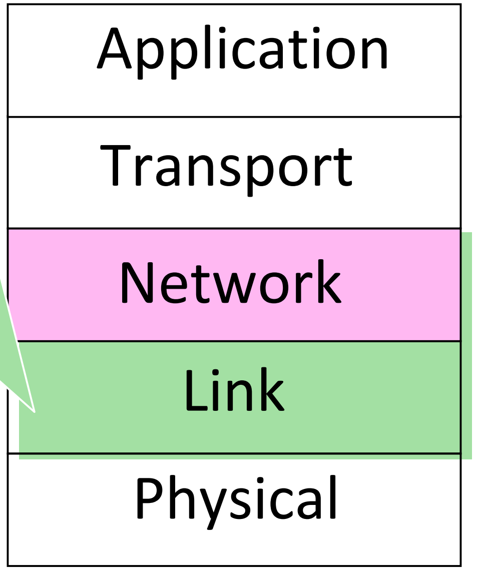
Where we are in the Course

Concerns how to transfer messages over connected *links*.

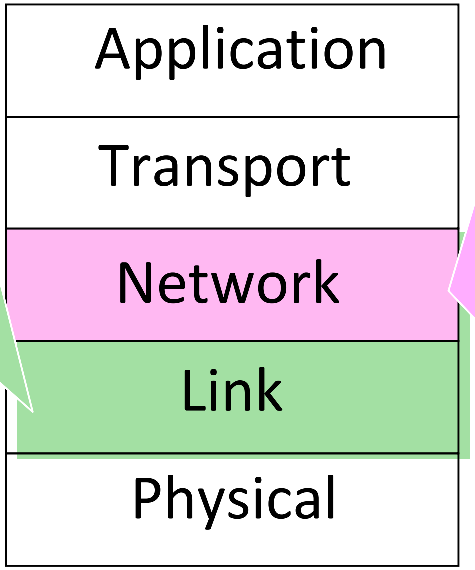
- Logic Link Control (LLC)
  - error and flow control, interfaces for high layers
- Media Access Control (MAC)
  - framing and access control



Where we are in the Course

Concerns how to transfer messages over connected ***links***.

- Logic Link Control (LLC)
  - error and flow control, interfaces for high layers
- Media Access Control (MAC)
  - framing and access control



Application

Transport

Network

Link

Physical

Concerns how to transfer messages across connected ***networks***.

- Router architecture
- Network service models
- IP: Internet Protocol
- Routing algorithms and protocols

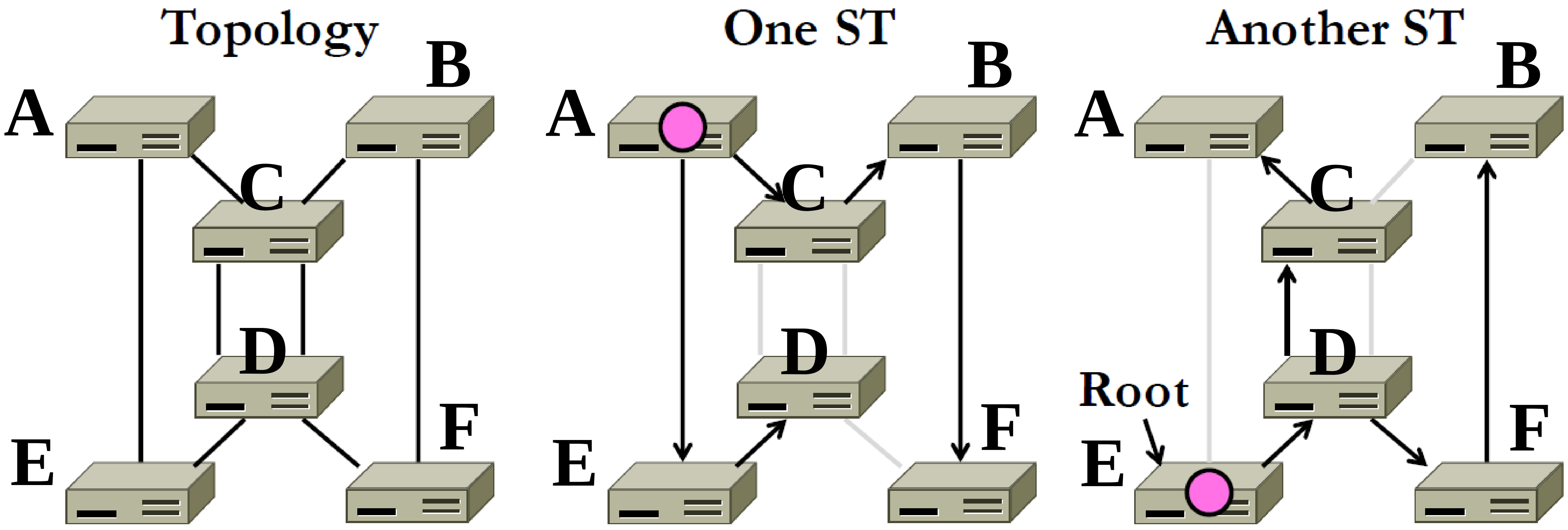
…

Roadmap

- Network layer Overview
  - network-layer approach
  - router architecture
  - network service models
- IP: Internet Protocol
- Routing

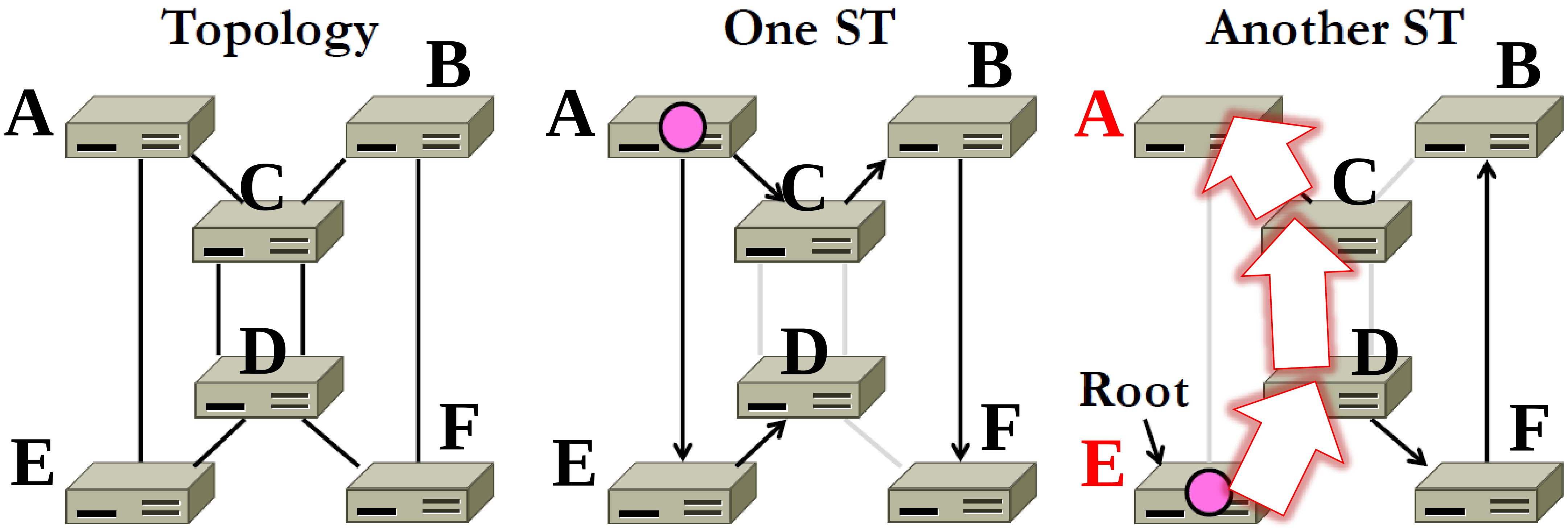
Why do we need a Network layer?

- We can already build networks with links and switches and send frames between hosts based on spanning tree …



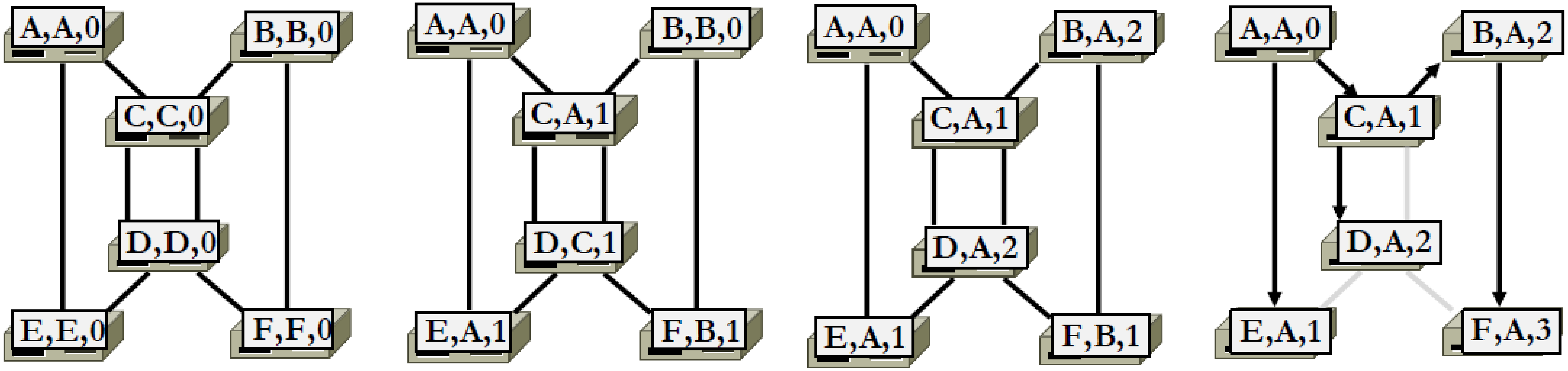
Why do we need a Network layer?

- We can already build networks with links and switches and send frames between hosts based on spanning tree …



Shortcomings of Switches

1. **Don’t scale to large networks**
   - Blow up of routing table, broadcast

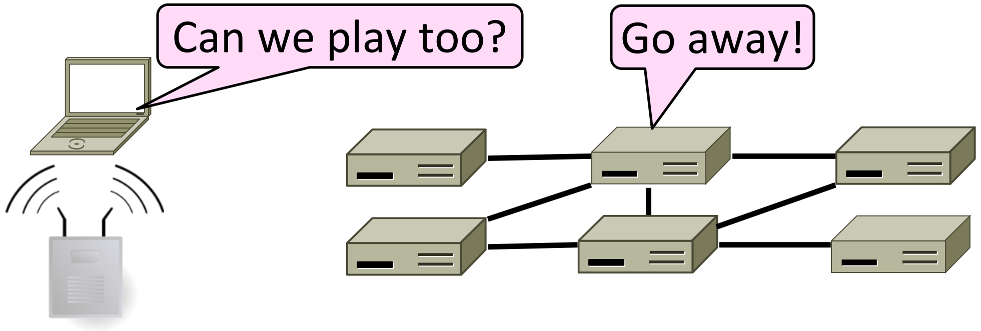


Need 4 rounds for the spanning tree to converge in this 6-node network …

What if there’re MILLIONS of nodes ? 😱

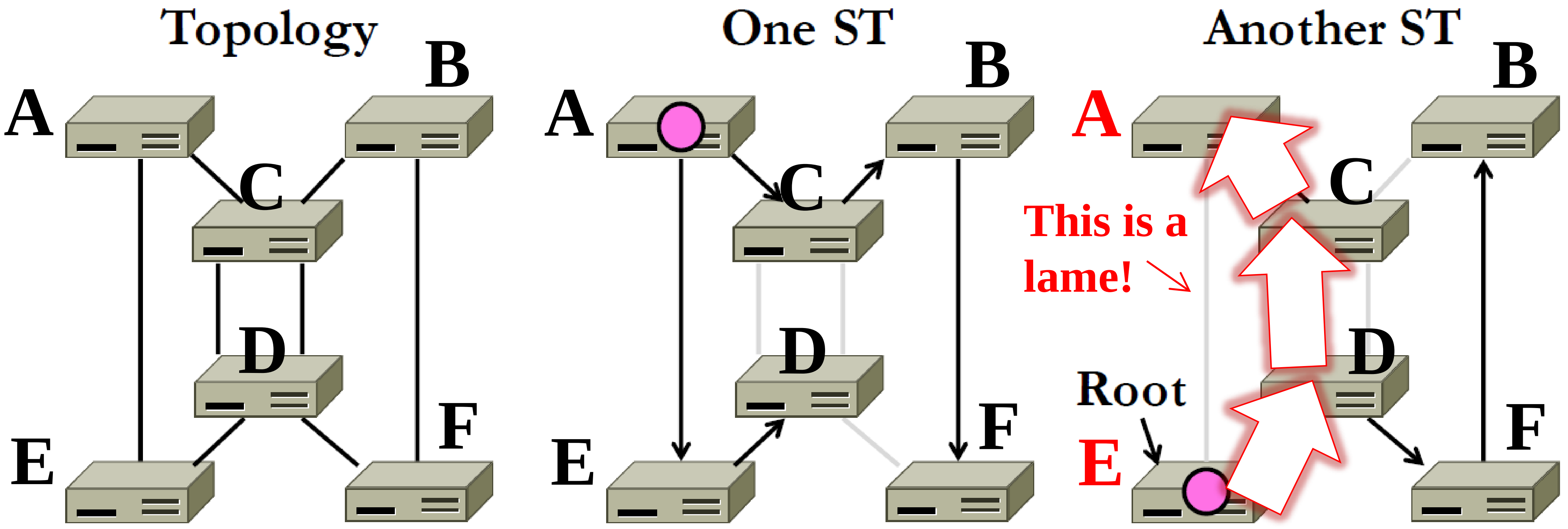
Shortcomings of Switches (2)

2. Don’t work across more than one link layer technology
   - Hosts on Ethernet + 3G + 802.11 …



Shortcomings of Switches (3)

3. Don’t give much traffic control
   - Want to plan routes / bandwidth



Network Layer Approach

- Scaling:
  - Hierarchy, in the form of prefixes
- Heterogeneity:
  - IP for internetworking
- Bandwidth Control:
  - Lowest-cost routing
  - Later QOS (Quality of Service)

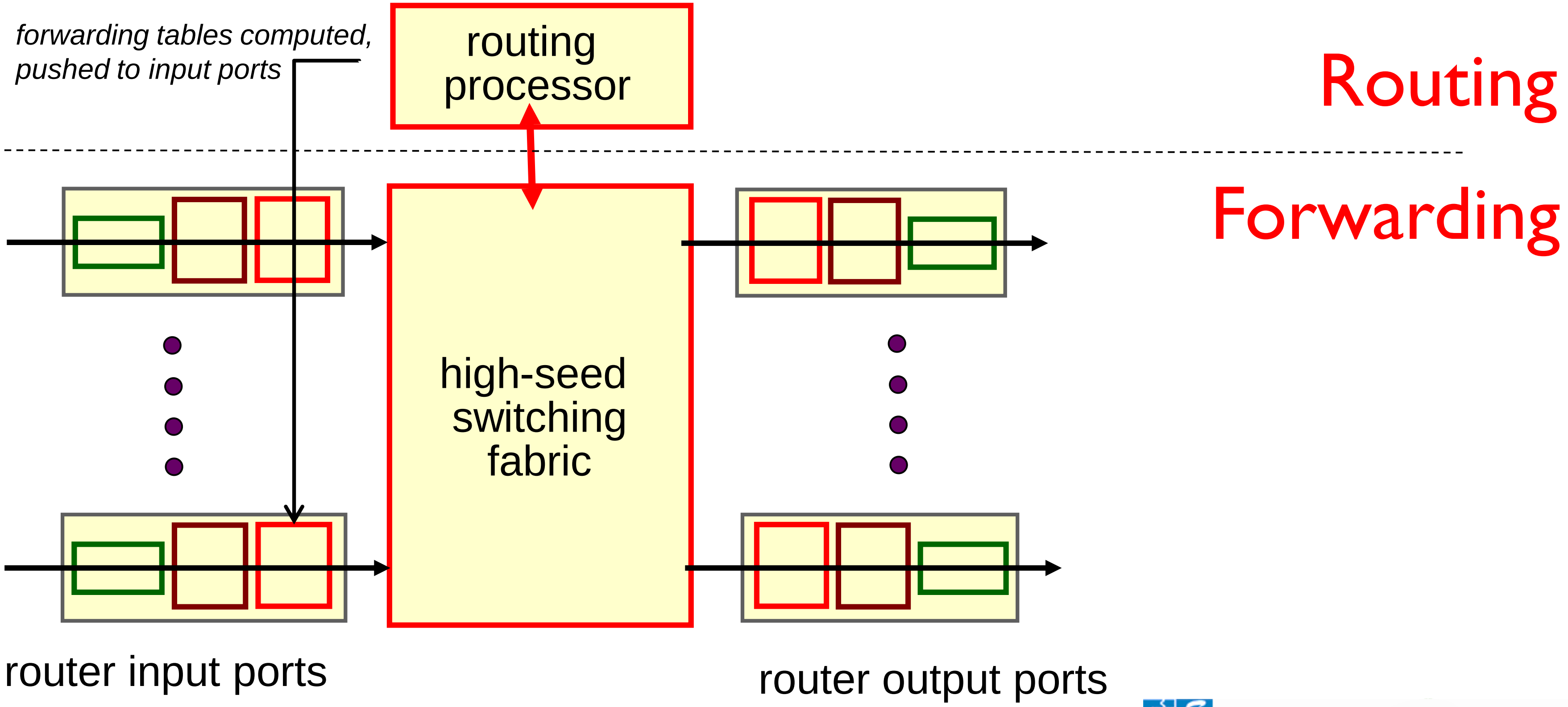
Roadmap

- Network layer Overview
  - network-layer approach
  - router architecture
  - network service models
- IP: Internet Protocol
- Routing

Router architecture overview

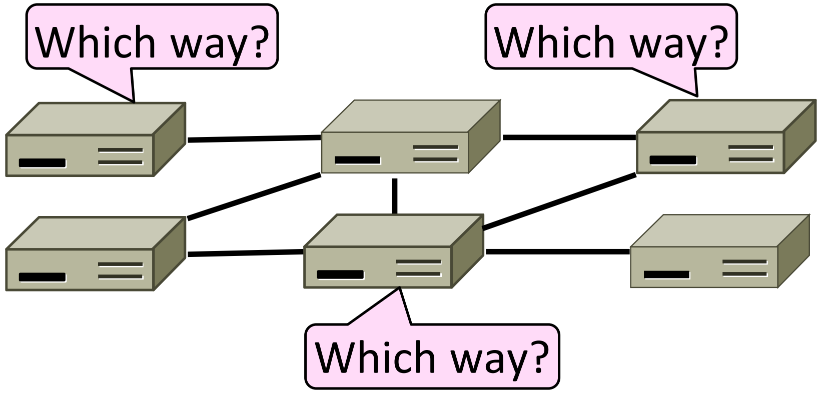
two key router functions:

- **running *routing* protocols (RIP, OSPF, BGP)**
- ***forwarding* datagrams from incoming to outgoing link**



Routing vs. Forwarding

- **Routing** is the process of deciding in which direction to send traffic
  - Network wide (global) and expensive

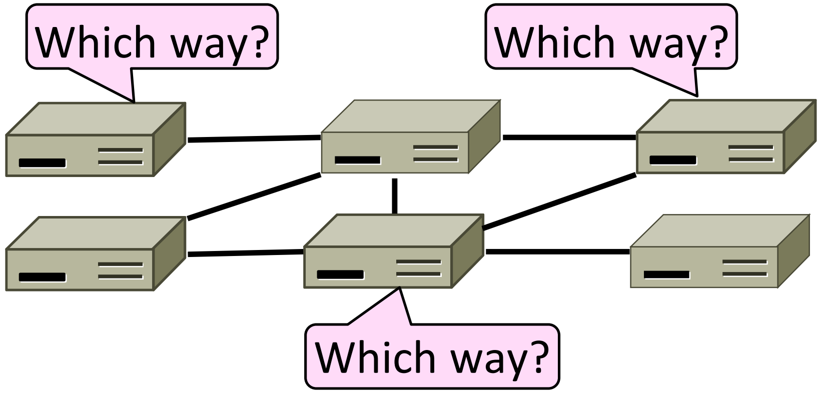


analogy:

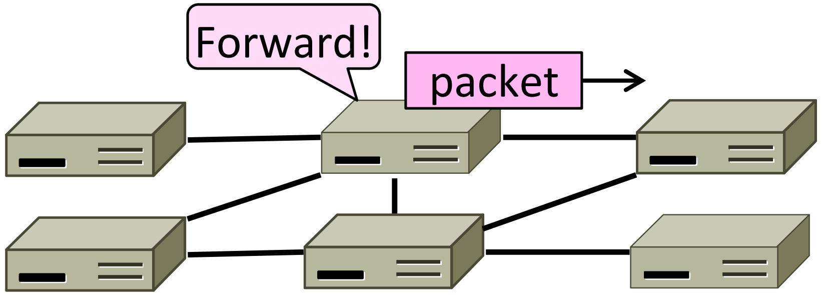
- ***routing:*** **process of planning trip from source to dest**

Routing vs. Forwarding

- Routing is the process of deciding in which direction to send traffic
  - Network wide (global) and expensive



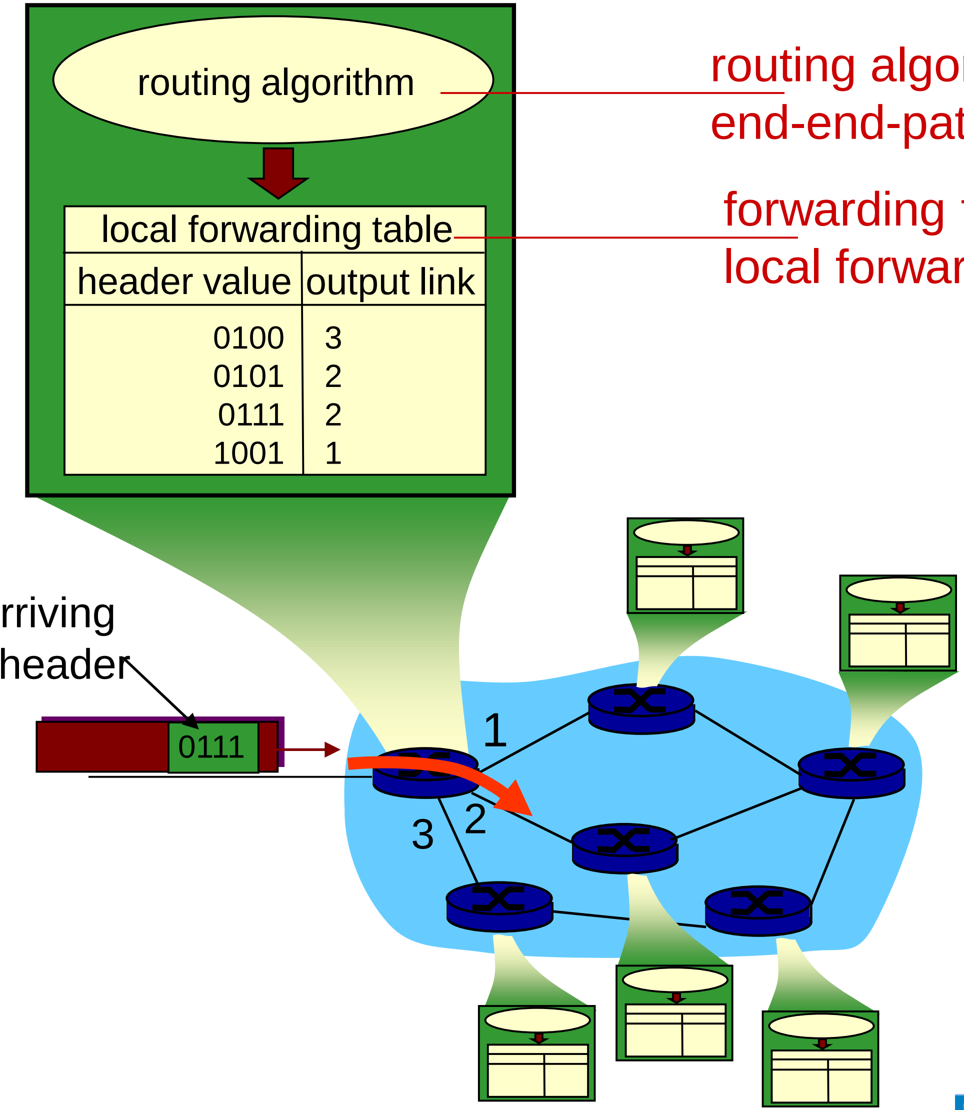
- Forwarding is the process of sending a packet on its way
  - Node process (local) and fast



analogy:

- *routing:* process of planning trip from source to dest
- *forwarding:* process of getting through single interchange

Interplay between routing and forwarding



local forwarding table	
header value	output link
0100	3
0101	2
0111	2
1001	1

routing algorithm determines
end-end-path through network

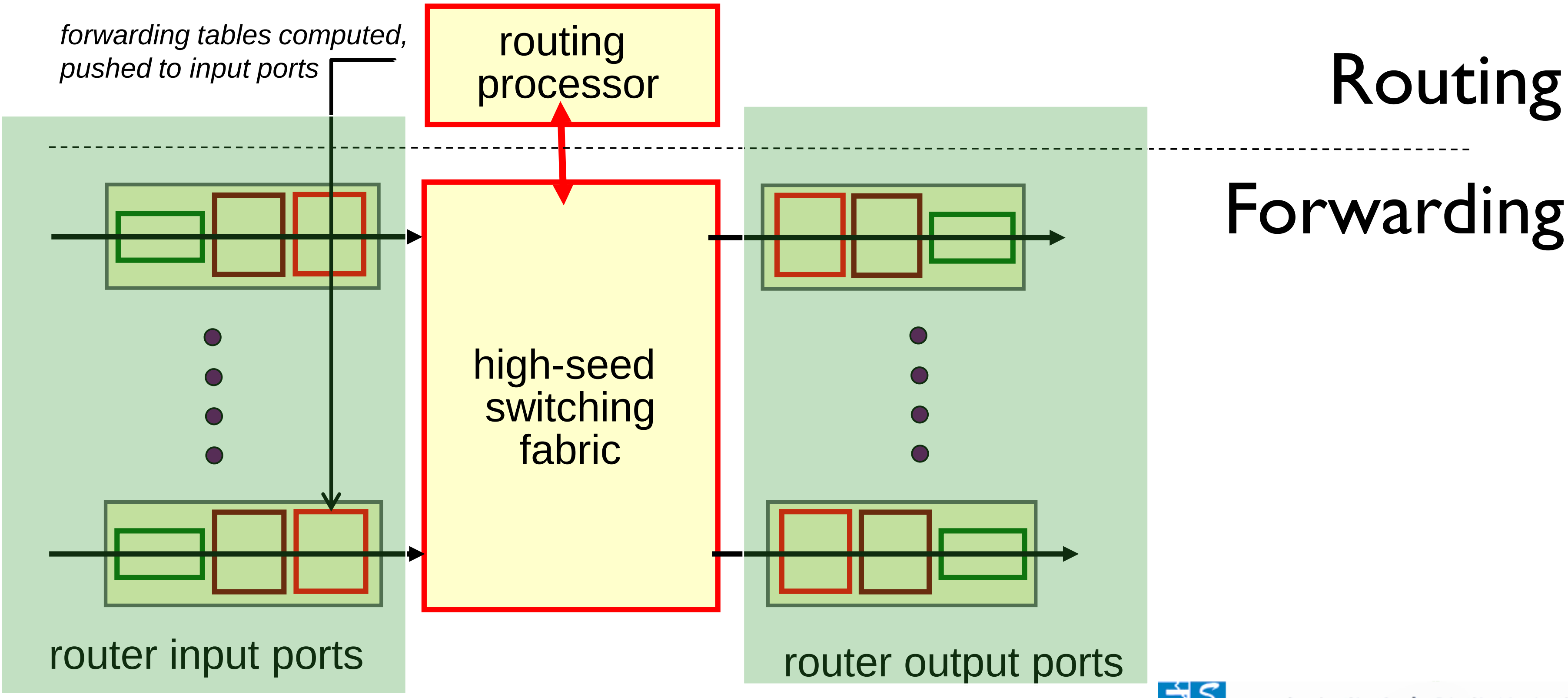
forwarding table determines
local forwarding at this router

value in arriving
packet’s header

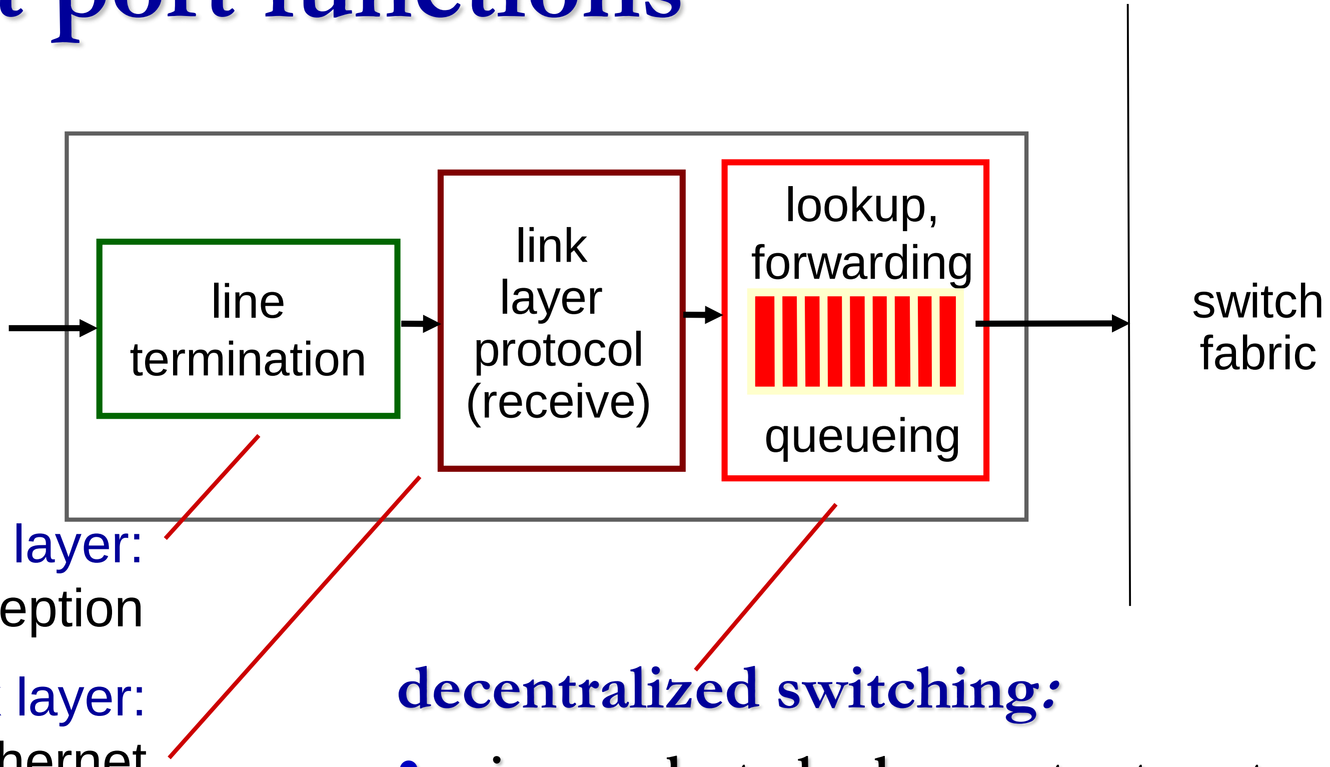
Router architecture overview

two key router functions:

- running *routing* protocols (RIP, OSPF, BGP)
- *forwarding* datagrams from incoming to outgoing link


forwarding tables computed, pushed to input ports
routing processor
Routing
Forwarding
high-seed switching fabric
router input ports
router output ports

Input port functions



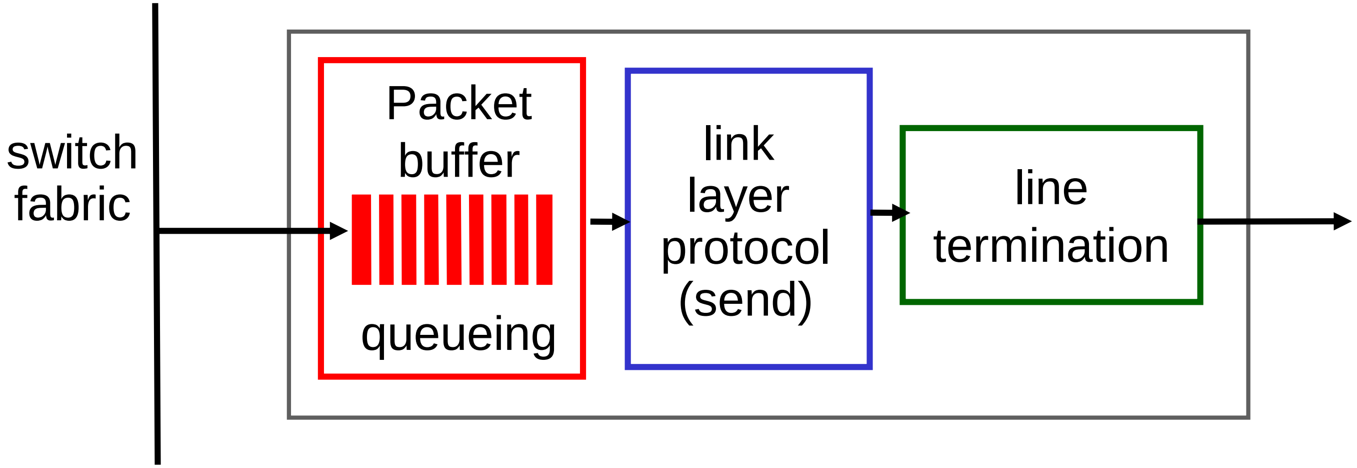
physical layer:
bit-level reception

data link layer:
e.g., Ethernet
see chapter 5

decentralized switching*:*

- **given a dest., lookup output port using forwarding table in input port memory *("match plus action")***
- **goal: complete input port processing at 'line speed'**
- **queuing: if packets arrive faster than forwarding rate into switch fabric**

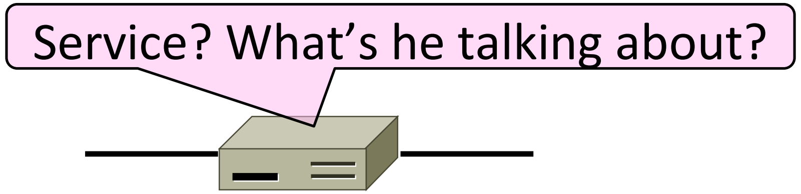
Output port functions



- *buffering* required when packets arrive from fabric faster than the transmission rate
- *scheduling policy* chooses among queued datagrams for transmission

Network-Layer Service Models

- **What kind of service does the Network layer provide to the Transport layer?**
  - Bottom-up view: interconnecting different networks
  - Top-down view: service provider



Two Network Service Models

- Datagrams, or *connectionless* service
  - Like postal letters
  - (This one is IP)

- Virtual circuits, or *connection-oriented* service
  - Like a telephone call

Who should maintain connection state?
The network or the end system?

