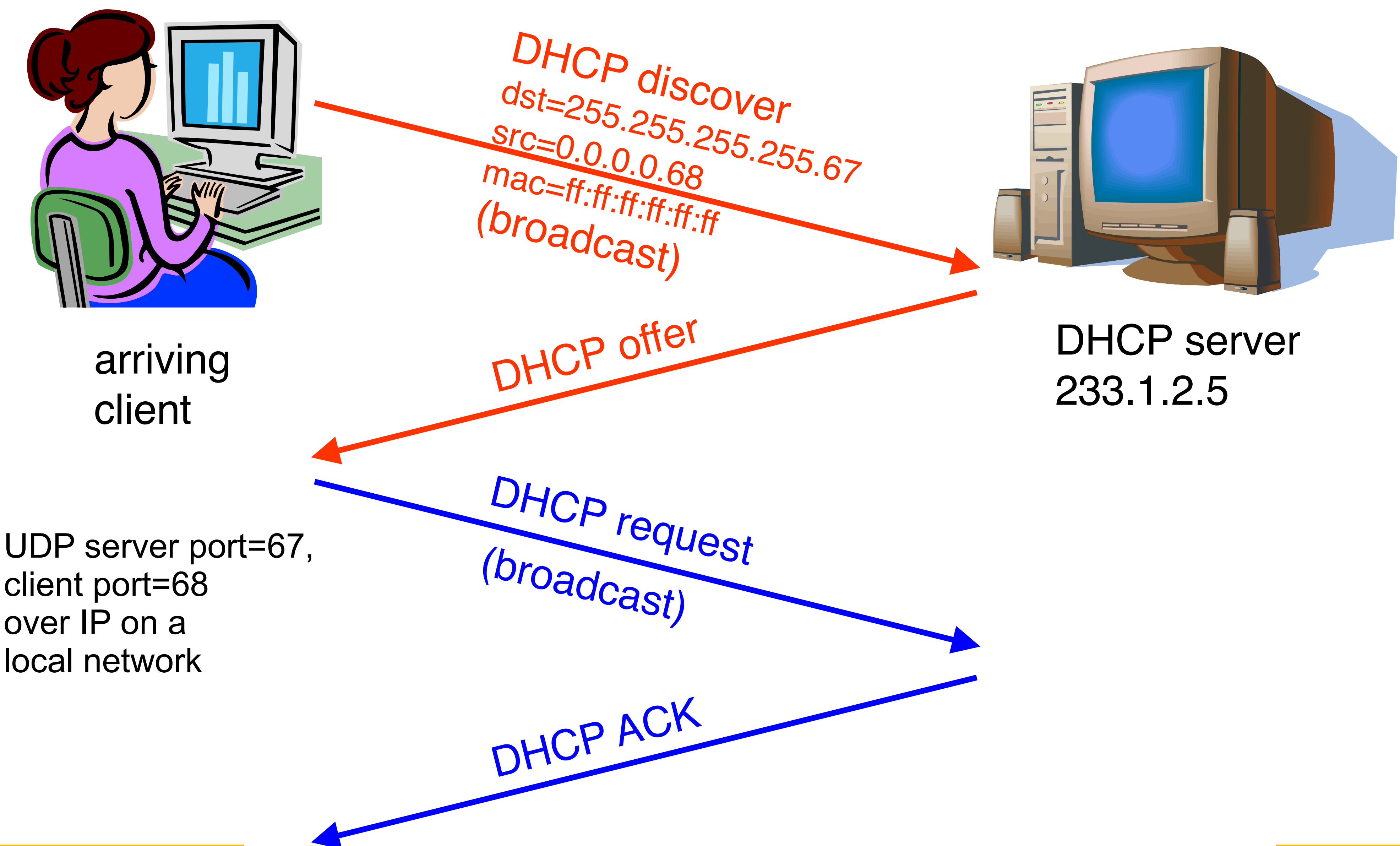
- Host doesn't have an IP address yet
  - So, host doesn't know what source to use
- Host doesn't know who to ask for an IP address
  - So, host doesn't know what destination to use
- Solution: discover a server who can help
  - Broadcast a DHCP server-discovery message
  - Server sends a DHCP "offer" offering an address



Response from the DHCP Server

- DHCP “offer message” from the server
  - Configuration parameters (proposed IP address, mask, gateway router, DNS server, ...)
  - Lease time (the time information remains valid)
- Multiple servers may respond with an offer
  - The client decides which offer to accept
    - Client sends a DHCP request echoing the parameters
  - The DHCP server responds with an ACK to confirm
    - + the other servers see they were not chosen

Dynamic Host Configuration Protocol



Deciding What IP Address to Offer

- Static allocation
  - All parameters are statically configured in the server
  - E.g., a dedicated IP address for each MAC address
  - Makes it easy to track a host over time
- Dynamic allocation
  - Server maintains a pool of available addresses
  - … and assigns them to hosts on demand
  - Enables more efficient use of the pool of addresses

Soft State: Refresh or Forget

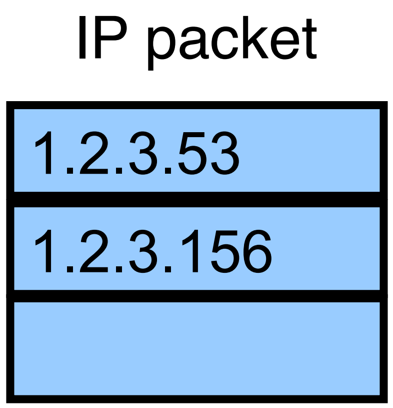
- Why is a lease time necessary?
  - Client can release the IP address (DHCP RELEASE)
    - E.g., "ipconfig /release" at the command line
    - E.g., clean shutdown of the computer
  - But, the host might not release the address
    - E.g., the host crashes (blue screen of death!), buggy client
  - Don't want the address to be allocated forever
- Performance trade-offs
  - Short lease: returns inactive addresses quickly
  - Long lease: avoids overhead of frequent renewals

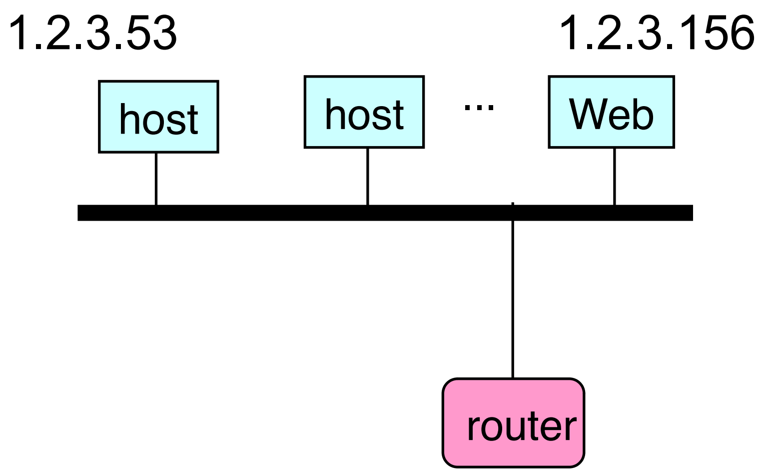
So, Now the Host Knows Things

- IP address
- Mask
- Gateway router
- DNS server

- And can send packets to other IP addresses
  - How to learn the MAC address of the destination?

Sending Packets Over a Link





- Adapters only understand MAC addresses
  - Translate the destination IP address to MAC address
  - Encapsulate the IP packet inside a link-level frame

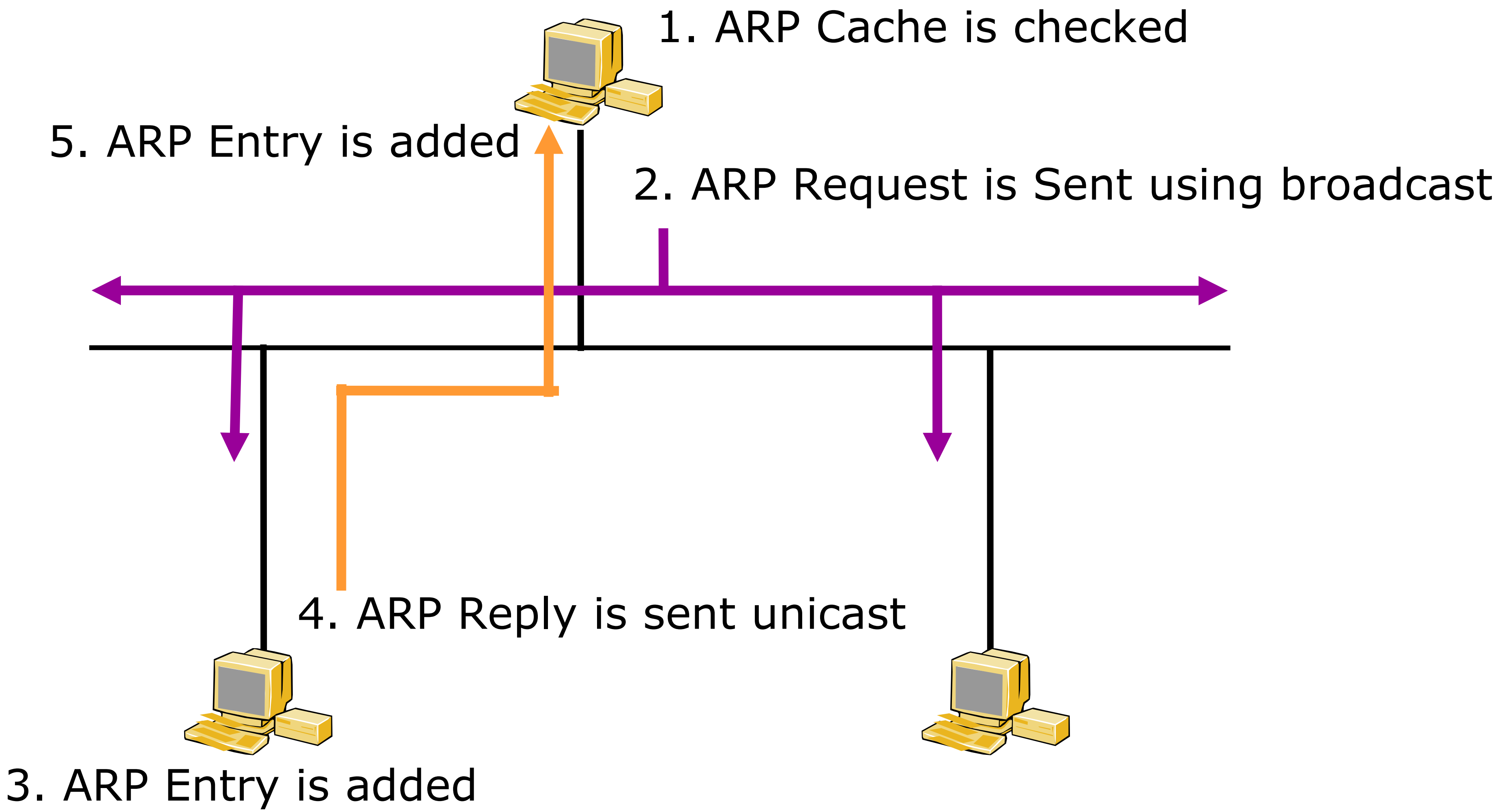
Address Resolution Protocol Table

- Every node maintains an ARP table
  - (IP address, MAC address) pair
- Consult the table when sending a packet
  - Map destination IP to destination MAC address
  - Encapsulate and transmit the data packet
- But, what if the IP address is not in the table?
  - Sender broadcasts: "Who has IP address 1.2.3.156?"
  - Receiver (owner) responds:
    "MAC address 58-23-D7-FA-20-B0"
  - Sender caches the result in its ARP table
  - Old cache entries removed by timeout

Types of ARP Messages

- ARP request
  - Who is IP addr X.X.X.X tell IP addr Y.Y.Y.Y
- ARP reply
  - IP addr X.X.X.X is Ethernet Address hh:hh:hh:hh:hh:hh

ARP Procedure



ARP Table

IP Address	Hardware Address	Age (Sec)
192.168.0.2	08-00-20-08-70-54	3
192.168.0.65	05-02-20-08-88-33	120
192.168.0.34	07-01-20-08-73-22	43

```
$ arp -n
Address          HWtype  HWaddress          Flags Mask   Iface
147.83.31.118    ether   00:21:28:f3:3b:8f  C            eth0
147.83.31.50     ether   52:54:00:1c:00:3b  C            eth0
```

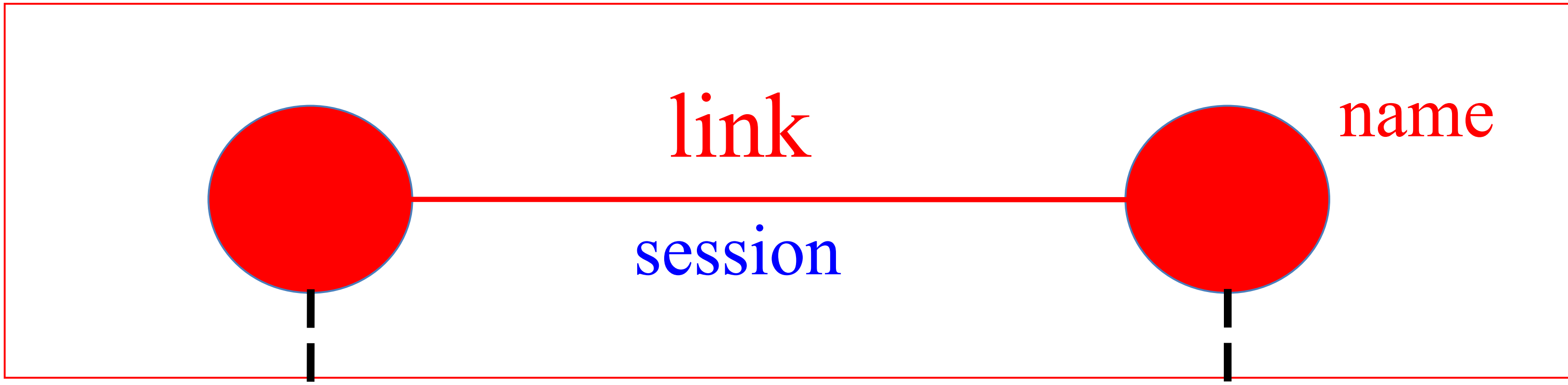
Gratuitous ARP

- ARP bcast request:
  Who is myIP addr, tell myIP addr?
  - Detect Conflict?
  - Update of other machines' ARP tables
  - Every time an interface goes up.

