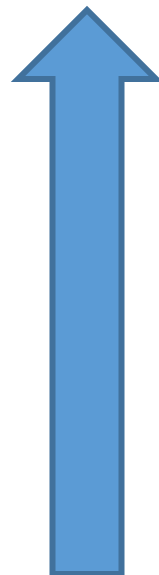
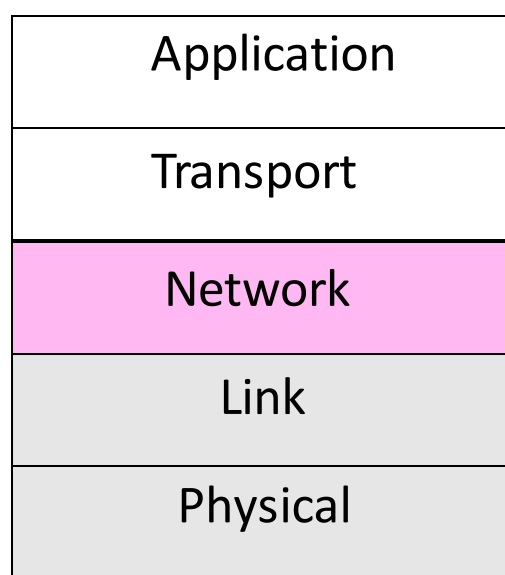


Network Layer

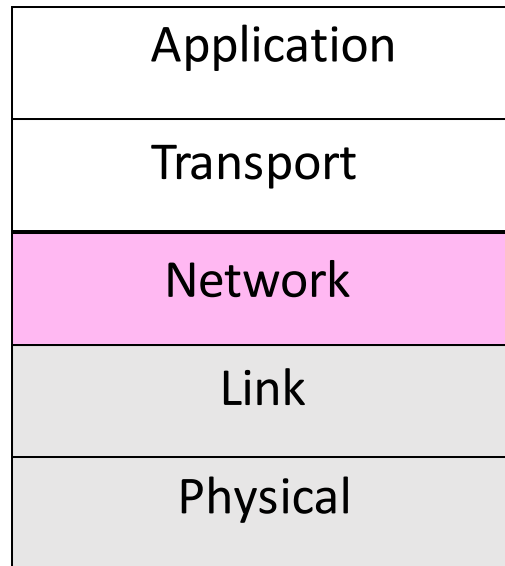
# Where we are in the Course

- Moving on up to the Network Layer!



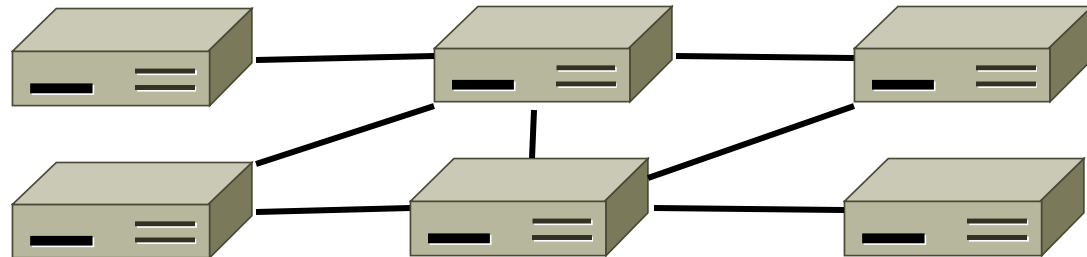
# Network Layer

- How to connect different link layer networks
  - Routing as the primary concern



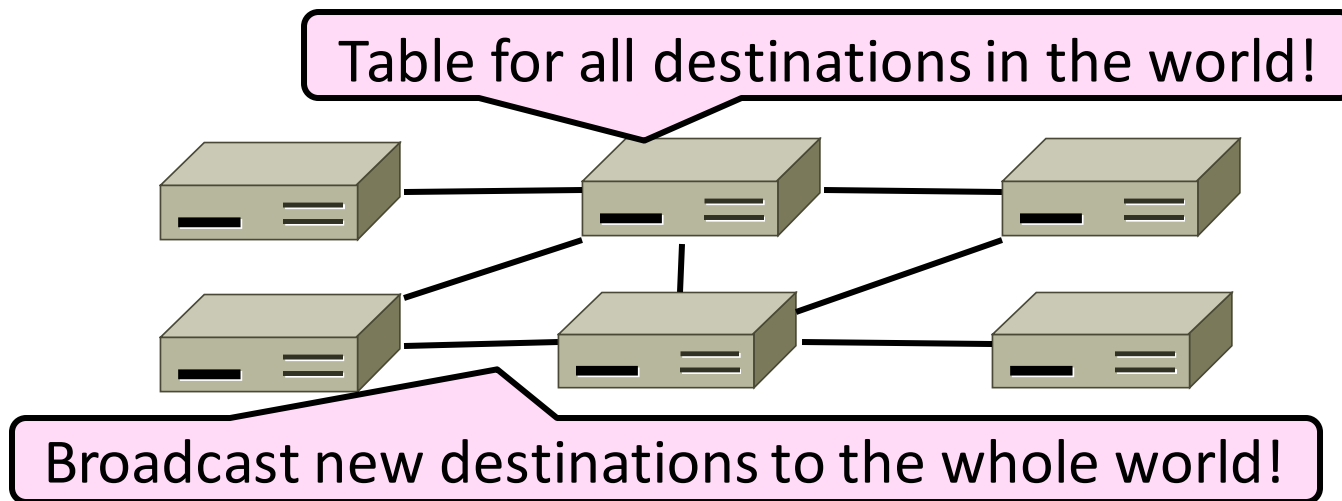
# Why do we need a Network layer?

- We can already build networks with links and switches and send frames between hosts ...



# Shortcomings of Switches

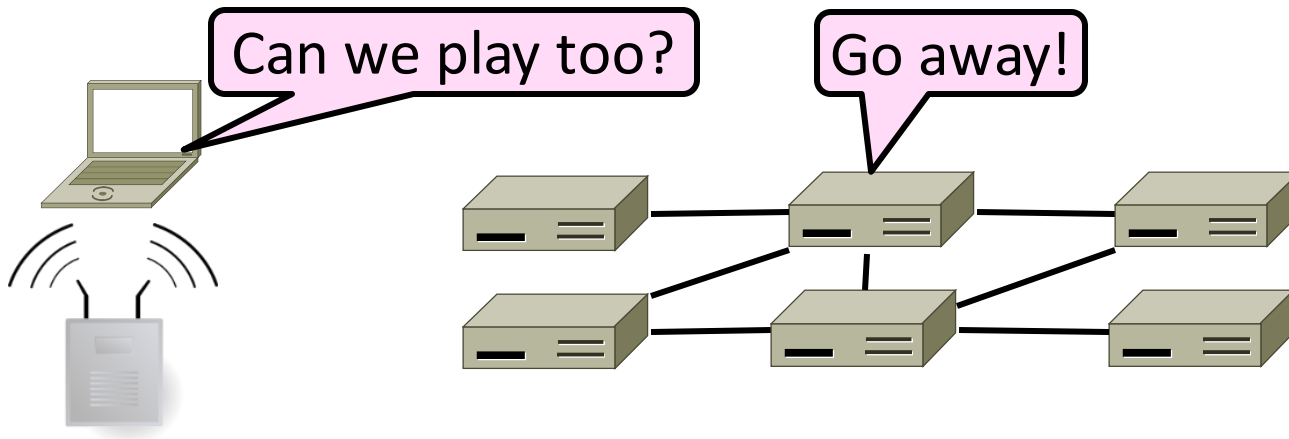
1. Don't scale to large networks
  - Blow up of routing table, broadcast