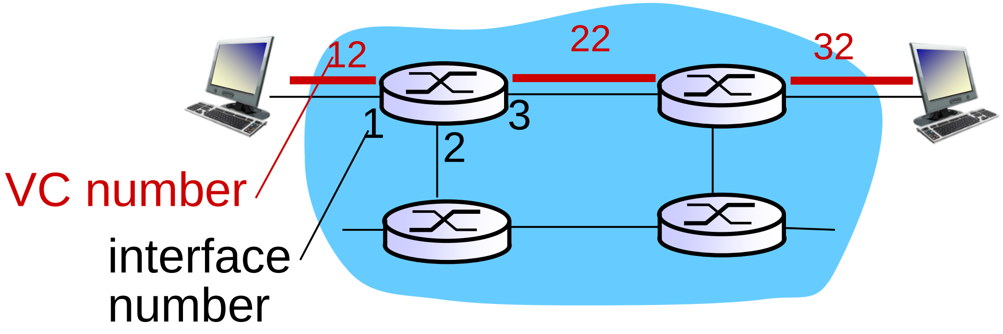
Virtual Circuit Model

- Three phases:
    1. Connection establishment, circuit is set up
        - Path is chosen, circuit information stored in routers
    2. Data transfer, circuit is used
        - Packets are forwarded along the path
    3. Connection teardown, circuit is deleted
        - Circuit information is removed from routers

- Just like a telephone circuit, but virtual in the sense that no bandwidth need be reserved; statistical sharing of links

Virtual circuit forwarding table

- **Packets only contain a short label to identify the circuit**
  - **Labels don't have any global meaning, only unique for a link**



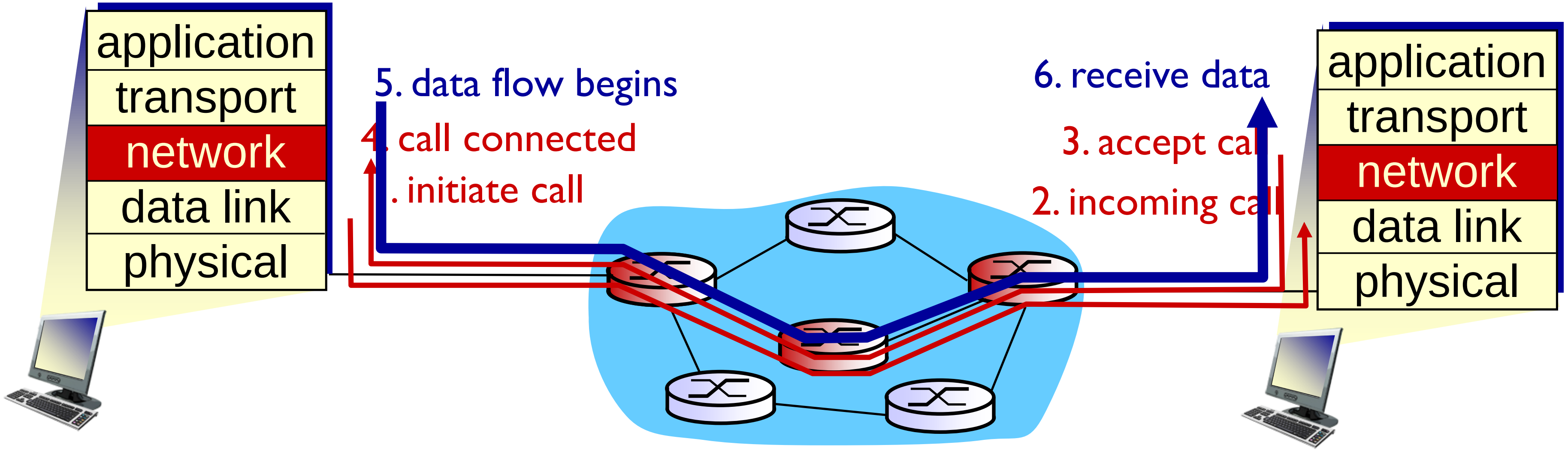
forwarding table in northwest router:

Incoming interface	Incoming VC #	Outgoing interface	Outgoing VC #
1	12	3	22
2	63	1	18
3	7	2	17
1	97	3	87
…	…	…	…

VC routers maintain connection state information!

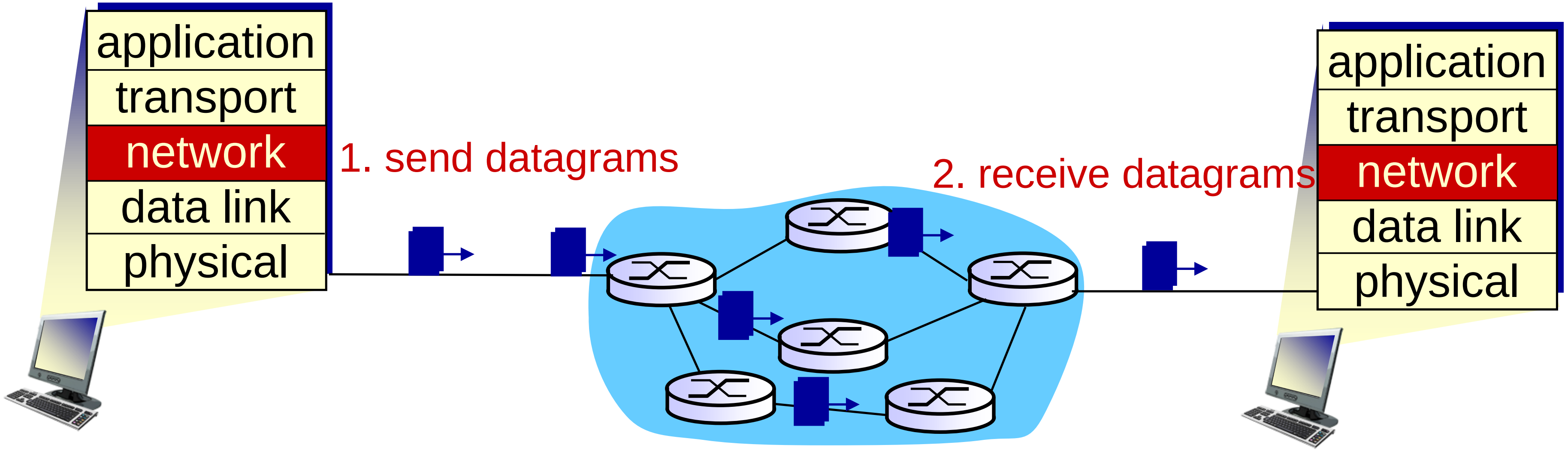
Virtual circuit signaling

- used to setup, maintain teardown VC
- used in ATM, frame-relay, X.25
- not used in today's Internet



Datagram networks (e.g., Internet)

- no call setup at network layer
- routers: no state about end-to-end connections
  - no network-level concept of "connection"
- packets forwarded using destination host address



Datagram or VC network: why?

Datagram (Internet)

- data exchange among computers
  - "elastic" service, no strict timing requirement
- many link types
  - different characteristics
  - uniform service difficult
- "smart" end systems
  - can adapt, perform control, error recovery
  - *simple inside network, complexity at "edge"*

VC (ATM)

- evolved from telephony
- human conversation:
  - strict timing, reliability requirements
  - need for guaranteed service
- "dumb" end systems
  - telephones
  - *complexity inside network*

Datagrams vs Virtual Circuits

- **Complementary strengths**

Issue	Datagrams	Virtual Circuits
Setup phase	Not needed	Required
Router state	Per destination	Per connection
Addresses	Packet carries full address	Packet carries short label
Routing	Per packet	Per circuit
Failures	Easier to mask	Difficult to mask
Quality of service	Difficult to add	Easier to add

Roadmap

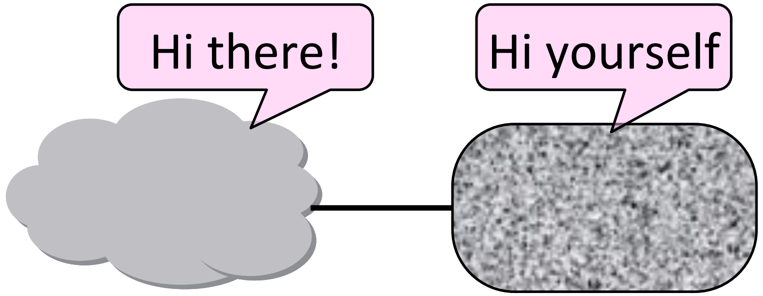
- Network layer Overview
- IP: Internet Protocol
  - internetworking and datagram format
  - IPv4 addressing and prefix-based forwarding
  - fragmentation and MTU discovery
  - ARP, DHCP, and NAT
  - ICMP
  - IPv6
- Routing

Roadmap

- Network layer Overview
- IP: Internet Protocol
  - internetworking and datagram format
  - IPv4 addressing and prefix-based forwarding
  - fragmentation and MTU discovery
  - ARP, DHCP, and NAT
  - ICMP
  - IPv6
- Routing

Topic

- How do we connect different networks together?
  - This is called internetworking
  - We'll look at how IP does it



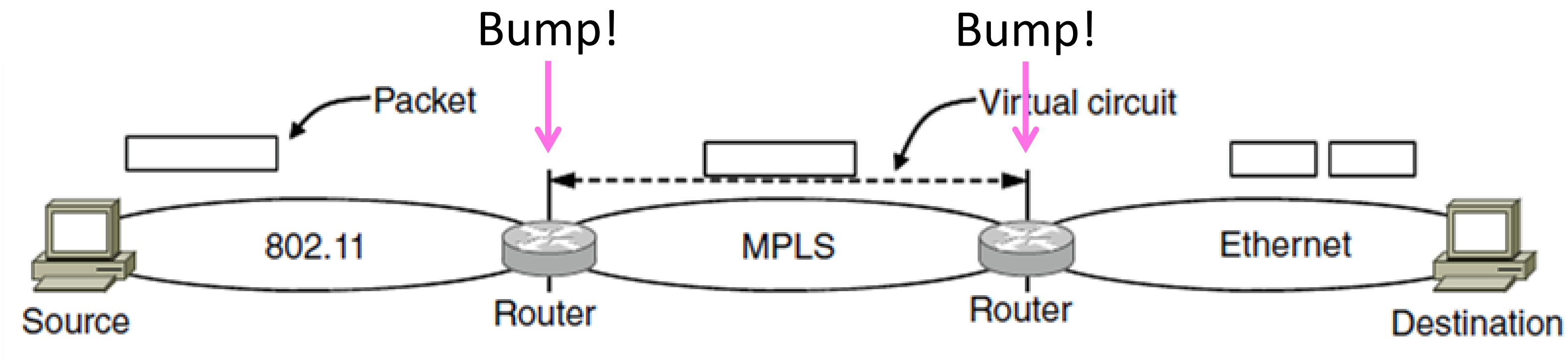
How Networks May Differ

- Basically, in a lot of ways:
  - Service model (datagrams, VCs)
  - Addressing (what kind)
  - QOS (priorities, no priorities)
  - Packet sizes
  - Security (whether encrypted)

- Internetworking hides the differences with a common protocol. (Uh oh.)

Connecting Datagram and VC networks

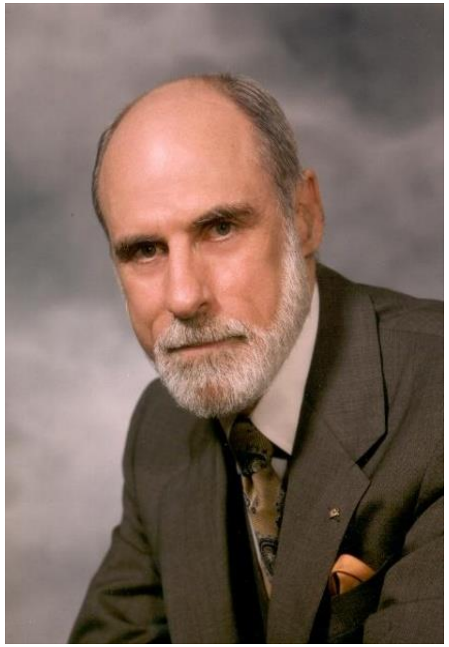
- An example to show that it’s not so easy
  - Need to map destination address to a VC and vice-versa
  - A bit of a “road bump”, e.g., might have to set up a VC



Internetworking – Cerf and Kahn

- Pioneered by Cerf and Kahn, the "fathers of the Internet"
  - In 1974, later led to TCP/IP
- Tackled the problems of interconnecting networks
  - Instead of mandating a single network technology

Vint Cerf



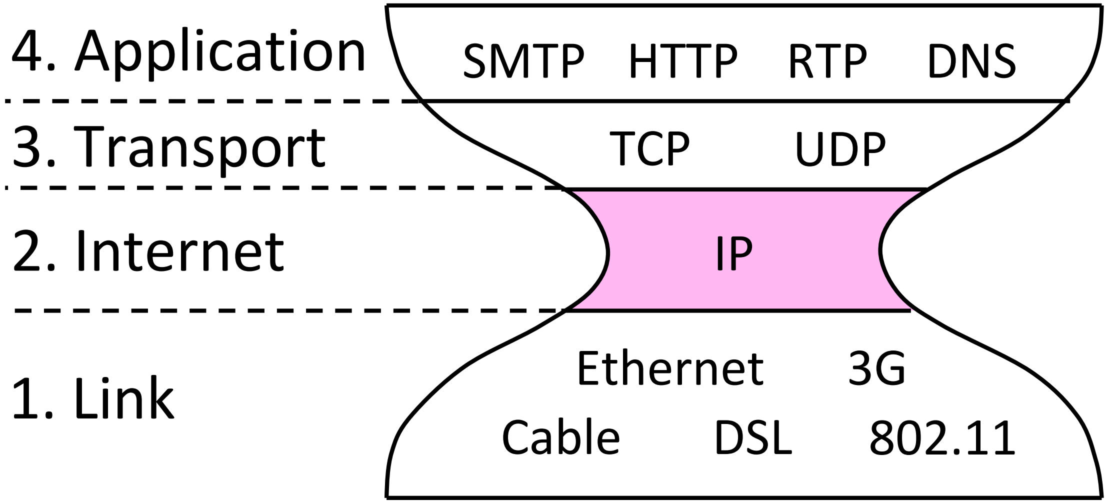
© 2009 IEEE

Bob Kahn

© 2009 IEEE

Internet Reference Model

- IP is the “narrow waist” of the Internet
  - Supports many different links below and apps above



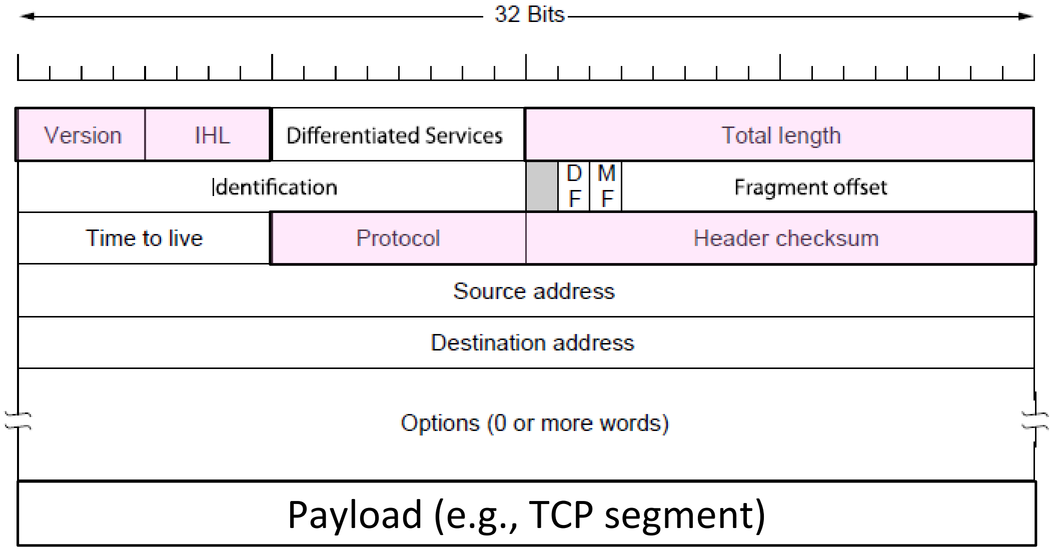
IP as a Lowest Common Denominator

- Suppose only some networks support QOS or security etc.
  - Difficult for internetwork to support

- Pushes IP to be a “lowest common denominator” protocol
  - Asks little of lower-layer networks
  - Gives little as a higher layer service

IPv4 Datagram Format

- Various fields to meet straightforward needs
  - Version, Header (IHL) and Total length, Protocol, and Header Checksum



32 Bits

Version | IHL | Differentiated Services | Total length

Identification | DF | MF | Fragment offset

Time to live | Protocol | Header checksum

Source address

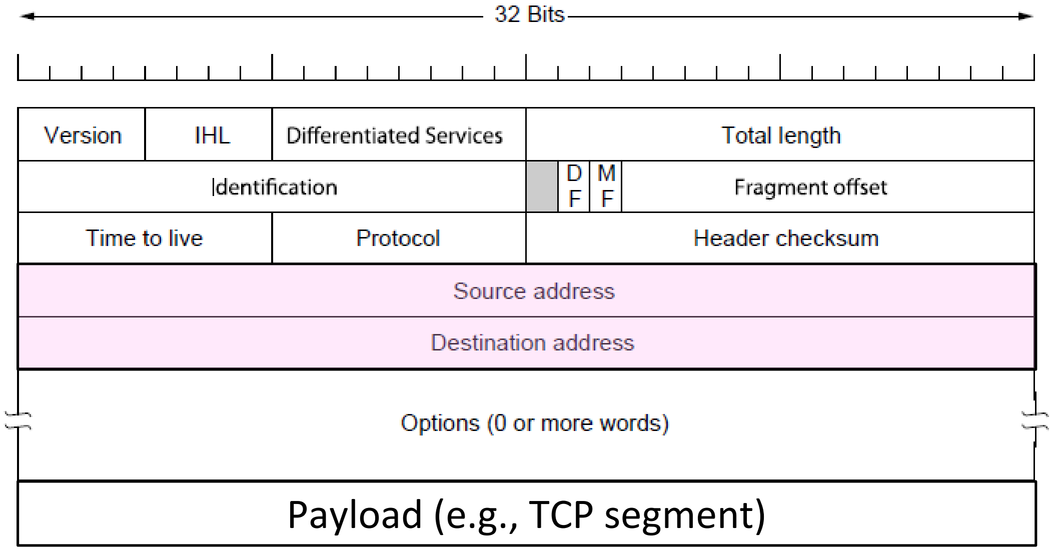
Destination address

Options (0 or more words)

Payload (e.g., TCP segment)

IPv4 Datagram Format (2)

- Network layer of the Internet, uses datagrams
  - Provides a layer of addressing above link addresses (next)



32 Bits

Version | IHL | Differentiated Services | Total length

Identification | DF | MF | Fragment offset

Time to live | Protocol | Header checksum

Source address

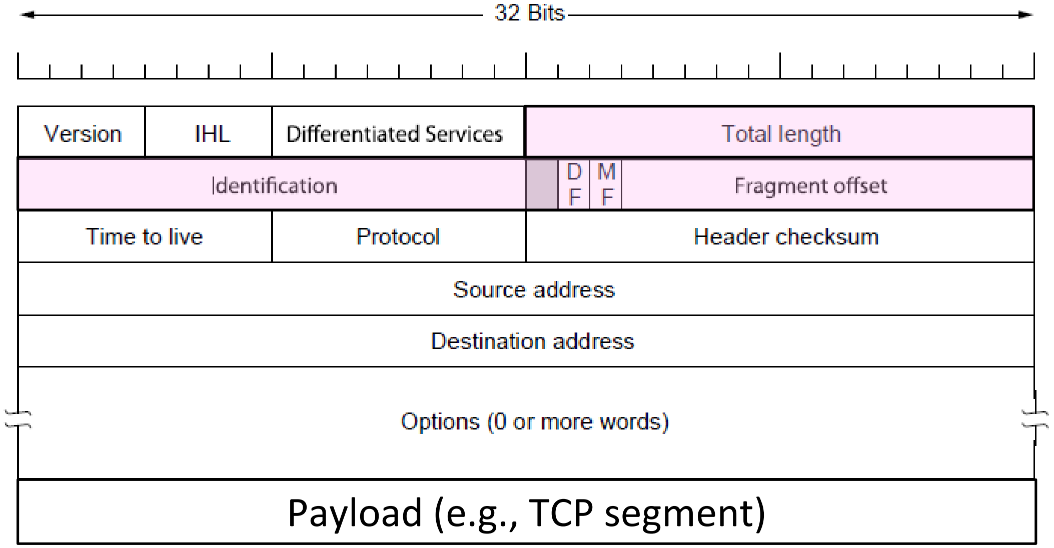
Destination address

Options (0 or more words)

Payload (e.g., TCP segment)

IPv4 Datagram Format (3)

- Some fields to handle packet size differences (later)
  - Identification, Fragment offset, Fragment control bits



32 Bits

Version | IHL | Differentiated Services | Total length

Identification | DF | MF | Fragment offset

Time to live | Protocol | Header checksum

Source address

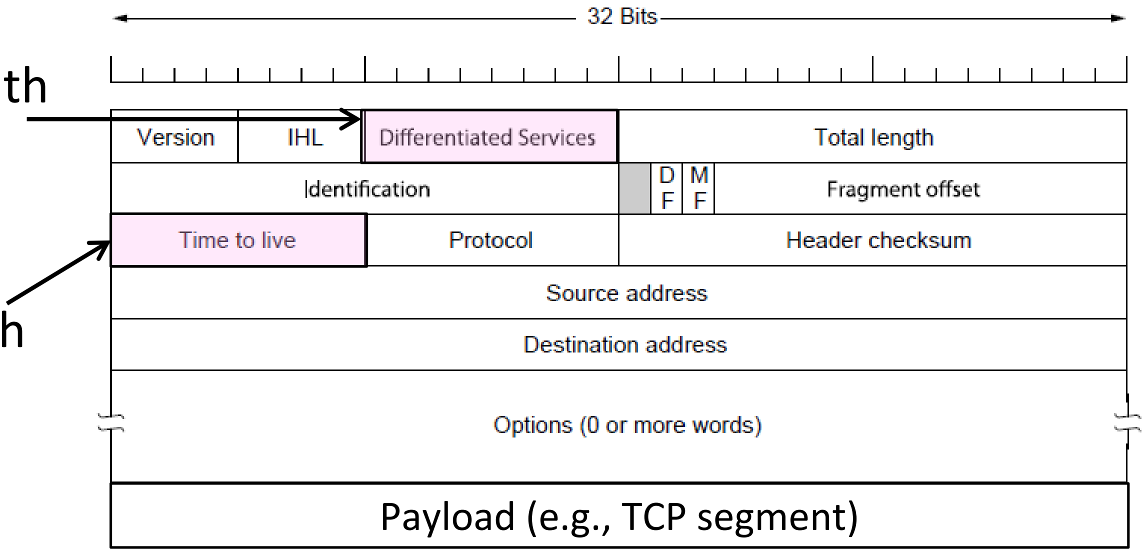
Destination address

Options (0 or more words)

Payload (e.g., TCP segment)

IPv4 Datagram Format (4)

- **Other fields to meet other needs (later, later)**
  - **Differentiated Services, Time to live (TTL)**



Later, with QOS

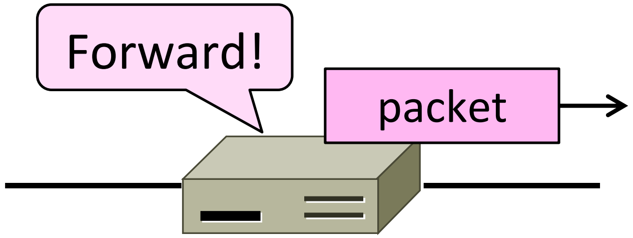
Later, with ICMP

Roadmap

- Network layer Overview
- IP: Internet Protocol
  - internetworking and datagram format
  - IPv4 addressing and prefix-based forwarding
  - fragmentation and MTU discovery
  - ARP, DHCP, and NAT
  - ICMP
  - IPv6
- Routing

Topic

- How do routers forward packets?
  - We'll look at how IP does it
  - (We'll cover routing later)



IP Addresses

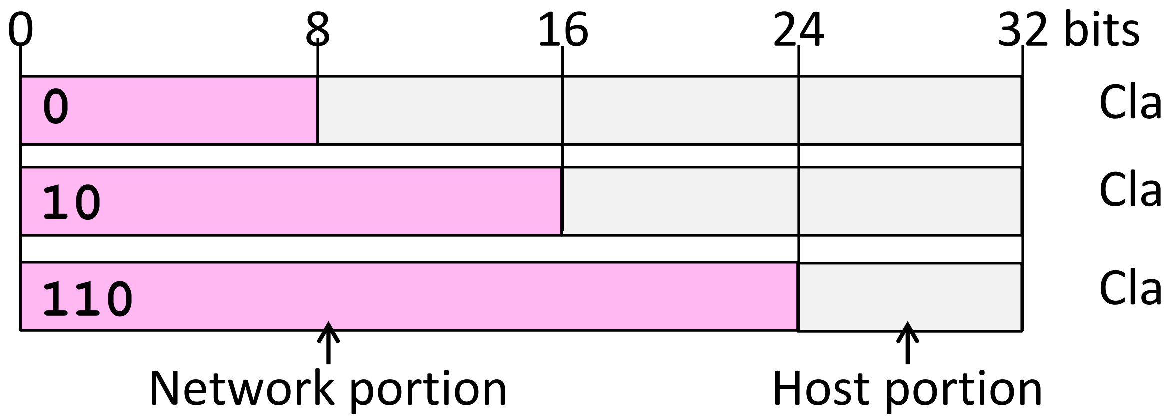
- IPv4 uses 32-bit addresses
  - Later we’ll see IPv6, which uses 128-bit addresses
- Written in “dotted quad” notation
  - Four 8-bit numbers separated by dots

$$\overbrace{\texttt{aaaaaaaa}}^{\text{8 bits}}\overbrace{\texttt{bbbbbbbb}}^{\text{8 bits}}\overbrace{\texttt{cccccccc}}^{\text{8 bits}}\overbrace{\texttt{dddddddd}}^{\text{8 bits}} \leftrightarrow \texttt{A.B.C.D}$$

$$\texttt{00010010}\mid\texttt{00011111}\mid\texttt{00000000}\mid\texttt{00000001} \leftrightarrow$$

IP Address Classes - Historical

- Originally, IP addresses came in fixed size blocks with the class/size encoded in the high-order bits
  - They still do, but the classes are now ignored



Class A, 2^{24} addresses

Class B, 2^{16} addresses

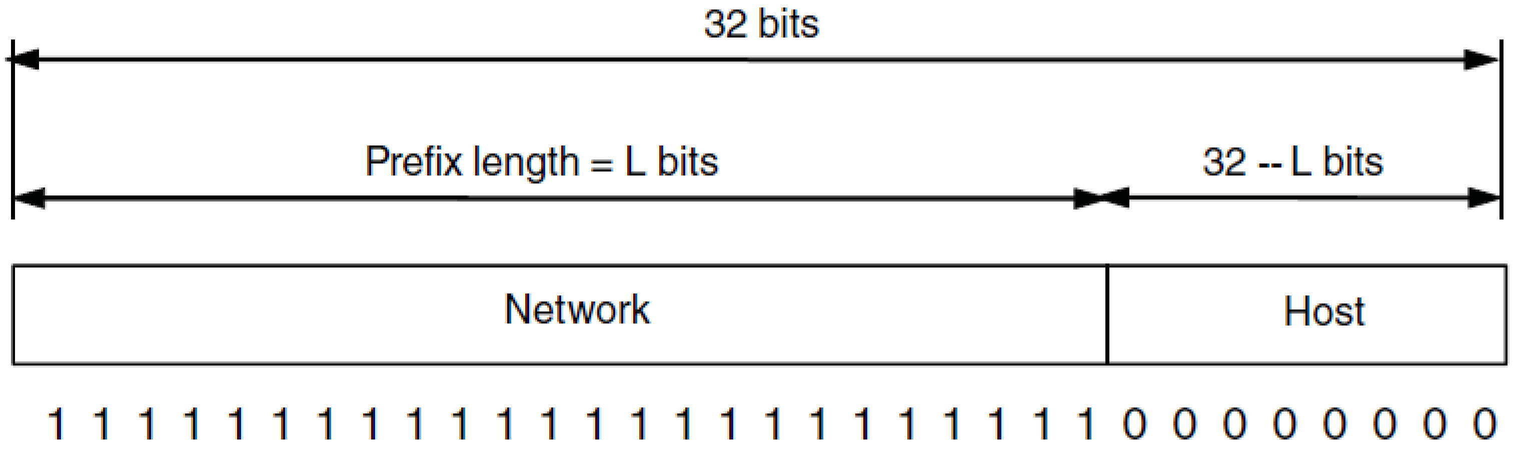
Class C, 2^{8} addresses

Public / Private IP Addresses

- Public IP addresses, e.g., 18.31.0.1
  - Valid destination on the global Internet
  - Must be allocated to you before use
  - Mostly exhausted … time for IPv6
- Private IP addresses
  - Can be used freely within private networks (home, small company)
  - 10.0.0.0/8, 172.16.0.0/12, 192.168.0.0/16
  - Need public IP address(es) and NAT to connect to global Internet

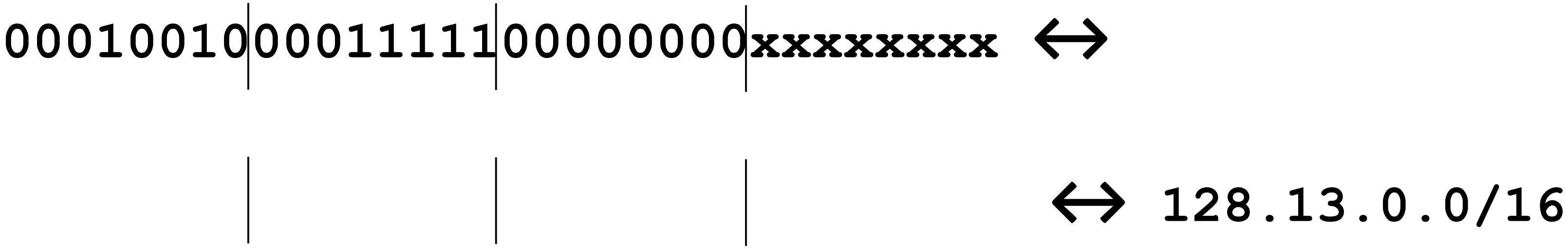
IP Prefixes - Modern

- Addresses are allocated in blocks called prefixes
  - Addresses in an L-bit prefix have the same top L bits
  - There are 2^{32-L} addresses aligned on 2^{32-L} boundary



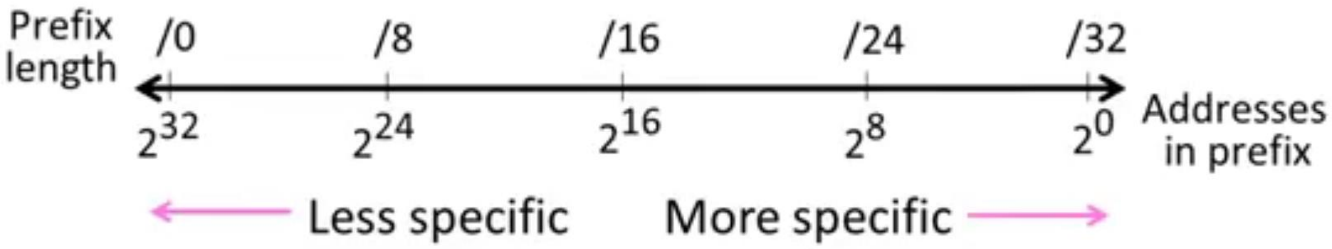
IP Prefixes (2)