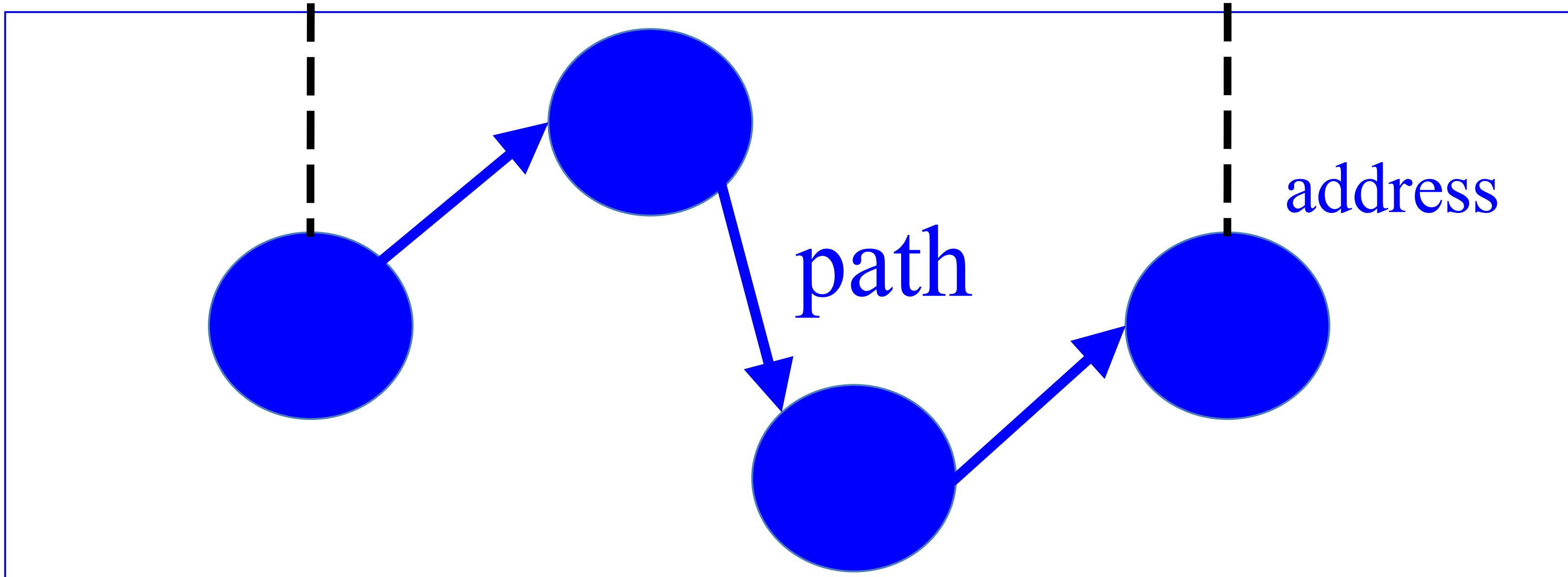
```
Ethernet II, Src: 02:02:02:02:02:02, Dst: ff:ff:ff:ff:ff:ff
    Destination: ff:ff:ff:ff:ff:ff (Broadcast)
    Source: 02:02:02:02:02:02 (02:02:02:02:02:02)
    Type: ARP (0x0806)
Address Resolution Protocol (request/gratuitous ARP)
    Hardware type: Ethernet (0x0001)
    Protocol type: IP (0x0800)
    Opcode: request (0x0001)
    Sender MAC address: 02:02:02:02:02:02 (02:02:02:02:02:02)
    Sender IP address: 192.168.1.1 (192.168.1.1)
    Target MAC address: ff:ff:ff:ff:ff:ff (Broadcast)
    Target IP address: 192.168.1.1 (192.168.1.1)
```

Routing

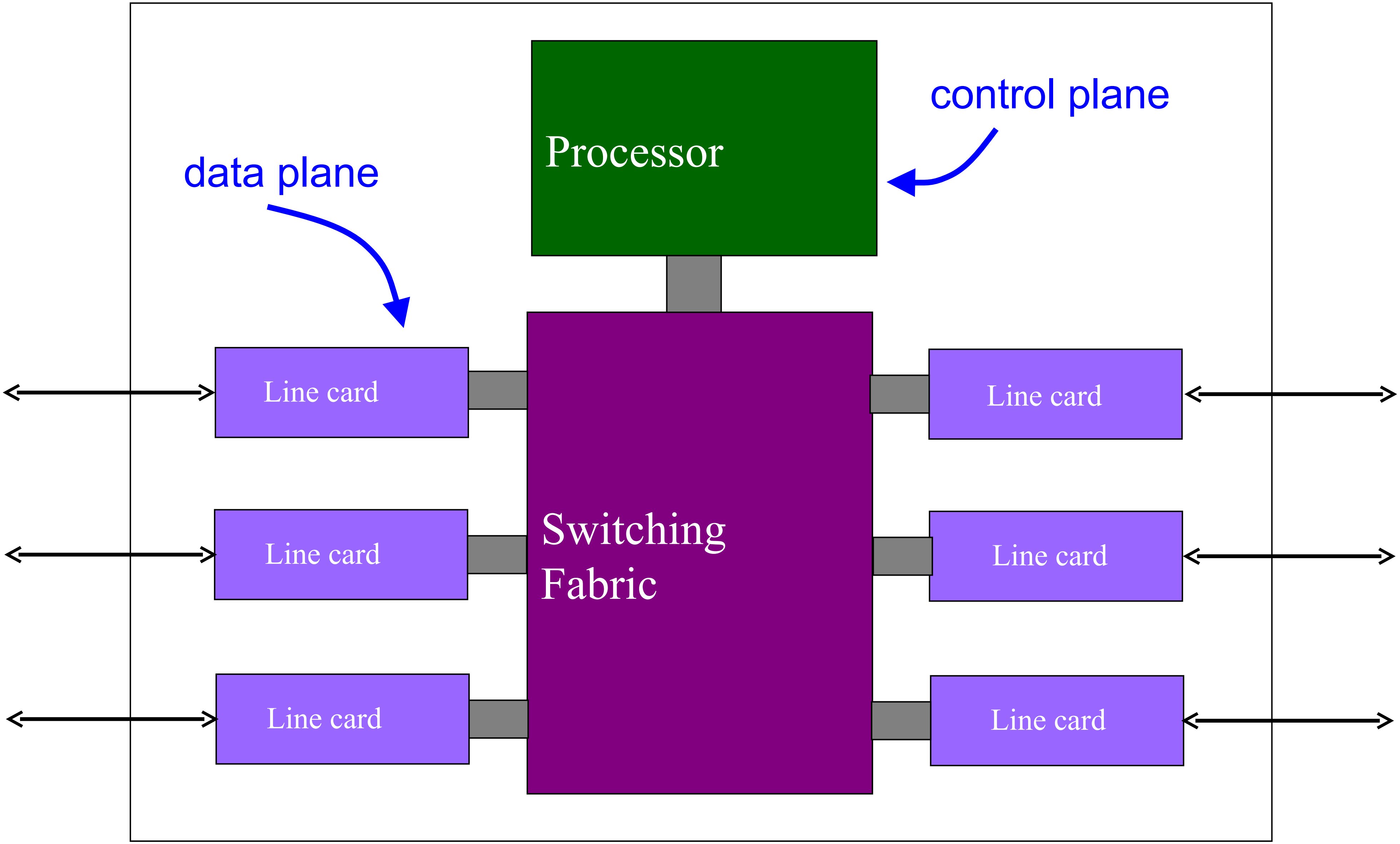
Routing: Mapping Link to Path

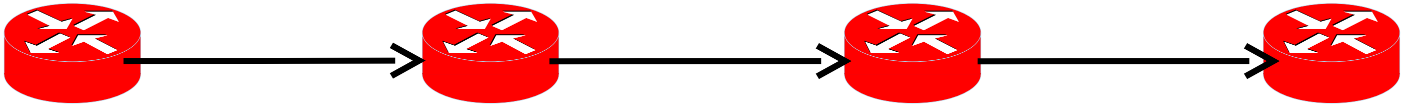


- Configuring routing tables at each node



Data and Control Planes





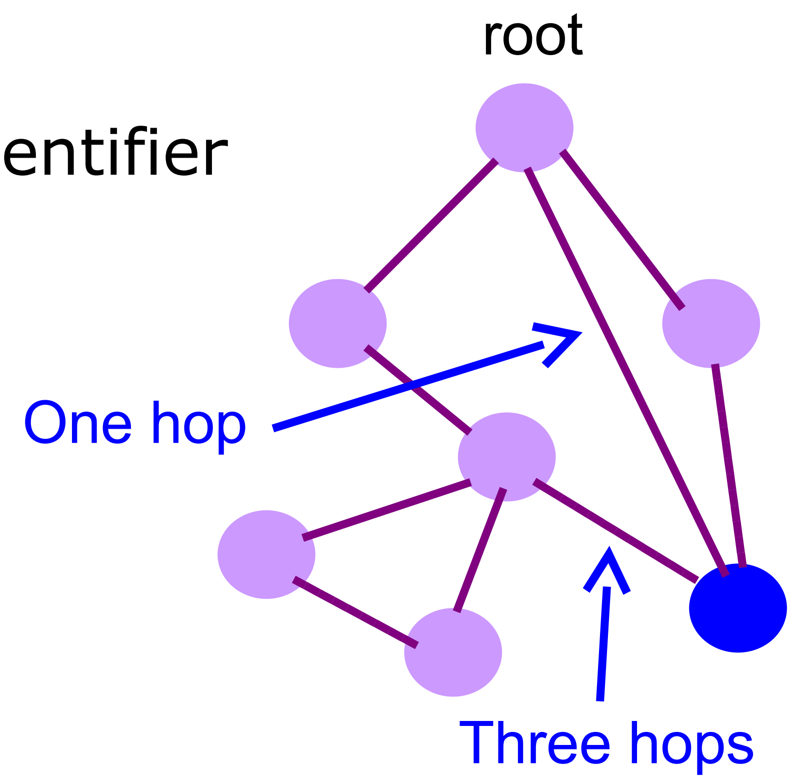
Routing vs. Forwarding

- Routing: control plane
  - Computing paths the packets will follow
  - Routers talking amongst themselves
  - Creating the forwarding tables
- Forwarding: data plane
  - Directing a data packet to an outgoing link
  - Using the forwarding tables

How to Compute Paths?