# Shortcomings of Switches (2)

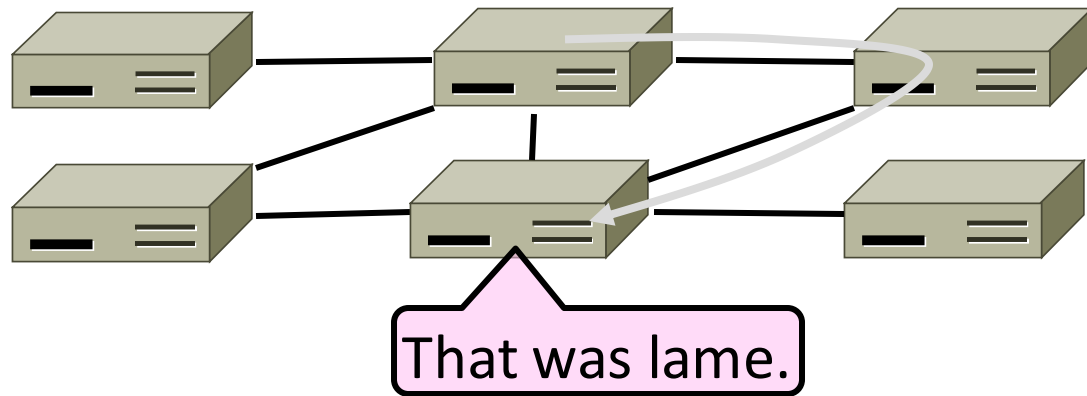
## 2. Don't work across more than one link layer technology

- 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)

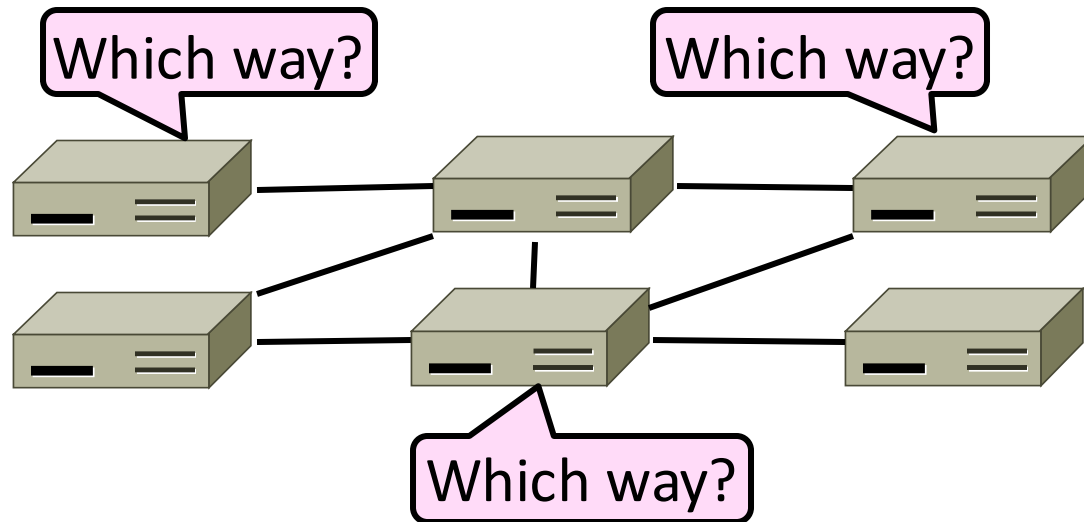


# Topics

- Network service models
  - Datagrams (packets), virtual circuits
- IP (Internet Protocol)
  - Internetworking
  - Forwarding (Longest Matching Prefix)
  - Helpers: ARP and DHCP
  - Errors: ICMP (traceroute!)
  - IPv6, scaling IP to the world
  - NAT, and “middleboxes”
- Routing Algorithms

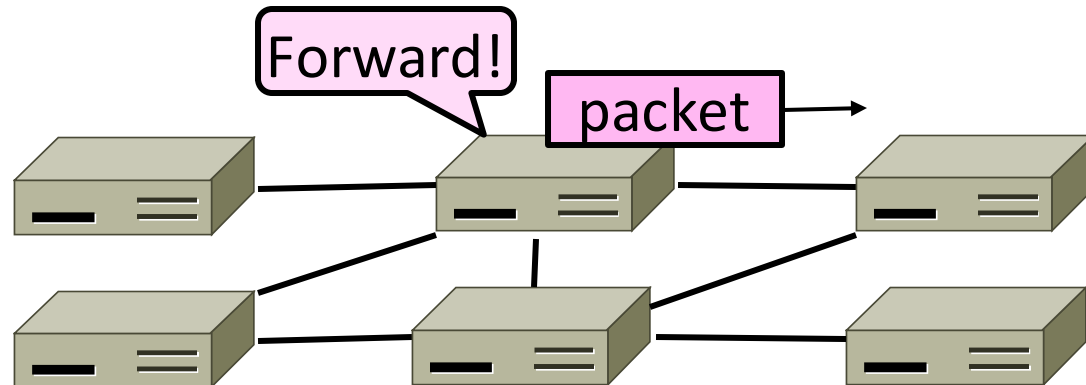
# Routing vs. Forwarding

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



# Routing vs. Forwarding (2)

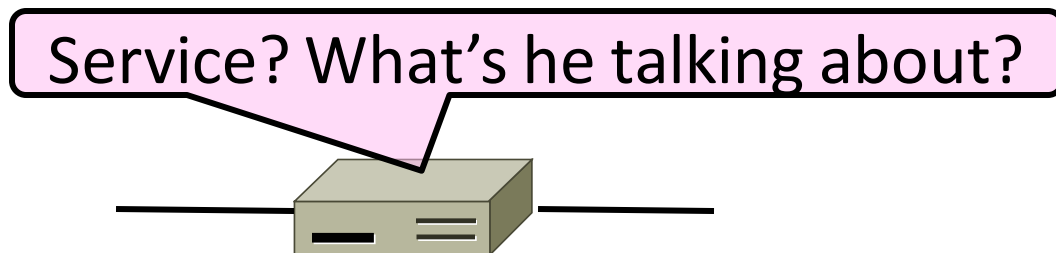
- Forwarding is the process of sending a packet
  - Node process (local) and fast



# Networking Services

# Topic

- What kind of service does the Network layer provide to the Transport layer?
  - How is it implemented at routers?



# Two Network Service Models

- Datagrams, or connectionless service

- Like postal letters

- (IP as an example)



- Virtual circuits, or connection-oriented service

- Like a telephone call

- **Cut for space**

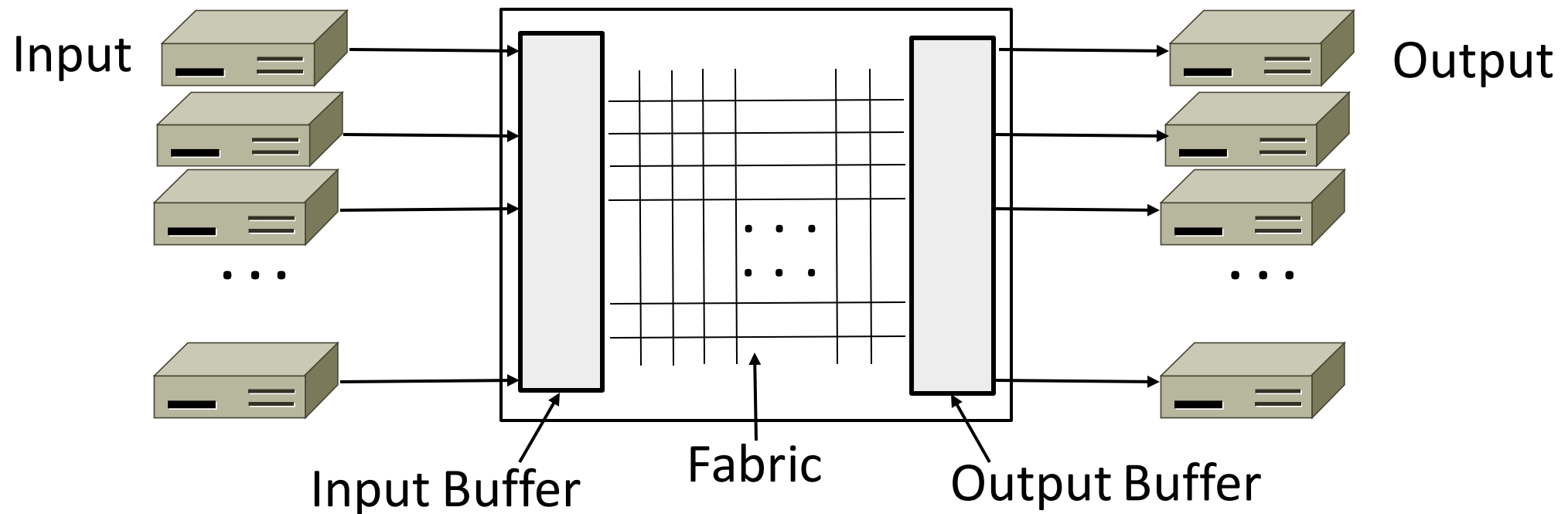


# Store-and-Forward Packet Switching

- Implemented with store-and-forward packet switching
  - Routers receive a complete packet, storing it temporarily if necessary before forwarding it onwards
  - We use statistical multiplexing to share link bandwidth over time