- Written in “IP address/length” notation
  - Address is lowest address in the prefix, length is prefix bits
  - E.g., 128.13.0.0/16 is 128.13.0.0 to 128.13.255.255
  - So a /24 (“slash 24”) is 256 addresses, and a /32 is one address



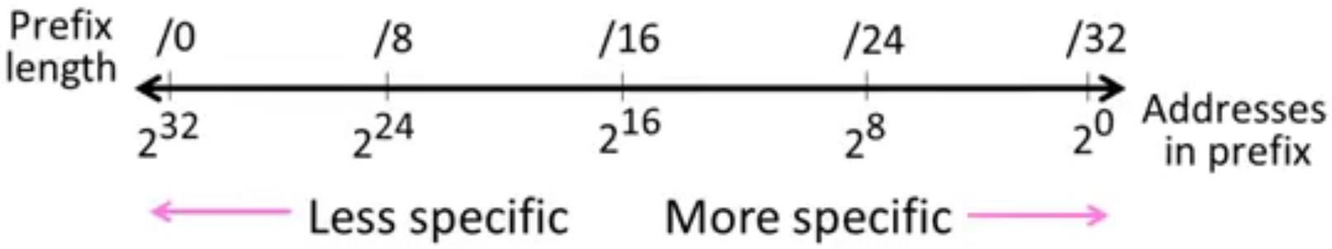
IP Prefixes (3)

- More specific prefix
  - Has longer prefix, hence a smaller number of IP addresses
- Less specific prefix
  - Has shorter prefix, hence a larger number of IP addresses



IP Prefixes (3)

- More specific prefix
  - Has longer prefix, hence a smaller number of IP addresses
- Less specific prefix
  - Has shorter prefix, hence a larger number of IP addresses

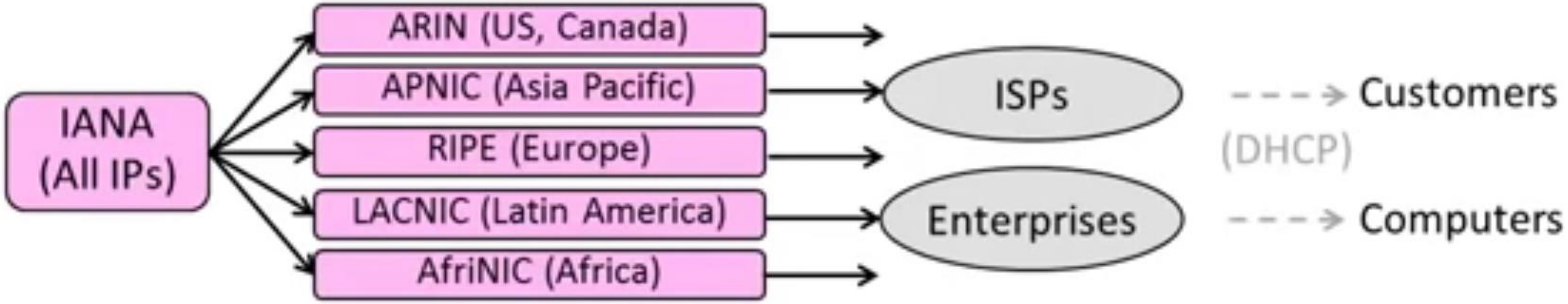


✓ Flexibility

✓ Making table compact

Allocating Public IP Addresses

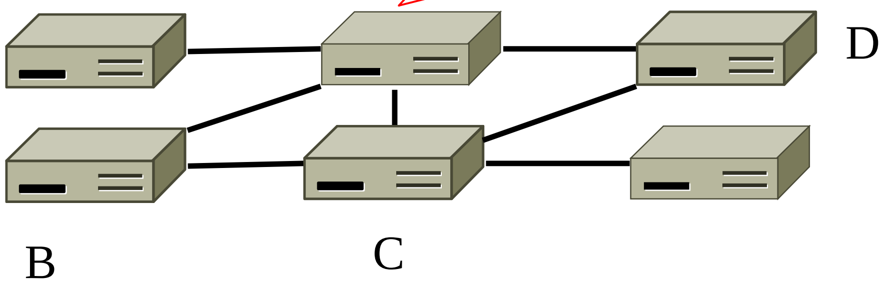
- Follows a hierarchical process
  - IANA delegates to regional bodies (RIRs)
  - RIRs delegate to companies in their region
  - Companies assign to their customers/computers (later, DHCP)



IP Forwarding

- IP addresses on one network belong to the same prefix
- Node uses a table that lists the next hop for IP prefixes

Prefix	Next Hop
192.24.0.0/18	D
192.24.12.0/22	B

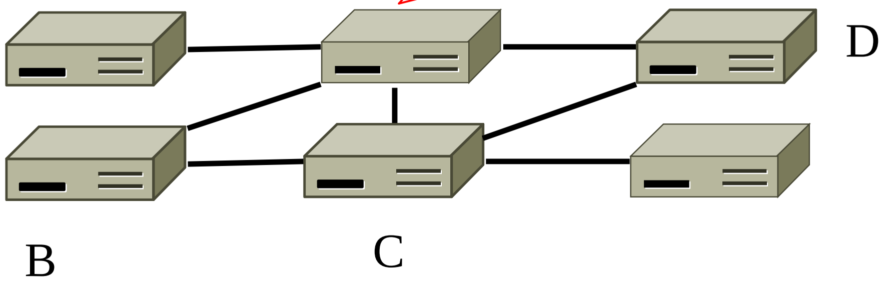


IP Forwarding

- IP addresses on one network belong to the same prefix
- Node uses a table that lists the next hop for IP prefixes

Prefix	Next Hop
192.24.0.0/18	D
192.24.12.0/22	B

192.24.32.1 → ?



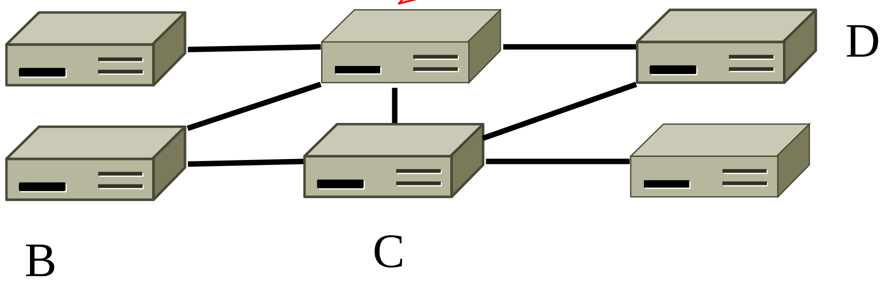
IP Forwarding

- IP addresses on one network belong to the same prefix
- Node uses a table that lists the next hop for IP prefixes

Prefix	Next Hop
192.24.0.0/18	D
192.24.12.0/22	B

192.24.32.1 → D

192.24.64.1 → ?



Longest Matching Prefix

- Prefixes in the table might overlap!
  - Combines hierarchy with flexibility

- Longest matching prefix forwarding rule:
  - For each packet, find the longest prefix that contains the destination address, i.e., the most specific entry
  - Forward the packet to the next hop router for that prefix

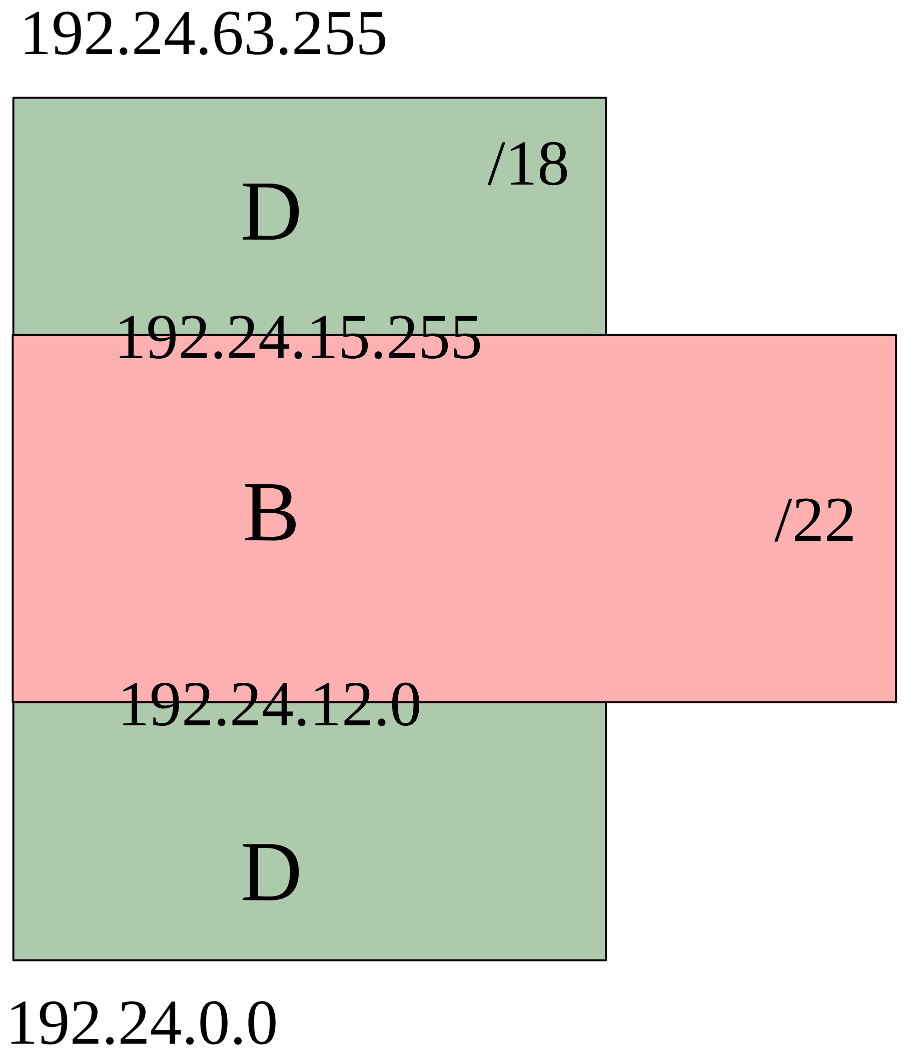
Longest Matching Prefix (2)

Prefix	Next Hop
192.24.0.0/18	D
192.24.12.0/22	B

192.24.6.0

192.24.14.32

192.24.54.0



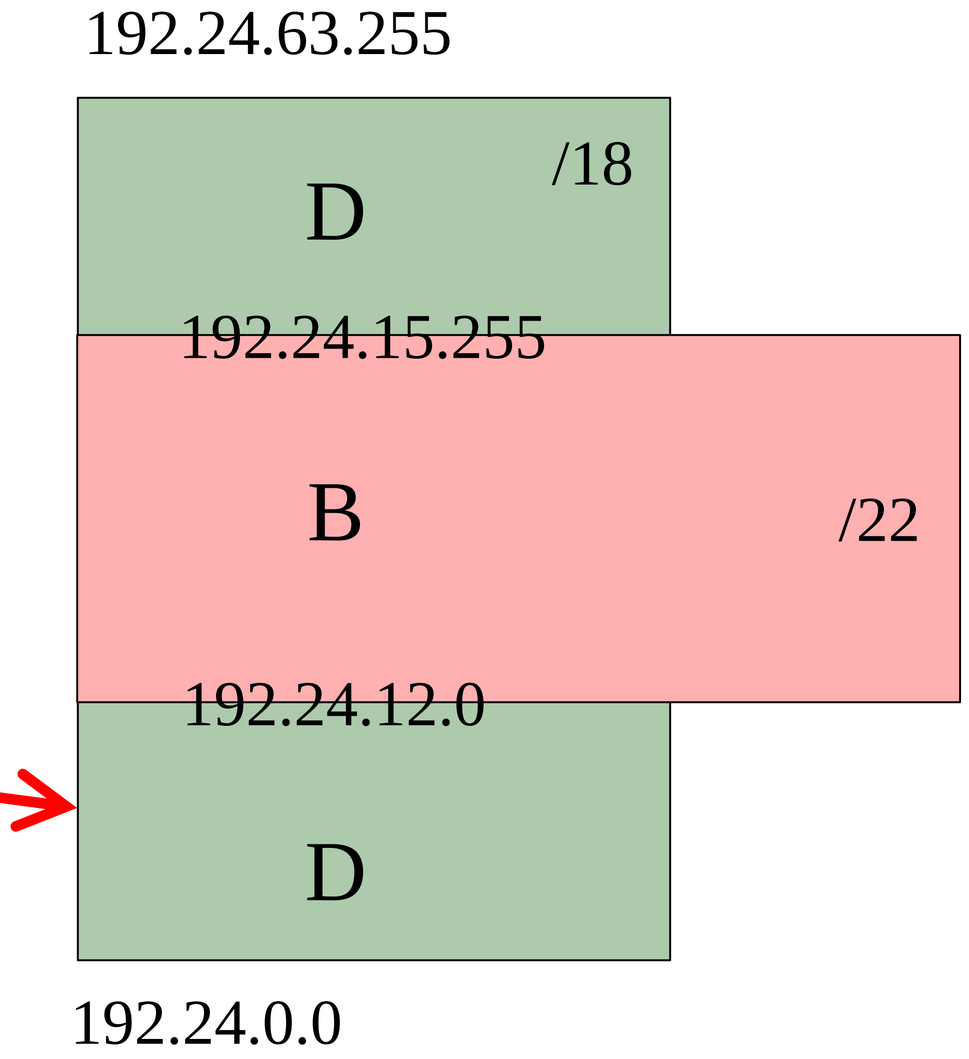
Longest Matching Prefix (2)

Prefix	Next Hop
192.24.0.0/18	D
192.24.12.0/22	B

192.24.6.0

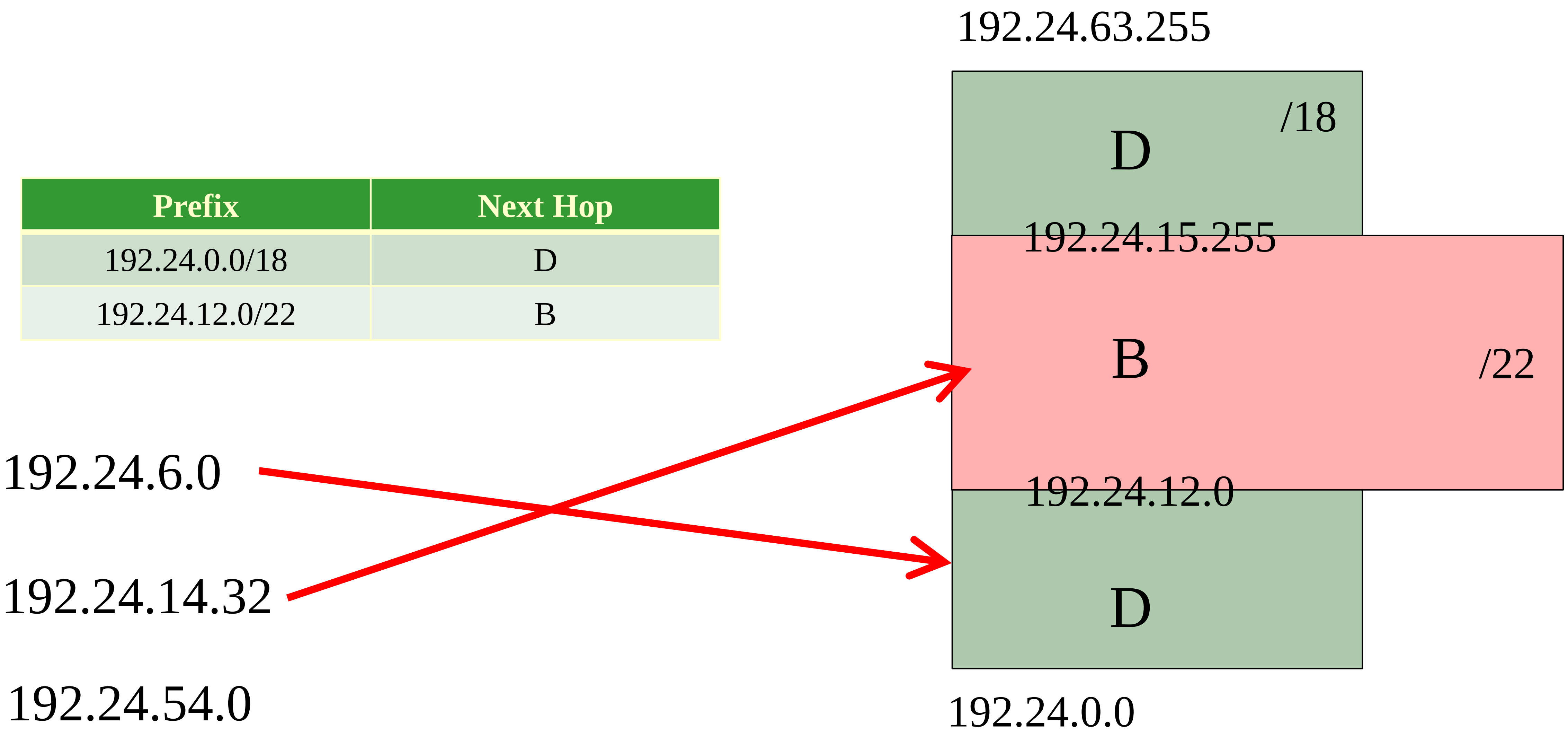
192.24.14.32

192.24.54.0



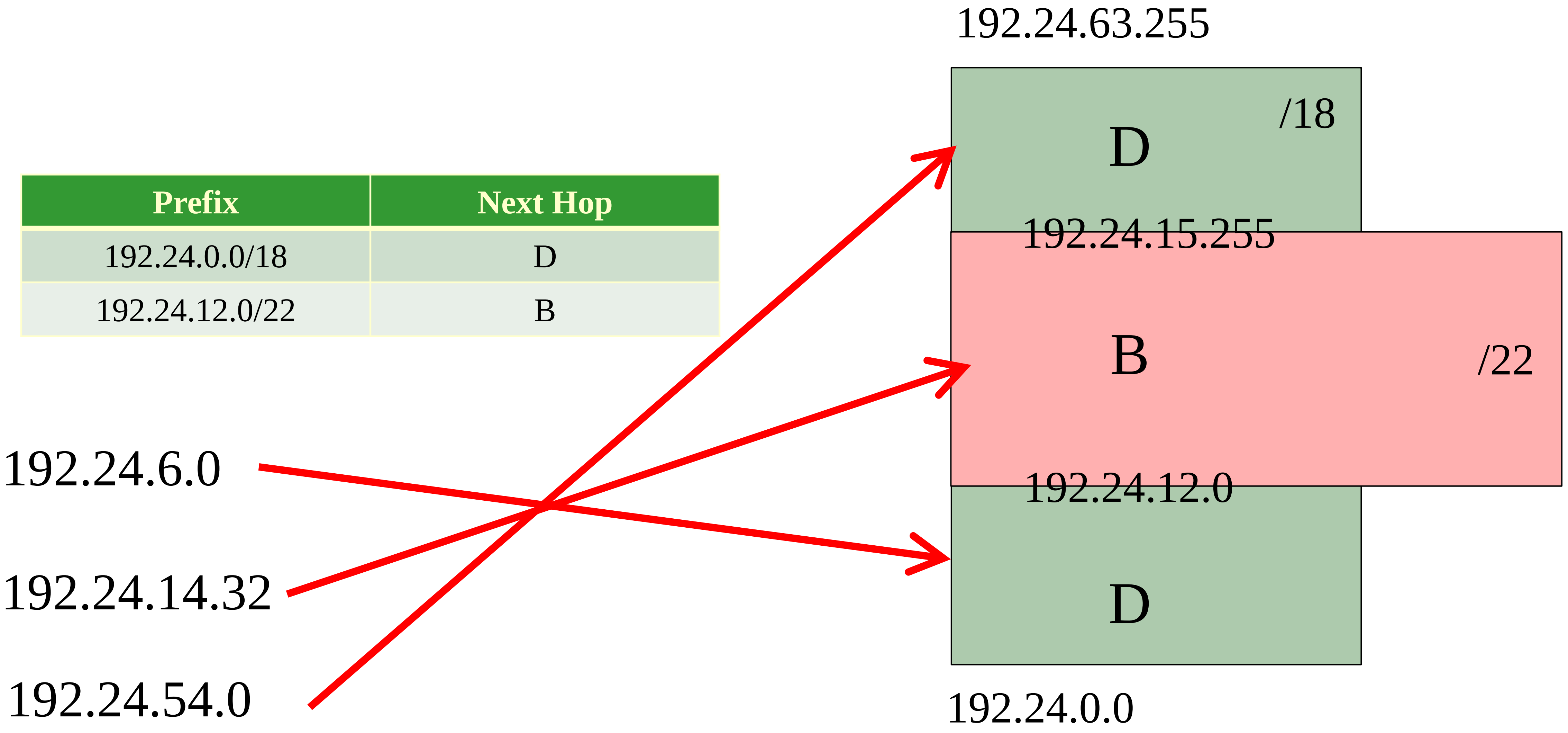
Longest Matching Prefix (2)

Prefix	Next Hop
192.24.0.0/18	D
192.24.12.0/22	B



Longest Matching Prefix (2)

Prefix	Next Hop
192.24.0.0/18	D
192.24.12.0/22	B

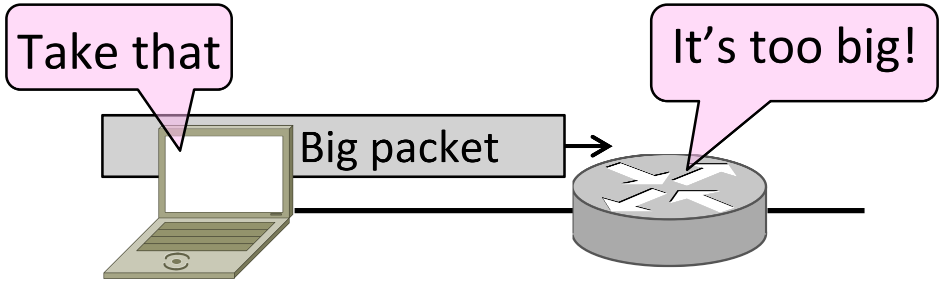


Roadmap

- Network layer Overview
- IP: Internet Protocol
  - internetworking and datagram format
  - IPv4 addressing and prefix-based forwarding
  - fragmentation and MTU discovery
  - ARP, DHCP, and NAT
  - ICMP
  - IPv6
- Routing

Topic

- How do we connect networks with different maximum packet sizes?
  - Need to split up packets, or discover the largest size to use



Packet Size Problem

- Different networks have different maximum packet sizes
  - Or MTU (Maximum Transmission Unit)
  - E.g., Ethernet 1.5K, WiFi 2.3K

- Prefer large packets for efficiency
  - But what size is too large?
  - Difficult because node does not know complete network path

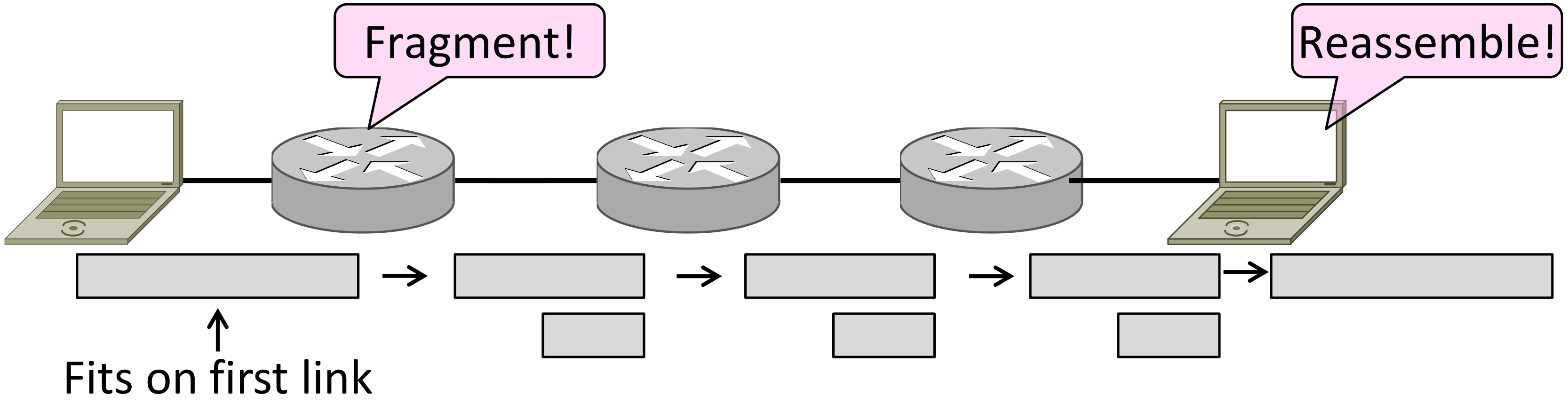
Packet Size Solutions

- Fragmentation (now)
  - Split up large packets in the network if they are too big to send
  - Classic method, dated

- Discovery (next)
  - Find the largest packet that fits on the network path and use it
  - IP uses today instead of fragmentation

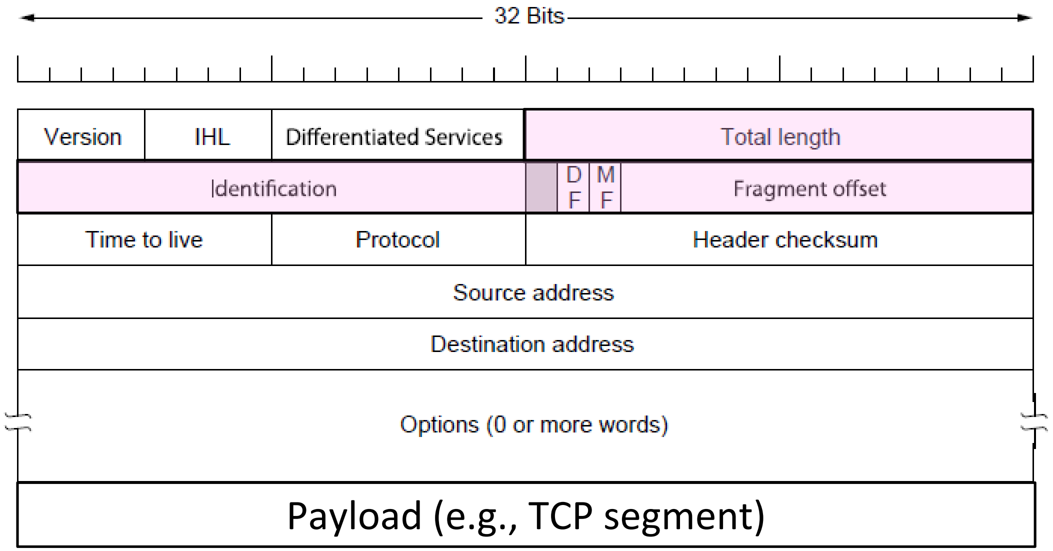
IPv4 Fragmentation

- Routers fragment packets that are too large to forward
- Receiving host reassembles to reduce load on routers



IPv4 Fragmentation Fields

- **Header fields used to handle packet size differences**
  - **Identification, Fragment offset, MF/DF control bits**



32 Bits

Version | IHL | Differentiated Services | Total length

Identification | DF | MF | Fragment offset

Time to live | Protocol | Header checksum

Source address

Destination address

Options (0 or more words)

Payload (e.g., TCP segment)

IPv4 Fragmentation Procedure

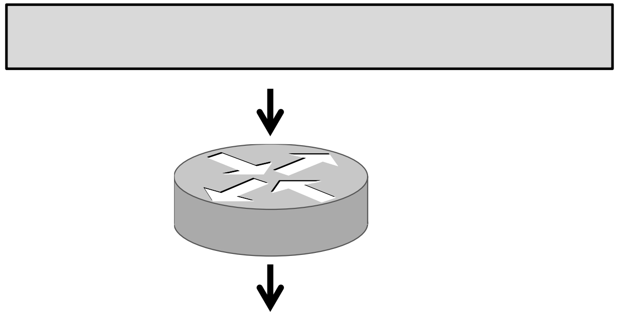
- Routers split a packet that is too large:
  - Typically break into large pieces
  - Copy IP header to pieces
  - Adjust length on pieces
  - Set offset to indicate position
  - Set MF (More Fragments) on all pieces except last

- Receiving hosts reassembles the pieces:
  - Identification field links pieces together, MF tells receiver when it has all pieces

IPv4 Fragmentation (2)

Before
MTU = 2300

ID = 0x12ef
Data Len = 2300
Offset = 0
MF = 0



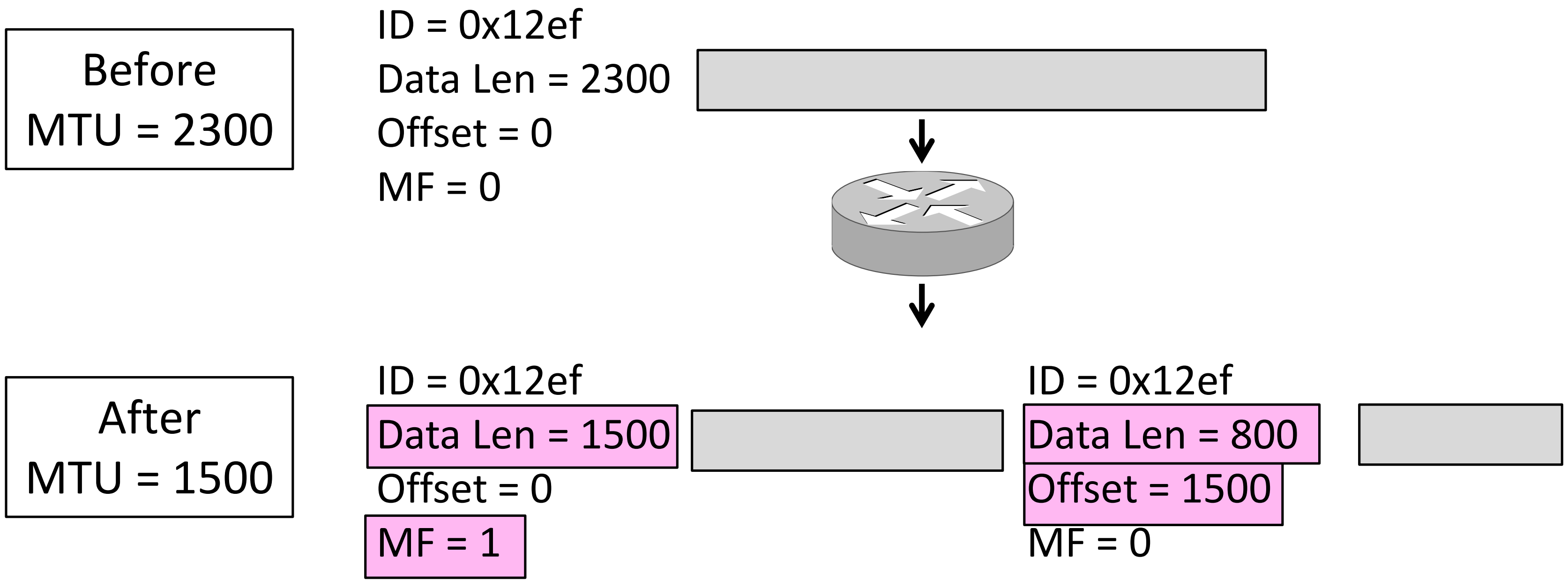
(Ignore length of headers)

After
MTU = 1500

ID =
Data Len =
Offset =
MF =

ID =
Data Len =
Offset =
MF =

IPv4 Fragmentation (3)



Before
MTU = 2300

ID = 0x12ef
Data Len = 2300
Offset = 0
MF = 0

After
MTU = 1500

ID = 0x12ef
Data Len = 1500
Offset = 0
MF = 1

ID = 0x12ef
Data Len = 800
Offset = 1500
MF = 0

IPv4 Fragmentation (4)

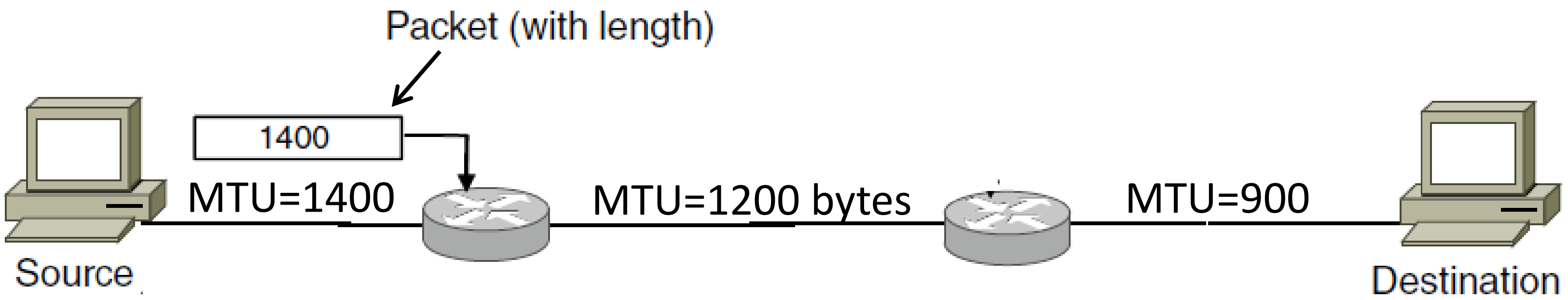
- **It works!**
  - Allows repeated fragmentation
- **But fragmentation is undesirable**
  - More work for routers, hosts
  - Tends to magnify loss rate
  - Security vulnerabilities too