Spanning Tree Algorithm

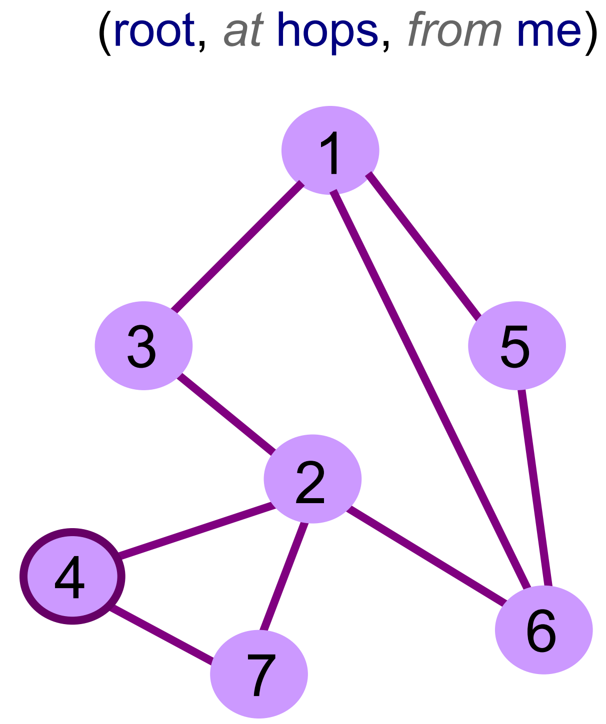
- Elect a root
  - The switch with the smallest identifier
  - And form a tree from there
- Algorithm
  - Repeatedly talk to neighbors
    - "I think node Y is the root"
    - "My distance from Y is d"
  - Update based on neighbors
    - Smaller id as the root
    - Smaller distance d+1



Used in Ethernet LANs

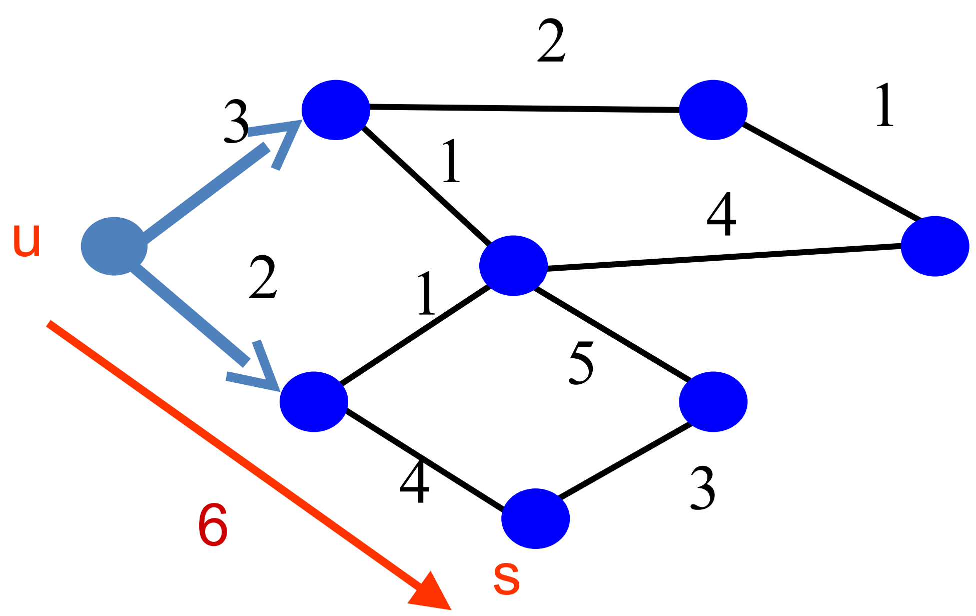
Spanning Tree Example: Switch #4

- Switch #4 thinks it is the root
  - Sends (4, 0, 4) message to 2 and 7
- Switch #4 hears from #2
  - Receives (2, 0, 2) message from 2
  - … and thinks that #2 is the root
  - And realizes it is just one hop away
- Switch #4 hears from #7
  - Receives (2, 1, 7) from 7
  - But, this is a longer path, so 4 prefers 4-2 over 4-7-2
  - And removes 4-7 link from the tree



Shortest-Path Problem

- Compute: *path costs* to all nodes
  - From a given source u to all other nodes
  - Cost of the path through each outgoing link
  - Next hop along the least-cost path to s



Link State: Dijkstra's Algorithm

- Flood the topology information to all nodes
- Each node computes shortest paths to other nodes

Initialization

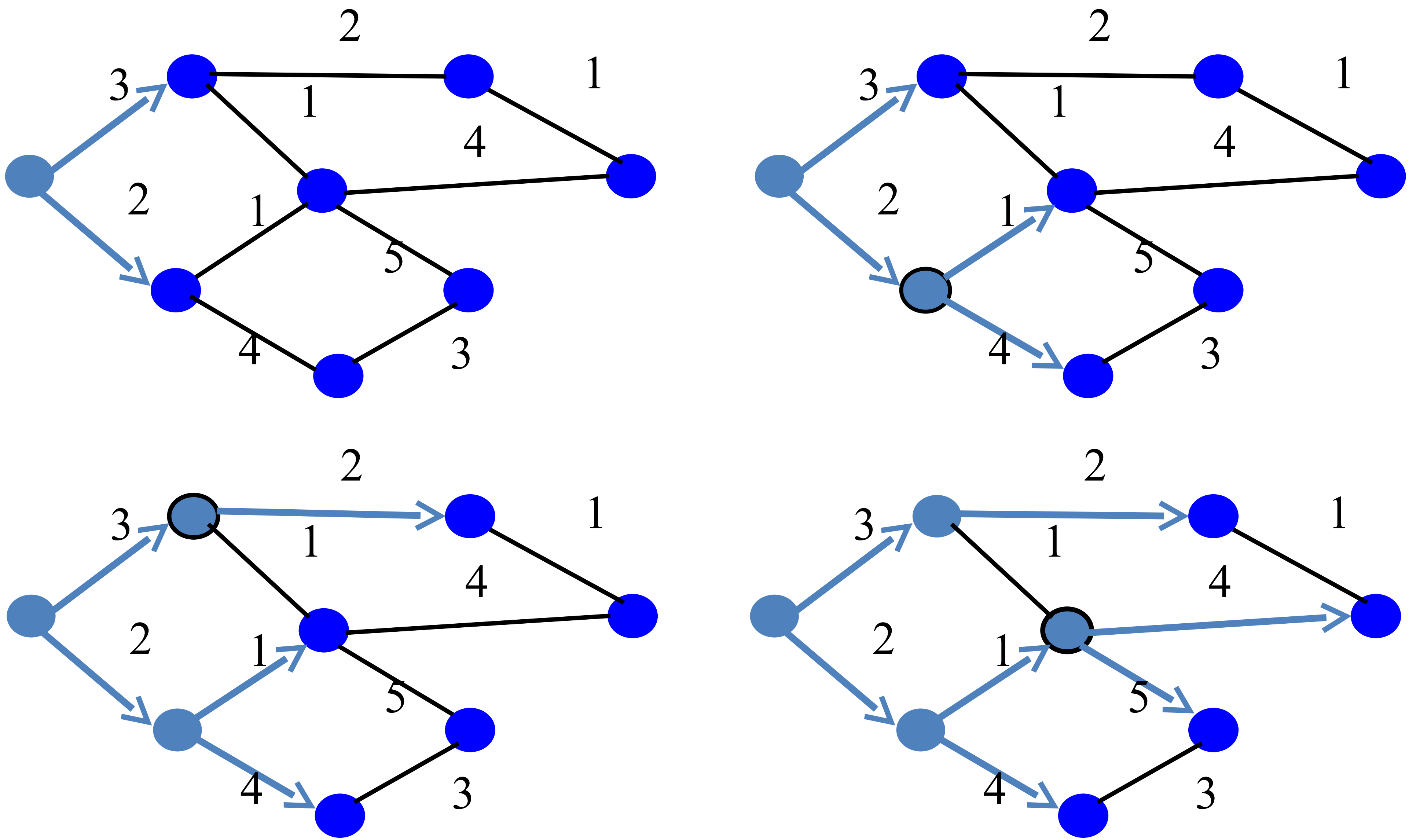
```
S = {u}
for all nodes v
  if (v is adjacent to u)
    D(v) = c(u,v)
  else D(v) = ∞
```

Loop

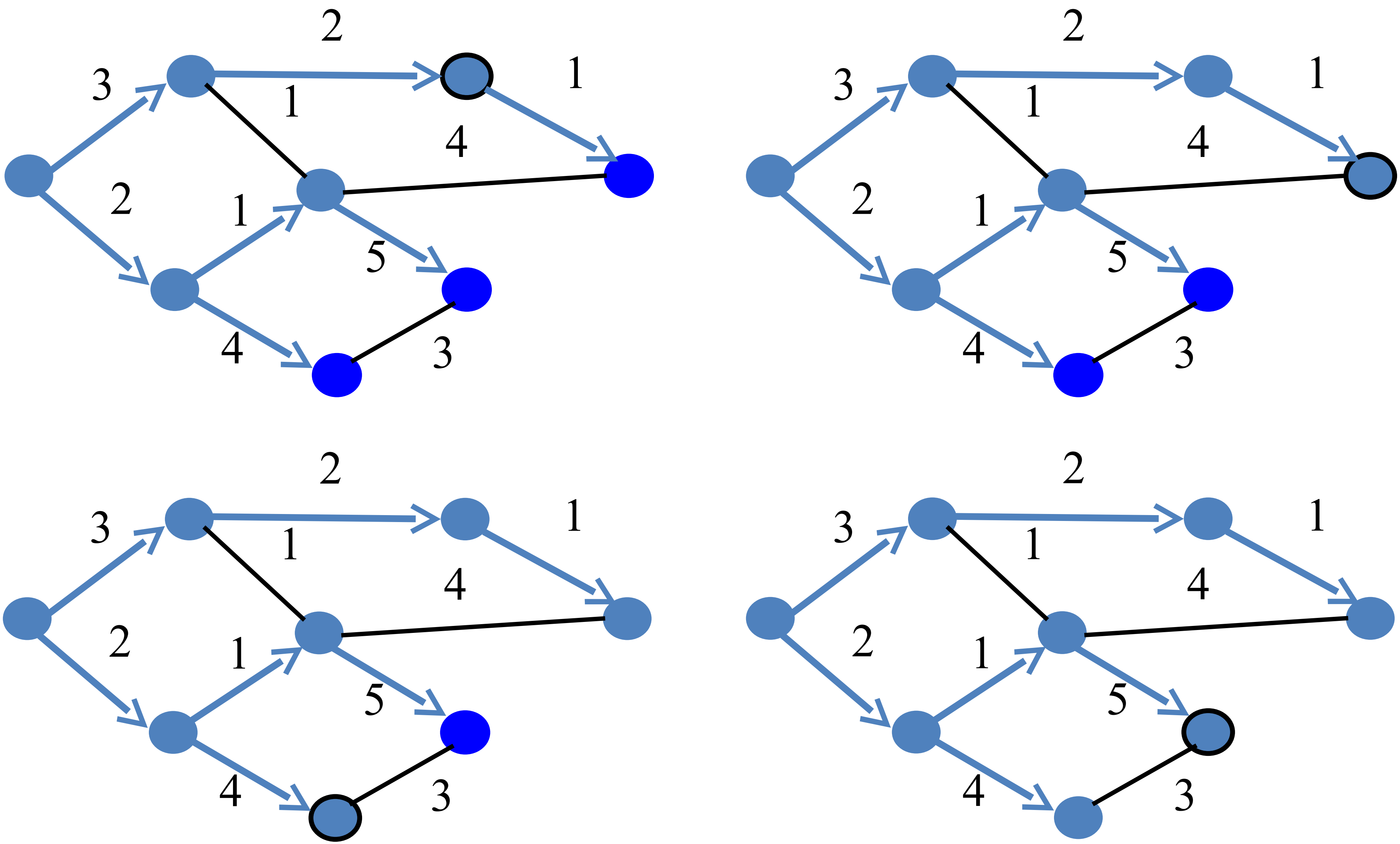
```
add w with smallest D(w) to S
update D(v) for all adjacent v:
  D(v) = min{D(v), D(w) + c(w,v)}
until all nodes are in S
```

Used in OSPF and IS-IS

Link-State Routing Example

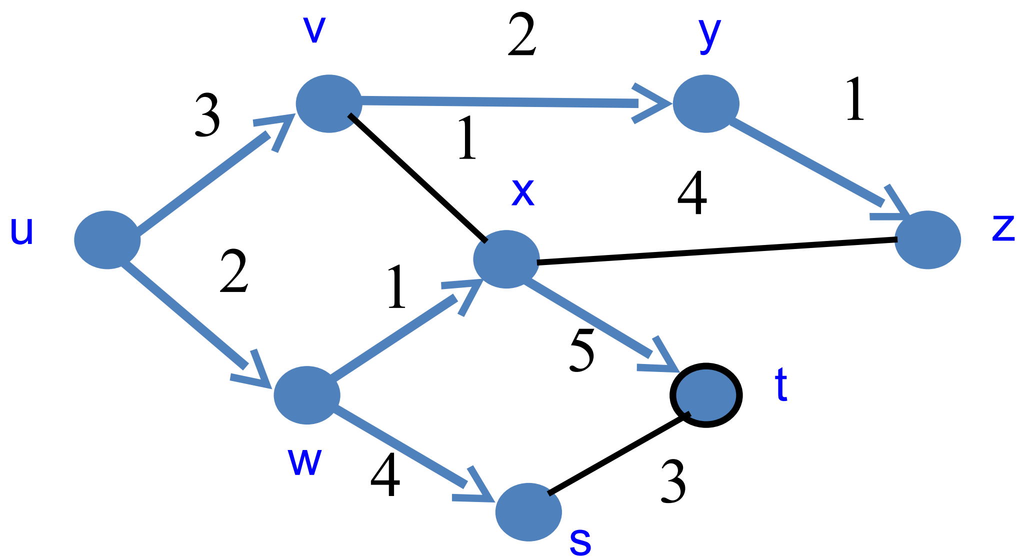


Link-State Routing Example (cont.)