# Store-and-Forward (2)

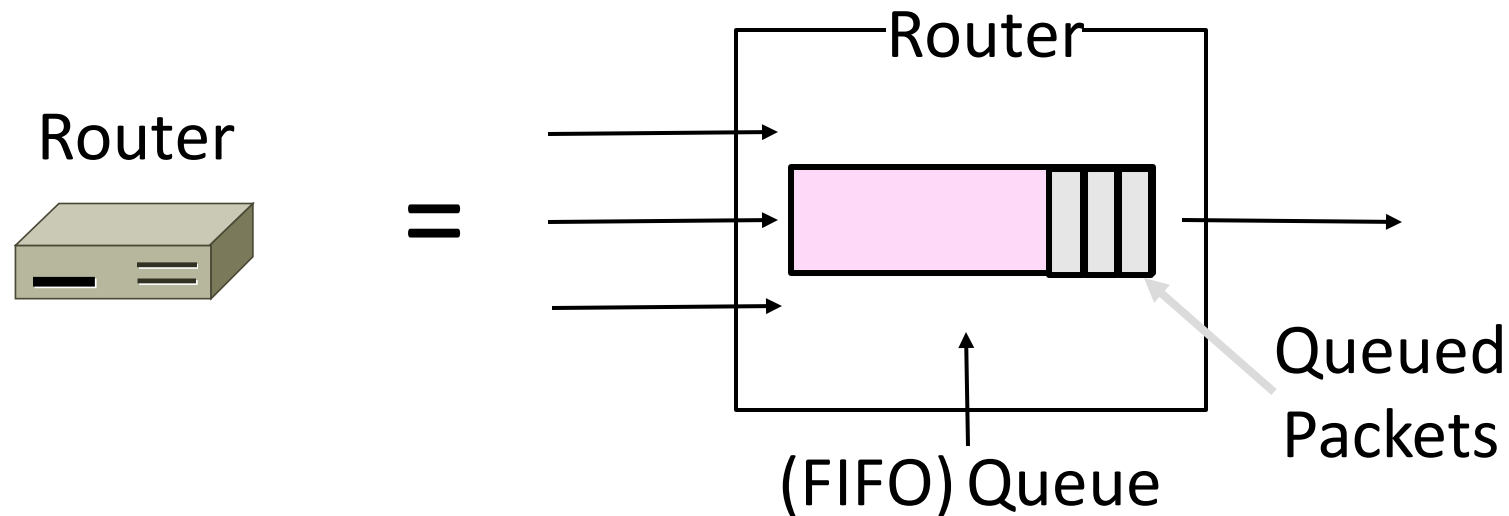
- Switching element has internal buffering for contention





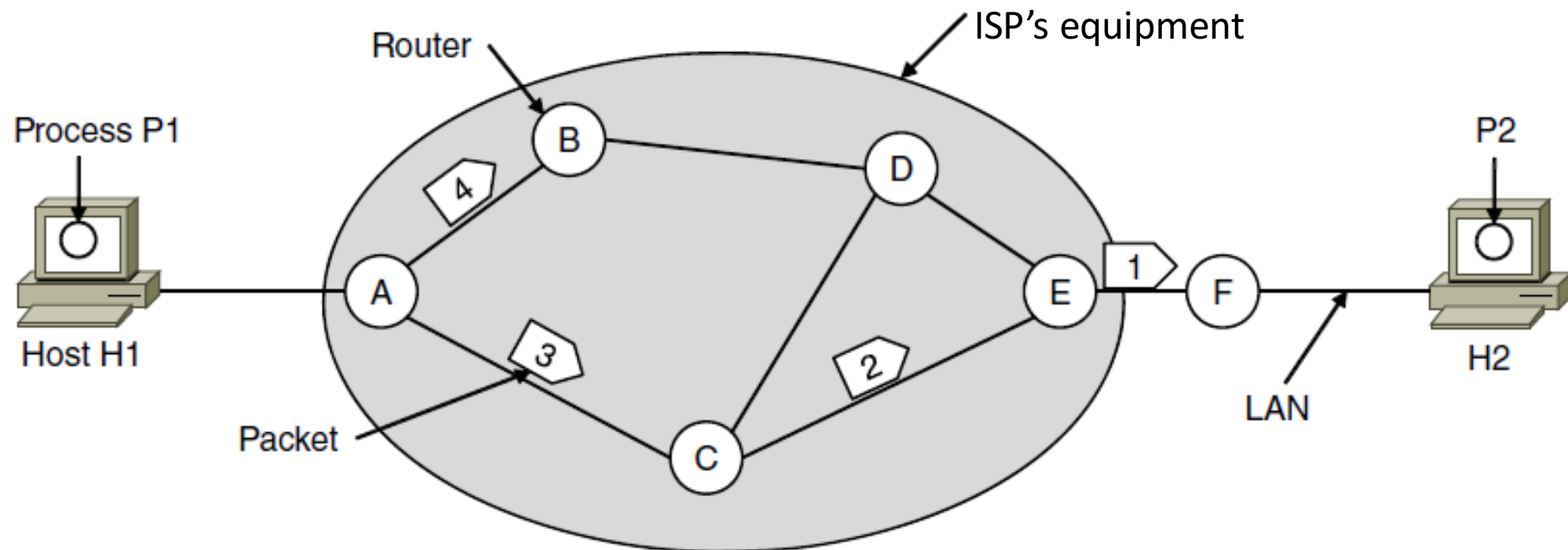
# Store-and-Forward (3)

- Simplified view with per port output buffering
  - Buffer is typically a FIFO (First In First Out) queue
  - If full, packets are discarded (congestion, later)



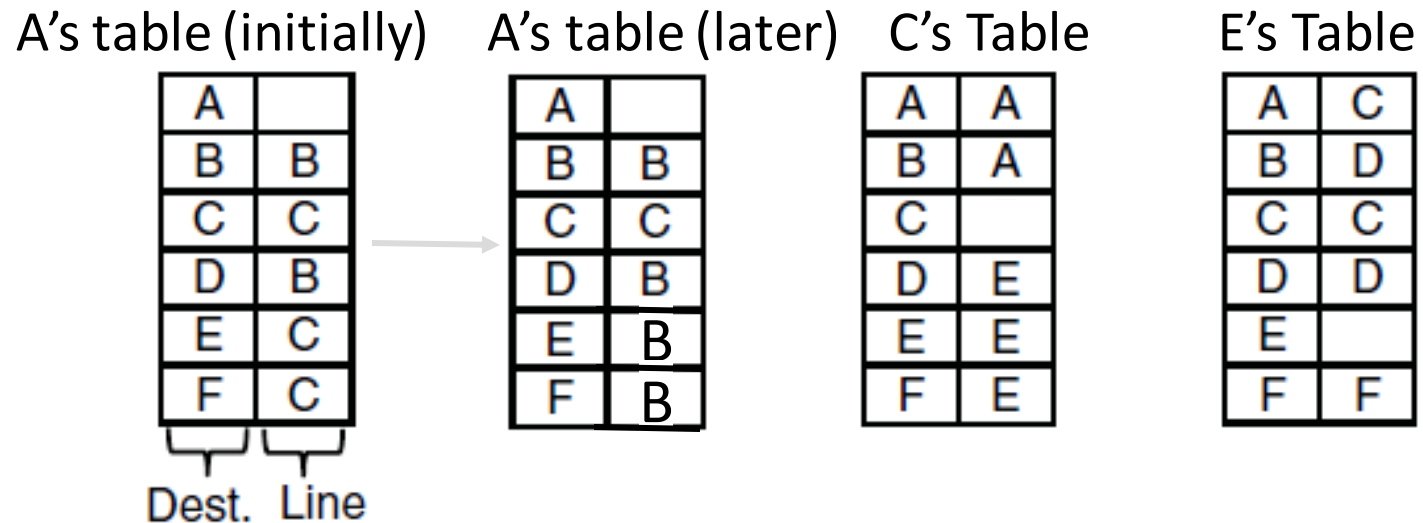
# Datagram Model

- Packets contain a destination address; each router uses it to forward packets, maybe on different paths