Path MTU Discovery

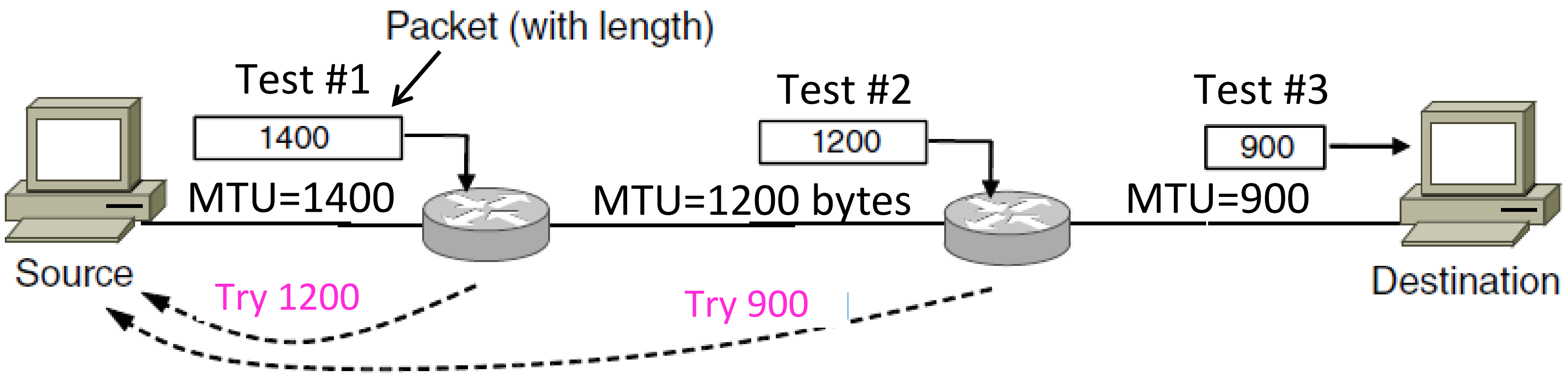
- Discover the MTU that will fit
  - So we can avoid fragmentation
  - The method in use today

- Host tests path with large packet
  - Routers provide feedback if too large; they tell host what size would have fit

Path MTU Discovery (2)



Path MTU Discovery (3)



Path MTU Discovery (4)

- Process may seem involved
  - But usually quick to find right size

- Path MTU depends on the path and so can change over time
  - Search is ongoing

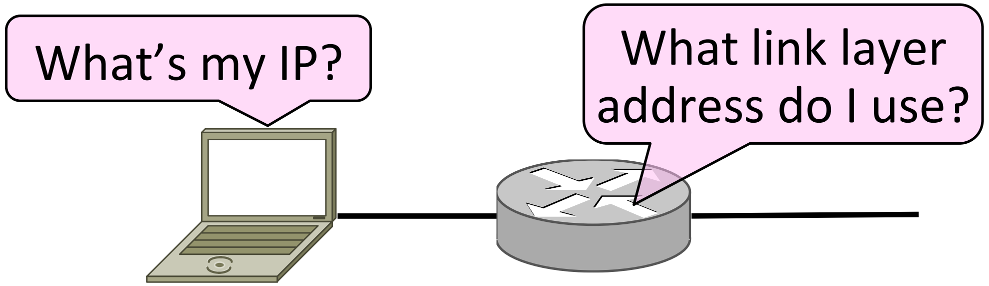
- Implemented with ICMP (next)
  - Set DF (Don’t Fragment) bit in IP header to get feedback messages

Roadmap

- Network layer Overview
- IP: Internet Protocol
  - internetworking and datagram format
  - IPv4 addressing and prefix-based forwarding
  - fragmentation and MTU discovery
  - ARP, DHCP, and NAT
  - ICMP
  - IPv6
- Routing

Topic

- Filling in the gaps we need to make for IP forwarding work in practice
    - Getting IP addresses (DHCP) »
    - Mapping IP to link addresses (ARP) »

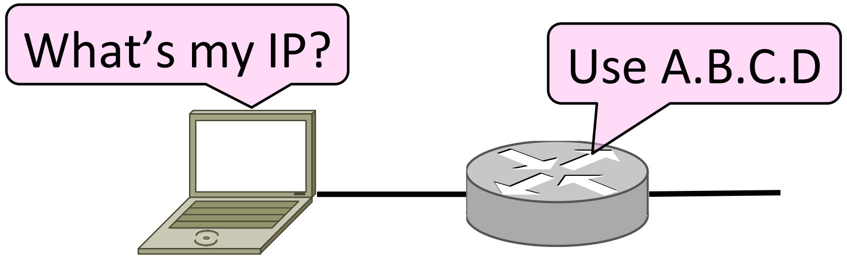


Getting IP Addresses

- **Problem:**
  - A node wakes up for the first time …
  - What is its IP address? What's the IP address of its router? Etc.
  - At least Ethernet address is on NIC

Getting IP Addresses (2)

1. Manual configuration (old days)
   - Can't be factory set, depends on use
2. A protocol for automatically configuring addresses (DHCP)
   - Shifts burden from users to IT folk

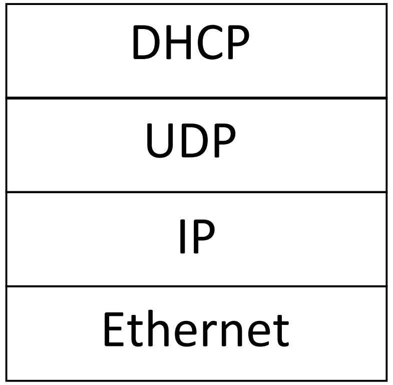


DHCP

- DHCP (Dynamic Host Configuration Protocol), from 1993, widely used
- It leases IP address to nodes
- Provides other parameters too
  - Network prefix
  - Address of local router
  - DNS server, time server, etc.

DHCP Protocol Stack

- **DHCP is a client-server application**
  - **Uses UDP ports 67, 68**



DHCP

UDP

IP

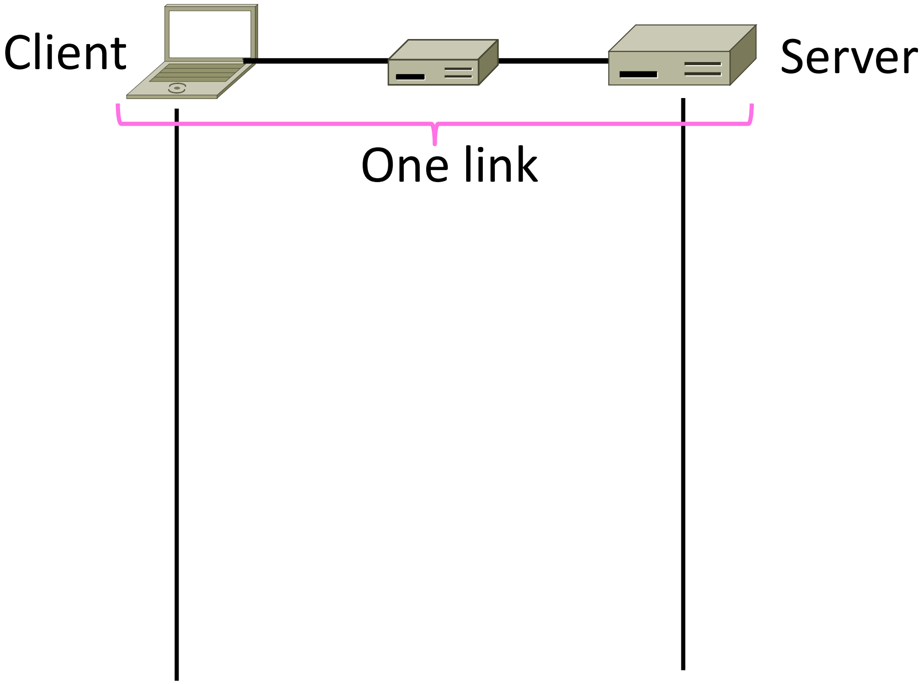
Ethernet

DHCP Addressing

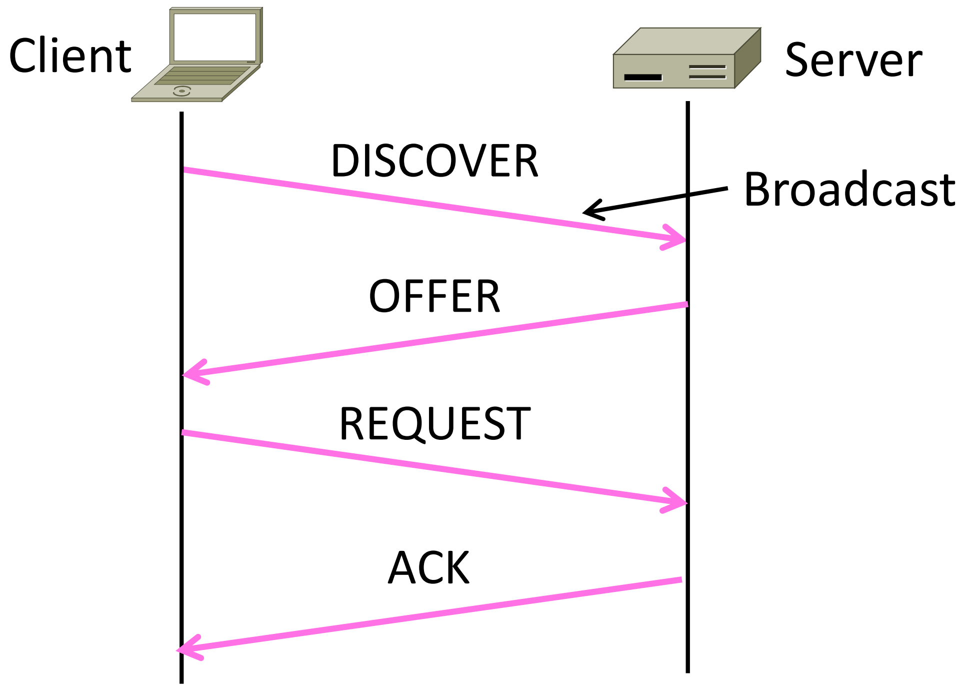
- Bootstrap issue:
  - How does node send a message to DHCP server before it is configured?

- Answer:
  - Node sends broadcast messages that delivered to all nodes on the network
  - Broadcast address is all 1s
  - IP (32 bit): 255.255.255.255
  - Ethernet (48 bit): ff:ff:ff:ff:ff:ff

DHCP Messages



DHCP Messages (2)



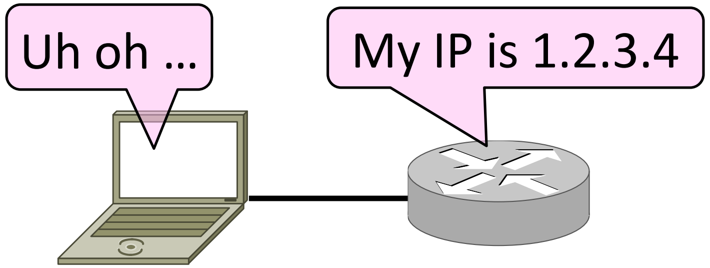
DHCP Messages (3)

- To renew an existing lease, an abbreviated sequence is used:
  - REQUEST, followed by ACK

- Protocol also supports replicated servers for reliability

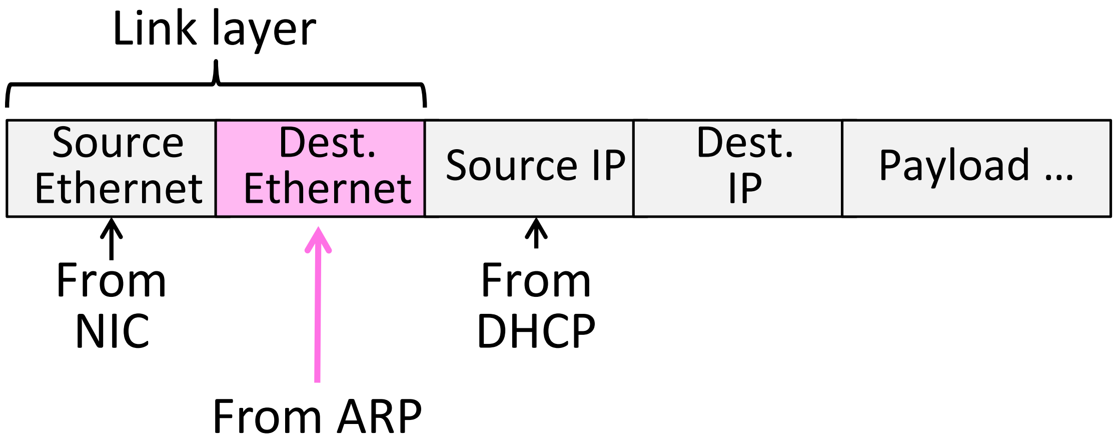
Sending an IP Packet

- **Problem:**
  - A node needs Link layer addresses to send a frame over the local link
  - How does it get the destination link address from a destination IP address?



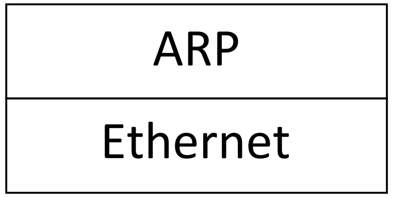
ARP (Address Resolution Protocol)

- **Node uses to map a local IP address to its Link layer addresses**



ARP Protocol Stack

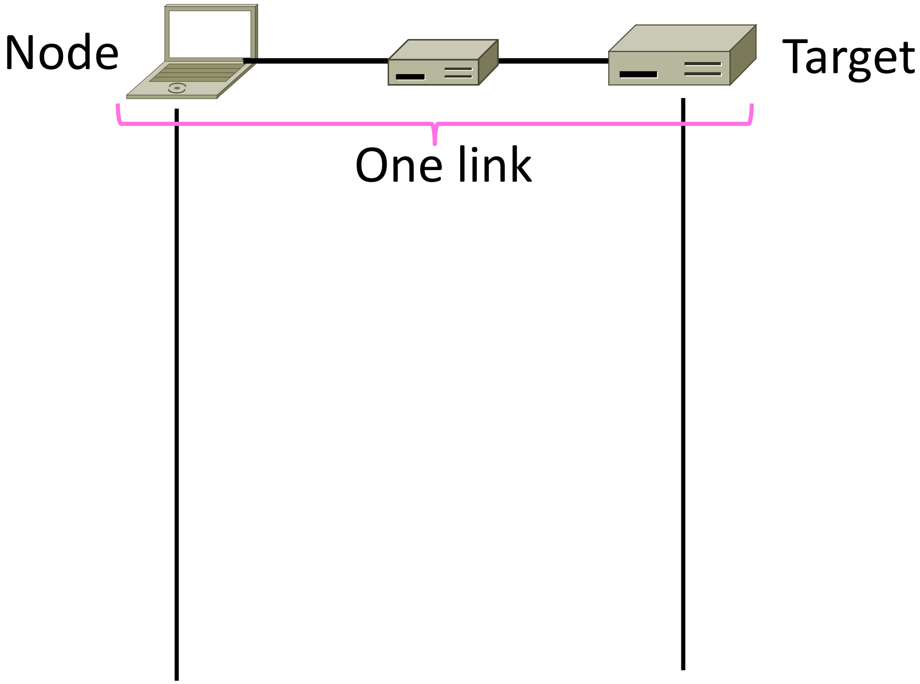
- **ARP sits right on top of link layer**
  - **No servers, just asks node with target IP to identify itself**
  - **Uses broadcast to reach all nodes**



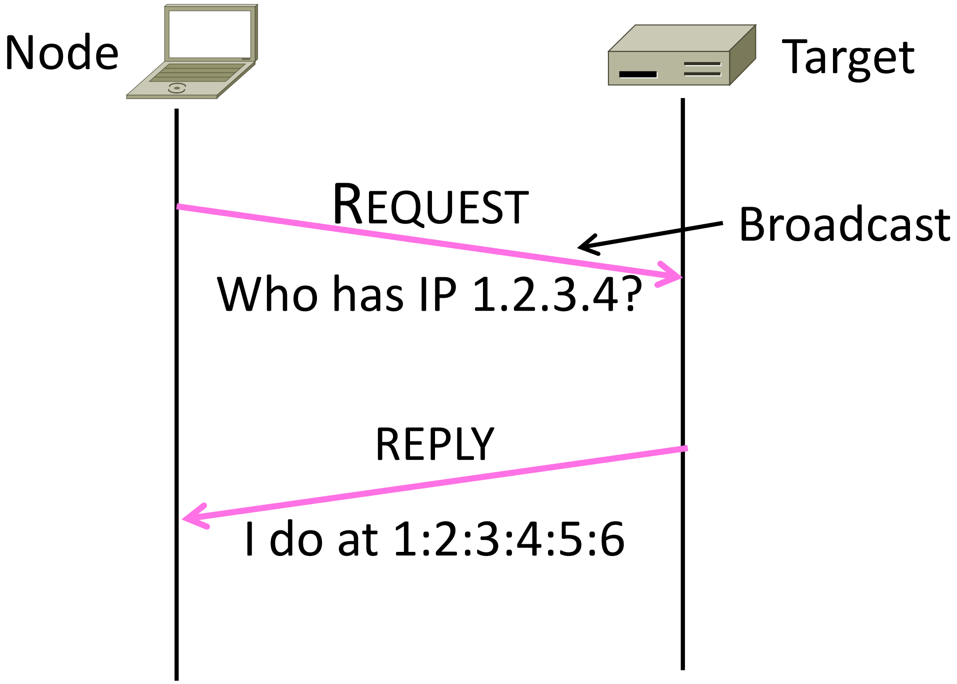
ARP

Ethernet

ARP Messages



ARP Messages (2)



Node

Target

REQUEST

Broadcast

Who has IP 1.2.3.4?

REPLY

I do at 1:2:3:4:5:6

Discovery Protocols

- Help nodes find each other
  - There are more of them!
    - E.g., zeroconf, Bonjour
- Often involve broadcast
  - Since nodes aren't introduced
  - Very handy glue

Roadmap

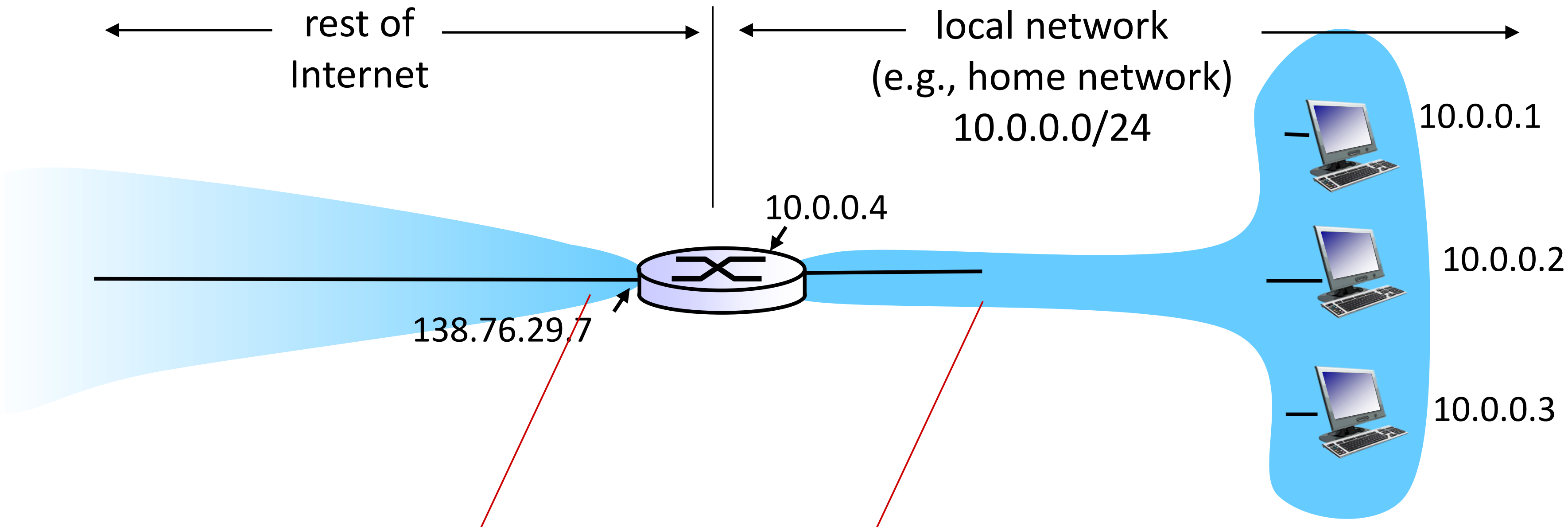
- Network layer Overview
- IP: Internet Protocol
  - internetworking and datagram format
  - IPv4 addressing and prefix-based forwarding
  - fragmentation and MTU discovery
  - ARP, DHCP, and NAT
  - ICMP
  - IPv6
- Routing

Public / Private IP Addresses

- Public IP addresses, e.g., 18.31.0.1
  - Valid destination on the global Internet
  - Must be allocated to you before use

- Private IP addresses
  - 10.0.0.0/8, 172.16.0.0/12, 192.168.0.0/16
  - Can be used freely within private networks (home, small company)

NAT: network address translation



all datagrams *leaving* local network have *same* single source NAT IP address: 138.76.29.7,different source port numbers

datagrams with source or destination in this network have 10.0.0.0/24 address for source, destination (as usual)

NAT: network address translation

implementation: NAT router must:

- *outgoing datagrams: replace* (source IP address, port #) of every outgoing datagram to (NAT IP address, new port #)

  . . . remote clients/servers will respond using (NAT IP address, new port #) as destination addr

- *remember (in NAT translation table)*

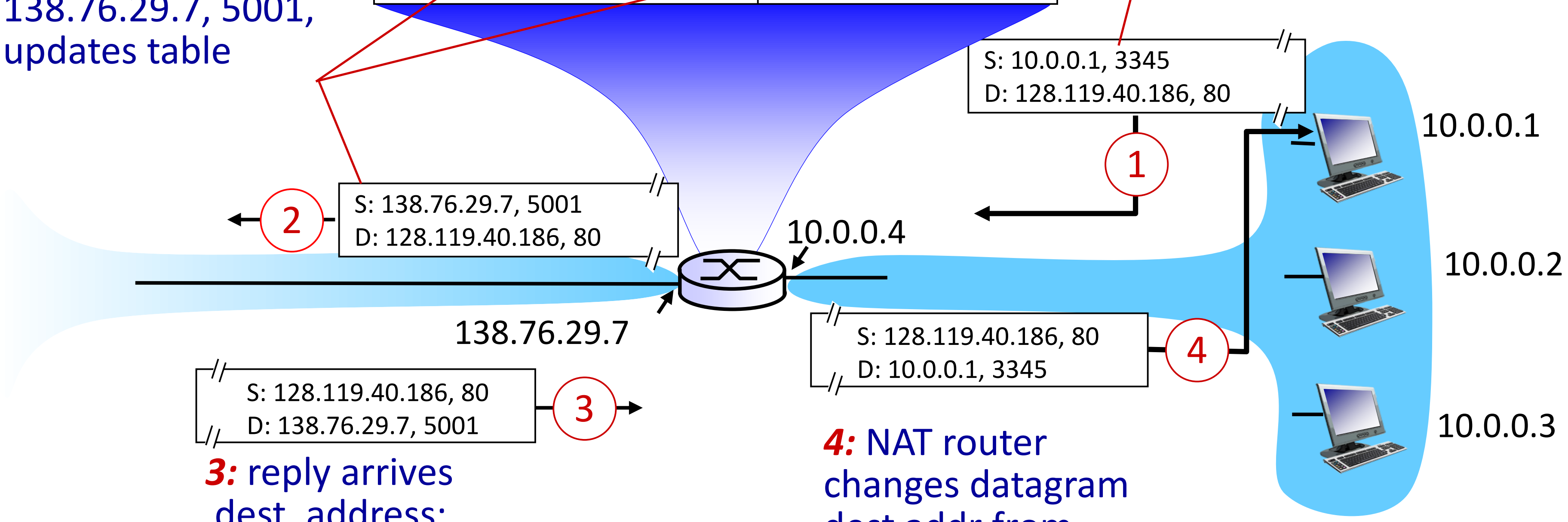
every translation pair, i.e., (source IP address, port #) to (NAT IP address, new port #)

- *incoming datagrams: replace* (NAT IP address, new port #) in dest fields of every incoming datagram with corresponding (source IP address, port #) stored in NAT table

NAT: network address translation

NAT translation table

WAN side addr	LAN side addr
138.76.29.7, 5001	10.0.0.1, 3345
……	……



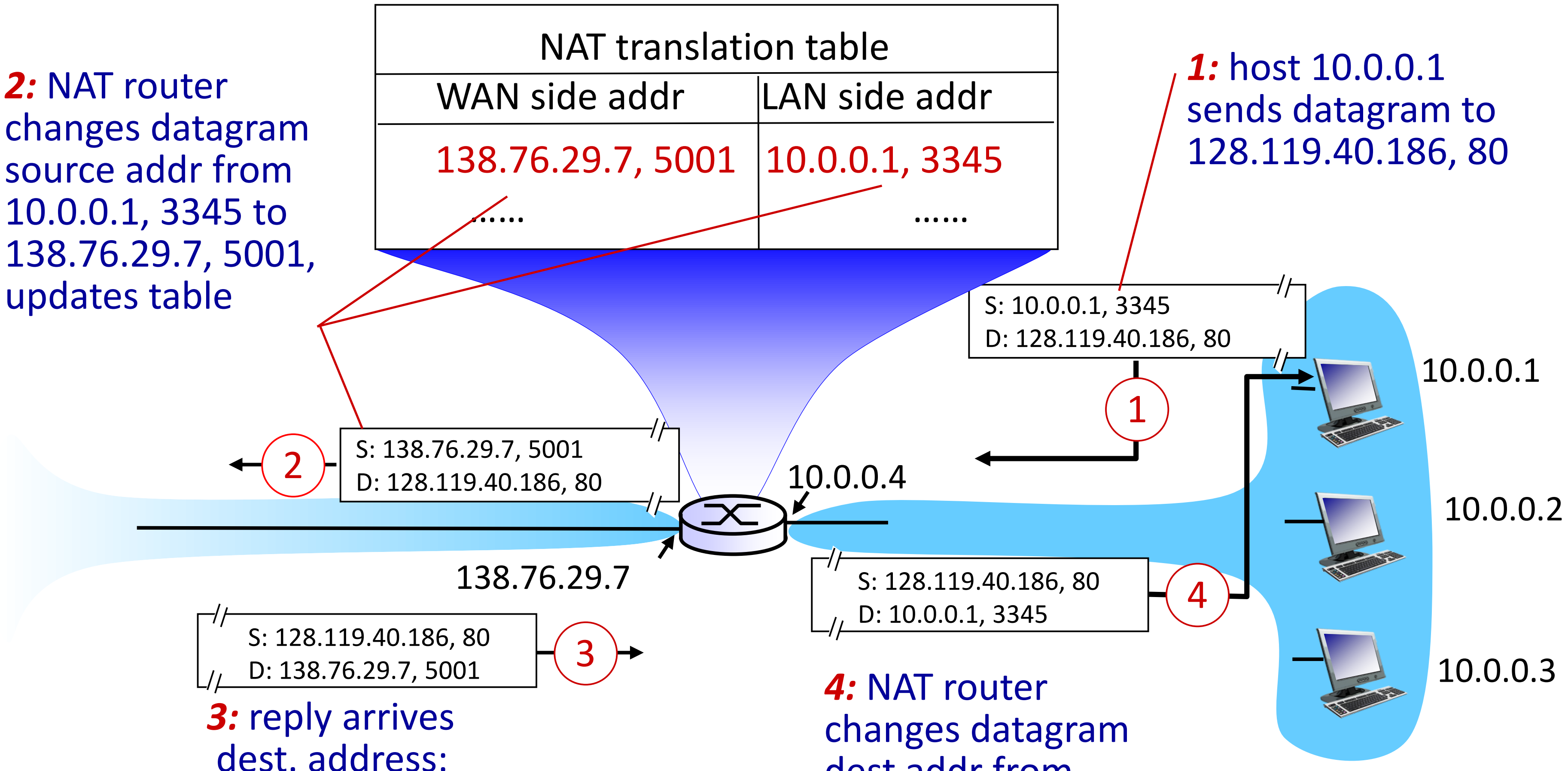
1: host 10.0.0.1 sends datagram to 128.119.40.186, 80

2: NAT router changes datagram source addr from 10.0.0.1, 3345 to 138.76.29.7, 5001, updates table

3: reply arrives dest. address: 138.76.29.7, 5001

4: NAT router changes datagram dest addr from 138.76.29.7, 5001 to 10.0.0.1, 3345

NAT: network address translation



NAT translation table

WAN side addr	LAN side addr
138.76.29.7, 5001	10.0.0.1, 3345
……	……

2: NAT router changes datagram source addr from 10.0.0.1, 3345 to 138.76.29.7, 5001, updates table

1: host 10.0.0.1 sends datagram to 128.119.40.186, 80

3: reply arrives dest. address: 138.76.29.7, 5001

4: NAT router changes datagram dest addr from 138.76.29.7, 5001 to 10.0.0.1, 3345

只有当内网主机访问外网时才会建立一个NAT translation pair =>外网无法主动访问内网主机

NAT: network address translation

Upside: local network uses just one IP address as far as outside world is concerned:

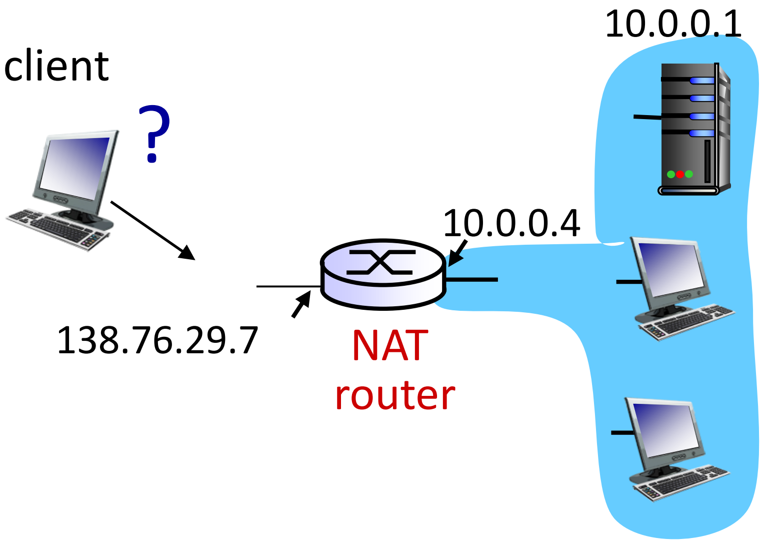
- range of addresses not needed from ISP: just one IP address for all devices
- 16-bit port-number field = 60,000 simultaneous connections with a single LAN-side address!
- devices inside local net not explicitly addressable, visible by outside world (a security plus)

NAT: network address translation

- *Downside*: NAT is controversial,
  - routers should only process up to layer 3
  - hinder peer-to-peer communication
  - address shortage should instead be solved by IPv6

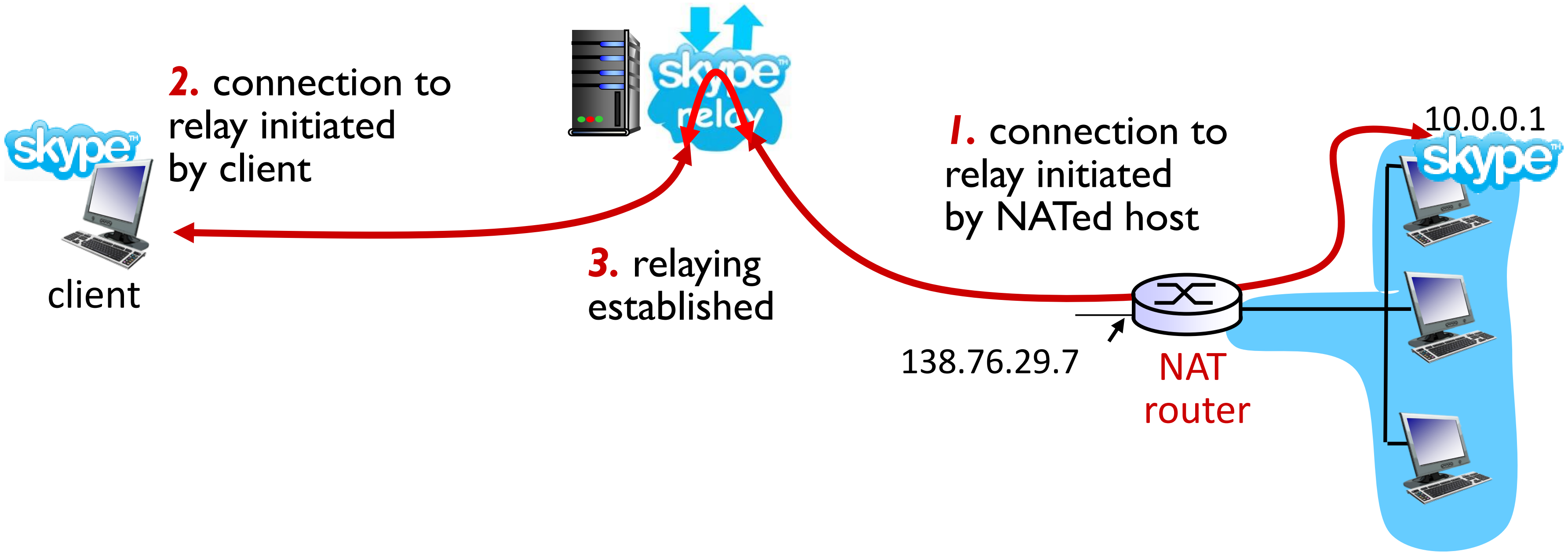
NAT traversal problem

- client wants to connect to server with address 10.0.0.1
  - server address 10.0.0.1 local to LAN (client can't use it as destination addr)
  - only one externally visible NATed address: 138.76.29.7
- *solution1:* statically configure NAT to forward incoming connection requests at given port to server
  - e.g., (123.76.29.7, port 2500) always forwarded to 10.0.0.1 port 25000