Link State: Shortest-Path Tree

Shortest-path tree from u

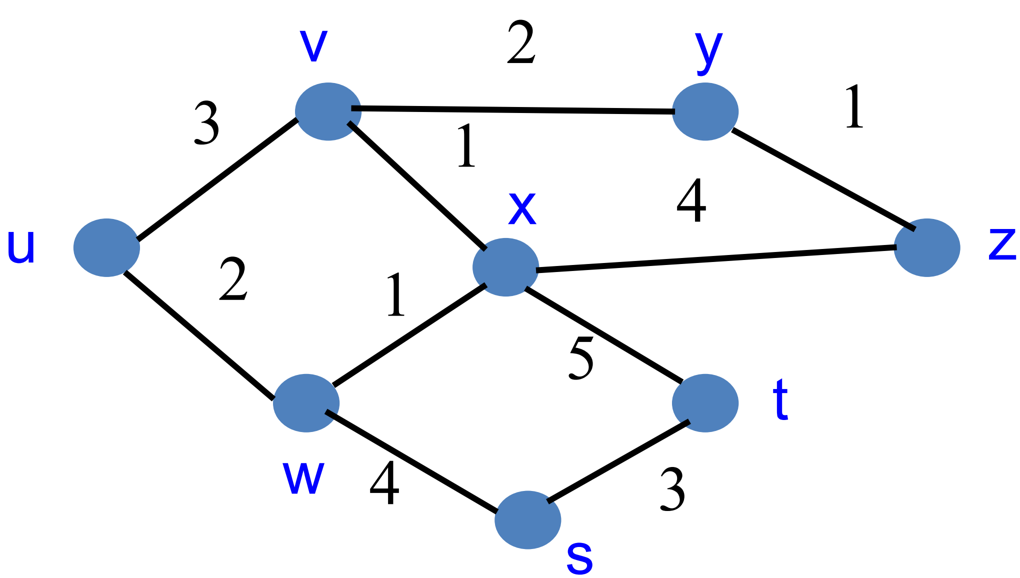


Forwarding table at u

	link
v	(u,v)
w	(u,w)
x	(u,w)
y	(u,v)
z	(u,v)
s	(u,w)
t	(u,w)

Distance Vector: Bellman-Ford Alg

- Define distances at each node x
  - $d_x(y)$ = cost of least-cost path from x to y
- Update distances based on neighbors
  - $d_x(y) = \min \{c(x,v) + d_v(y)\}$ over all neighbors v

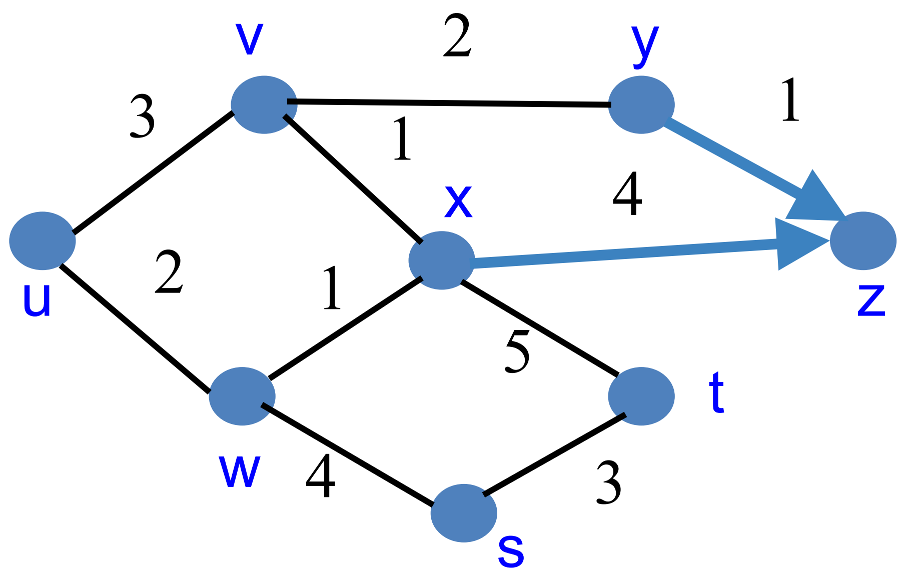


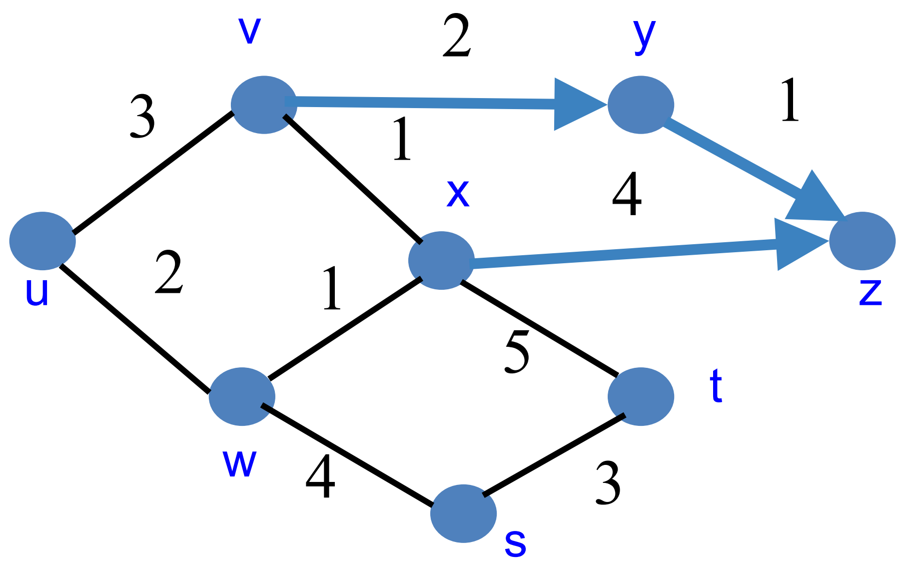
$$d_u(z) = \min\{c(u,v) + d_v(z),\ c(u,w) + d_w(z)\}$$

Used in RIP and EIGRP

Distance Vector Example

To z:



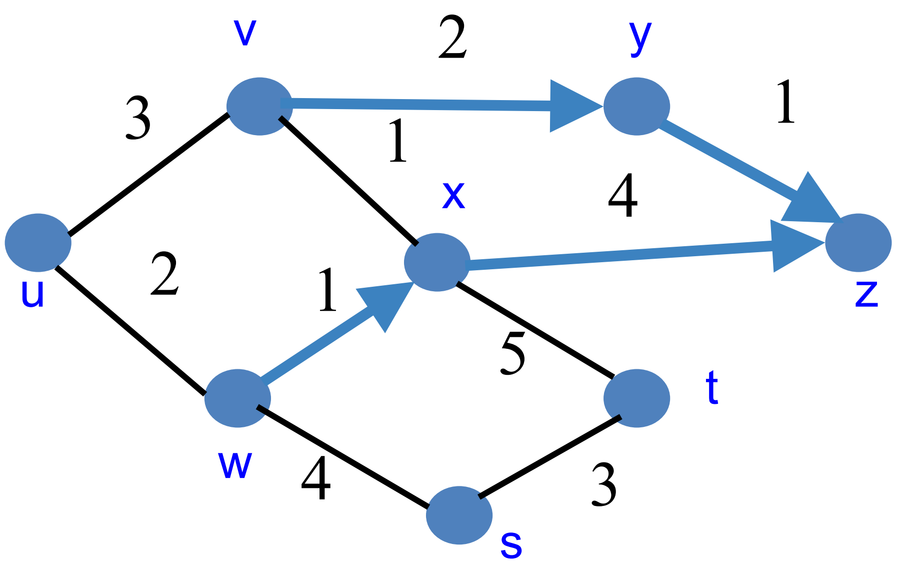


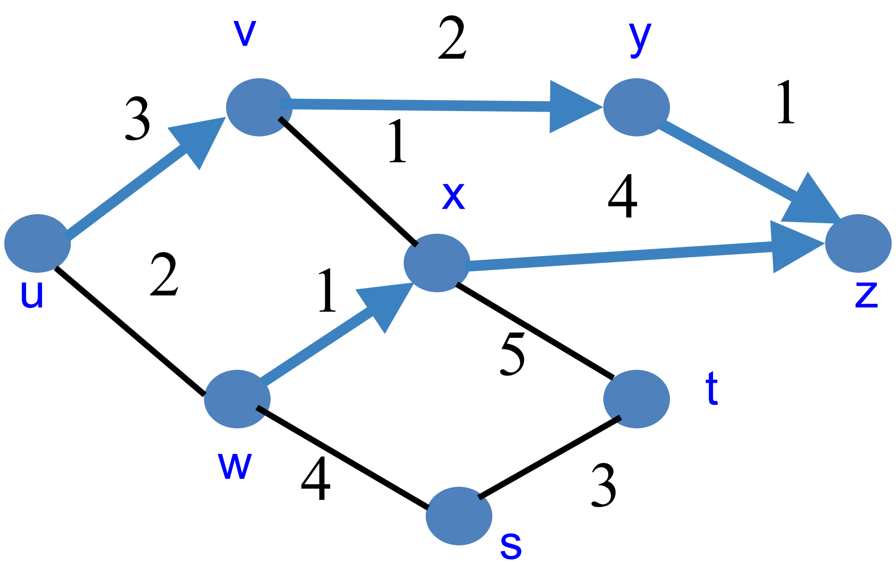
$d_y(z)=1$

$d_x(z)=4$

$d_v(z)= \min\{2+d_y(z),$
$1+d_x(z)\}$
$= 3$

Distance Vector Example (Cont.)





$d_w(z) = \min\{1+d_x(z),$
$4+d_s(z),$
$2+d_u(z)\}$
$= 5$

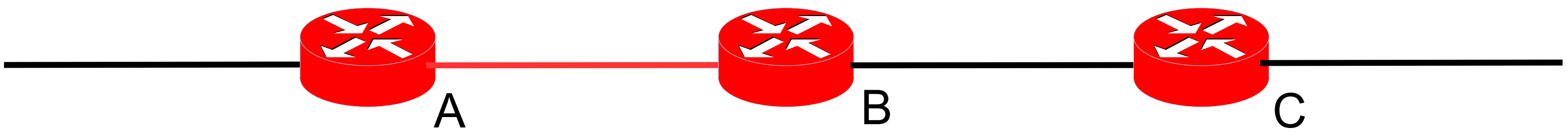
$d_u(z) = \min\{3+d_v(z),$
$2+d_w(z)\}$
$= 6$

Comparison

- Link state
  - All routers know (all paths, all routers) of the net
  - Link state info flooded → all routers have a consistent copy of the link-state database
  - Each router constructs its own relative shortest-path tree, itself as root, for all routes
- Distance vector
  - Involves: distance (metric) to destination + vector (direction) to get there
  - Routing info only exchanged among direct neighb
  - Short-sighted: ignores where neighb learned route

Routing Information Protocol (RIP)

- Distance Vector protocol
  - Number of hops, $\infty = 16$
  - UDP, port 520
  - Routers send RIP updates every 30 secs
  - Neighbor router is down: 180 secs
- RIP versions:
  - RIPv1 (1988) no subnet info, no auth, bcast table
  - RIPv2 (1993-1998), mcast table to 224.0.0.9
  - RIPng (1997), IPv6



Convergence: Count to infinity

- A-B fails, B knows, C still says A is 2 hops away from C (through B), B believes B-C-?-A
- Split horizon:
  - Prohibiting a router from advertising a route back onto the interface from which it was learned
  - Poisoned reverse: adding entries with ∞ cost
  - Send updates when change (before 30 sec timer)
  - Hold down timer (CISCO), 280 secs w/o updates

Open Shortest Path First (OSPF)

- Link State protocol

  Routers monitor neighbour routers and nets

  When any change, Link State Advertisement sent to all routers (IP, mcast 224.0.0.5)

  Routers maintain LS database with LSAs
- Metric can include link bitrate, delay, etc
- No convergence (count to infinity) problems
- Interior Gateway Protocol