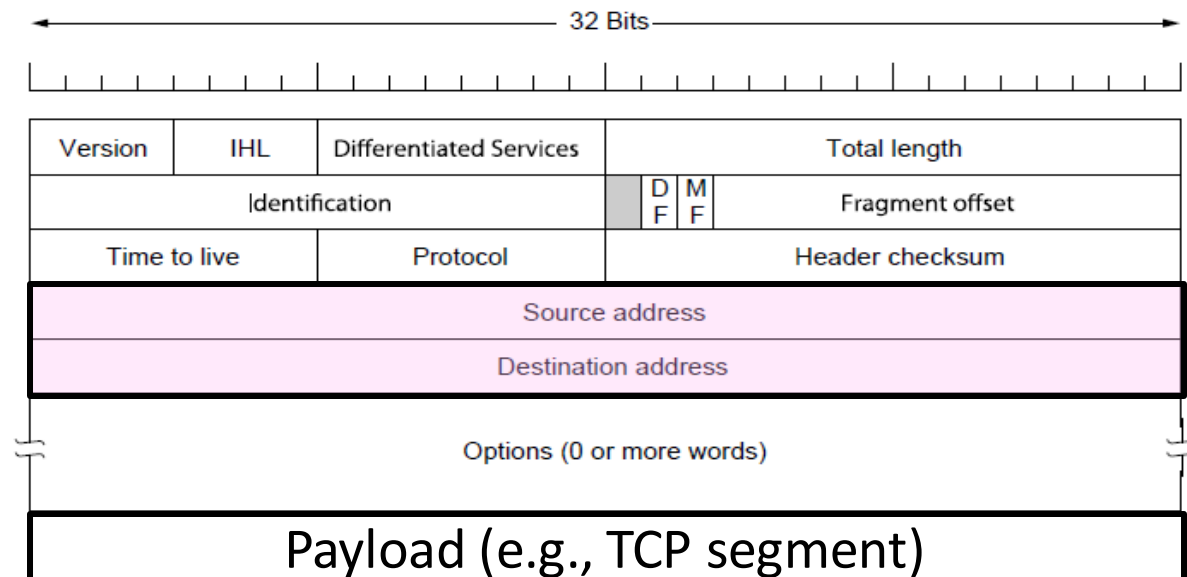
# Datagram Model (2)

- Each router has a forwarding table keyed by address
  - Gives next hop for each destination address; may change



# IP (Internet Protocol)

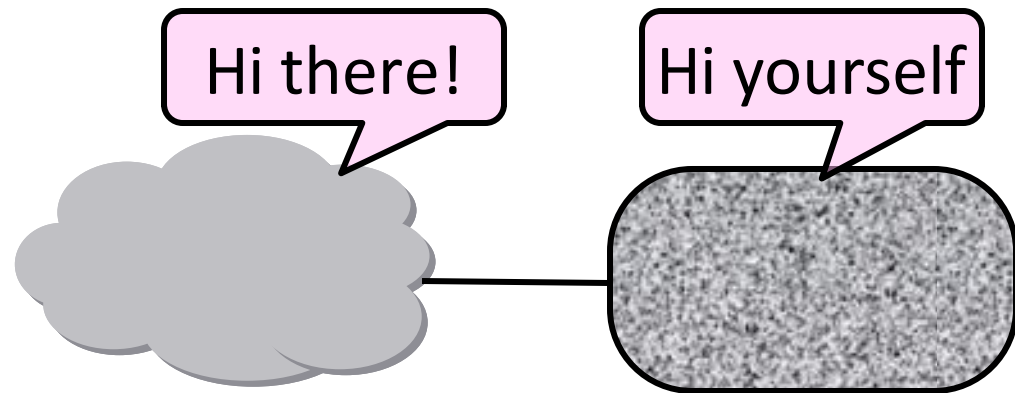
- Network layer of the Internet, uses datagrams (next)
  - IPv4 carries 32 bit addresses on each packet (often 1.5 KB)



Internetworking (IP)

# Topic

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



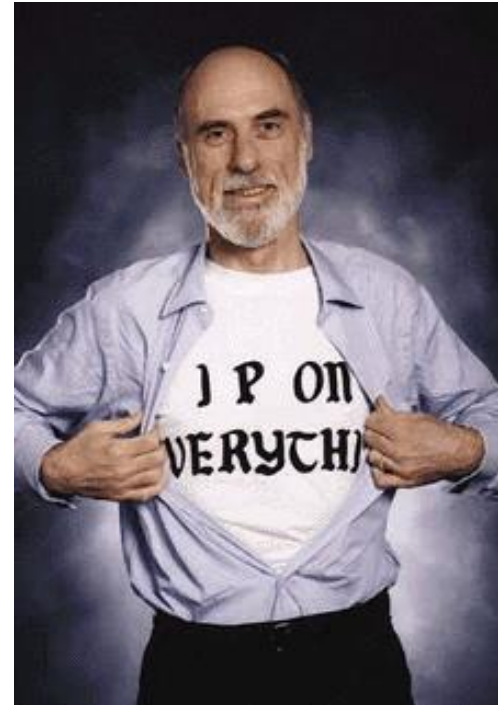
# How Networks May Differ

- Basically, in a lot of ways:
  - Service model (datagrams, Virtual Circuits)
  - Addressing (what kind)
  - QOS (priorities, no priorities)
  - Packet sizes
  - Security (whether encrypted)
- Internetworking hides the differences with a common protocol. (Uh oh.)

# Internetworking – Cerf and Kahn

- Pioneers: Cerf and Kahn
  - “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