NAT traversal problem

- *solution 2:* relaying (used in Skype)
  - NATed client establishes connection to relay
  - external client connects to relay
  - relay bridges packets between two connections



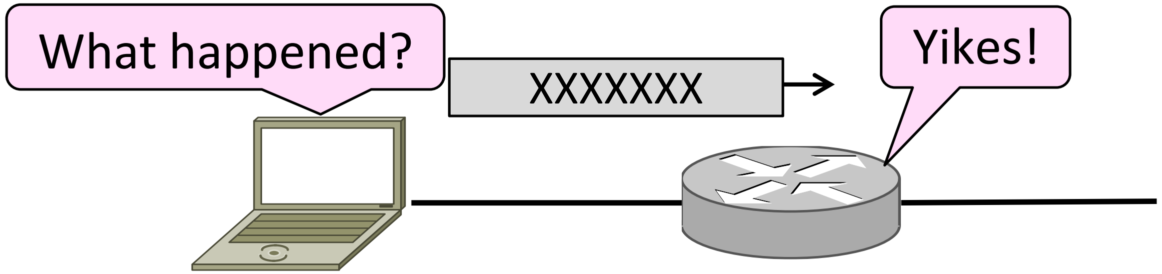
2. connection to relay initiated by client
skype
skype relay
client
3. relaying established
1. connection to relay initiated by NATed host
10.0.0.1
skype
138.76.29.7
NAT router

Roadmap

- Network layer Overview
- IP: Internet Protocol
  - internetworking and datagram format
  - IPv4 addressing and prefix-based forwarding
  - fragmentation and MTU discovery
  - ARP, DHCP, and NAT
  - ICMP
  - IPv6
- Routing

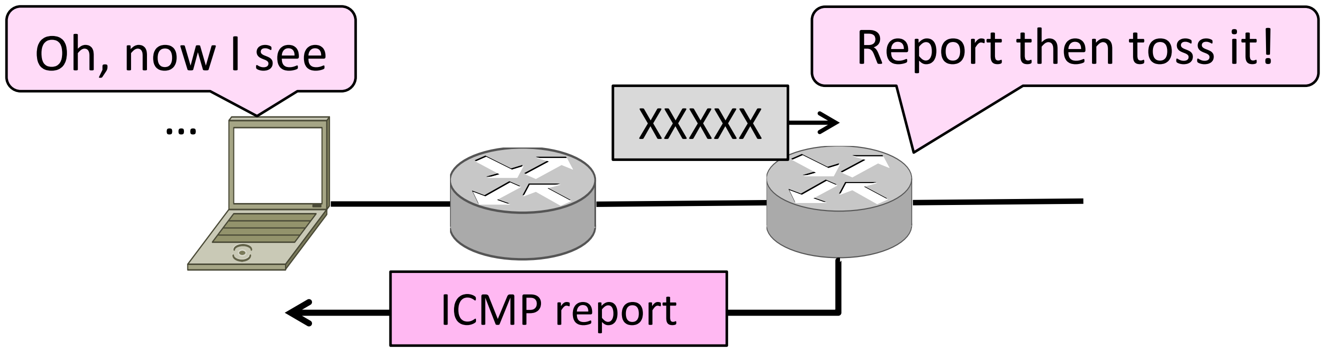
Topic

- What happens when something goes wrong during forwarding?
  - Need to be able to find the problem



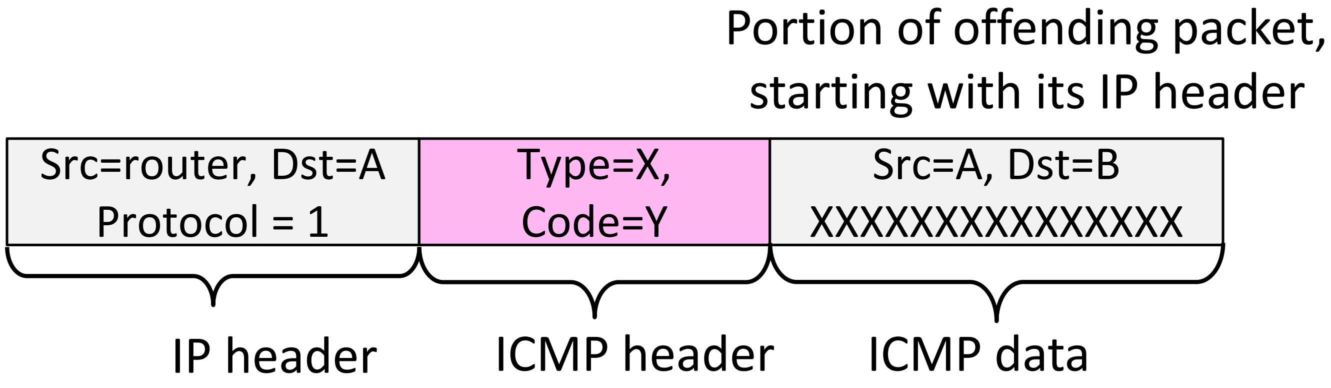
Internet Control Message Protocol

- When router encounters an error while forwarding:
  - It sends an ICMP error report back to the IP source address
  - It discards the problematic packet; host needs to rectify



ICMP Message Format

- Each ICMP message has a Type, Code, and Checksum
- Often carry the start of the offending packet as payload
- Each message is carried in an IP packet



Example ICMP Messages

Name	**Type / Code**	**Usage**
Dest. Unreachable (Net or Host)	3 / 0 or 1	Lack of connectivity
Dest. Unreachable (Fragment)	3 / 4	Path MTU Discovery
Time Exceeded (Transit)	11 / 0	Traceroute
Echo Request or Reply	8 or 0 / 0	Ping

Testing, not a forwarding error: Host sends Echo Request, and destination responds with an Echo Reply

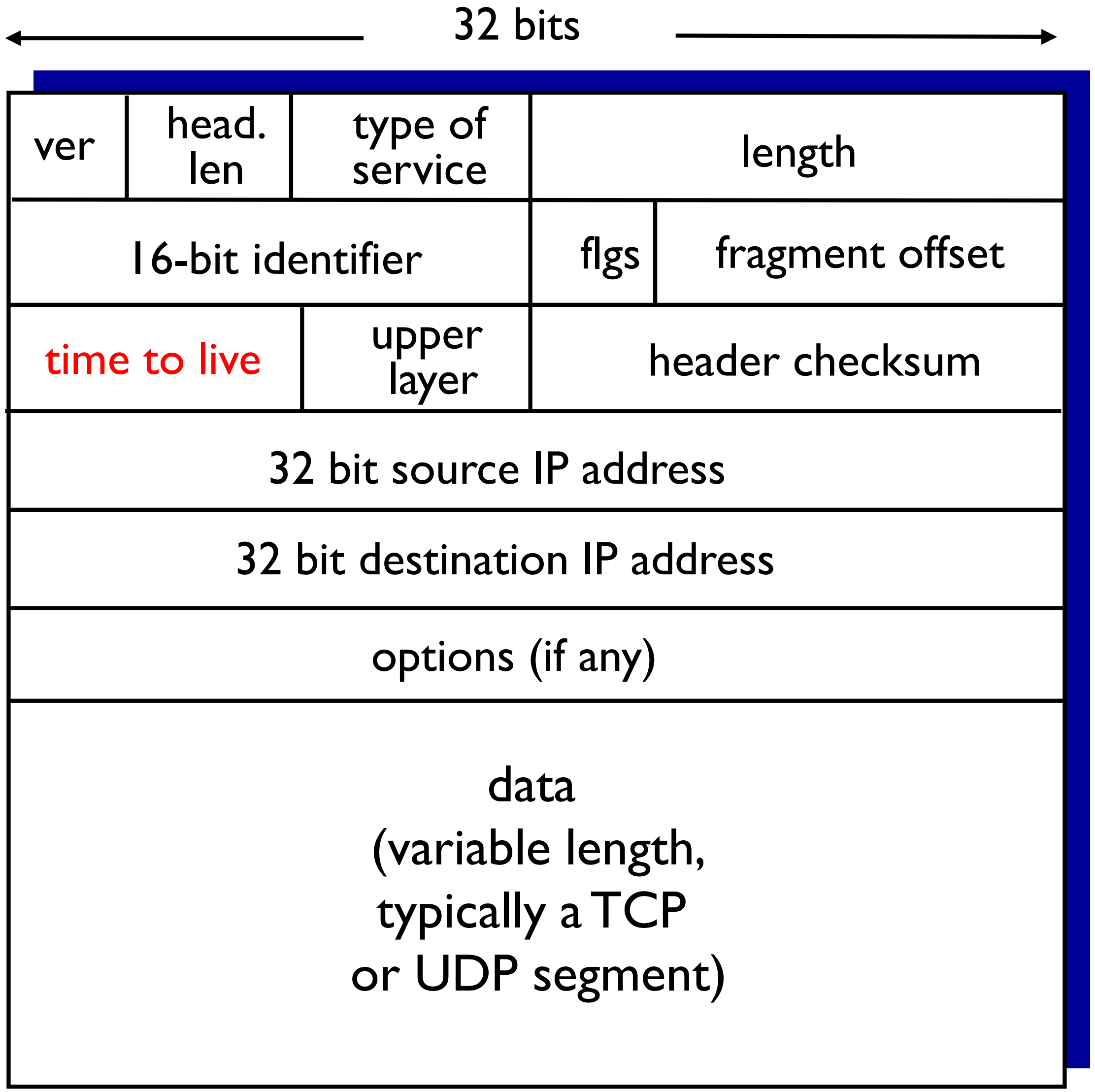
Traceroute

Goal

- for displaying the route (path) and measuring transit delays of packets across an Internet Protocol (IP) network.

How to do it?

- IP header contains TTL (Time to live) field
  - Decremented every router hop, with ICMP error if it hits zero
  - Protects against forwarding loops



32 bits

ver

head. len

type of service

length

16-bit identifier

flgs

fragment offset

time to live

upper layer

header checksum

32 bit source IP address

32 bit destination IP address

options (if any)

data (variable length, typically a TCP or UDP segment)

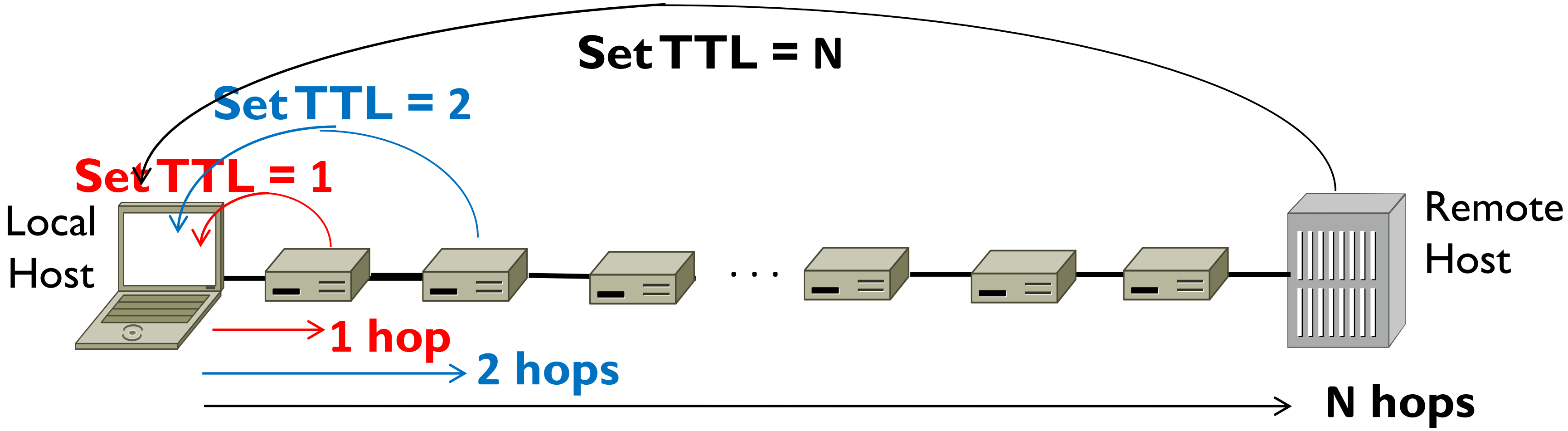
Traceroute

- Source sends series of UDP segments to dest
  - first set has TTL =1
  - second set has TTL=2, etc.
  - unlikely port number
- When nth datagram arrives to nth router:
  - router discards datagrams
  - sends source ICMP msg (type 11, code 0)
  - ICMP msgincludes name of router & IP addr

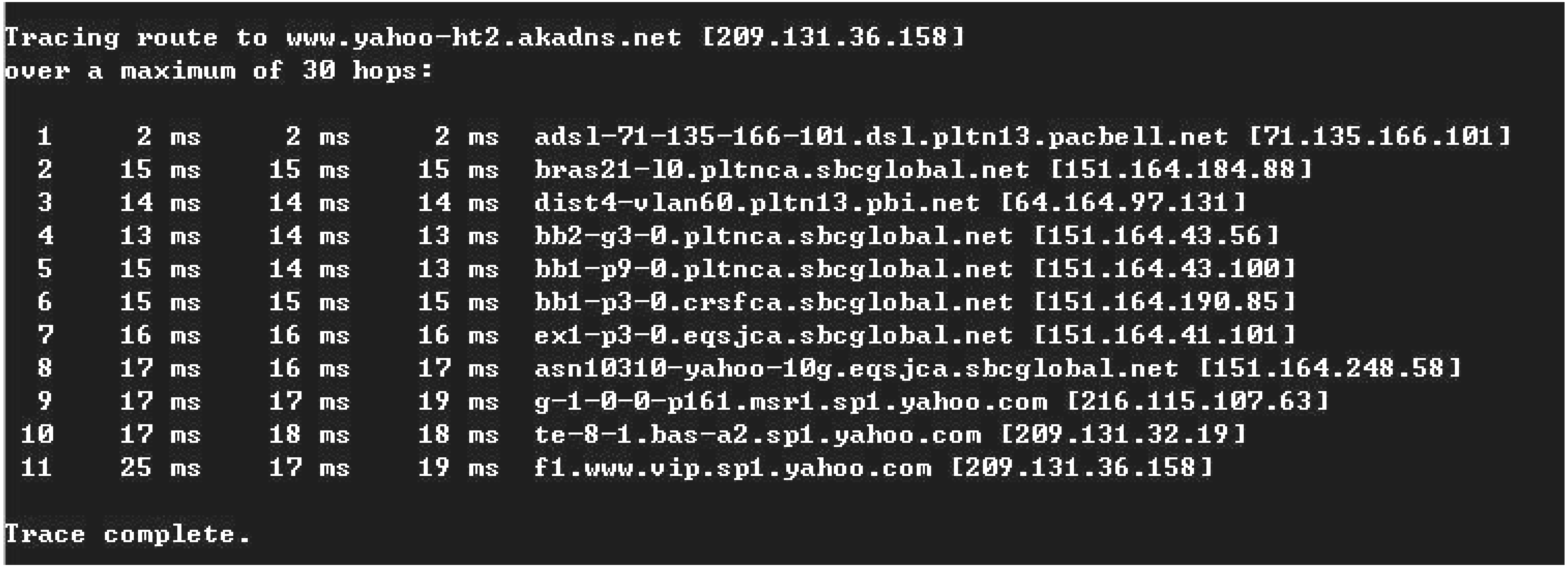
- when ICMP messages arrives, source records RTTs (Round of Trip Times)

stopping criteria:

- UDP segment eventually arrives at destination host
- destination returns ICMP "port unreachable" message (type 3, code 3)
- source stops



Traceroute

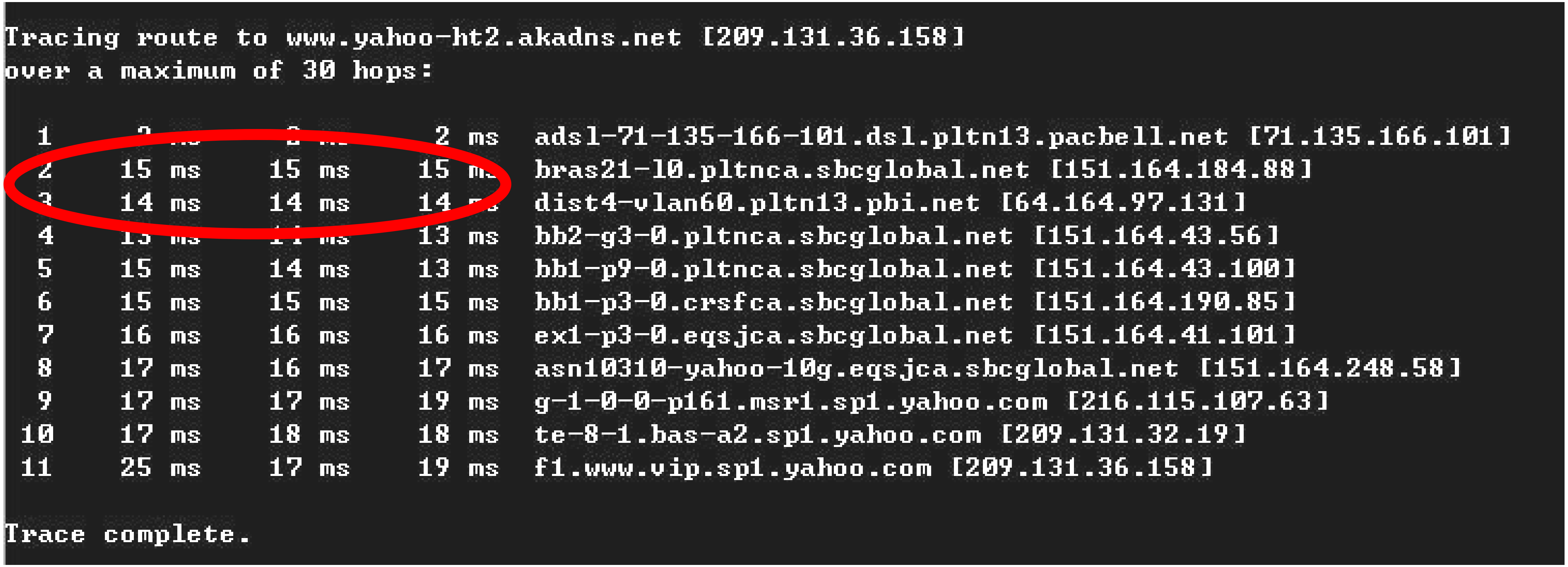


```
Tracing route to www.yahoo-ht2.akadns.net [209.131.36.158]
over a maximum of 30 hops:

  1     2 ms     2 ms     2 ms  adsl-71-135-166-101.dsl.pltn13.pacbell.net [71.135.166.101]
  2    15 ms    15 ms    15 ms  bras21-l0.pltnca.sbcglobal.net [151.164.184.88]
  3    14 ms    14 ms    14 ms  dist4-vlan60.pltn13.pbi.net [64.164.97.131]
  4    13 ms    14 ms    13 ms  bb2-g3-0.pltnca.sbcglobal.net [151.164.43.56]
  5    15 ms    14 ms    13 ms  bb1-p9-0.pltnca.sbcglobal.net [151.164.43.100]
  6    15 ms    15 ms    15 ms  bb1-p3-0.crsfca.sbcglobal.net [151.164.190.85]
  7    16 ms    16 ms    16 ms  ex1-p3-0.eqsjca.sbcglobal.net [151.164.41.101]
  8    17 ms    16 ms    17 ms  asn10310-yahoo-10g.eqsjca.sbcglobal.net [151.164.248.58]
  9    17 ms    17 ms    19 ms  g-1-0-0-p161.msr1.sp1.yahoo.com [216.115.107.63]
 10    17 ms    18 ms    18 ms  te-8-1.bas-a2.sp1.yahoo.com [209.131.32.19]
 11    25 ms    17 ms    19 ms  f1.www.vip.sp1.yahoo.com [209.131.36.158]

Trace complete.
```

Traceroute



```
Tracing route to www.yahoo-ht2.akadns.net [209.131.36.158]
over a maximum of 30 hops:

  1     2 ms     2 ms     2 ms  adsl-71-135-166-101.dsl.pltn13.pacbell.net [71.135.166.101]
  2    15 ms    15 ms    15 ms  bras21-l0.pltnca.sbcglobal.net [151.164.184.88]
  3    14 ms    14 ms    14 ms  dist4-vlan60.pltn13.pbi.net [64.164.97.131]
  4    13 ms    14 ms    13 ms  bb2-g3-0.pltnca.sbcglobal.net [151.164.43.56]
  5    15 ms    14 ms    13 ms  bb1-p9-0.pltnca.sbcglobal.net [151.164.43.100]
  6    15 ms    15 ms    15 ms  bb1-p3-0.crsfca.sbcglobal.net [151.164.190.85]
  7    16 ms    16 ms    16 ms  ex1-p3-0.eqsjca.sbcglobal.net [151.164.41.101]
  8    17 ms    16 ms    17 ms  asn10310-yahoo-10g.eqsjca.sbcglobal.net [151.164.248.58]
  9    17 ms    17 ms    19 ms  g-1-0-0-p161.msr1.sp1.yahoo.com [216.115.107.63]
 10    17 ms    18 ms    18 ms  te-8-1.bas-a2.sp1.yahoo.com [209.131.32.19]
 11    25 ms    17 ms    19 ms  f1.www.vip.sp1.yahoo.com [209.131.36.158]

Trace complete.
```

Roadmap

- Network layer Overview
- IP: Internet Protocol
  - internetworking and datagram format
  - IPv4 addressing and prefix-based forwarding
  - fragmentation and MTU discovery
  - ARP, DHCP, and NAT
  - ICMP
  - IPv6
- Routing

IPv6: motivation

- *initial motivation:* 32-bit address space soon to be completely allocated.
- additional motivation:
  - header format helps speed processing/forwarding
  - header changes to facilitate QoS

IPv6 datagram format:

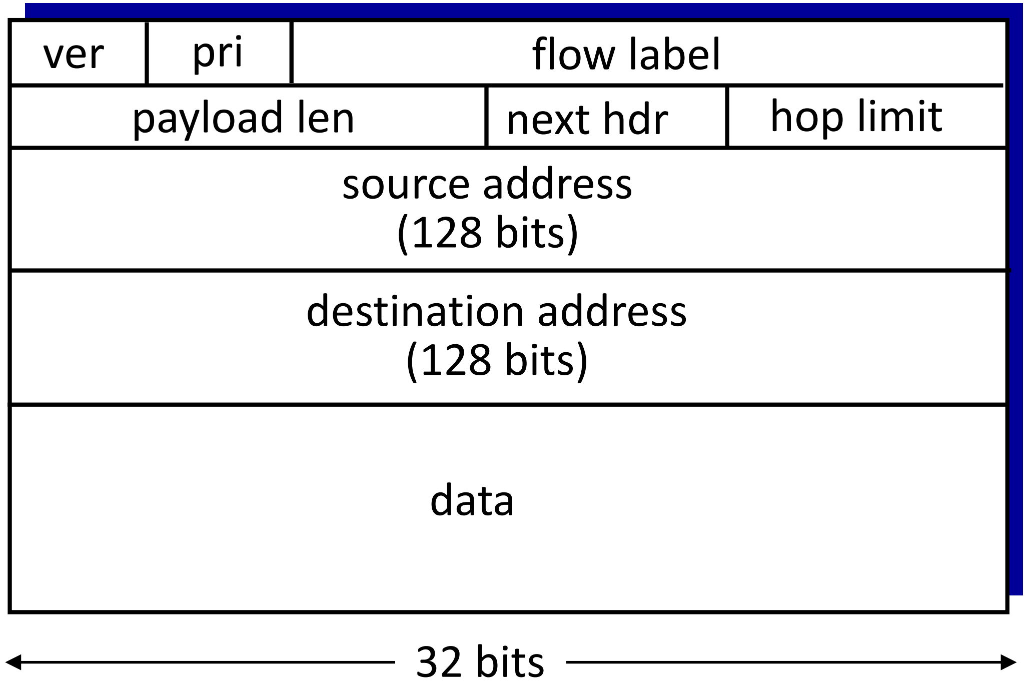
- fixed-length 40 byte header
- no fragmentation allowed

IPv6 datagram format

priority: identify priority among datagrams in flow

flow Label: identify datagrams in same “flow.”
(concept of“flow” not well defined).

next header: identify upper layer protocol for data



ver | pri | flow label

payload len | next hdr | hop limit

source address
(128 bits)

destination address
(128 bits)

data

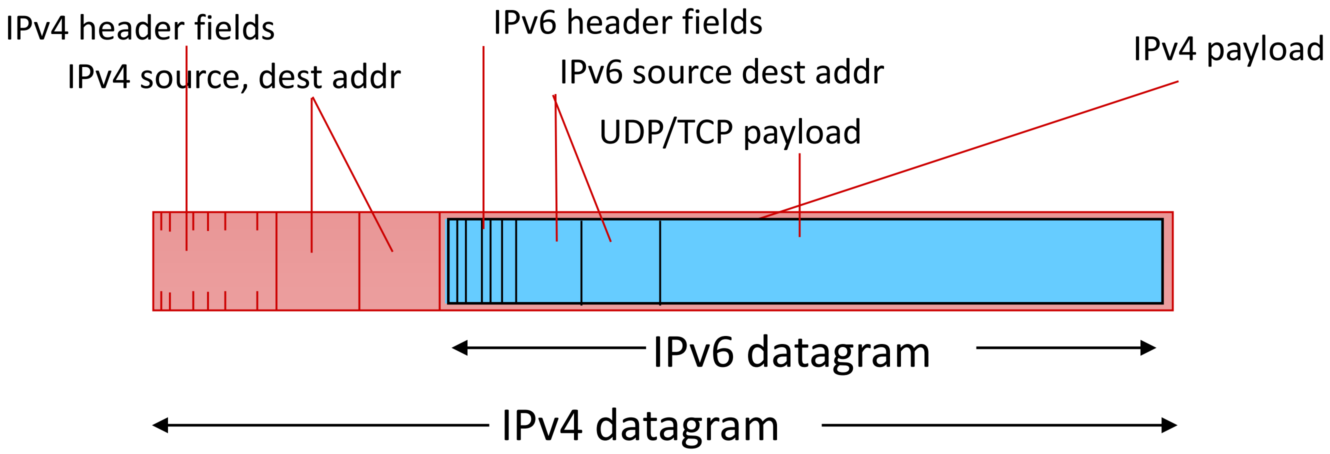
32 bits

Other changes from IPv4

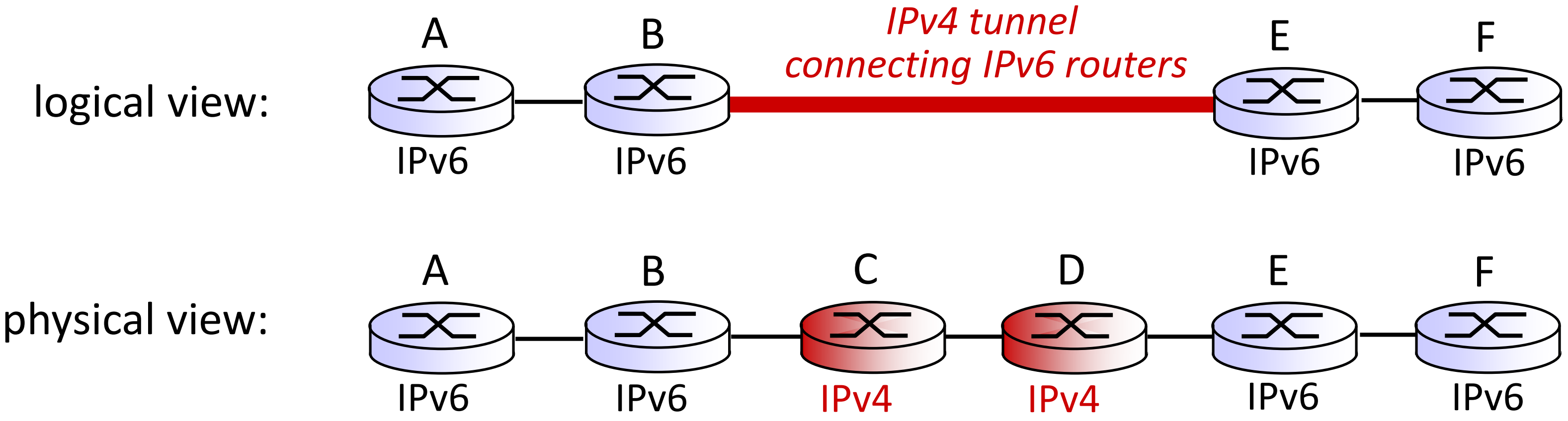
- *checksum:* removed entirely to reduce processing time at each hop
- *options:* allowed, but outside of header, indicated by "Next Header" field
- *ICMPv6:* new version of ICMP
  - additional message types, e.g. "Packet Too Big"
  - multicast group management functions

Transition from IPv4 to IPv6

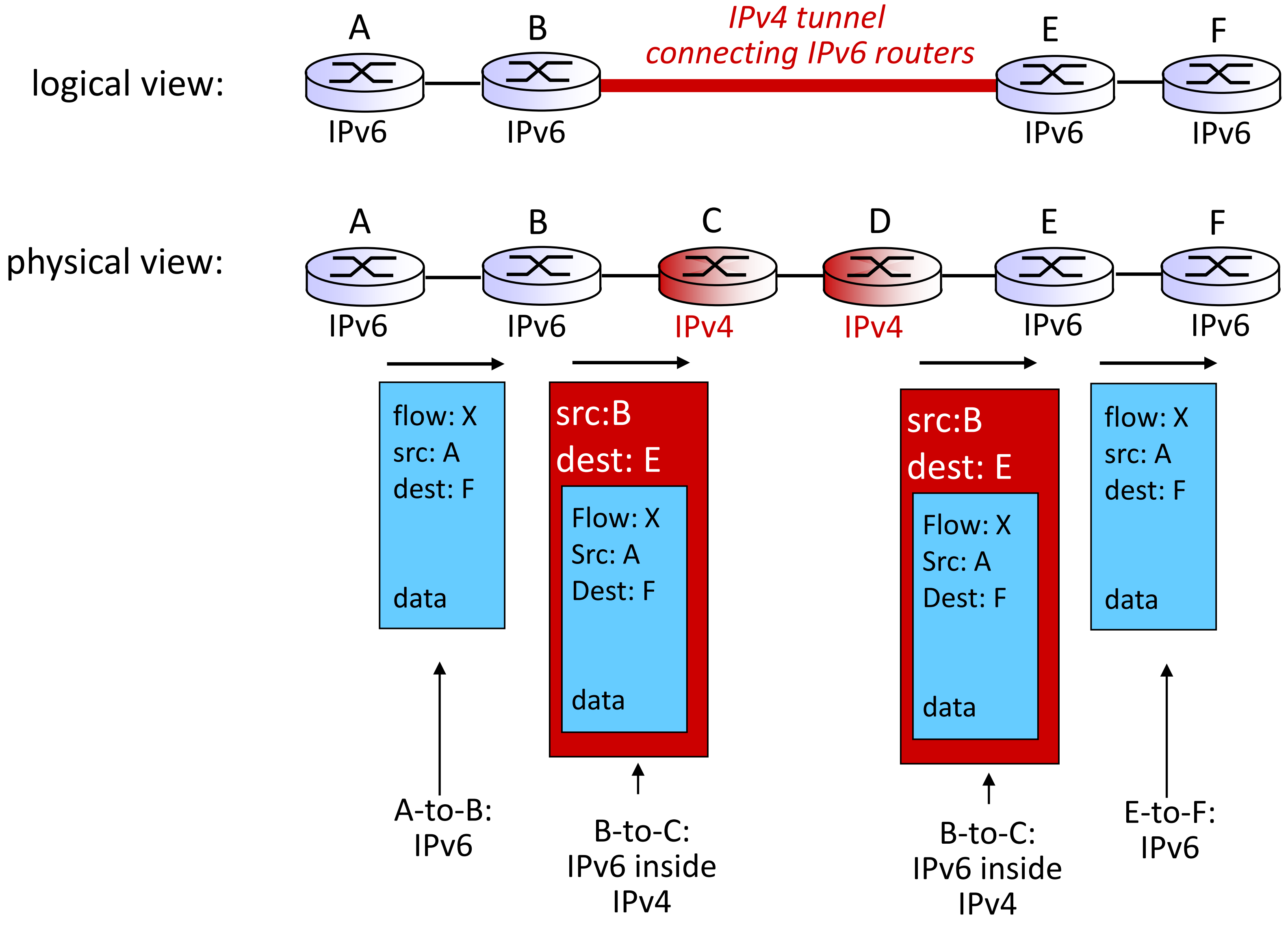
- not all routers can be upgraded simultaneously
  - no "flag days"
  - how will network operate with mixed IPv4 and IPv6 routers?
- *tunneling:* IPv6 datagram carried as *payload* in IPv4 datagram among IPv4 routers



Tunneling


logical view:
A
B
IPv4 tunnel connecting IPv6 routers
E
F
IPv6
IPv6
IPv6
IPv6
physical view:
A
B
C
D
E
F
IPv6
IPv6
IPv4
IPv4
IPv6
IPv6

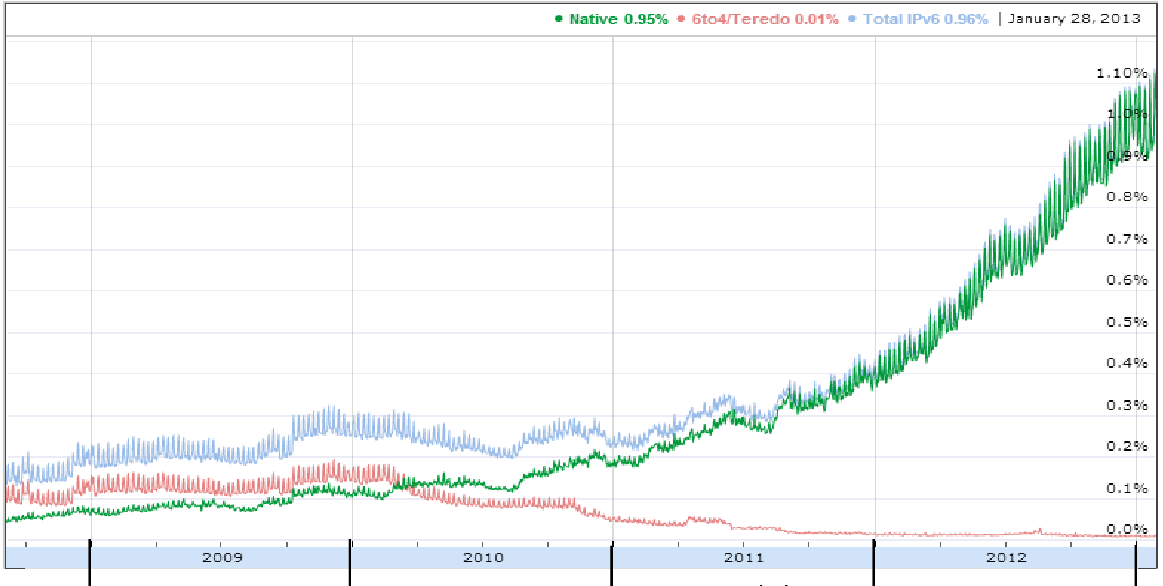
Tunneling



logical view:
A
B
E
F
IPv6
IPv6
IPv6
IPv6
IPv4 tunnel
connecting IPv6 routers
physical view:
A
B
C
D
E
F
IPv6
IPv6
IPv4
IPv4
IPv6
IPv6
flow: X
src: A
dest: F
data
src:B
dest: E
Flow: X
Src: A
Dest: F
data
src:B
dest: E
Flow: X
Src: A
Dest: F
data
flow: X
src: A
dest: F
data
A-to-B:
IPv6
B-to-C:
IPv6 inside
IPv4
B-to-C:
IPv6 inside
IPv4
E-to-F:
IPv6

IPv6: adoption

- US National Institutes of Standards estimate [2013]:
  - ~3% of industry IP routers
  - ~11% of US gov't routers



~1%

Time for growth!