About the Internet

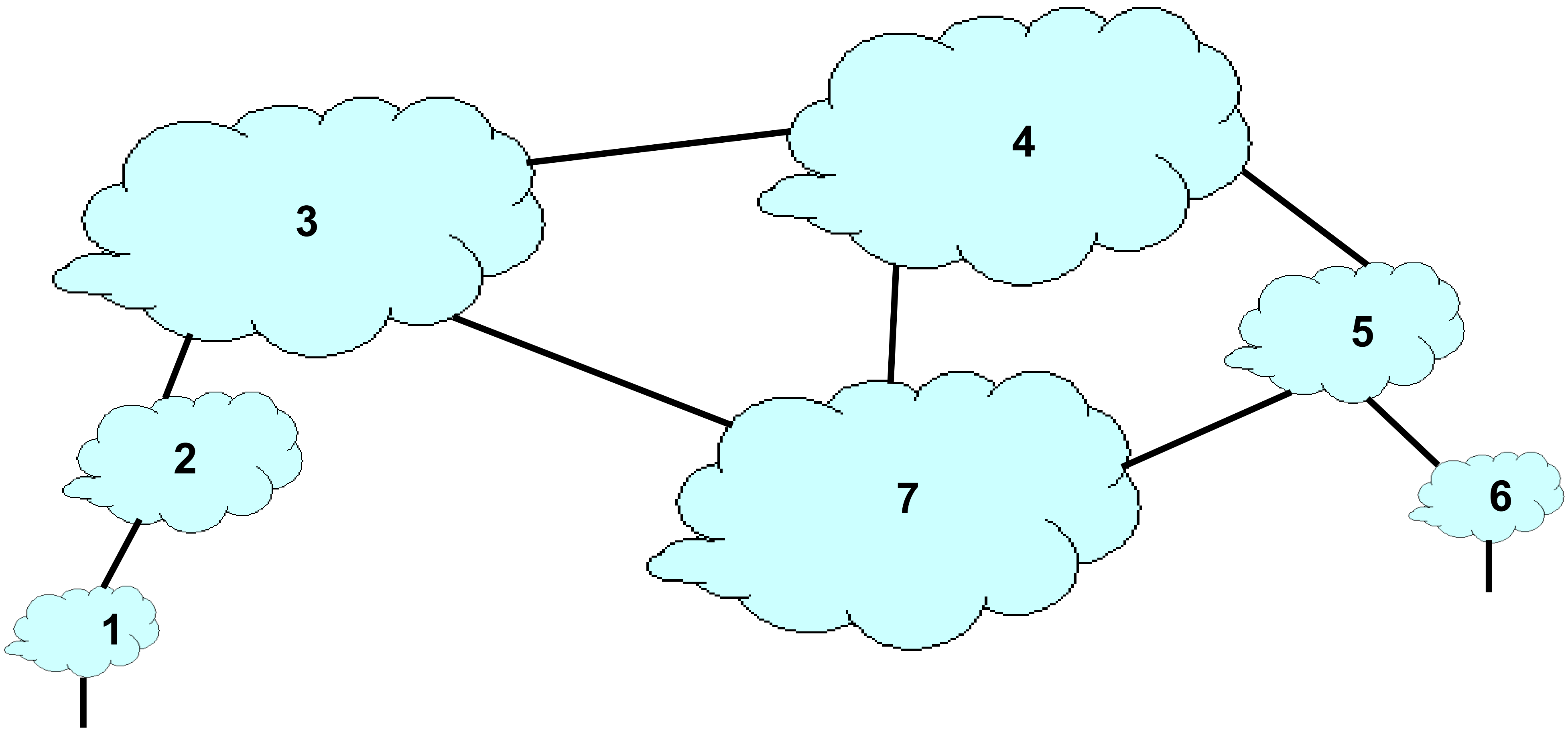
- Internet's two-level topology
  - Autonomous Systems + connections between them
  - Routers + links between them
- AS-level topology
  - Autonomous System (AS) numbers
  - Business relationships between ASs
  - Tier-1 providers
- Routing:
  - Interior Gateway Protocols: RIP, OSPF, CISCO IGRP
  - Exterior: Among AS: BGPv4

Internet Routing Architecture

- Divided into Autonomous Systems
  - Distinct regions of administrative control
  - Routers/links managed by a single “institution”
  - IP prefixes w/ single routing policy
  - Service provider, company, university, …
- Hierarchy of Autonomous Systems
  - Tier-1 providers with nation/continental wide backbone
  - Medium-sized regional provider with smaller backbone
  - Small network run by a single company or university
- Interaction between Autonomous Systems
  - Internal topology is not shared between ASes
  - … but, neighboring ASes interact to coordinate routing

AS Topology

- Node: Autonomous System
- Edge: Two ASes that connect to each other



Autonomous System Numbers

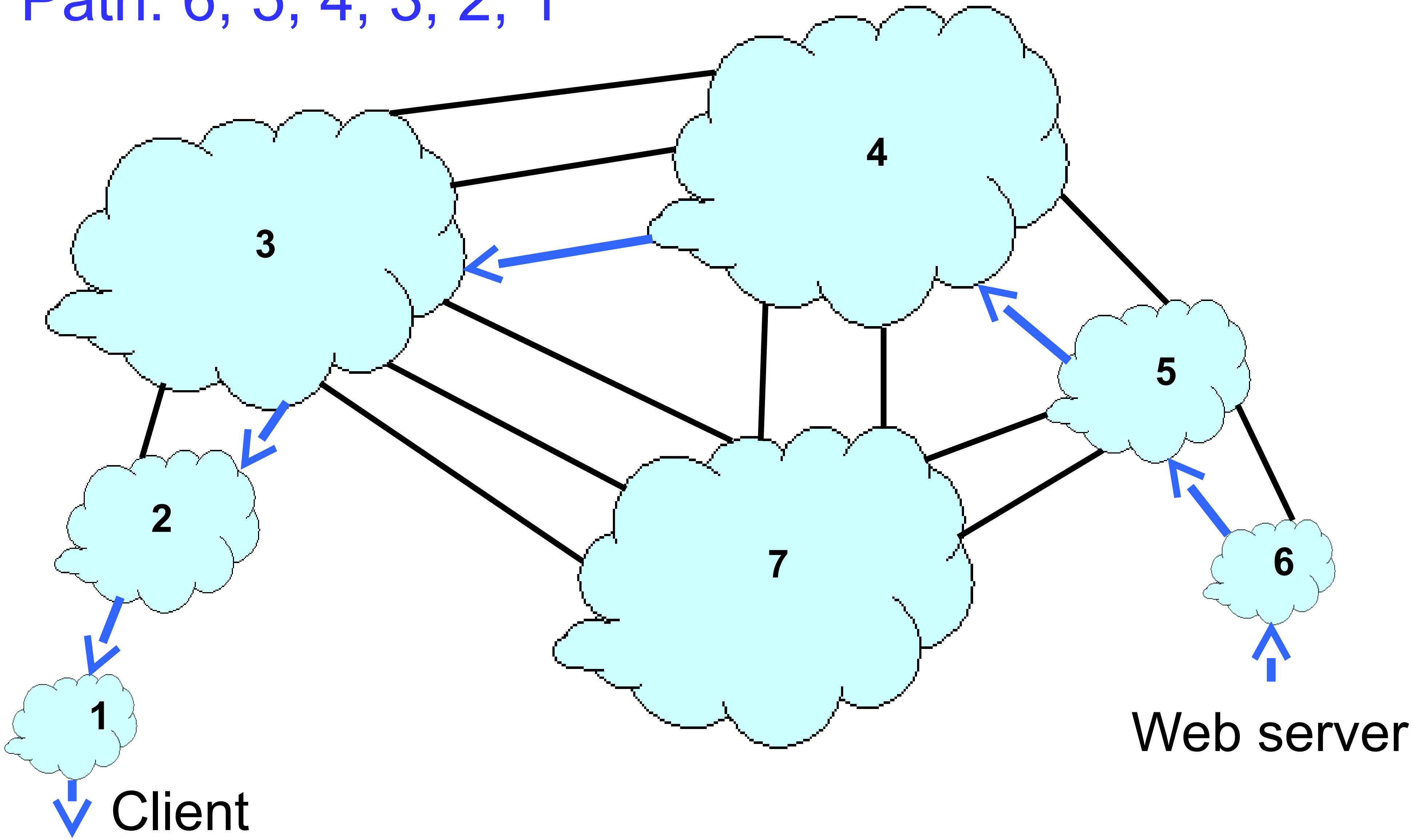
- AS Numbers are 16 or 32 bit values.
  - AS1 LVLT-1 - Level 3
  - AS2 University of Delaware
  - AS3 MIT

    …
  - AS766 RedIRIS
  - AS3352 Telefonica
  - AS6752 Andorra Telecom

    …
  - AS13041 CSUC.CAT

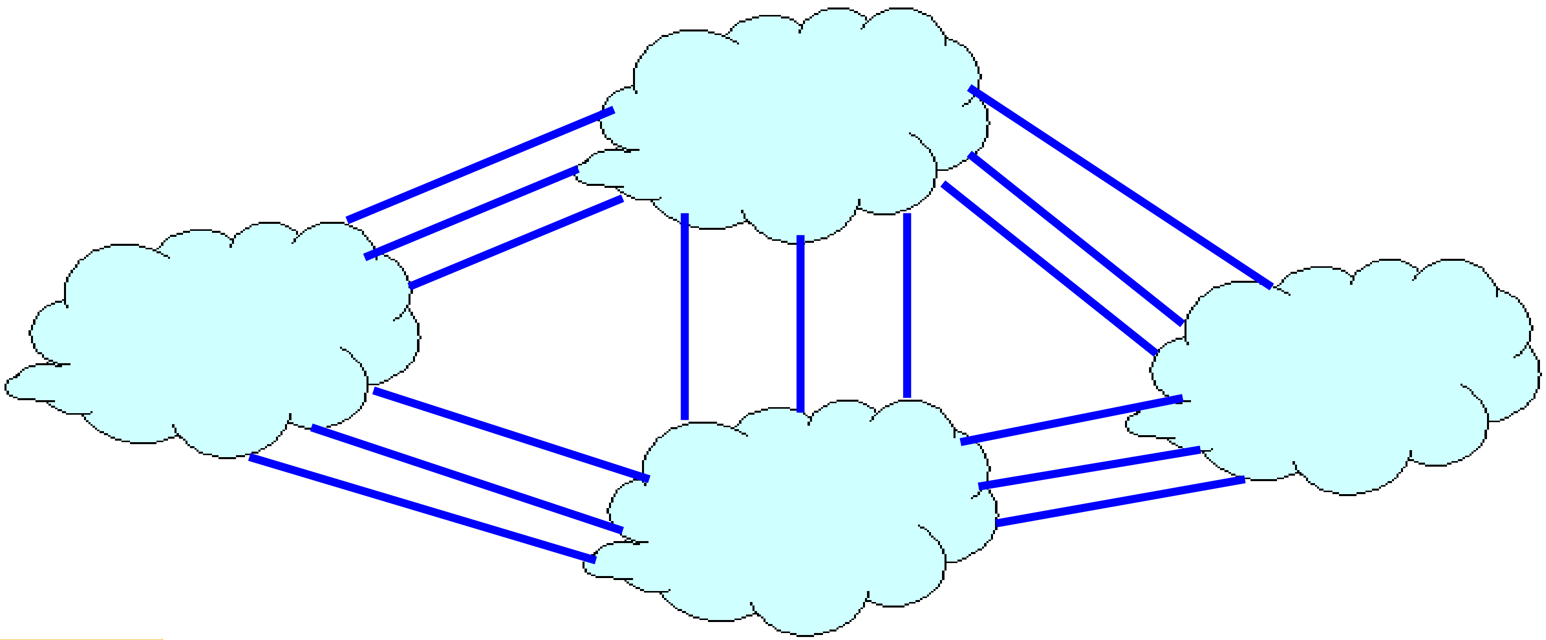
Interdomain Paths

Path: 6, 5, 4, 3, 2, 1

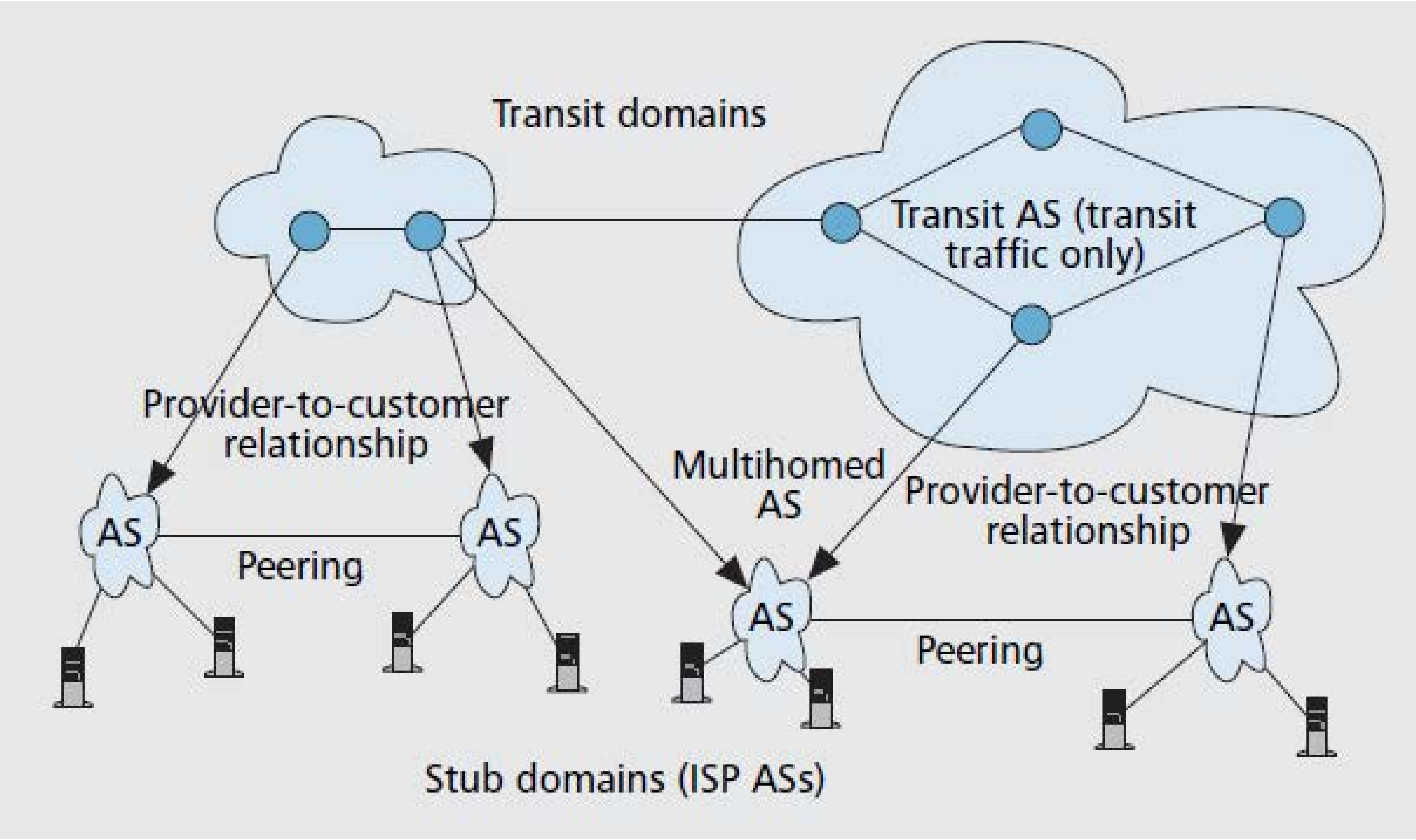


AS Structure: Tier-1 Providers

- Tier-1 provider
  - Has no upstream provider of its own
  - Typically has a national or international backbone
  - UUNET, Sprint, AT&T, Level 3, …
- Top of the Internet hierarchy of 12-20 ASes
  - Full peer-peer connections between tier-1 providers