Bob Kahn

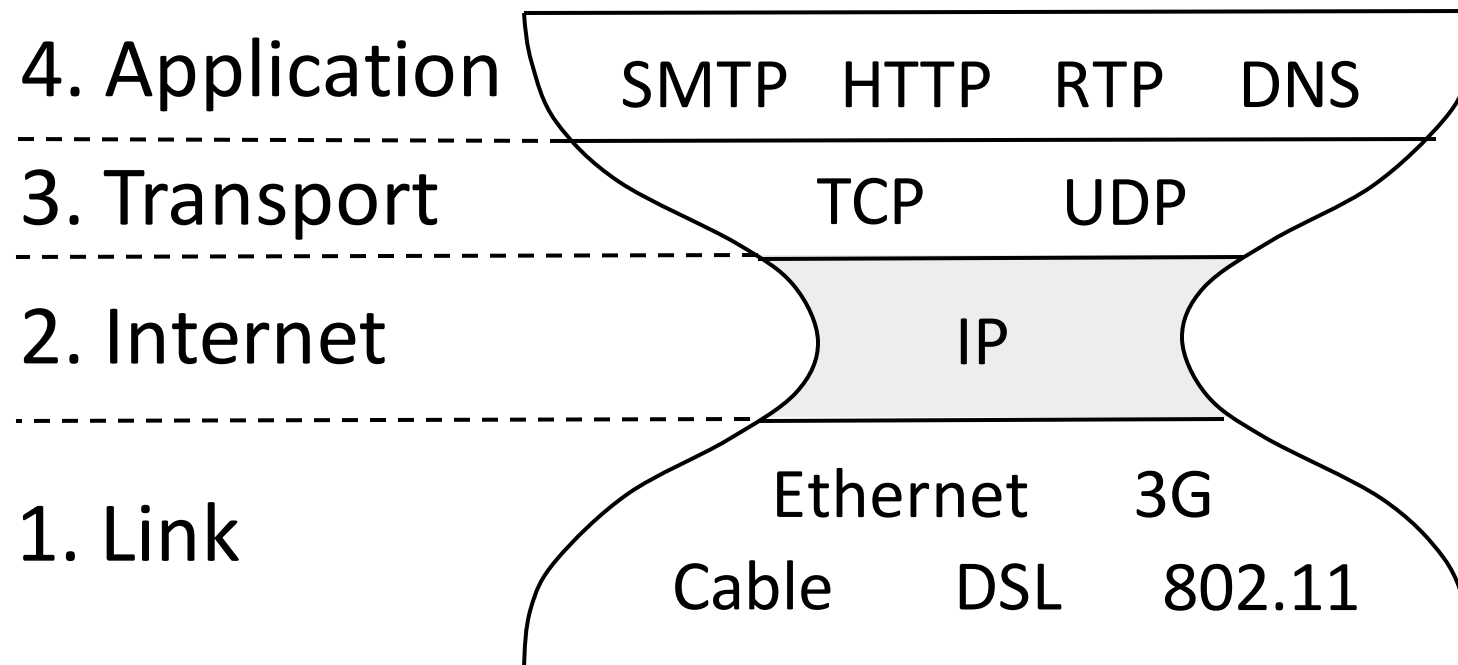


© 2009 IEEE



# Internet Reference Model

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

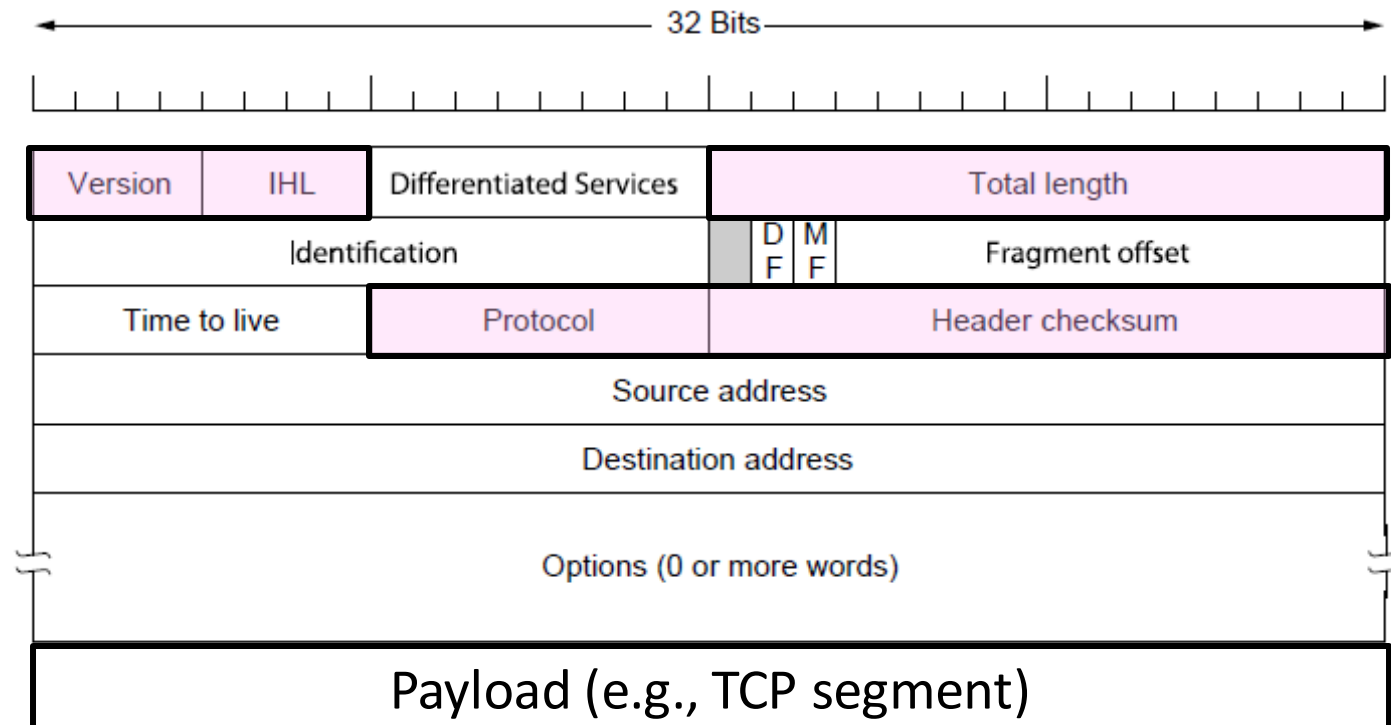


# IP as a Lowest Common Denominator

- Suppose only some networks support QOS or security etc.
  - Difficult for internet network to support
- Pushes IP to be a “lowest common denominator”
  - Asks little of lower-layer networks
  - Gives little as a higher layer service

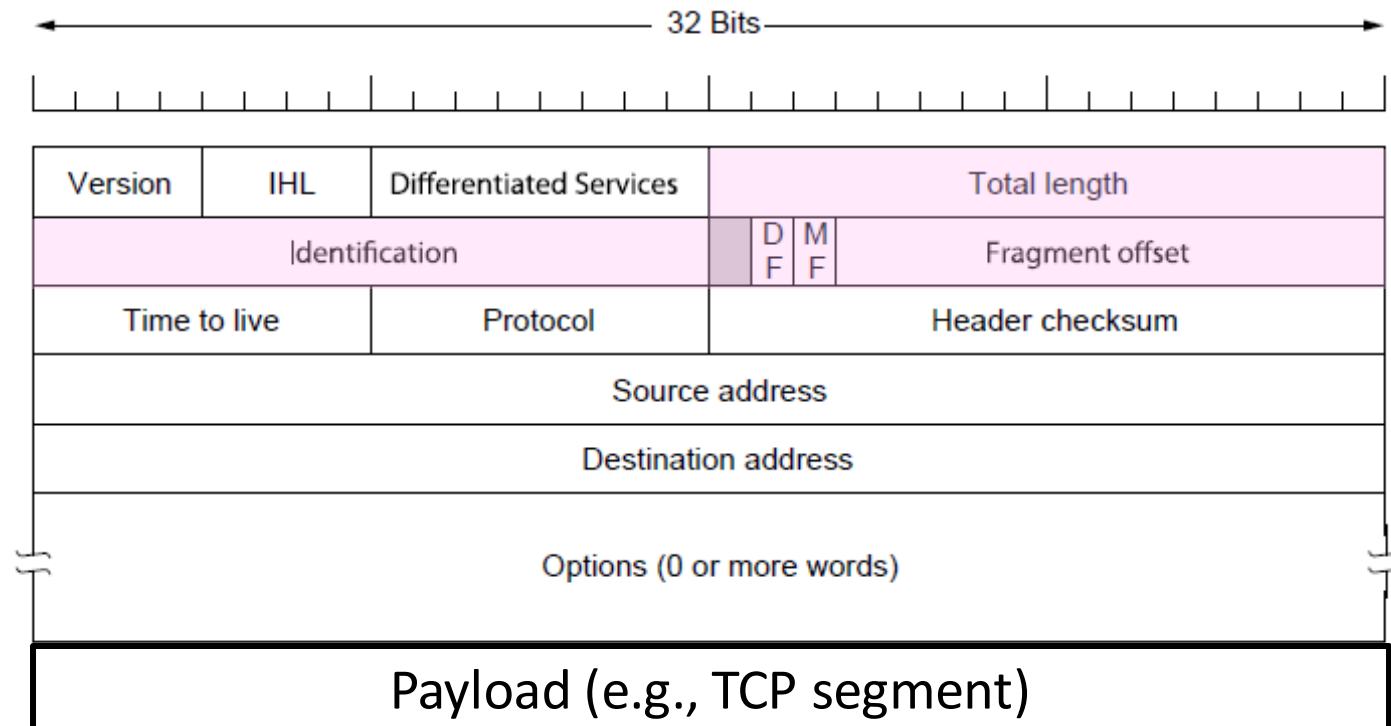
# IPv4 (Internet Protocol)