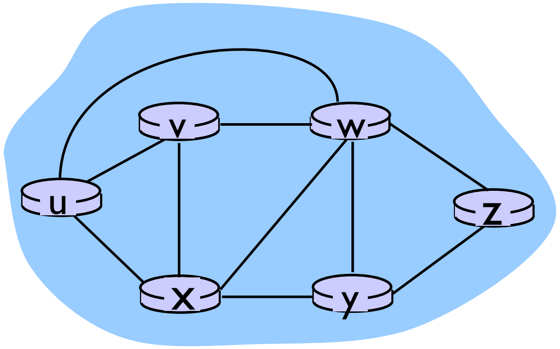
Source: Google IPv6 Statistics, 30/1/13

Percentage of users accessing Google via IPv6

Roadmap

- Network layer Overview
- Router architecture
- Network service models
- IP: Internet Protocol
- Routing
  - Routing algorithms
    - link State (LS)
    - distance vector (DV)
    - hierarchical routing
  - Routing protocols
    - RIP
    - OSPF
    - BGP
  - Broadcast and multicast routing

Graph abstraction



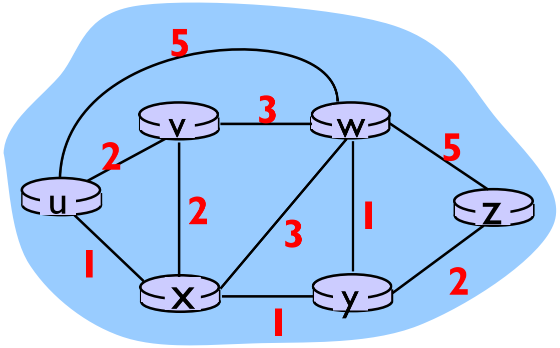
graph: G = (N,E)

N = set of routers = { u, v, w, x, y, z }

E = set of links ={ (u,v), (u,x), (v,x), (v,w), (x,w), (x,y), (w,y), (w,z), (y,z) }

aside: graph abstraction is useful in other network contexts, e.g., P2P, where *N* is set of peers and *E* is set of TCP connections

Graph abstraction: costs



c(x,x') = cost of link (x,x')

e.g., c(w,z) = 5

cost could always be 1, or inversely related to bandwidth, or inversely related to congestion

cost of path $(x_1, x_2, x_3, \ldots, x_p) = c(x_1,x_2) + c(x_2,x_3) + \ldots + c(x_{p-1},x_p)$

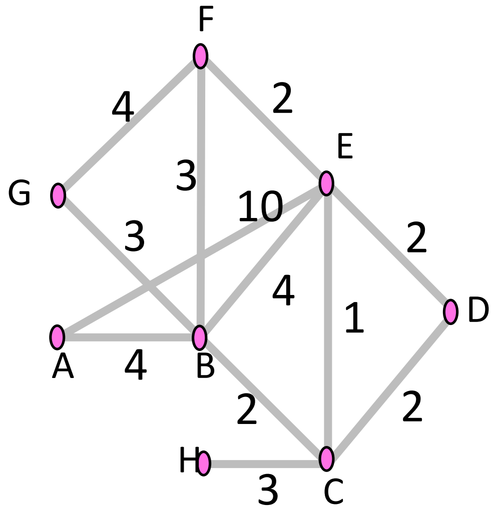
key question: what is the least-cost path between u and z ?

routing algorithm: algorithm that finds that *least cost path*

An Example of Least Cost Path

- Find the shortest path A → E

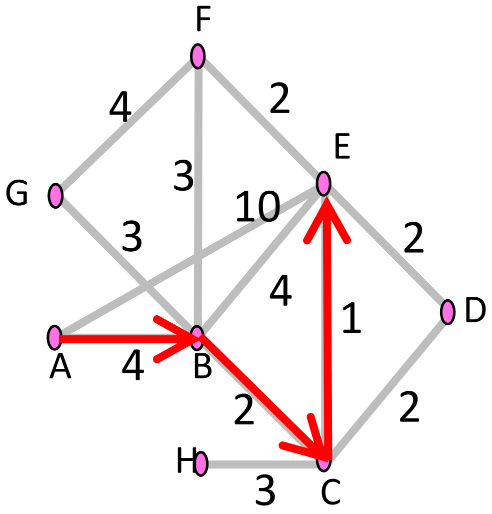
- All links are bidirectional, with equal costs in each direction
  - Can extend model to unequal costs if needed



An Example of Least Cost Path

- ABCE is a shortest path
- dist(ABCE) = 4 + 2 + 1 = 7

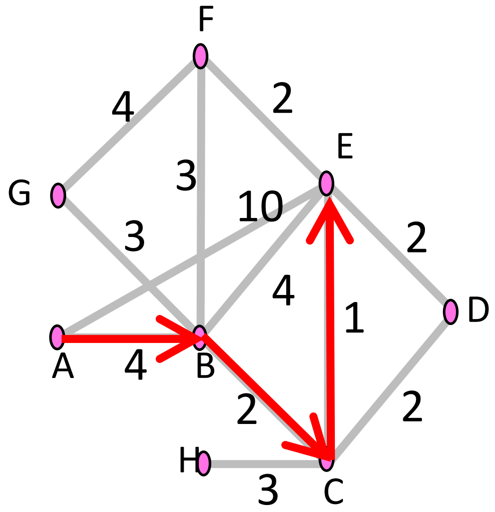
- This is less than:
  - dist(ABE) = 8
  - dist(ABFE) = 9
  - dist(AE) = 10
  - dist(ABCDE) = 10



An Example of Least Cost Path

- Optimality property:
  - Subpaths of shortest paths are also shortest paths
- ABCE is a shortest path

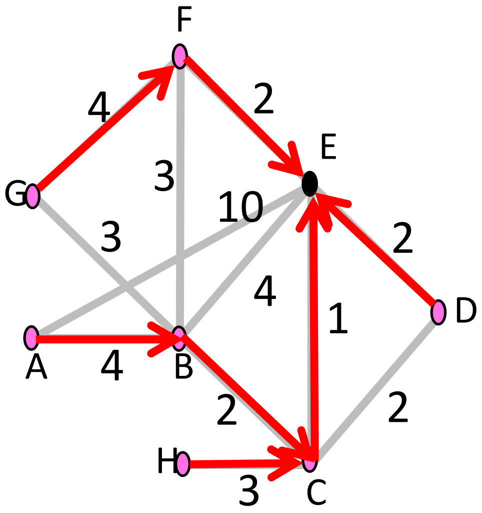
  →So are ABC, AB, BCE, BC, CE



Sink Trees

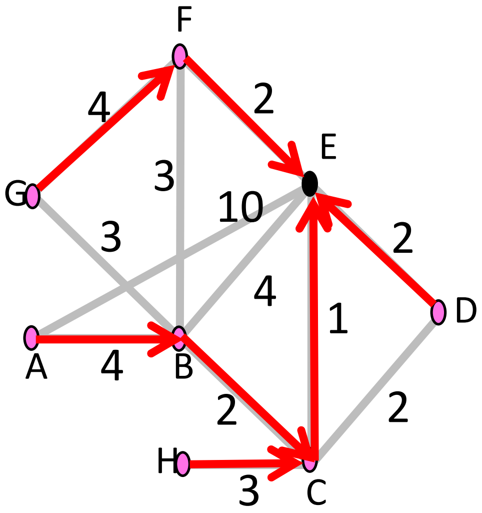
- Sink tree for a destination is the union of all shortest paths towards the destination
  - Similarly source tree

- Find the sink tree for E



Sink Trees

- Implications:
  - Only need to use destination to follow shortest paths
  - Each node only need to send to the next hop



- Forwarding table at a node
  - Lists next hop for each destination
  - Routing table may know more

Goals of Routing Algorithms

We want several properties of any routing scheme:

Property	**Meaning**
Correctness	Finds paths that work
Efficient paths	Uses network bandwidth well
Fair paths	Doesn’t starve any nodes
Fast convergence	Recovers quickly after changes
Scalability	Works well as network grows large

Routing algorithm classification

Q: global or decentralized info?

global:

- all routers have complete topology, link cost info
- "link state" algorithms

decentralized:

- router knows physically-connected neighbors, link costs to neighbors
- iterative process of computation, exchange of info with neighbors
- "distance vector" algorithms

Q: static or dynamic?

static:

- routes change slowly over time

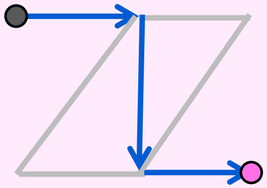
dynamic:

- routes change more quickly
  - periodic update
  - in response to link cost changes

Routing algorithm classification

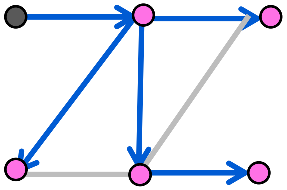
Unicast

(§ 5.2)



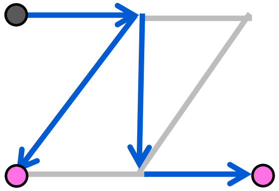
Broadcast

(§ 5.2.7)



Multicast

(§ 5.2.8)



Roadmap

- Network layer Overview
- Router architecture
- Network service models
- IP: Internet Protocol
- Routing
  - Routing algorithms
    - link State (LS)
    - distance vector (DV)
    - hierarchical routing
  - Routing protocols
    - RIP
    - OSPF
    - BGP
  - Broadcast and multicast routing

A Link-State Routing Algorithm

Dijkstra's algorithm

- net topology, link costs known to all nodes
  - accomplished via "link state broadcast"
  - all nodes have same info
- computes least cost paths from one node ('source") to all other nodes
  - gives *forwarding table* for that node
- iterative: after k iterations, know least cost path to k dest.'s

notation:

- c(x,y): link cost from node x to y; = ∞ if not direct neighbors
- D(v): current value of cost of path from source to dest. v
- p(v): predecessor node along path from source to v
- N': set of nodes whose least cost path definitively known

Dijsktra' s Algorithm

```
Initialization:
  N' = {u}
  for all nodes v
    if v adjacent to u
      then $D(v) = c(u,v)$
    else $D(v) = \infty$

Loop
  find w not in N' such that $D(w)$ is a minimum
  add w to N'
  update $D(v)$ for all v adjacent to w and not in N' :
      $D(v) = \min( D(v), D(w) + c(w,v) )$
  /* new cost to v is either old cost to v or known
   shortest path cost to w plus cost from w to v */
until all nodes in N'
```

notation:

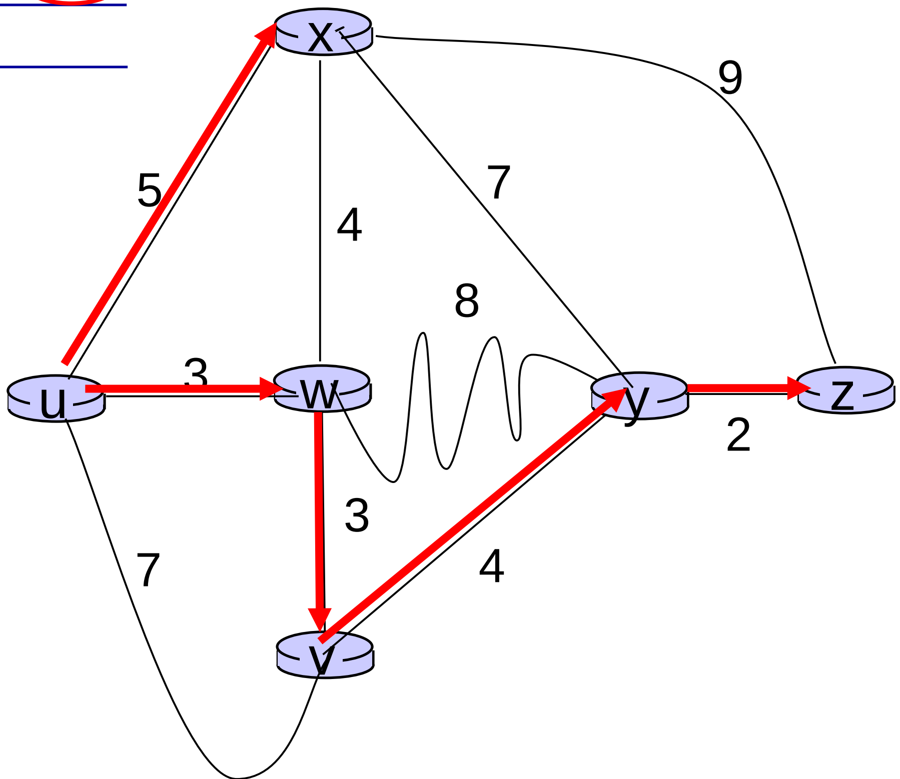
- $c(x,y)$: link cost from node x to y; $= \infty$ if not direct neighbors
- $D(v)$: current value of cost of path from source to dest. v
- $p(v)$: predecessor node along path from source to v
- N': set of nodes whose least cost path definitively known

Dijkstra’s algorithm: example

		D(v)	D(w)	D(x)	D(y)	D(z)
Step	N'	p(v)	p(w)	p(x)	p(y)	p(z)
0	u	7,u	3,u	5,u	∞	∞
1	uw	6,w		5,u	11,w	∞
2	uwx	6,w			11,w	14,x
3	uwxv				10,v	14,x
4	uwxvy					12,y
5	uwxvyz					

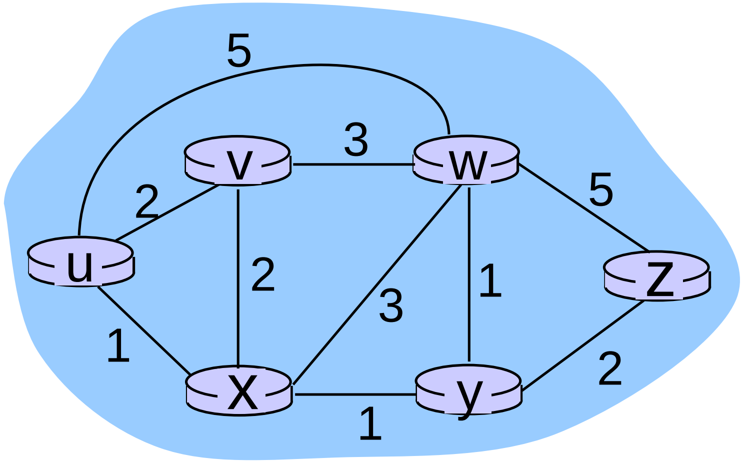
notes:

- construct shortest path tree by tracing predecessor nodes
- ties can exist (can be broken arbitrarily)



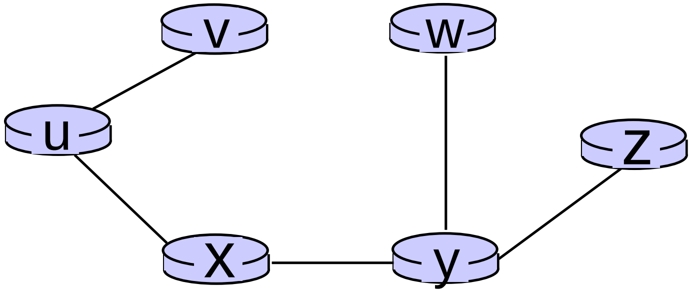
Dijkstra’ s algorithm: another example

Step	N'	D(v),p(v)	D(w),p(w)	D(x),p(x)	D(y),p(y)	D(z),p(z)
0	u	2,u	5,u	1,u	∞	∞
1	ux	2,u	4,x		2,x	∞
2	uxy	2,u	3,y			4,y
3	uxyv		3,y			4,y
4	uxyvw					4,y
5	uxyvwz					



Dijkstra's algorithm: example (2)

resulting shortest-path tree from u:



resulting forwarding table in u:

destination	link
v	(u,v)
x	(u,x)
y	(u,x)
w	(u,x)
z	(u,x)

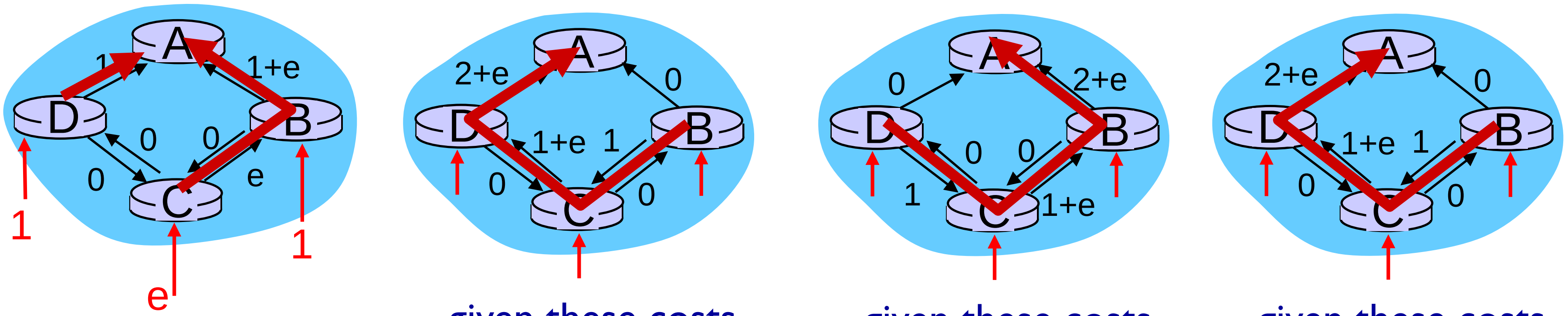
Dijkstra' s algorithm, discussion

algorithm complexity: n nodes

- each iteration: need to check all nodes, w, not in N
- n(n+1)/2 comparisons: O(n^2)
- more efficient implementations possible: O(nlogn)

oscillations possible:

- e.g., support link cost equals amount of carried traffic:



initially

given these costs, find new routing…. resulting in new costs

given these costs, find new routing…. resulting in new costs

given these costs, find new routing…. resulting in new costs

Roadmap

- Network layer Overview
- Router architecture
- Network service models
- IP: Internet Protocol
- Routing
  - Routing algorithms
    - link State (LS)
    - distance vector (DV)
    - hierarchical routing
  - Routing protocols
    - RIP
    - OSPF
    - BGP
  - Broadcast and multicast routing

Distance vector algorithm

Bellman-Ford equation (dynamic programming)

Divide path xy into two parts {xv, vy}, where v is a neighbor of x

let

$d_x(y)$:= cost of least-cost path from x to y

then

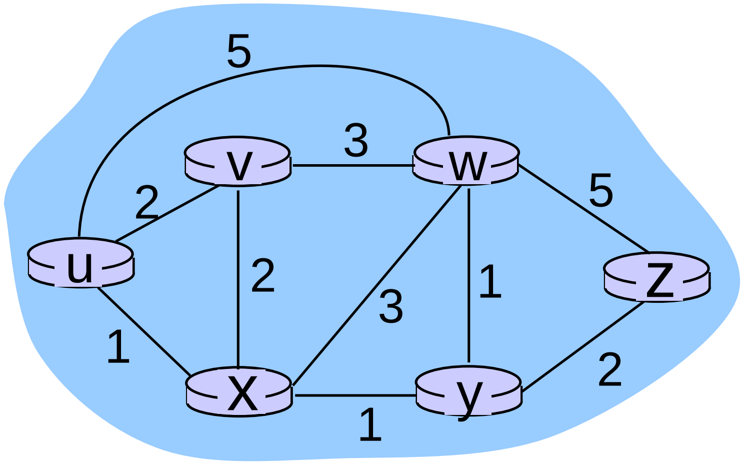
$$d_x(y) = \min_v \{c(x,v) + d_v(y)\}$$

cost from neighbor v to destination y

cost to neighbor v

min taken over all neighbors v of x

Bellman-Ford example



clearly, $d_v(z) = 5$, $d_x(z) = 3$, $d_w(z) = 3$

B-F equation says:

$$\begin{aligned} d_u(z) &= \min \{ c(u,v) + d_v(z), \\ &\qquad c(u,x) + d_x(z), \\ &\qquad c(u,w) + d_w(z) \} \\ &= \min \{2 + 5, \\ &\qquad 1 + 3, \\ &\qquad 5 + 3\} = 4 \end{aligned}$$

the neighbor achieving minimum is the next hop in shortest path, used in forwarding table

Distance vector algorithm

- $D_x(y)$ = estimate of least cost from x to y
  - x maintains distance vectors to all other nodes

  $\mathbf{D}_x = [D_x(y): y \in N\ ]$
- node x:
  - knows cost to each neighbor v: $c(x,v)$
  - maintains its neighbors' distance vectors. For each neighbor v, x receives
    $\mathbf{D}_v = [D_v(y): y \in N\ ]$

Distance vector algorithm

key idea:

- from time-to-time, each node sends its own distance vector estimate to neighbors
- when x receives new DV estimate from neighbor, it updates its own DV using B-F equation:

$$D_x(y) \leftarrow \min_v\{c(x,v) + D_v(y)\} \quad \text{for each node } y \in N$$

- under minor, natural conditions, the estimate $D_x(y)$ *converge to the actual least cost* $d_x(y)$

Distance vector algorithm

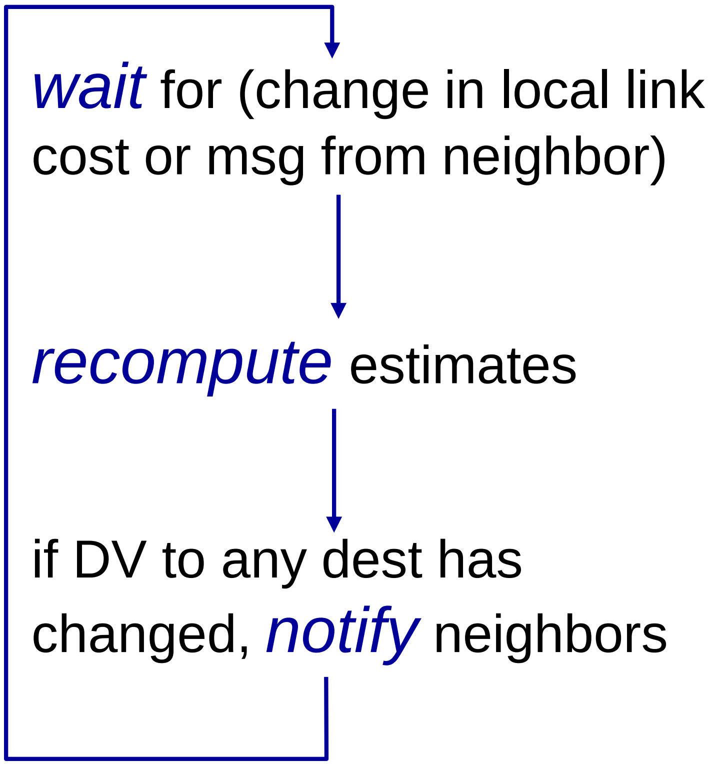
iterative, asynchronous:
each local iteration caused by:

- local link cost change
- DV update message from neighbor

distributed:

- each node notifies neighbors *only* when its DV changes
  - neighbors then notify their neighbors if necessary

each node:



wait for (change in local link cost or msg from neighbor)

recompute estimates

if DV to any dest has changed, *notify* neighbors

$$D_x(y) = \min\{c(x,y) + D_y(y), c(x,z) + D_z(y)\}$$
$$= \min\{2+0 , 7+1\} = 2$$

$$D_x(z) = \min\{c(x,y) +$$
$$D_y(z), c(x,z) + D_z(z)\}$$
$$= \min\{2+1 , 7+0\} = 3$$

node x table

	cost to		
from	x	y	z
x	0	2	7
y	∞	∞	∞
z	∞	∞	∞

	cost to		
from	x	y	z
x	0	2	3
y	2	0	1
z	7	1	0

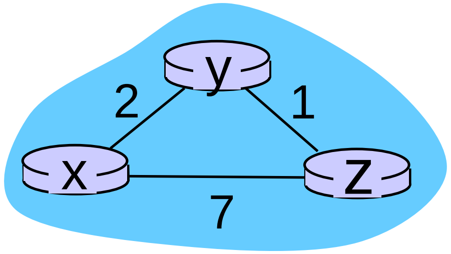
node y table

	cost to		
from	x	y	z
x	∞	∞	∞
y	2	0	1
z	∞	∞	∞

node z table

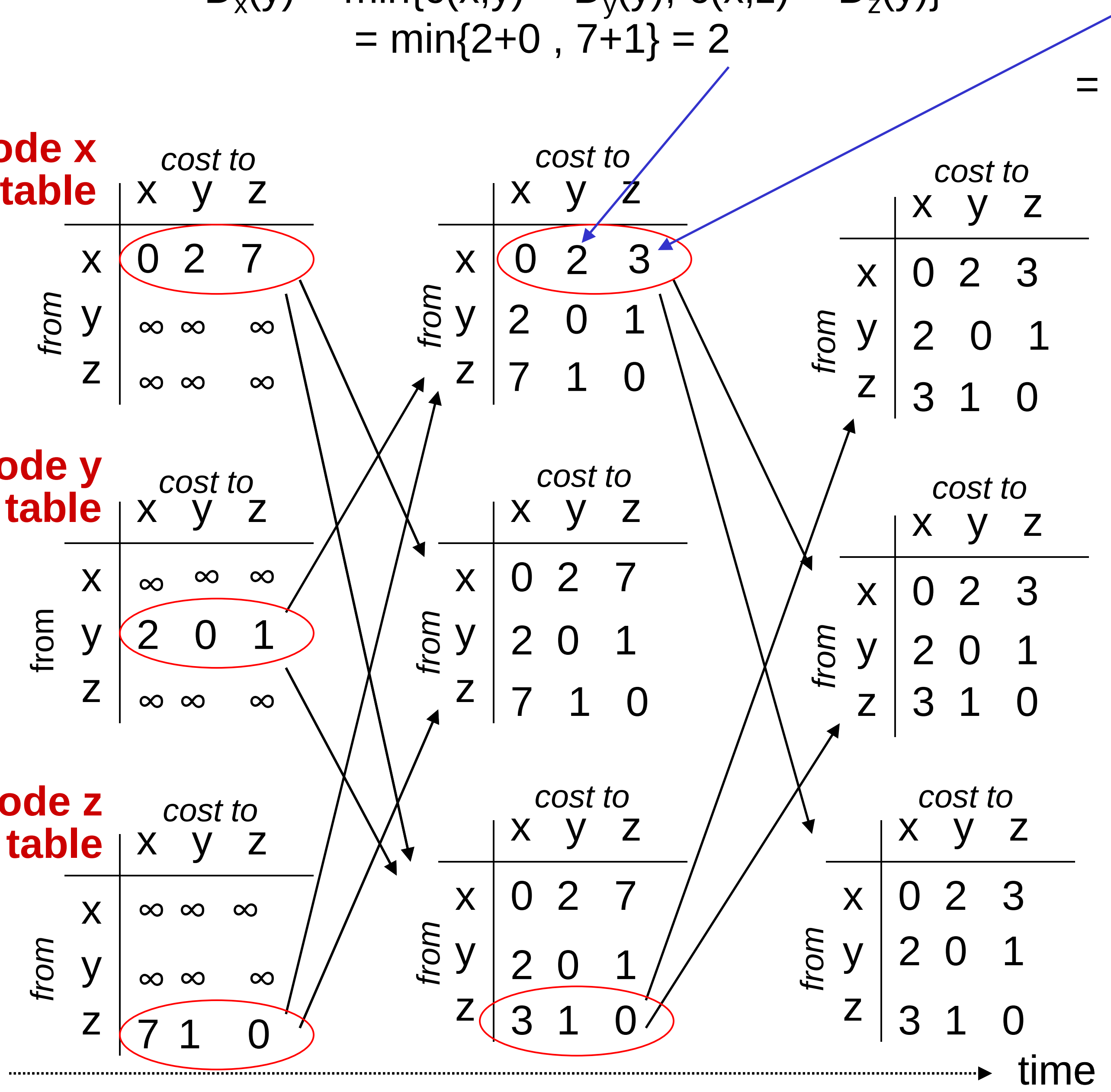
	cost to		
from	x	y	z
x	∞	∞	∞
y	∞	∞	∞
z	7	1	0

time



$D_x(y) = \min\{c(x,y) + D_y(y), c(x,z) + D_z(y)\}$
$= \min\{2+0 , 7+1\} = 2$

$D_x(z) = \min\{c(x,y) +$
$D_y(z), c(x,z) + D_z(z)\}$
$= \min\{2+1 , 7+0\} = 3$



node x table

cost to

from	x	y	z
x	0	2	7
y	∞	∞	∞
z	∞	∞	∞

cost to

from	x	y	z
x	0	2	3
y	2	0	1
z	7	1	0

cost to

from	x	y	z
x	0	2	3
y	2	0	1
z	3	1	0

node y table

cost to

from	x	y	z
x	∞	∞	∞
y	2	0	1
z	∞	∞	∞

cost to

from	x	y	z
x	0	2	7
y	2	0	1
z	7	1	0

cost to

from	x	y	z
x	0	2	3
y	2	0	1
z	3	1	0

node z table

cost to

from	x	y	z
x	∞	∞	∞
y	∞	∞	∞
z	7	1	0

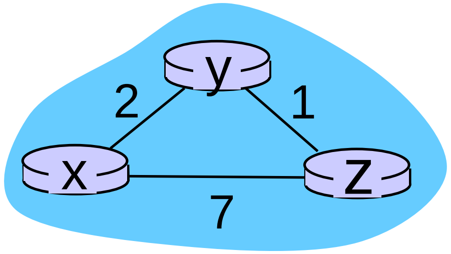
cost to

from	x	y	z
x	0	2	7
y	2	0	1
z	3	1	0

cost to

from	x	y	z
x	0	2	3
y	2	0	1
z	3	1	0

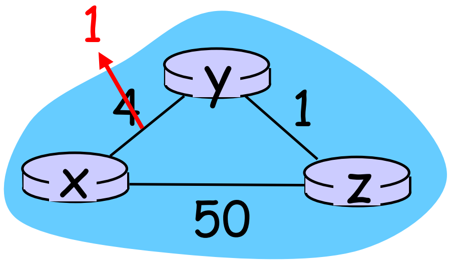
time



Distance vector: link cost changes

link cost changes:

- node detects local link cost change
- updates routing info, recalculates distance vector
- if DV changes, notify neighbors



"good news travels fast"

t_0 : *y* detects link-cost change, updates its DV, informs its neighbors.