The structure of the Internet



Middleboxes

Internet Ideal: simple network model

- Globally unique identifiers
  - Each node has a unique, fixed IP address
  - … reachable from everyone and everywhere
- Simple packet forwarding
  - Network nodes simply forward packets
  - … rather than modifying or filtering them

Internet Reality

- Host mobility
  - Host changing address as it moves
- IP address depletion
  - Multiple hosts using the same address
- Security concerns
  - Detecting and blocking unwanted traffic
- Replicated services
  - Load balancing over server replicas
- Performance concerns
  - Allocating bandwidth, caching content, …
- Incremental deployment
  - New technology deployed in stages

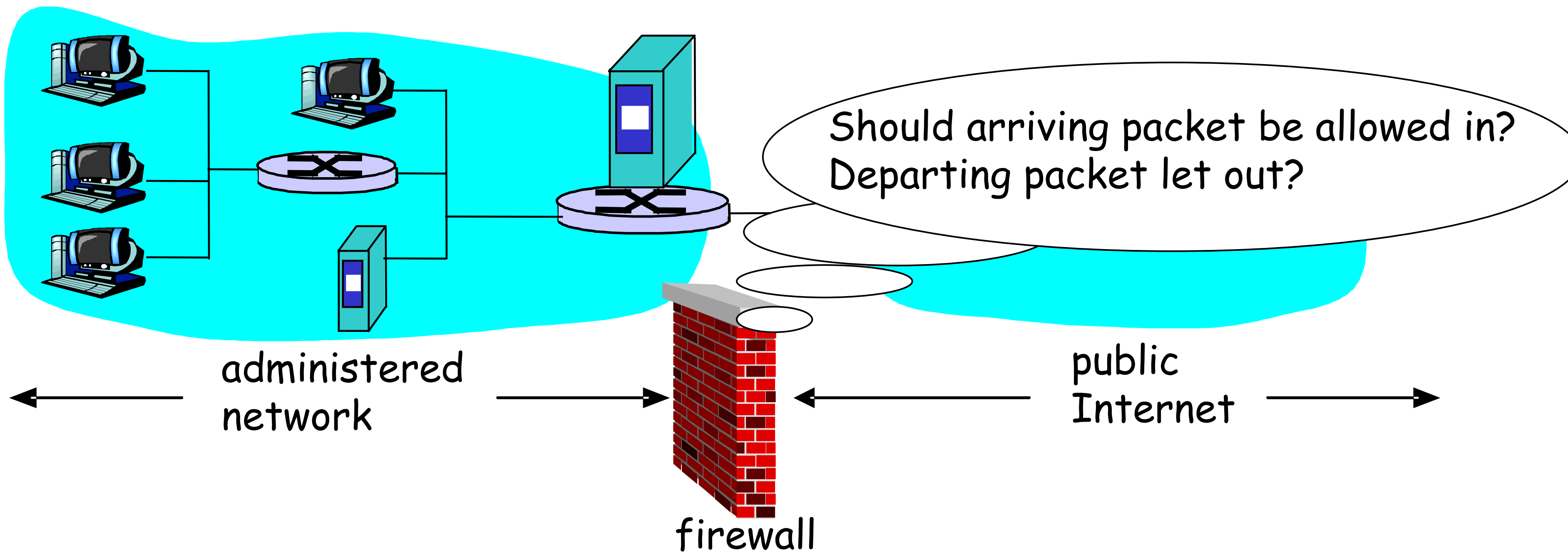
Middleboxes

- Middleboxes are intermediaries
  - Interposed between communicating hosts
  - Often without knowledge of one or both parties
- Myriad uses
  - Address translators
  - Firewalls
  - Traffic shapers
  - Intrusion detection
  - Transparent proxies
  - Application accelerators

"An abomination!"
- Violation of layering
- Hard to reason about
- Responsible for subtle bugs

"A practical necessity!"
- Solve real/pressing problems
- Needs not likely to go away

Firewalls



- Firewall filters packet-by-packet, based on:
  - Source & destination IP addresses, port numbers
  - TCP SYN and ACK bits; ICMP message type
  - Deep packet inspection on packet contents (DPI)

Packet Filtering Examples

- Block all packets with IP protocol field = 17 and with either source or dest port = 23.
  - All incoming and outgoing UDP flows blocked
  - All Telnet connections are blocked
- Block inbound TCP packets with SYN but no ACK
  - Prevents external clients from making TCP connections with internal clients
  - But allows internal clients to connect to outside
- Block all packets with TCP port of Quake

Firewall Configuration

- Firewall applies a set of rules to each packet
  - To decide whether to permit or deny the packet
- Each rule is a test on the packet
  - Comparing IP and TCP/UDP header fields
  - … and deciding whether to permit or deny
- Order matters
  - Once packet matches a rule, the decision is done

Firewall Configuration Example

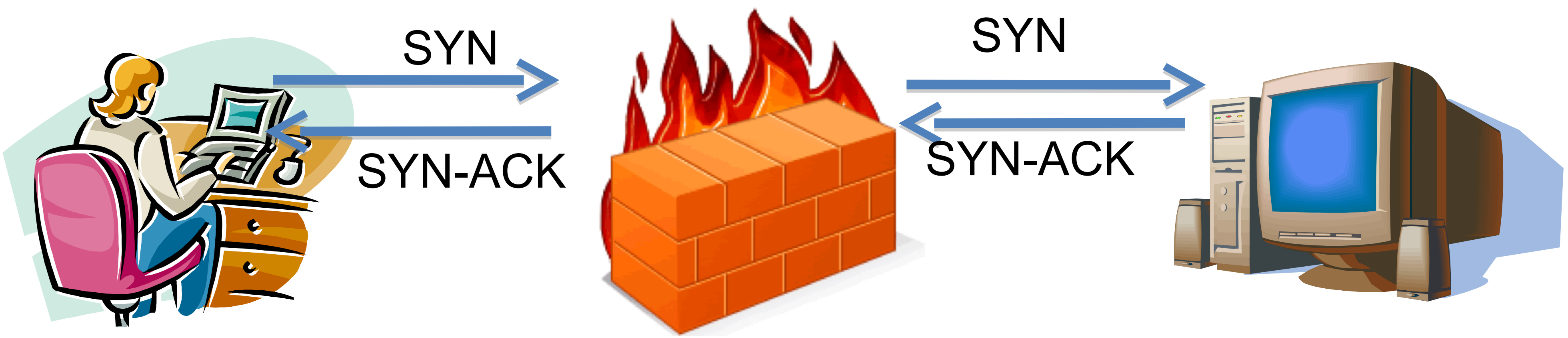
- Alice runs a network in 222.22.0.0/16
- Wants to let Bob's school access certain hosts
  - Bob is on 111.11.0.0/16
  - Alice's special hosts on 222.22.22.0/24
- Alice doesn't trust Trudy, inside Bob's network
  - Trudy is on 111.11.11.0/24
- Alice doesn't want any other Internet traffic

Firewall Configuration Rules

- #1: Don’t let Trudy’s machines in
  - Deny (src = 111.11.11.0/24, dst = 222.22.0.0/16)
- #2: Let rest of Bob’s network in to special dsts
  - Permit (src=111.11.0.0/16, dst = 222.22.22.0/24)
- #3: Block the rest of the world
  - Deny (src = 0.0.0.0/0, dst = 0.0.0.0/0)

Stateful Firewall

- Stateless firewall:
  - Treats each packet independently
- Stateful firewall
  - Remembers connection-level information
  - E.g., client initiating connection with a server
  - … allows the server to send return traffic



A Variation: Traffic Management

- Permit vs. deny is too binary a decision
  - Classify the traffic based on rules
  - … and handle each class differently
- Traffic shaping (rate limiting)
  - Limit the amount of bandwidth for certain traffic
- Separate queues
  - Use rules to group related packets
  - And then do weighted fair scheduling across groups

Clever Users Subvert Firewalls

- Example: filtering dorm access to a server
  - Firewall rule based on IP addresses of dorms
  - … and the server IP address and port number
  - Problem: users may log in to another machine
- Example: filtering P2P based on port #s
  - Firewall rule based on TCP/UDP port numbers
    - E.g., allow only port 80 (e.g., Web) traffic
  - Problem: software using non-traditional ports
    - E.g., write P2P client to use port 80 instead

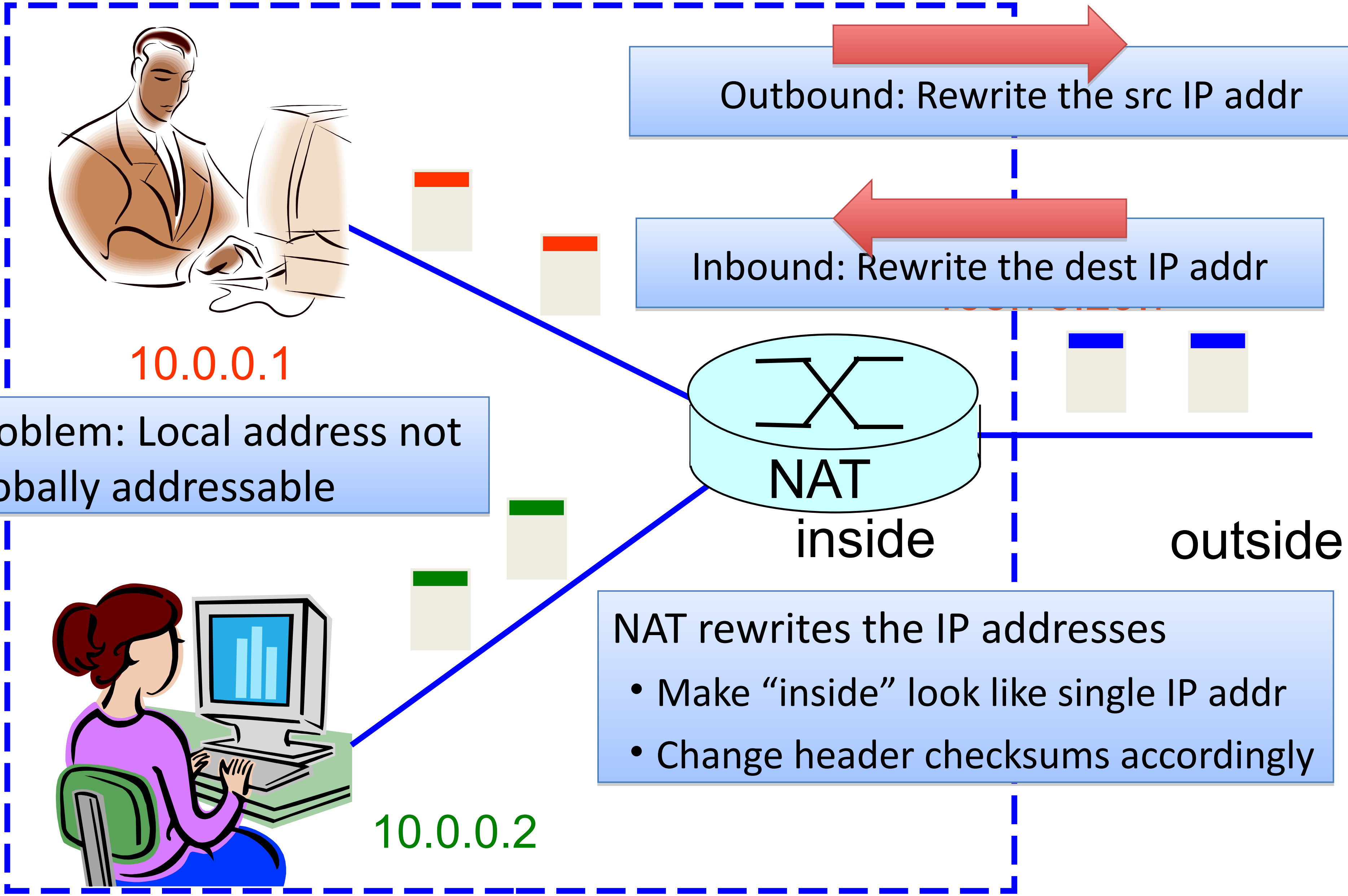
Address Translation

- Scenario:
  - Growth: scarcity of IPv4 addresses,
  - Security: protecting internal network,
  - Administration: isolating changes ISP addressing
- Solution:
  - External net (1, pool), router, internal net
  - Mapping addresses (NAT), and ports (PAT)
  - Static or dynamic
- Patterns:
  - Initiated internally, externally; dyn/static