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



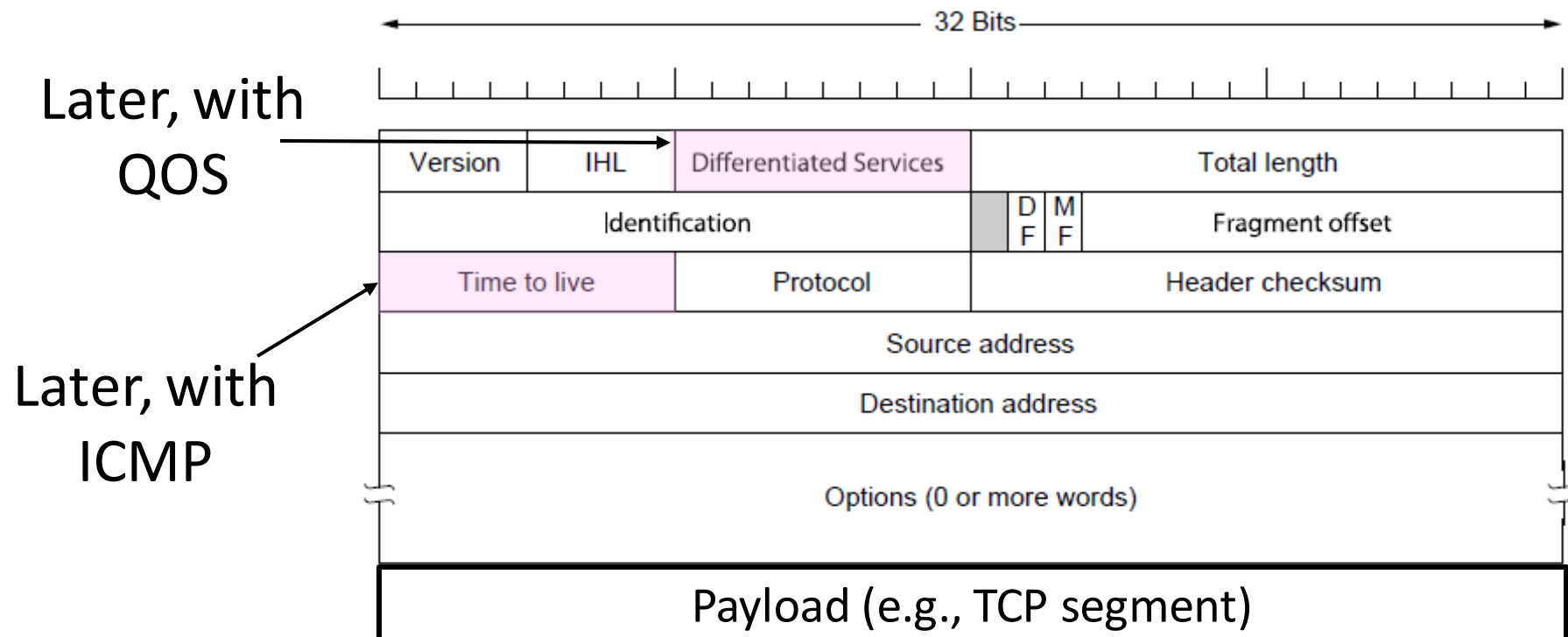
# IPv4 (2)

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



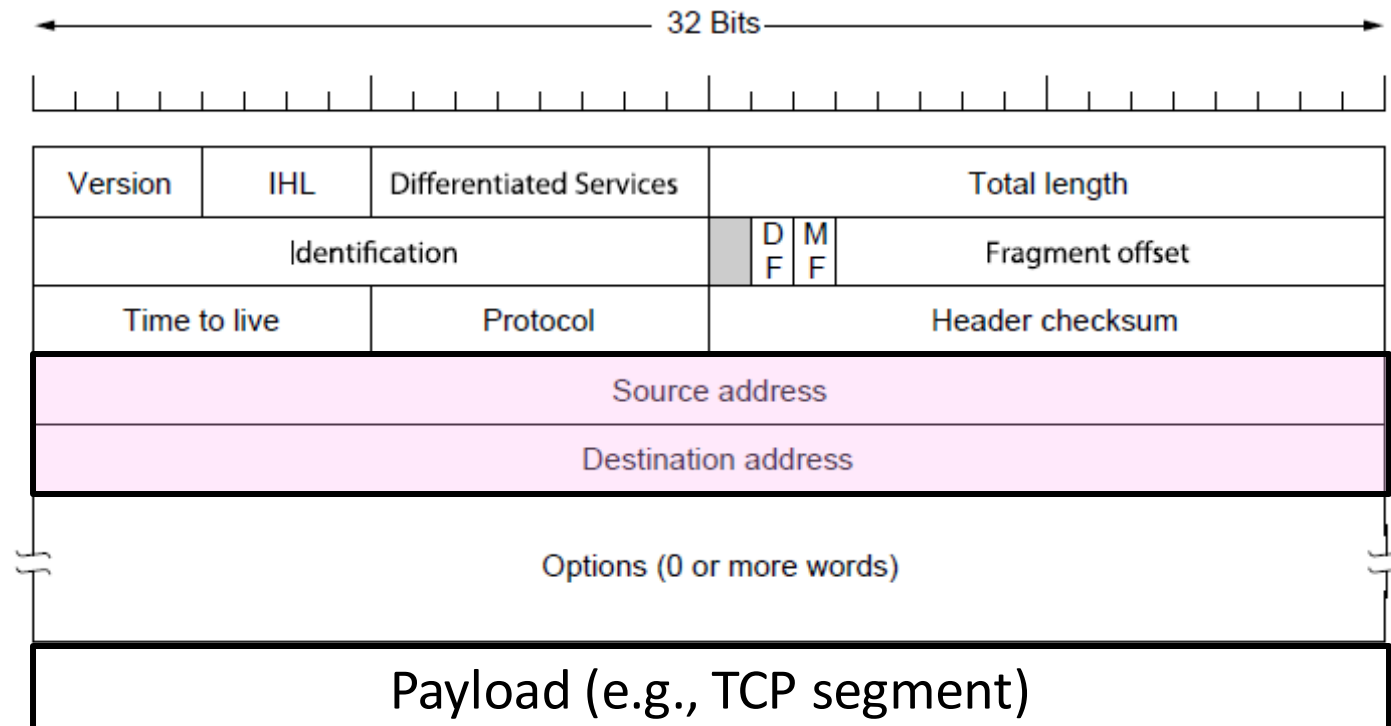
# IPv4 (3)

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



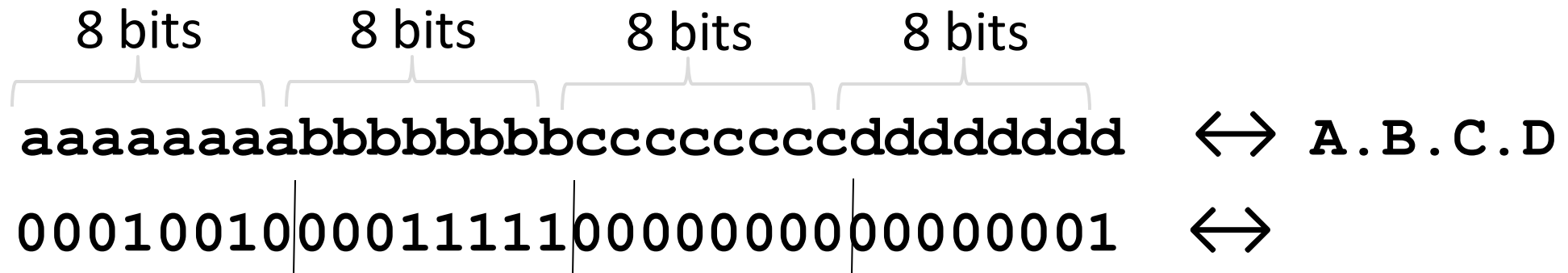
# IPv4 (4)