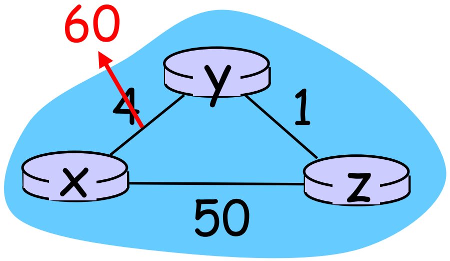
t_1 : *z* receives update from *y*, updates its table, computes new least cost to *x* , sends its neighbors its DV.

t_2 : *y* receives *z*' s update, updates its distance table. *y*' s least costs do *not* change, so *y* does *not* send a message to *z*.

Distance vector: link cost changes

link cost changes:

- node detects local link cost change
- *bad news travels slow* - "count to infinity" problem!



y sets $D_x(y)=4$

z sets $D_x(z)=50$

y says $D_x(y)=4$

z sets $D_x(z)=5$, z sets y as the next hop to x

time

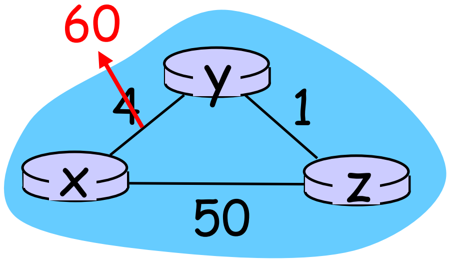
t0

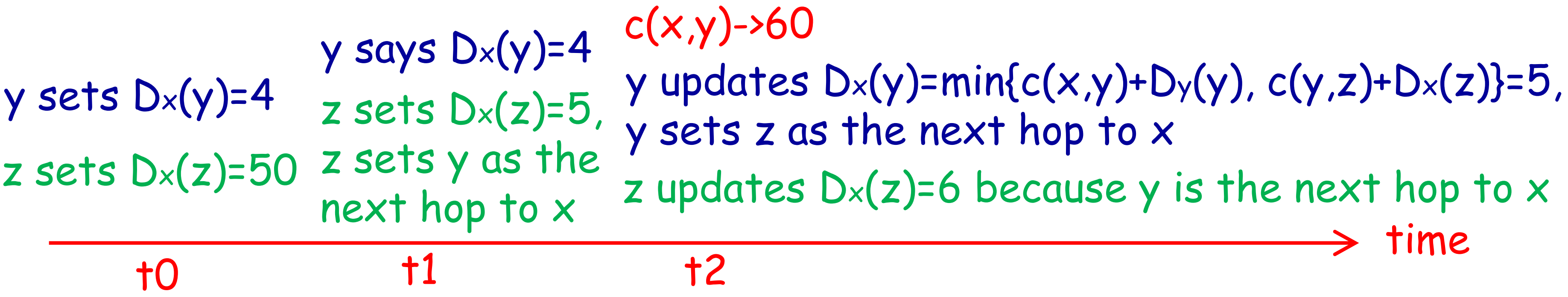
t1

Distance vector: link cost changes

link cost changes:

- node detects local link cost change
- *bad news travels slow* - "count to infinity" problem!





y sets $D_x(y)$=4

z sets $D_x(z)$=50

y says $D_x(y)$=4

z sets $D_x(z)$=5, z sets y as the next hop to x

c(x,y)->60

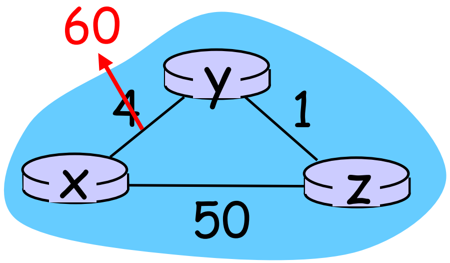
y updates $D_x(y)=\min\{c(x,y)+D_y(y), c(y,z)+D_x(z)\}=5$, y sets z as the next hop to x

z updates $D_x(z)$=6 because y is the next hop to x

Distance vector: link cost changes

link cost changes:

- node detects local link cost change
- *bad news travels slow* - "count to infinity" problem!



c(x,y)->60

t0
y sets $D_x(y)=4$
z sets $D_x(z)=50$

t1
y says $D_x(y)=4$
z sets $D_x(z)=5$, z sets y as the next hop to x

t2
y updates $D_x(y)=5$, y sets z as the next hop to x
z updates $D_x(z)=6$

t3
z says $D_x(z)=6$
y update $D_x(y)=7$ because z is the next hop to x …

time

poisoned reverse:

- If C routes through B to get to A :
  - C tells B its (C' s) distance to A is infinite (so B won' t route to A via C)
- will this completely solve count to infinity problem?

Comparison of LS and DV algorithms

message complexity

- **LS:** with n nodes, E links, O(nE) msgs sent
- **DV:** exchange between neighbors only
  - convergence time varies

speed of convergence

- **LS:** O(n^2) algorithm requires O(nE) msgs
  - may have oscillations
- **DV:** convergence time varies
  - may be routing loops
  - count-to-infinity problem

robustness: what happens if router malfunctions?

LS:

- node can advertise incorrect *link* cost
- each node computes only its *own* table

DV:

- DV node can advertise incorrect *path* cost
- each node' s table used by others
  - error propagate thru network

Roadmap

- Network layer Overview
- Router architecture
- Network service models
- IP: Internet Protocol
- Routing
  - Routing algorithms
    - link State (LS)
    - distance vector (DV)
    - hierarchical routing
  - Routing protocols
    - RIP
    - OSPF
    - BGP
  - Broadcast and multicast routing

Hierarchical routing

our routing study thus far - idealization

- all routers identical
- network "flat"

… *not* true in practice

scale: with 600 million destinations:

- can't store all dest's in routing tables!
- routing table exchange would swamp links!
- table lookup will be time consuming

administrative autonomy

- internet = network of networks
- each network admin may want to control routing in its own network

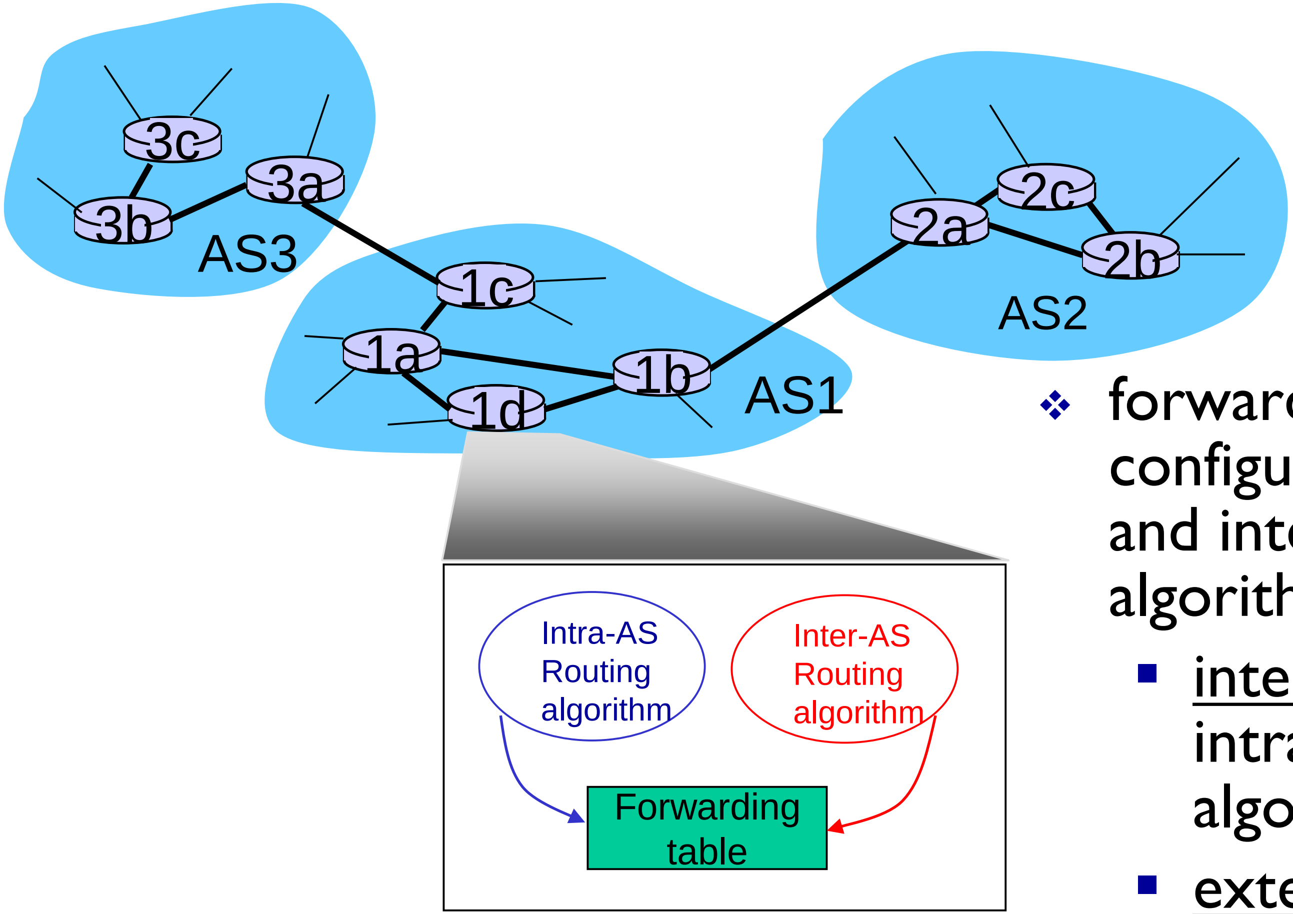
Hierarchical routing

- aggregate routers into regions, "autonomous systems" (AS)
- routers in same AS run same routing protocol
  - "intra-AS" routing protocol
  - routers in different AS can run different intra-AS routing protocol

gateway router:

- at "edge" of its own AS
- has link to router in another AS

Interconnected ASes



- forwarding table configured by both intra- and inter-AS routing algorithm
  - internal dests set by intra-AS routing algorithm
  - external dests set by inter-AS & intra-AS routing algorithm

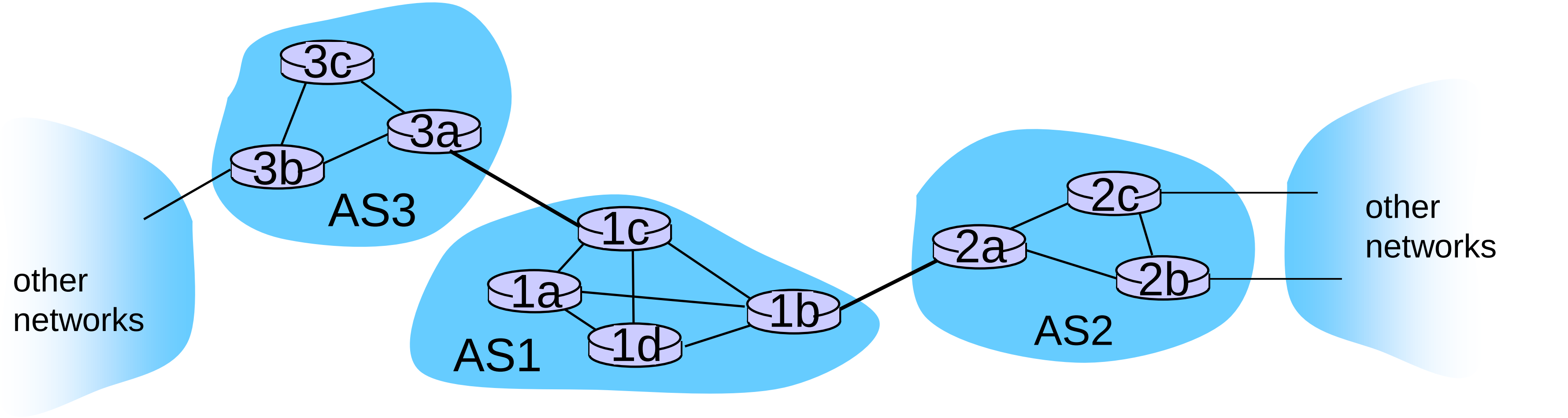
Inter-AS tasks

- suppose router in AS1 receives datagram destined outside of AS1:
  - router should forward packet to gateway router, but which one?

AS1 must:

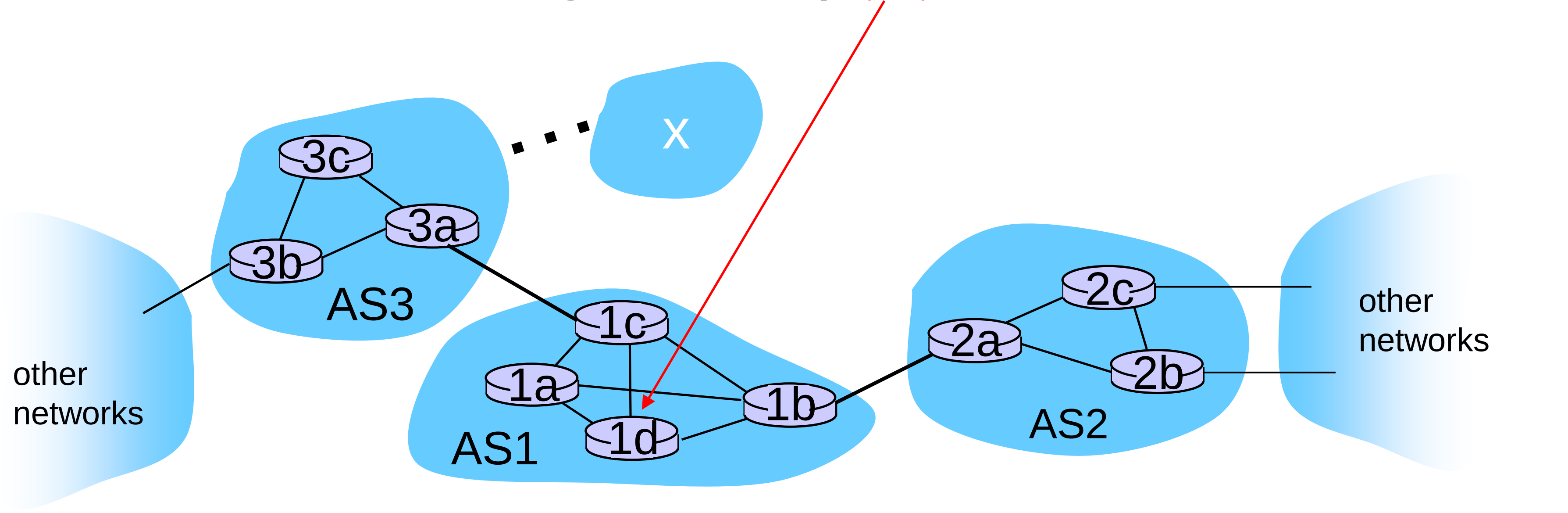
1. learn which dests are reachable through AS2, which through AS3
2. propagate this reachability info to all routers in AS1

job of inter-AS routing!



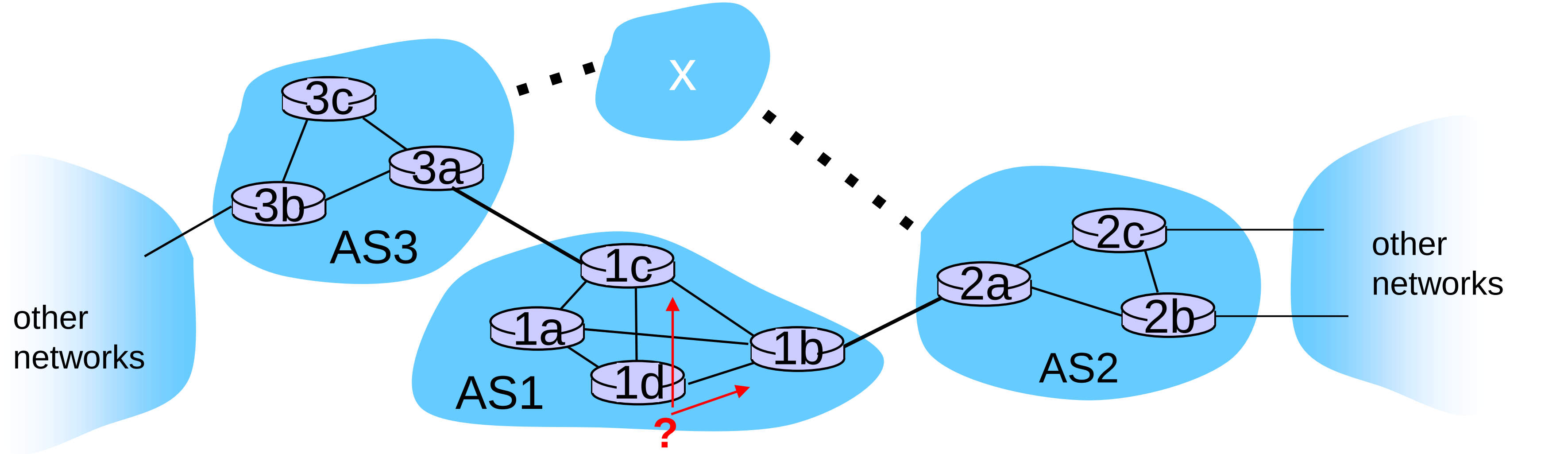
Example: setting forwarding table in router 1d

- suppose AS1 learns (via inter-AS protocol) that subnet *x* reachable via AS3 (gateway 1c), but not via AS2
  - inter-AS protocol propagates reachability info to all internal routers
- router 1d determines from intra-AS routing info that its interface *I* is on the least cost path to 1c
  - installs forwarding table entry *(x,I)*



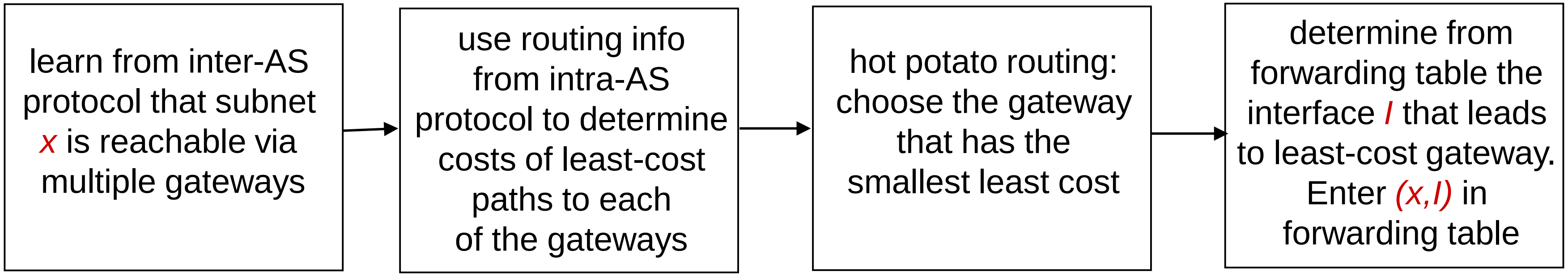
Example: choosing among multiple ASes

- now suppose AS1 learns from inter-AS protocol that subnet *x* is reachable from AS3 *and* from AS2.
- to configure forwarding table, router 1d must determine towards which gateway it should forward packets for dest x
  - this is also job of inter-AS routing protocol!



Example: choosing among multiple ASes

- now suppose AS1 learns from inter-AS protocol that subnet *x* is reachable from AS3 *and* from AS2.
- to configure forwarding table, router 1d must determine towards which gateway it should forward packets for dest x
  - this is also job of inter-AS routing protocol!
- *hot potato routing: send* packet towards closest of two routers.



learn from inter-AS protocol that subnet *x* is reachable via multiple gateways

use routing info from intra-AS protocol to determine costs of least-cost paths to each of the gateways

hot potato routing: choose the gateway that has the smallest least cost

determine from forwarding table the interface *I* that leads to least-cost gateway. Enter *(x,I)* in forwarding table

Roadmap

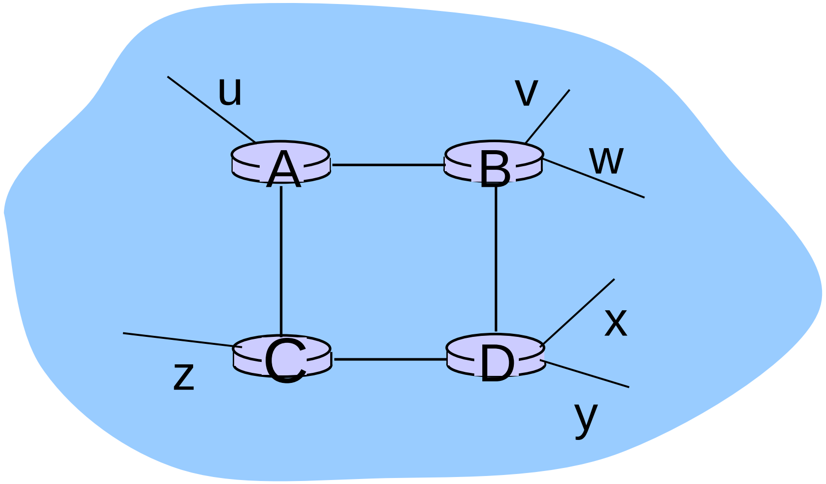
- Network layer Overview
- Router architecture
- Network service models
- IP: Internet Protocol
- Routing
  - Routing algorithms
    - link State (LS)
    - distance vector (DV)
    - hierarchical routing
  - Routing protocols
    - RIP
    - OSPF
    - BGP
  - Broadcast and multicast routing

Intra-AS Routing

- also known as *interior gateway protocols (IGP)*
- most common intra-AS routing protocols:
  - RIP: Routing Information Protocol
  - OSPF: Open Shortest Path First
  - IGRP: Interior Gateway Routing Protocol (Cisco proprietary)

RIP (Routing Information Protocol)

- included in BSD-UNIX distribution in 1982
- distance vector algorithm
  - distance metric: # hops (max = 15 hops), each link has cost 1
  - DVs exchanged with neighbors every 30 sec in response message (aka advertisement)



from router A to destination *subnets:*

subnet	hops
u	1
v	2
w	2
x	3
y	3
z	2

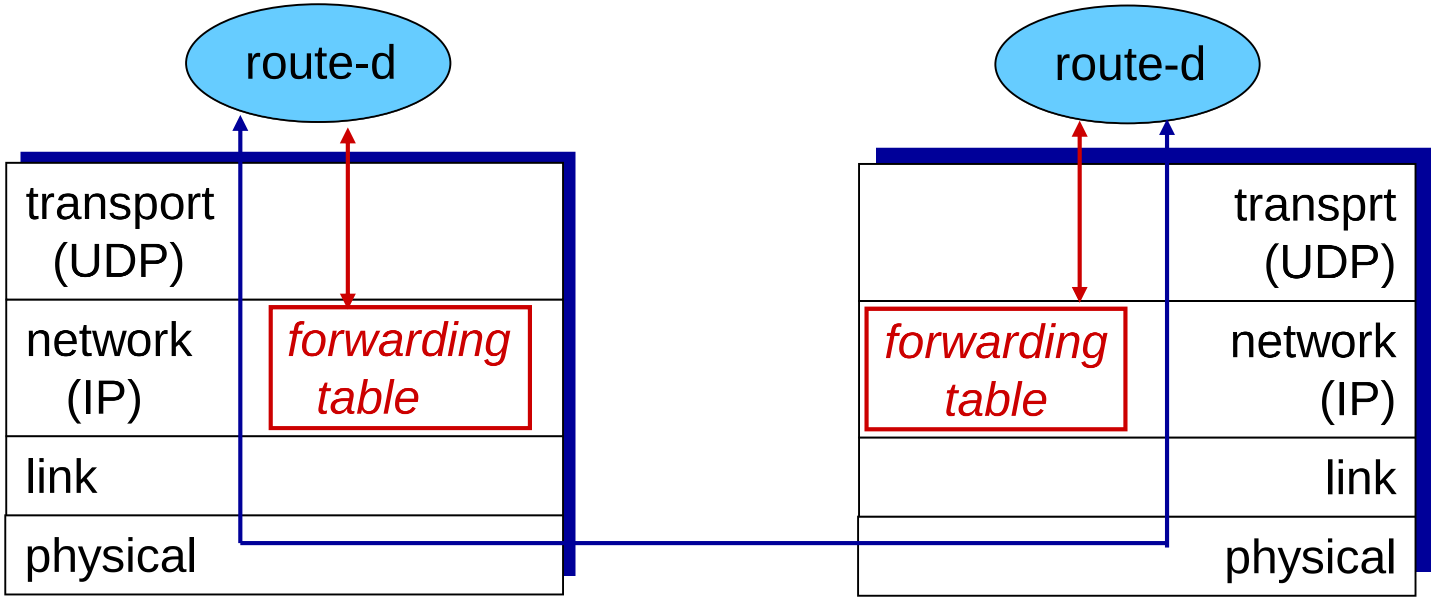
RIP: link failure, recovery

if no advertisement heard after 180 sec --> neighbor/link declared dead

- routes via neighbor invalidated
- new advertisements sent to neighbors
- neighbors in turn send out new advertisements (if tables changed)
- link failure info quickly propagates to entire net
- *poison reverse* used to prevent ping-pong loops (infinite distance = 16 hops)

RIP table processing

- RIP routing tables managed by *application-level* process called route-d (daemon)
- advertisements sent in *UDP* packets, periodically repeated



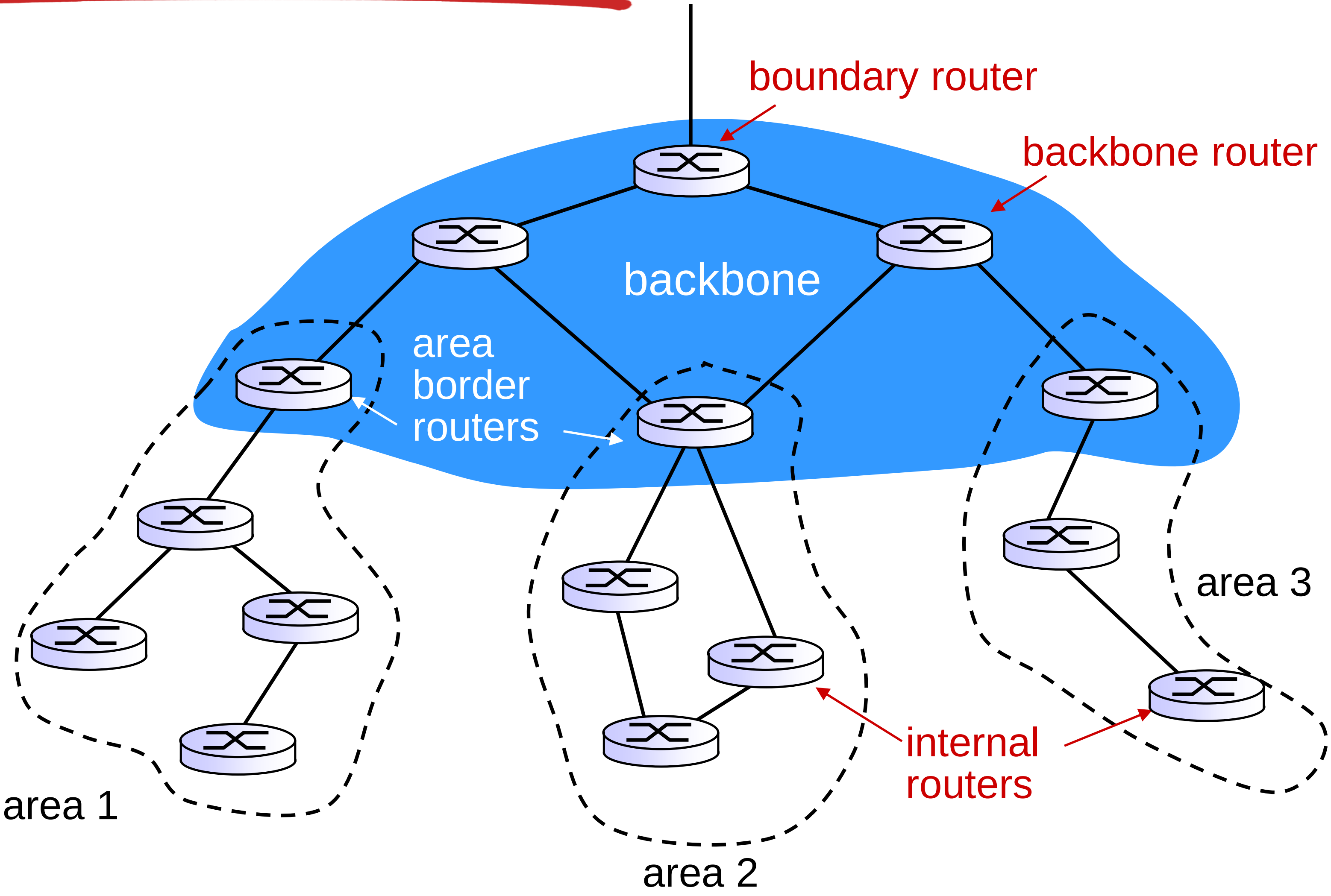
OSPF (Open Shortest Path First)

- “open”: publicly available
- uses link state algorithm
  - LS packet dissemination
  - topology map at each node
  - route computation using Dijkstra’s algorithm
- OSPF advertisement carries one entry per neighbor
- advertisements flooded to *entire* AS
  - carried in OSPF messages directly over IP (rather than TCP or UDP)

OSPF "advanced" features (not in RIP)

- *security:* all OSPF messages authenticated (to prevent malicious intrusion)
- multiple same-cost paths allowed (only one path in RIP)
- for each link, multiple cost metrics for different ToS (Type of Service) (e.g., satellite link cost set "low" for best effort ToS; high for real time ToS)
- integrated uni- and multicast support:
  - Multicast OSPF (MOSPF) uses same topology data base as OSPF
- hierarchical OSPF in large domains.

Hierarchical OSPF



Hierarchical OSPF

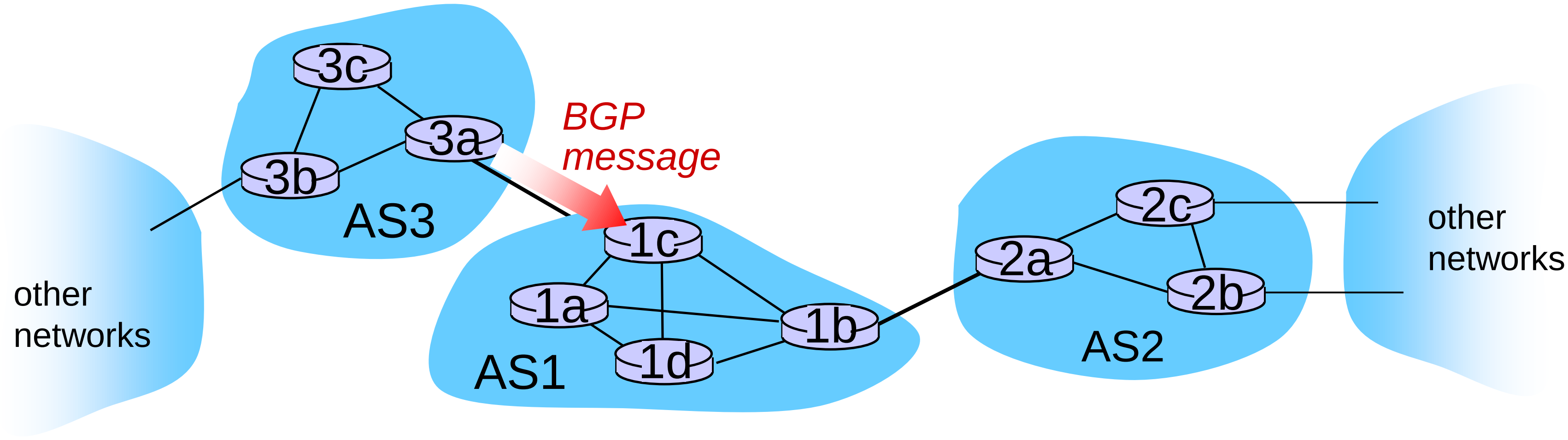
- *two-level hierarchy:* local area, backbone.
  - link-state advertisements only in area
  - each nodes has detailed area topology; only know direction (shortest path) to nets in other areas.
- *area border routers:* "summarize" distances to nets in own area, advertise to other Area Border routers.
- *backbone routers:* run OSPF routing limited to backbone.
- *boundary routers:* connect to other AS' s.

Internet inter-AS routing: BGP

- BGP (Border Gateway Protocol): *the* de facto inter-domain routing protocol
  - "glue that holds the Internet together"
- BGP provides each AS a means to:
  - eBGP: obtain subnet reachability information from neighboring ASs.
  - iBGP: propagate reachability information to all AS-internal routers.
  - determine "good" routes to other networks based on reachability information and *policy*.