Network Address Translation

- IP address space depletion
  - Clear in early 90s that 2^{32} addresses not enough
  - Work began on a successor to IPv4 but ...
- In the meantime…
  - Share addresses among numerous devices
  - … without requiring changes to existing hosts
- Meant as a short-term remedy
  - NAT is widely deployed, much more than IPv6

Network Address Translation



Outbound: Rewrite the src IP addr

Inbound: Rewrite the dest IP addr

Problem: Local address not globally addressable

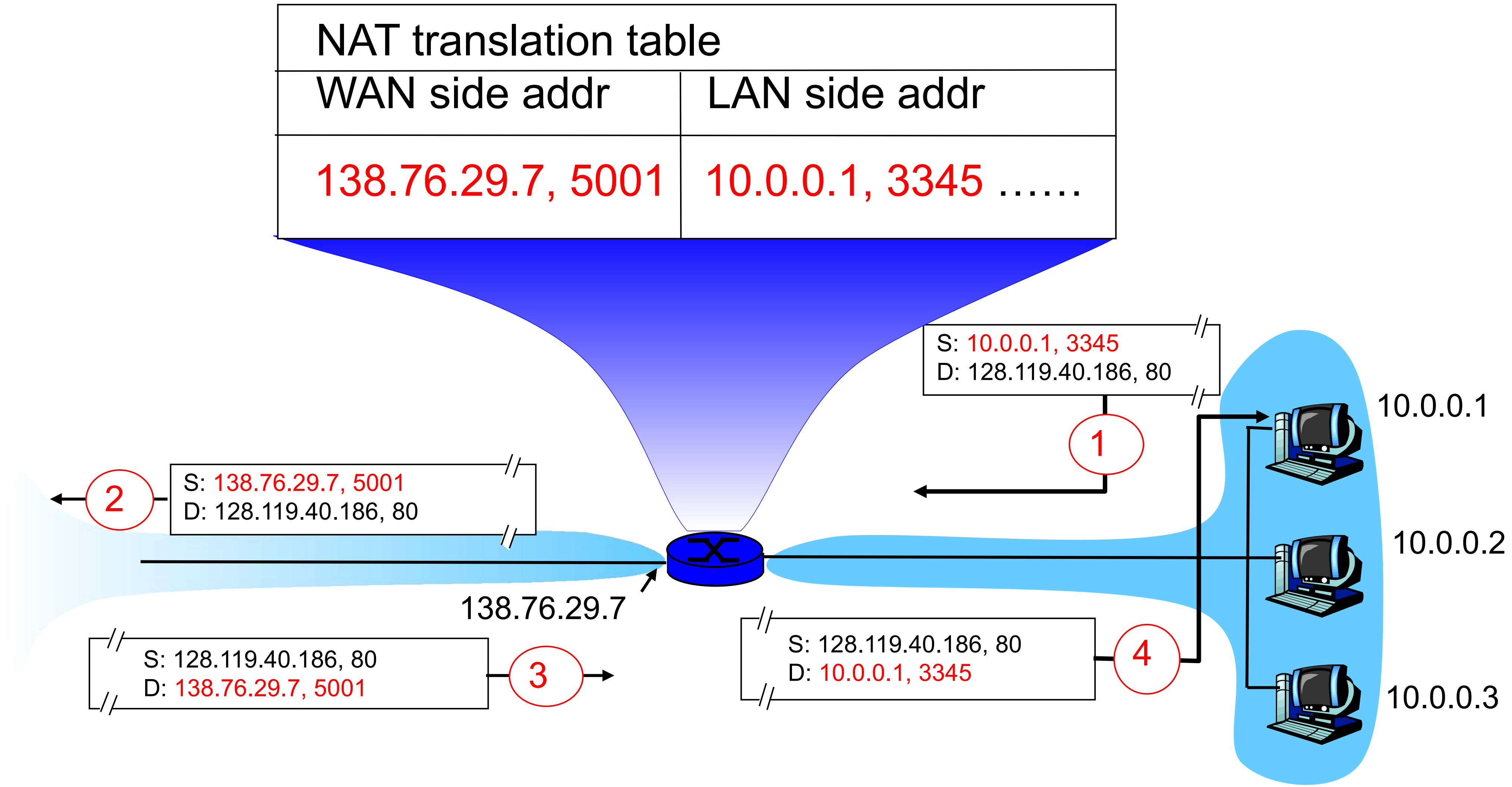
NAT rewrites the IP addresses

- Make “inside” look like single IP addr
- Change header checksums accordingly

Port-Translating NAT

- Two hosts communicate with same destination
  - Destination needs to differentiate the two
- Map outgoing packets
  - Change source address and source port
- Maintain a translation table
  - Map of (src addr, port #) to (NAT addr, new port #)
- Map incoming packets
  - Map the destination address/port to the local host

NAT Example


NAT translation table
WAN side addr
LAN side addr
138.76.29.7, 5001
10.0.0.1, 3345 ……
S: 10.0.0.1, 3345
D: 128.119.40.186, 80
1
10.0.0.1
2
S: 138.76.29.7, 5001
D: 128.119.40.186, 80
10.0.0.2
138.76.29.7
S: 128.119.40.186, 80
D: 10.0.0.1, 3345
4
S: 128.119.40.186, 80
D: 138.76.29.7, 5001
3
10.0.0.3

Maintaining the Mapping Table

- Create an entry upon seeing an outgoing packet
  - Packet with new (source addr, source port) pair
- Eventually, need to delete entries to free up #’s
  - When? If no packets arrive before a timeout
  - (At risk of disrupting a temporarily idle connection)
- Yet another example of “soft state”
  - I.e., removing state if not refreshed for a while

Where is NAT Implemented?

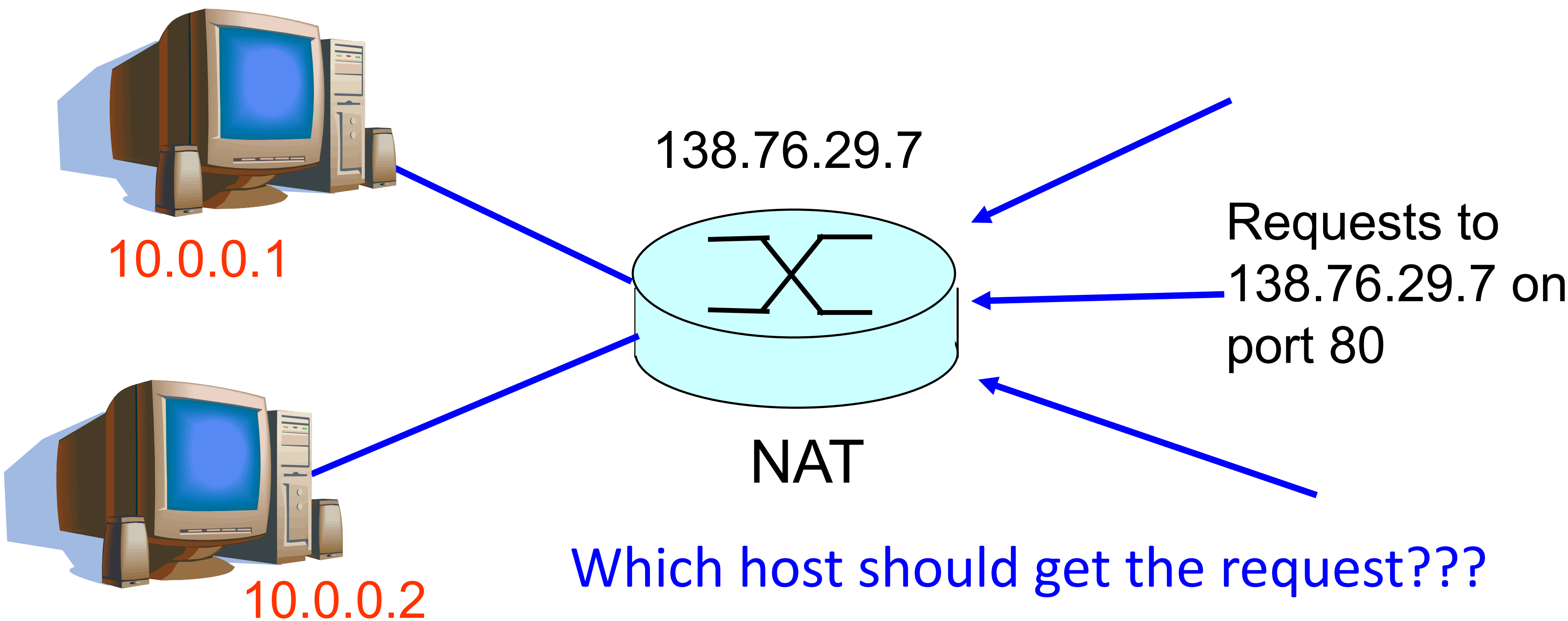
- Home router
  - Integrates router, DHCP server, NAT, etc.
  - Use single IP address from the service provider

- Campus or corporate network
  - NAT at the connection to the Internet
  - Share a collection of public IP addresses
  - Avoid complexity of renumbering hosts/routers when changing ISP (w/ provider-allocated IP prefix)

157

Practical Objections Against NAT

- Port #s are meant to identify sockets
  - Yet, NAT uses them to identify end hosts
  - Makes it hard to run a server behind a NAT



Which host should get the request???

- Explicit config at NAT for incoming conn's

Principled Objections Against NAT

- Routers are not supposed to look at port #s
  - Network layer should care only about *IP* header
  - … and not be looking at the *port numbers* at all
- NAT violates the end-to-end argument
  - Network nodes should not modify the packets
- IPv6 is a cleaner solution
  - Better to migrate than to limp along with a hack

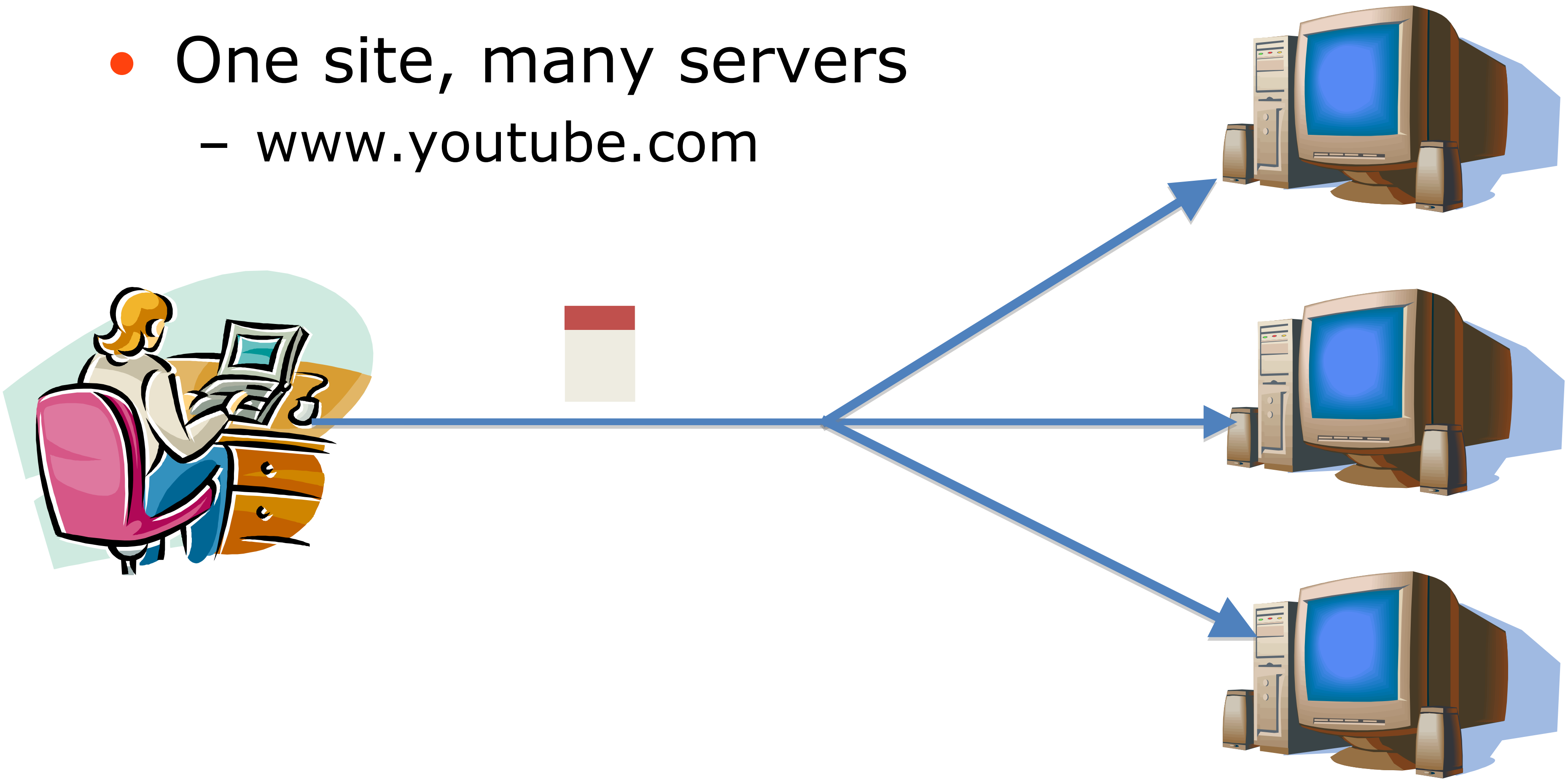
That’s what happens when network puts power in hands of end users!