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



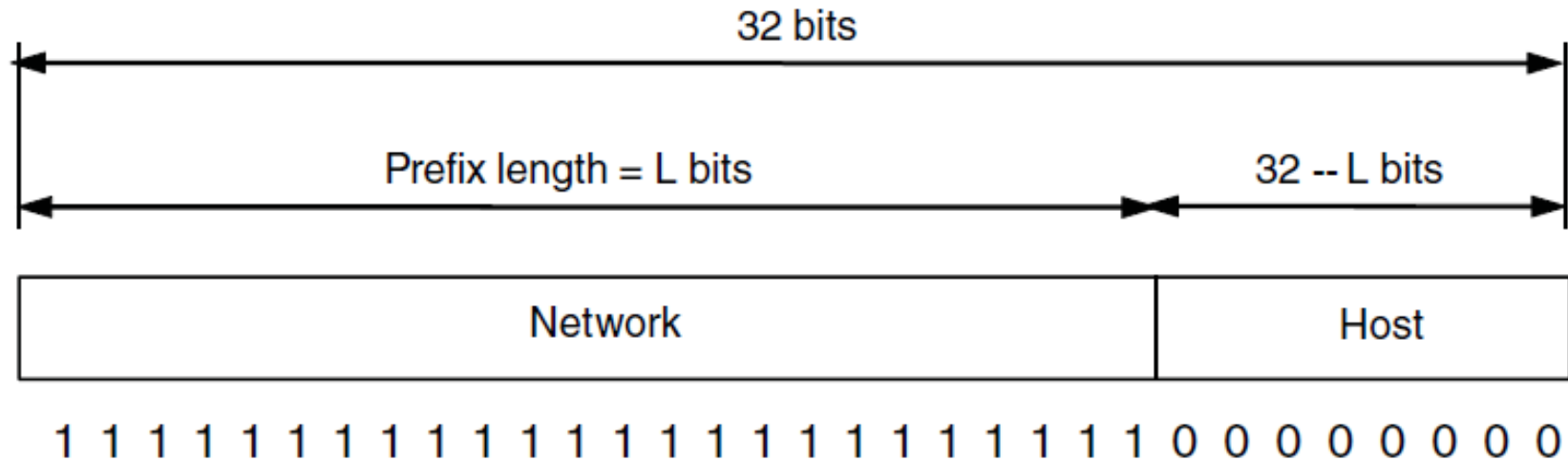
# IP Addresses

- 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



# IP Prefixes

- 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

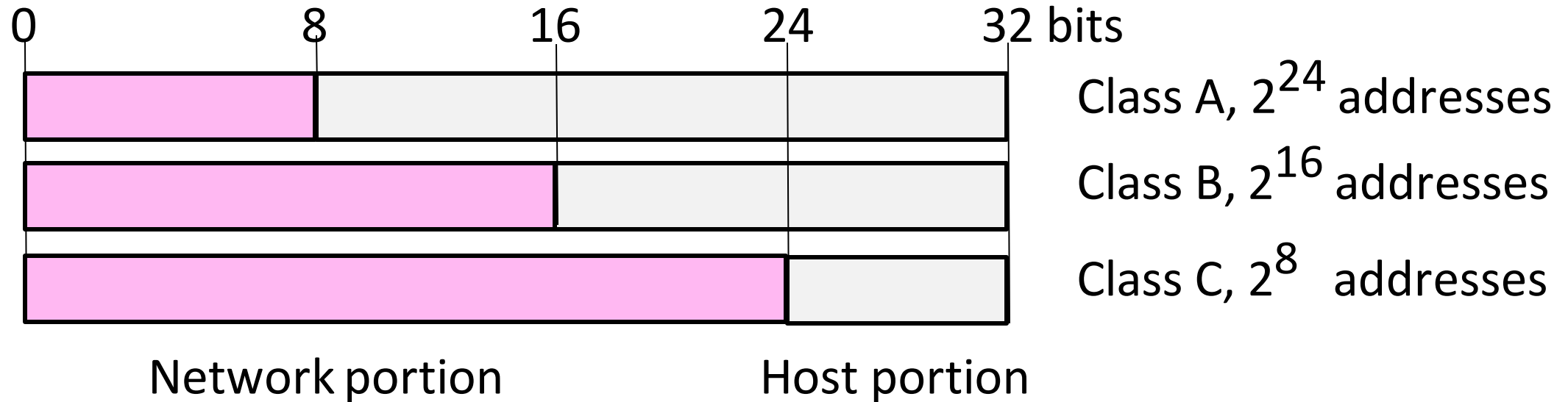






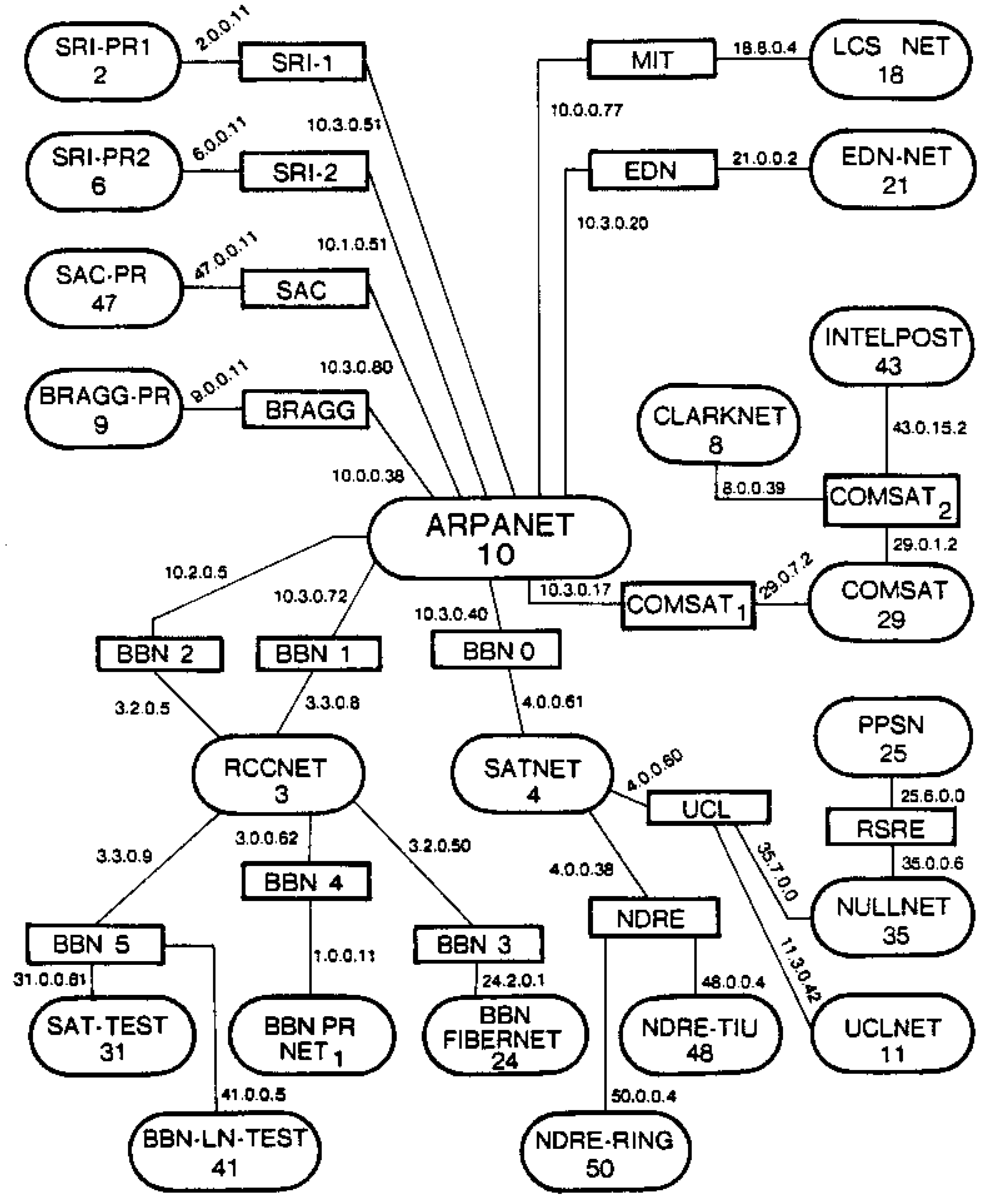
# Classful IP Addressing

- 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



# Classful IP Addressing

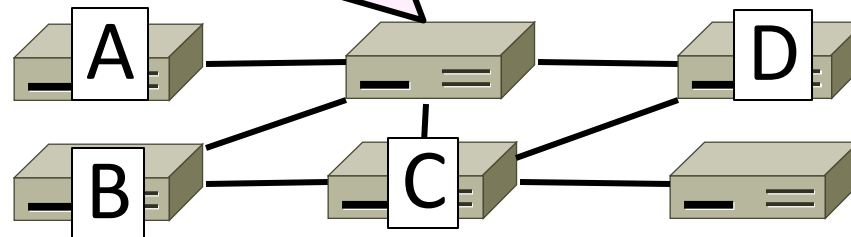
- This is an ARPANet assignment.