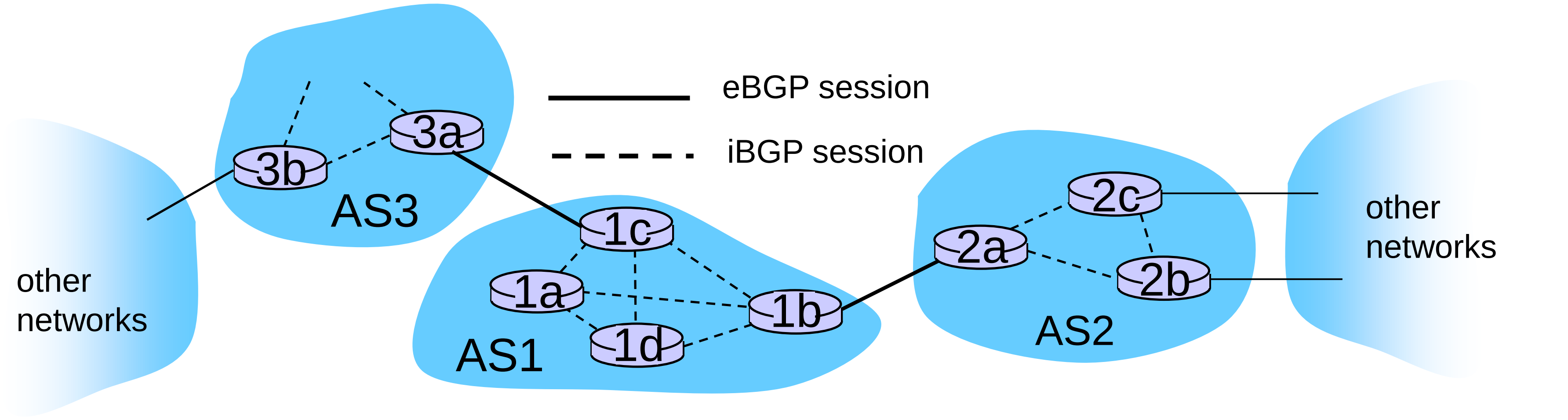
BGP basics

- BGP session: two BGP routers ("peers") exchange BGP messages:
  - advertising *paths* to different destination network prefixes ("path vector" protocol)
  - exchanged over TCP connections
- when AS3 advertises a prefix to AS1:
  - AS3 *promises* it will forward datagrams towards that prefix
  - AS3 can aggregate prefixes in its advertisement



BGP basics: distributing path information

- using eBGP session between 3a and 1c, AS3 sends prefix reachability info to AS1.
- 1c can then use iBGP to distribute new prefix info to all routers in AS1
- 1b can then re-advertise new reachability info to AS2 over 1b-to-2a eBGP session
- when router learns of new prefix, it creates entry for prefix in its forwarding table.



Path attributes and BGP routes

- advertised prefix includes BGP attributes
  - prefix + attributes = "route"
- two important attributes:
  - AS-PATH: contains ASs through which prefix advertisement has passed: e.g., AS 67, AS 17
  - NEXT-HOP: indicates specific internal-AS router to next-hop AS. (may be multiple links from current AS to next-hop-AS)
- gateway router receiving route advertisement uses policy to accept/decline
  - e.g., never route through AS x
  - *policy-based* routing

BGP route selection

- router may learn about more than 1 route to destination AS, selects route based on:
    1. local preference value attribute: policy decision
    2. shortest AS-PATH
    3. closest NEXT-HOP router: hot potato routing
    4. additional criteria

BGP messages

- BGP messages exchanged between peers over TCP connection
- BGP messages:
  - OPEN: opens TCP connection to peer and authenticates sender
  - UPDATE: advertises new path (or withdraws old)
  - KEEPALIVE: keeps connection alive in absence of UPDATES; also ACKs OPEN request
  - NOTIFICATION: reports errors in previous msg; also used to close connection

Putting it Altogether: *How Does an Entry Get Into a Router´s Forwarding Table?*

- Answer is complicated!

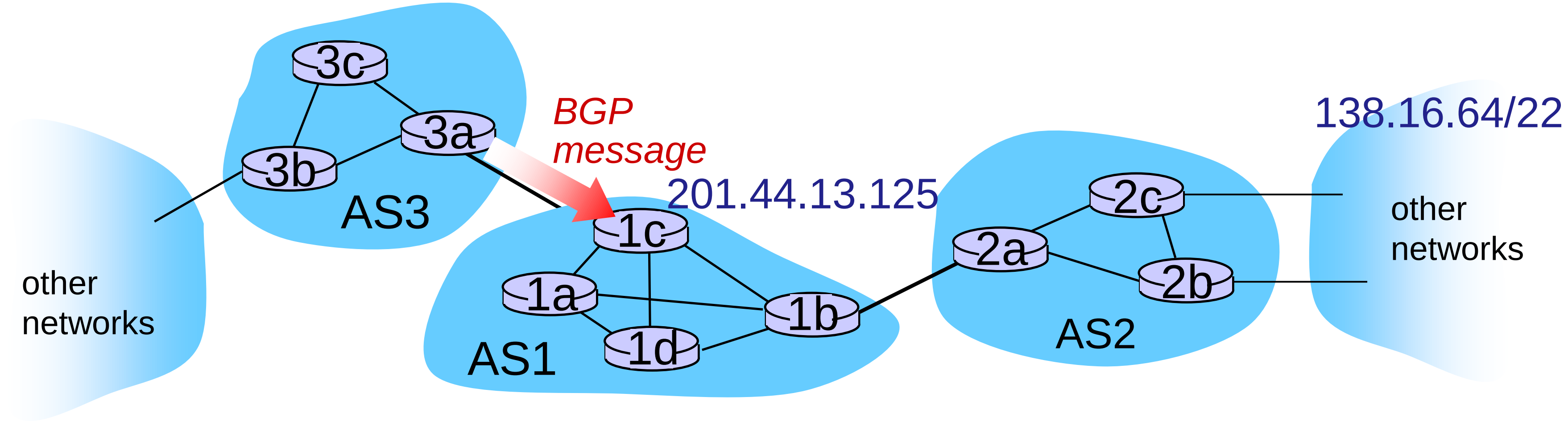
- Ties together hierarchical routing with BGP and OSPF.

How does entry get in forwarding table?

High-level overview

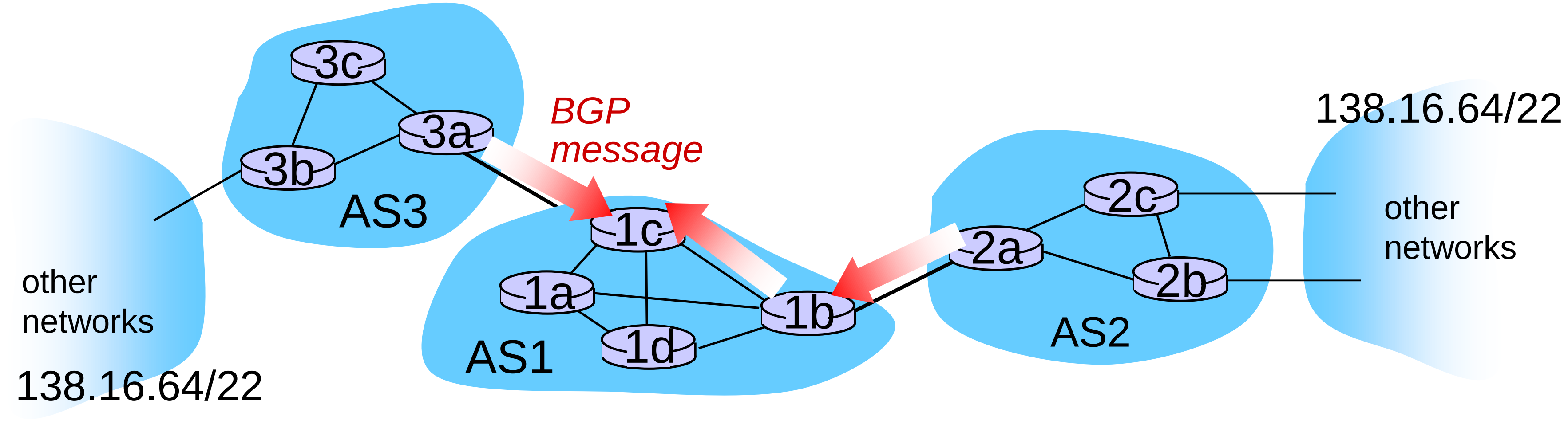
1. Router becomes aware of prefix
2. Router determines output port for prefix
3. Router enters prefix-port in forwarding table

Router becomes aware of prefix



- BGP message contains "routes"
- "route" is a prefix and attributes: AS-PATH, NEXT-HOP,…
- Example: route:
  - Prefix:138.16.64/22 ; AS-PATH: AS1 AS2 ; NEXT-HOP: 201.44.13.125

Router may receive multiple routes



- Router may receive multiple routes for <u>same</u> prefix
- Has to select one route

Select best *BGP route* to prefix

- Router selects route based on shortest AS-PATH

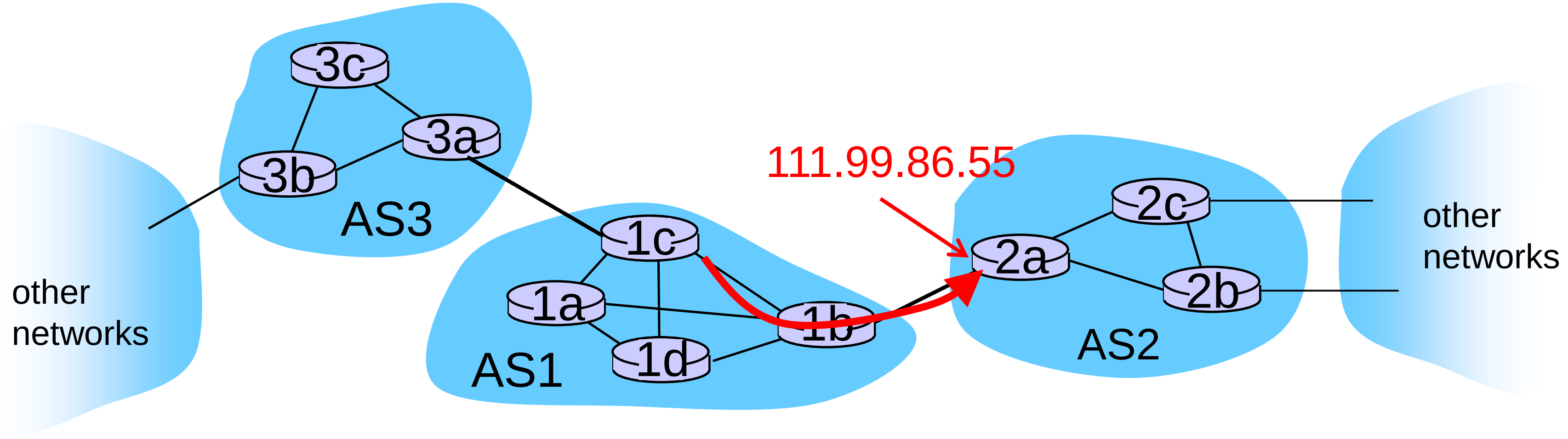
- Example:

  select

  - AS2 AS17 to 138.16.64/22
  - AS3 AS131 AS201 to 138.16.64/22

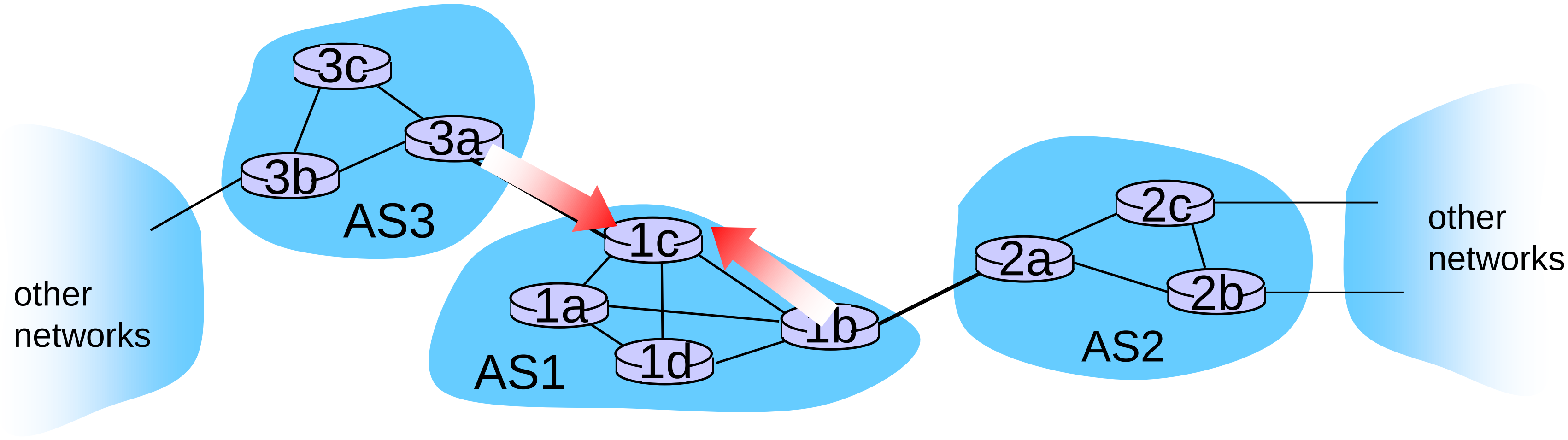
Find best *intra-route* to BGP route

- Use selected route's NEXT-HOP attribute
  - Route's NEXT-HOP attribute is the IP address of the router interface that begins the AS PATH.
- Example:
  - AS-PATH: AS2 AS17 ; NEXT-HOP: 111.99.86.55
- Router uses OSPF to find shortest path from 1c to 111.99.86.55



Hot Potato Routing

- Suppose there two or more best inter-routes.
- Then choose route with closest NEXT-HOP
  - Use OSPF to determine which gateway is closest
  - Q: From 1c, chose AS3 AS131 or AS2 AS17?
  - A: route AS3 AS201 since it is closer



How does entry get in forwarding table?

Summary

1. Router becomes aware of prefix
   - via BGP route advertisements from other routers
2. Determine router output port for prefix
   - Use BGP route selection to find best inter-AS route (*path vector algorithm, policy defined*)
   - Use OSPF to find best intra-AS route leading to best inter-AS route
   - Router identifies router port for that best route
3. Enter prefix-port entry in forwarding table

Why different Intra-, Inter-AS routing ?

policy:

- inter-AS: admin wants control over how its traffic routed, who routes through its net.
- intra-AS: single admin, so no policy decisions needed

scale:

- hierarchical routing saves table size, reduced update traffic

performance:

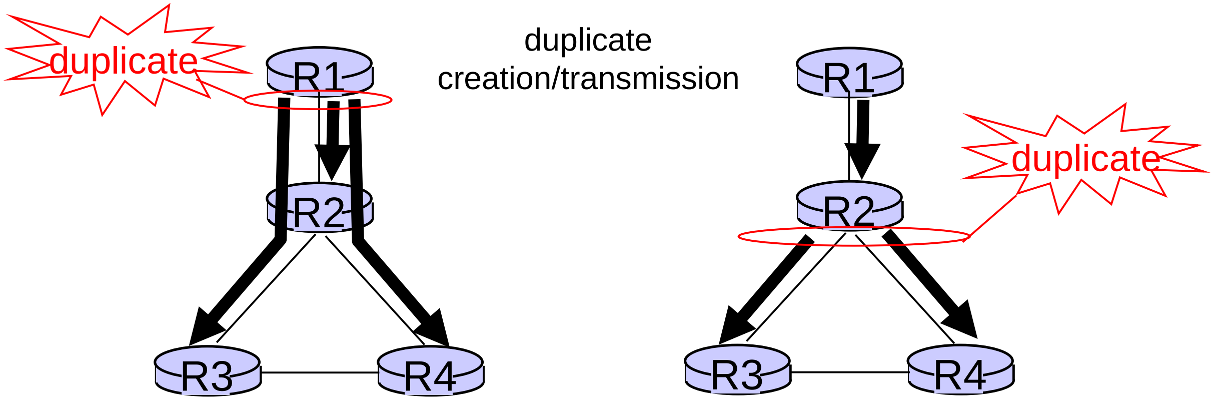
- intra-AS: can focus on performance
- inter-AS: policy may dominate over performance

Roadmap

- Network layer Overview
- Router architecture
- Network service models
- IP: Internet Protocol
- Routing
  - Routing algorithms
    - link State (LS)
    - distance vector (DV)
    - hierarchical routing
  - Routing protocols
    - RIP
    - OSPF
    - BGP
  - Broadcast and multicast routing

Broadcast routing

- deliver packets from source to all other nodes
- source duplication is inefficient:



source duplication

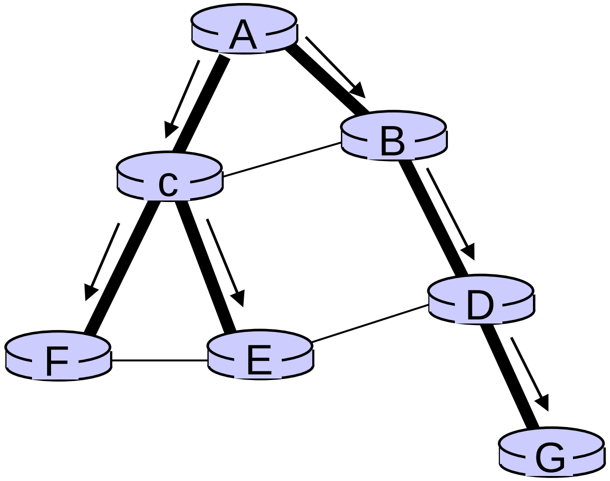
in-network duplication

In-network duplication

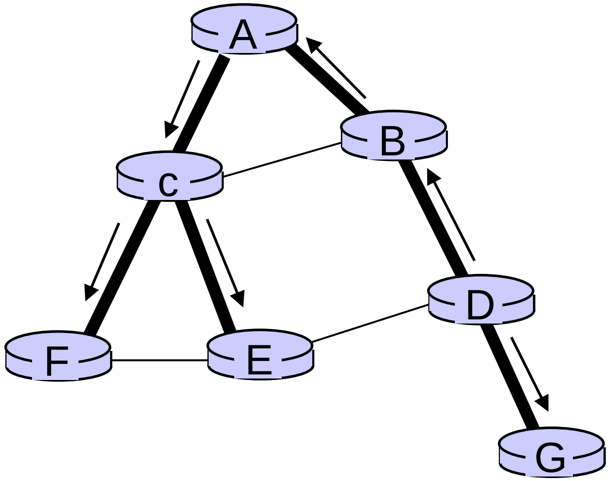
- *flooding:* when node receives broadcast packet, sends copy to all neighbors
  - problems: cycles & broadcast storm
- *controlled flooding:* node only broadcasts pkt if it hasn't broadcast same packet before
  - node keeps track of packet ids already broadacsted
  - or reverse path forwarding (RPF): only forward packet if it arrived on shortest path between node and source
- *spanning tree:*
  - no redundant packets received by any node

Spanning tree

- first construct a spanning tree
- nodes then forward/make copies only along spanning tree



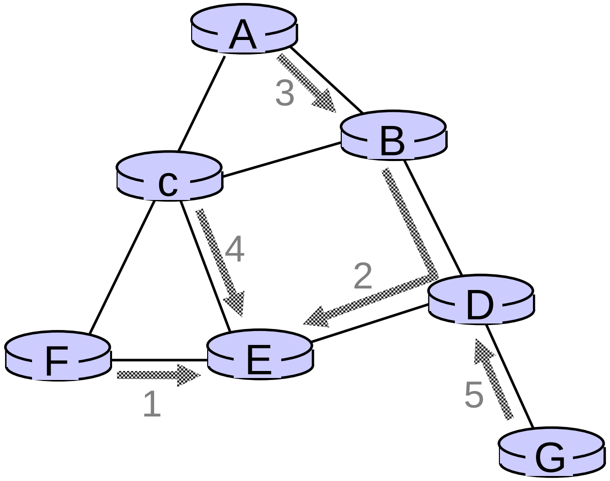
(a) broadcast initiated at A



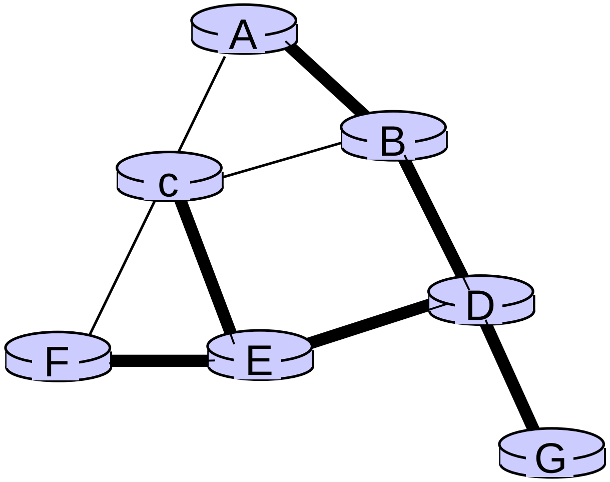
(b) broadcast initiated at D

Spanning tree: creation

- center node
- each node sends unicast join message to center node
  - message forwarded until it arrives at a node already belonging to spanning tree



(a) stepwise construction of spanning tree (center: E)

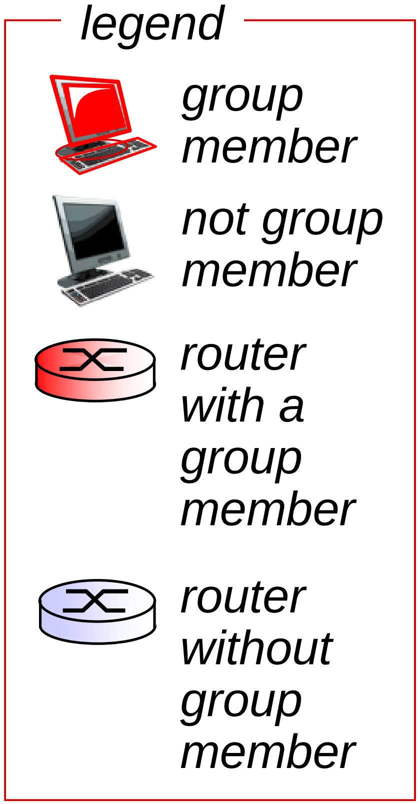


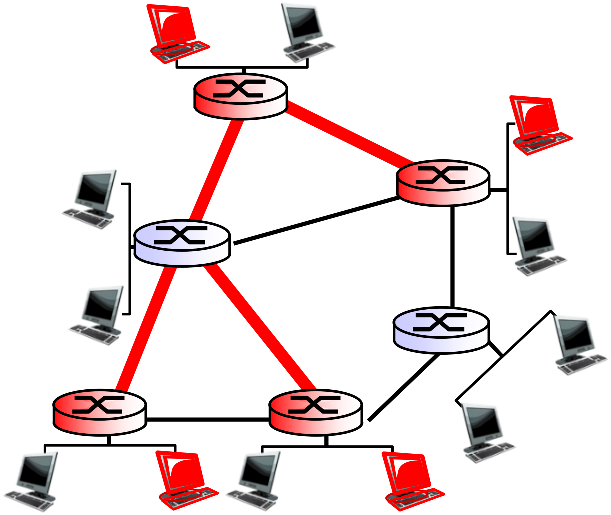
(b) constructed spanning tree

Multicast routing: problem statement

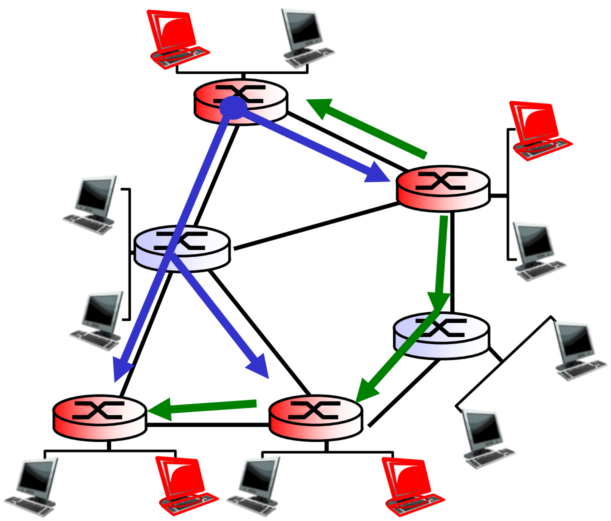
goal: find a tree (or trees) connecting routers having local mcast group members

- *shared-tree:* same tree used by all group members
- *source-based:* different tree from each sender to rcvrs





shared tree



source-based trees

Approaches for building mcast trees

approaches:

- *source-based tree:* one tree per source
  - shortest path trees (Dijkstra's algorithm)
  - reverse path forwarding
- *group-shared tree:* group uses one tree
  - minimal spanning (Steiner)
  - center-based trees

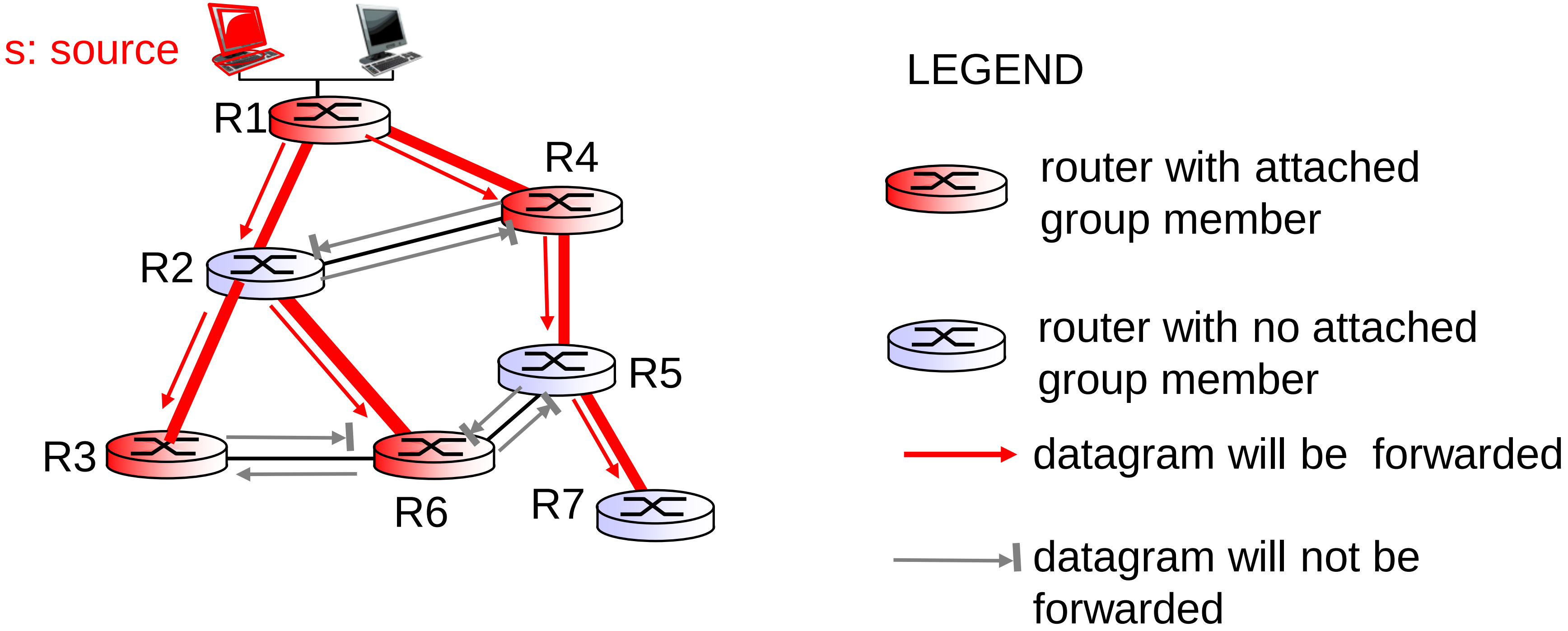
…we first look at basic approaches, then specific protocols adopting these approaches

Reverse path forwarding

- rely on router’ s knowledge of unicast shortest path from it to sender
- each router has simple forwarding behavior:

```
if (mcast datagram received on incoming link on
     shortest path back to center)
   then flood datagram onto all outgoing links
   else ignore datagram
```

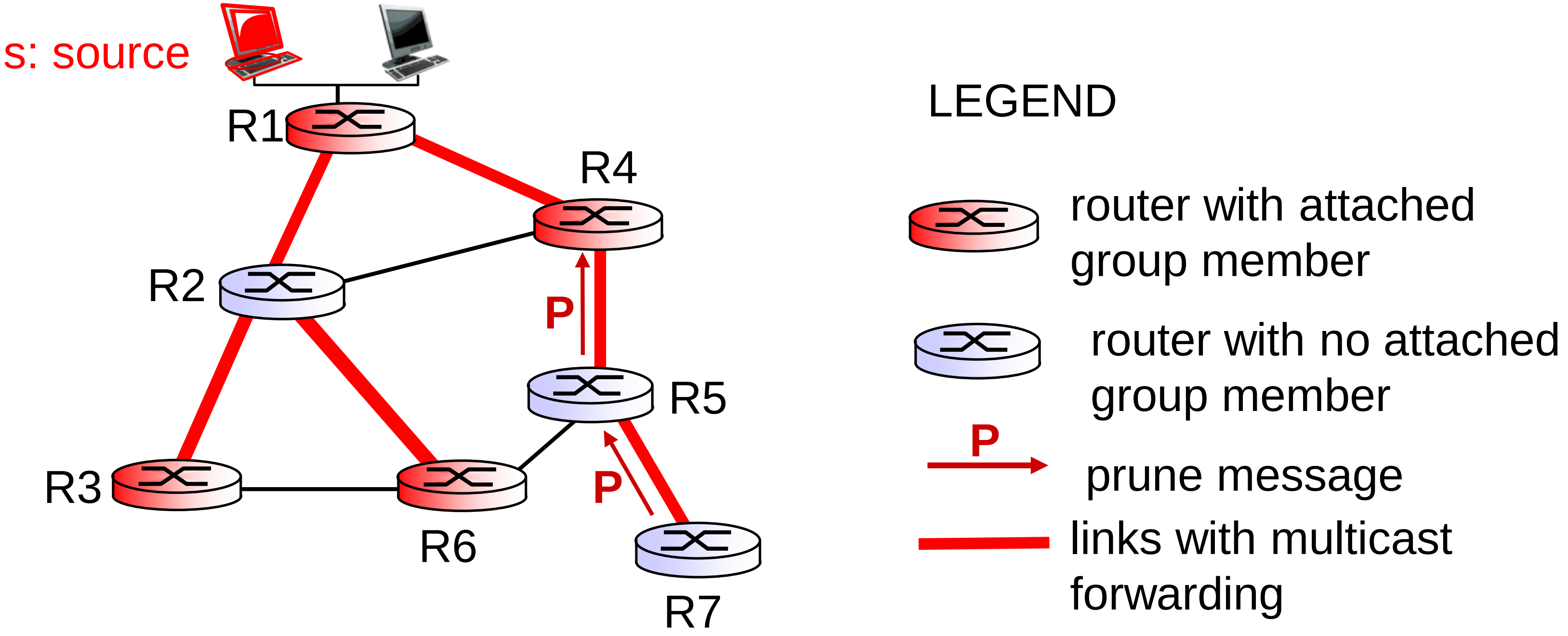
Reverse path forwarding: example


s: source
R1
R4
R2
R5
R3
R6
R7
LEGEND
router with attached group member
router with no attached group member
datagram will be forwarded
datagram will not be forwarded

- result is a source-specific *reverse* SPT
  - may be a bad choice with asymmetric links

Reverse path forwarding: pruning

- forwarding tree contains subtrees with no mcast group members
  - no need to forward datagrams down subtree
  - "prune" msgs sent upstream by router with no downstream group members


s: source
R1
R4
R2
P
R5
R3
R6
P
R7
LEGEND
router with attached group member
router with no attached group member
P
prune message
links with multicast forwarding

Shared-tree: steiner tree

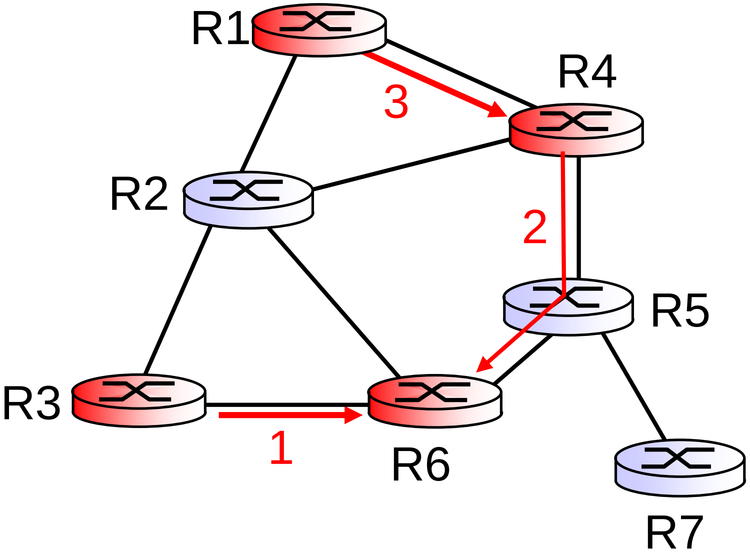
- *steiner tree:* minimum cost tree connecting all routers with attached group members
- problem is NP-complete
- excellent heuristics exists
- not used in practice:
  - computational complexity
  - information about entire network needed
  - monolithic: rerun whenever a router needs to join/leave

Center-based trees

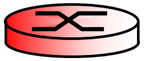
- single delivery tree shared by all
- one router identified as *"center"* of tree
- to join:
  - edge router sends unicast *join-msg* addressed to center router
  - *join-msg* "processed" by intermediate routers and forwarded towards center
  - *join-msg* either hits existing tree branch for this center, or arrives at center
  - path taken by *join-msg* becomes new branch of tree for this router

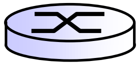
Center-based trees: example

suppose R6 chosen as center:



LEGEND


router with attached group member


router with no attached group member

path order in which join messages generated

Network Layer: A Summary