Types/names of NAT remappings

- Basic NAT (one-to-one)
  - Static allocation of a public to a private IP
  - usually for servers
- Dynamic NAT
  - Allocation range of public to private IPs on demand
  - Usually for clients
- **Port address translation (PAT)**
  - Same single public IP + range of ports to distinguish traffic from private IPs
- **Destination NAT (DNAT)**
  - Change destination IP address on request and inverse for replies
  - External connections to internal servers
  - Like NAT but connection initiated by external client
  - Static configuration needed

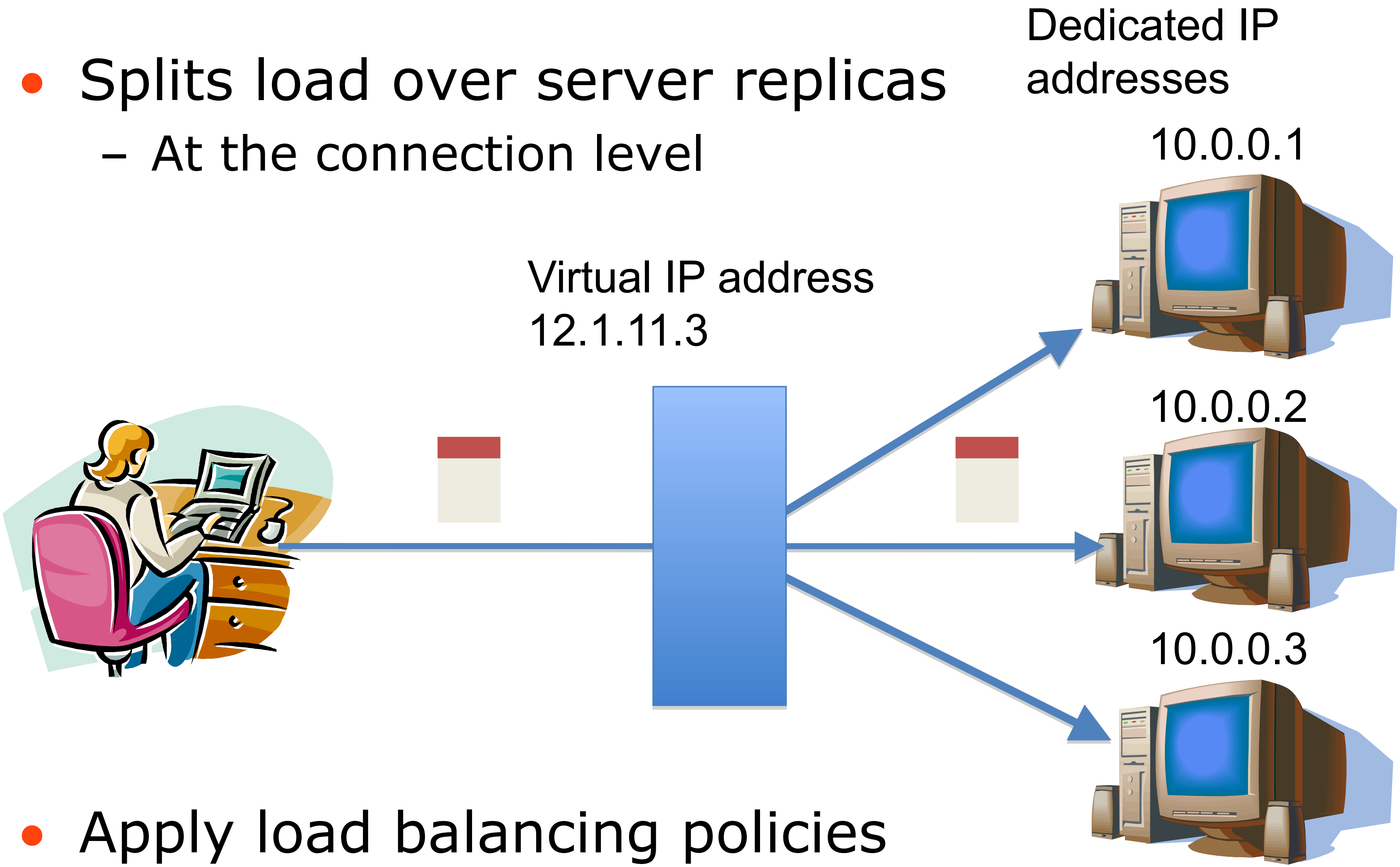
Replicated Servers

- One site, many servers
  - www.youtube.com



Load Balancer

- Splits load over server replicas
  - At the connection level

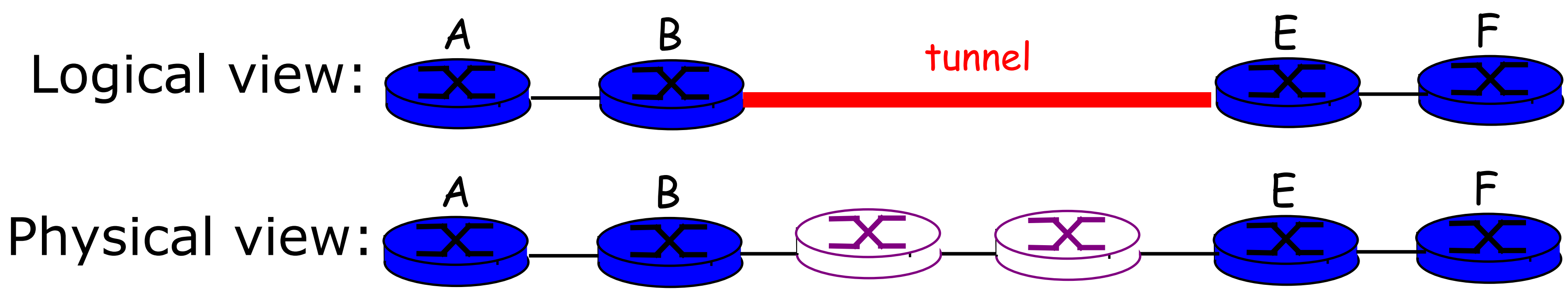


- Apply load balancing policies

Tunneling

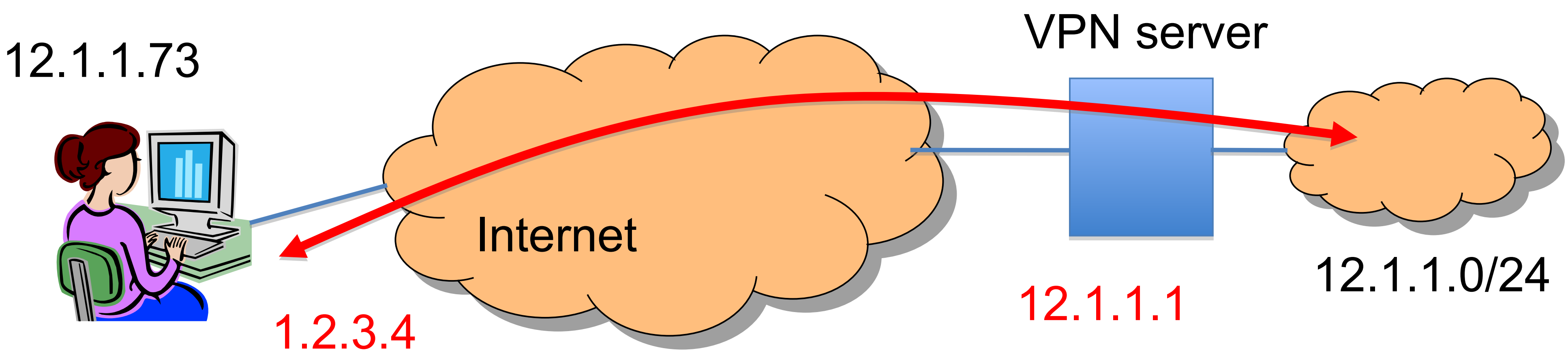
IP Tunneling

- IP tunnel is a virtual point-to-point link
  - Illusion of a direct link between two nodes



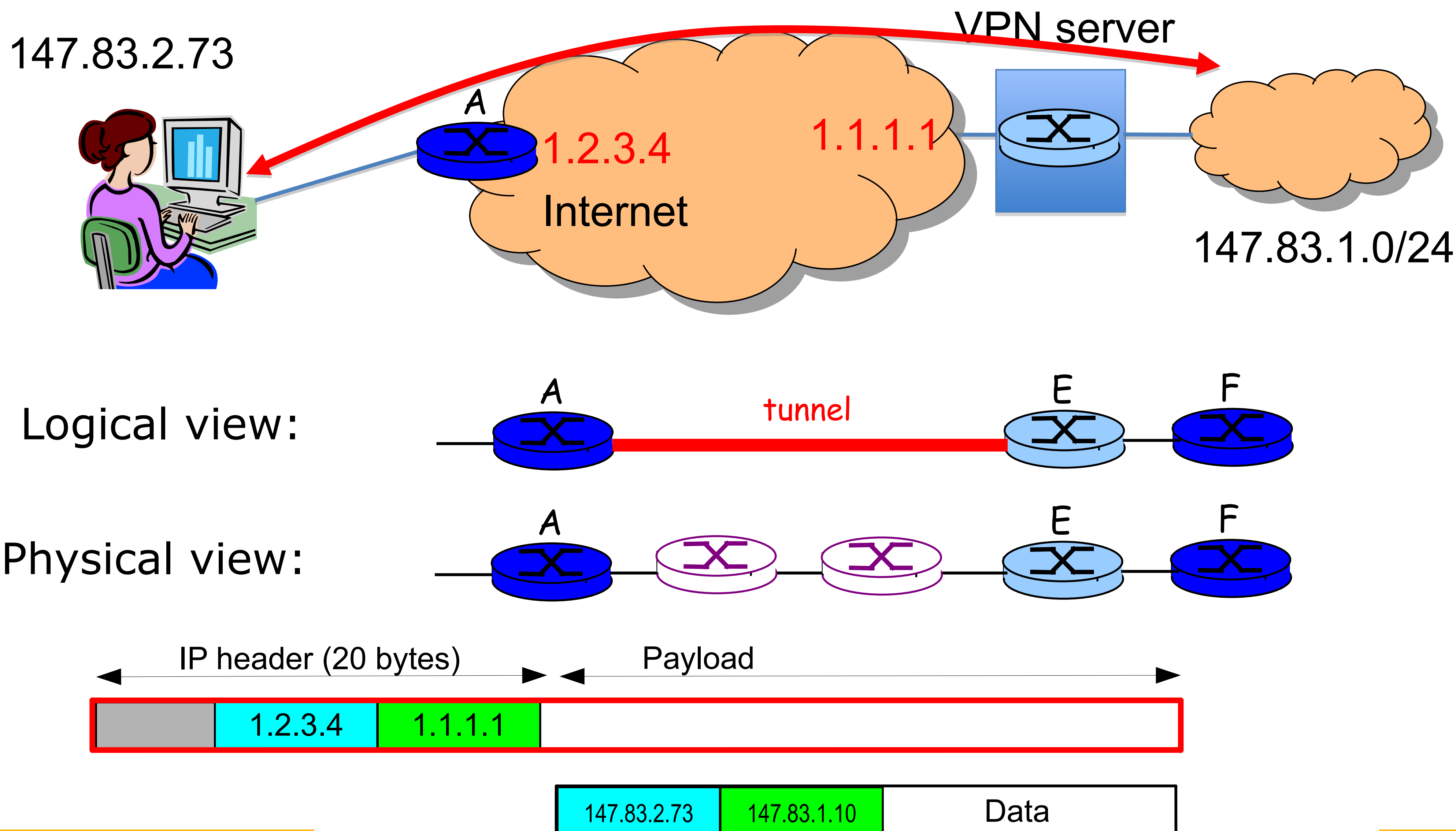
- Encapsulation of the packet inside IP datagram
  - Node B sends a packet to node E
  - ... containing another packet as the payload

Remote Access Virtual Private Network



- Tunnel from user machine to VPN server
  - A "link" across the Internet to the local network
  - A new virtual interface 'tun0' (net 12.1.1.0/24 gw 0.0.0.0 iface tun0)
- Encapsulates packets to/from the user
  - Packet from 12.1.1.73 to 12.1.1.100
  - Inside another IP packet from 1.2.3.4 to 12.1.1.1
- Authentication, encryption, compression

Remote Access Virtual Private Network



VPN Tunneling issues

- Fragmentation inside the tunnel ... the exit tunnel router might need to reassemble
- ICMP messages inside the tunnel.
- Tunnel entry router can fragment datagrams, if needed before encapsulation,
  to avoid the exit router having to reassemble

Conclusions

- Middleboxes address important problems
  - Getting by with fewer IP addresses
  - Blocking unwanted traffic
  - Making fair use of network resources
  - Improving end-to-end performance
- Middleboxes cause problems of their own
  - No longer globally unique IP addresses
  - Cannot assume network simply delivers packets