# IP Forwarding

- Addresses on one network belong to a unique prefix
- Node uses a table that lists the next hop for prefixes

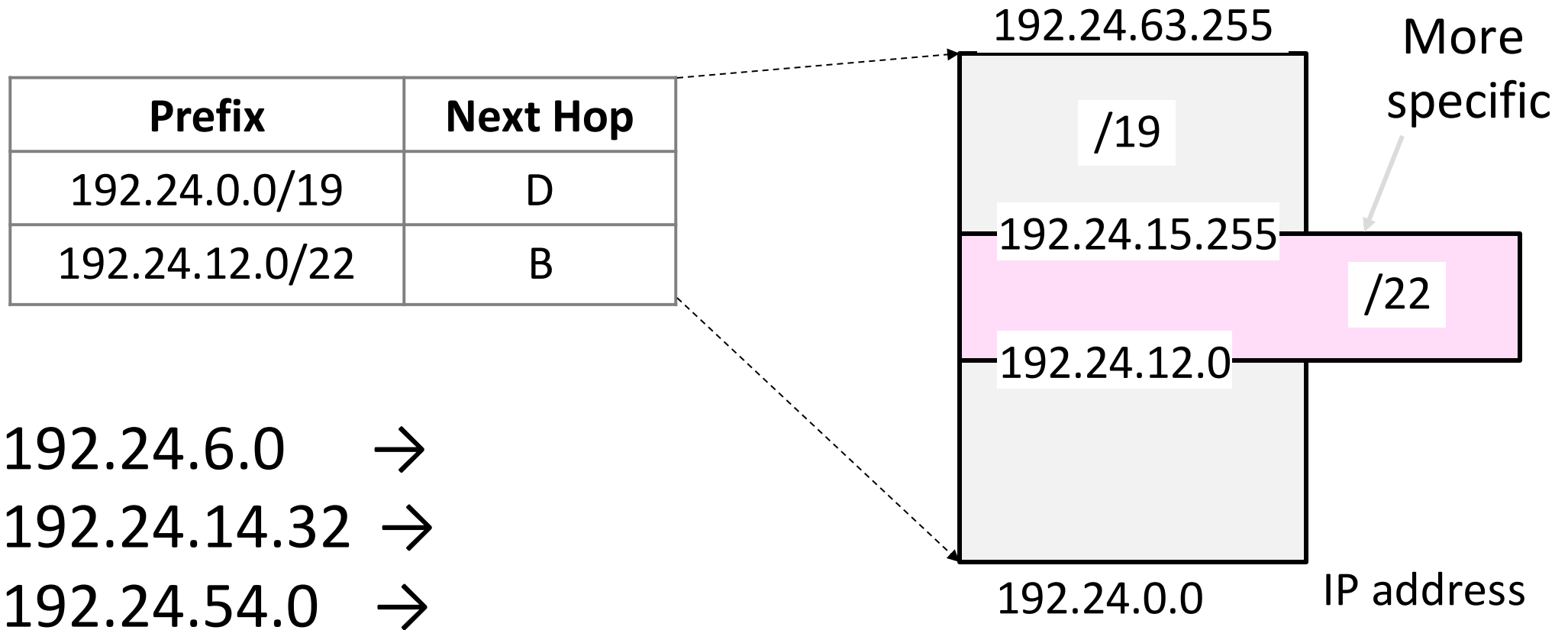
Prefix	Next Hop
192.24.0.0/19	D
192.24.12.0/22	B



# 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

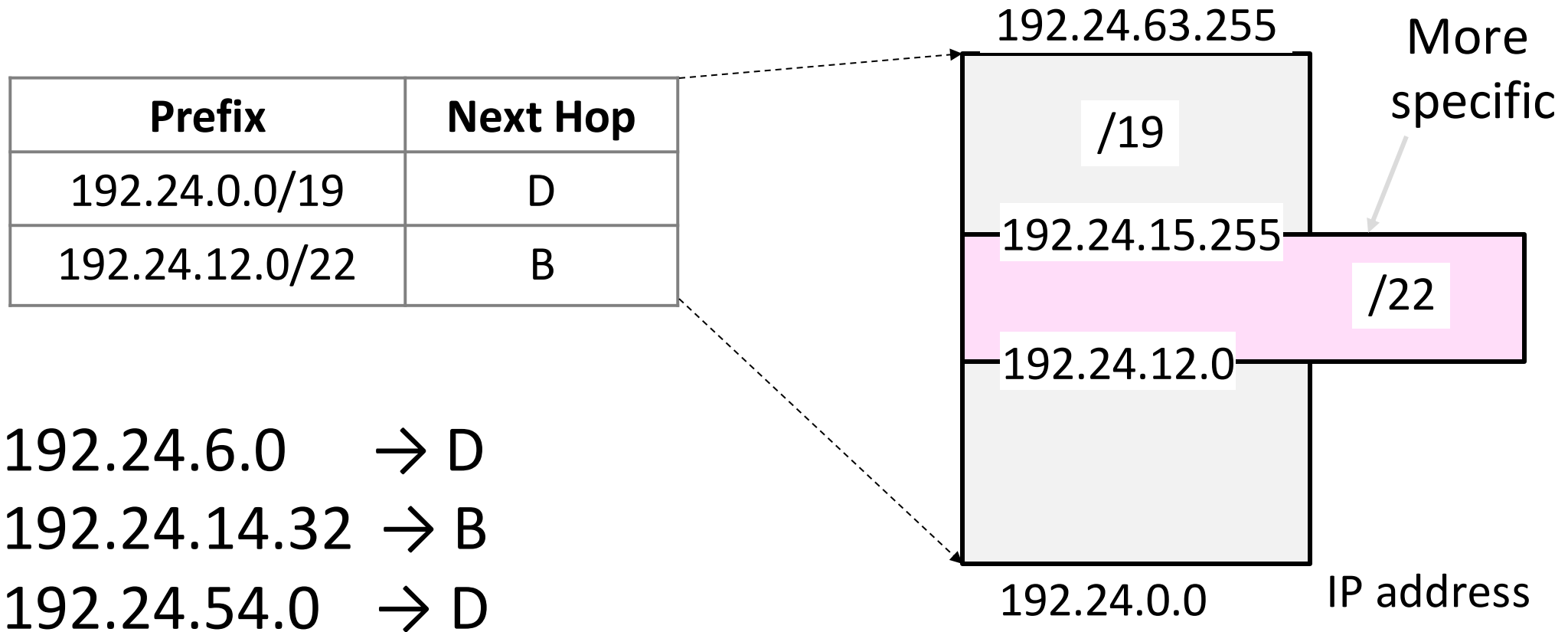
# Longest Matching Prefix (2)



# IP Address Work Slide:

- Route to D = 192.00011x.x.x
- Route to B = 192.00011000.000011x.x
- 192.24.6.0 = 192.00011000.00000110.00000000
- 192.24.14.32 = 192.00011000.00001110.00010000
- 192.24.54.0 = 192.00011000.00110110.00000000

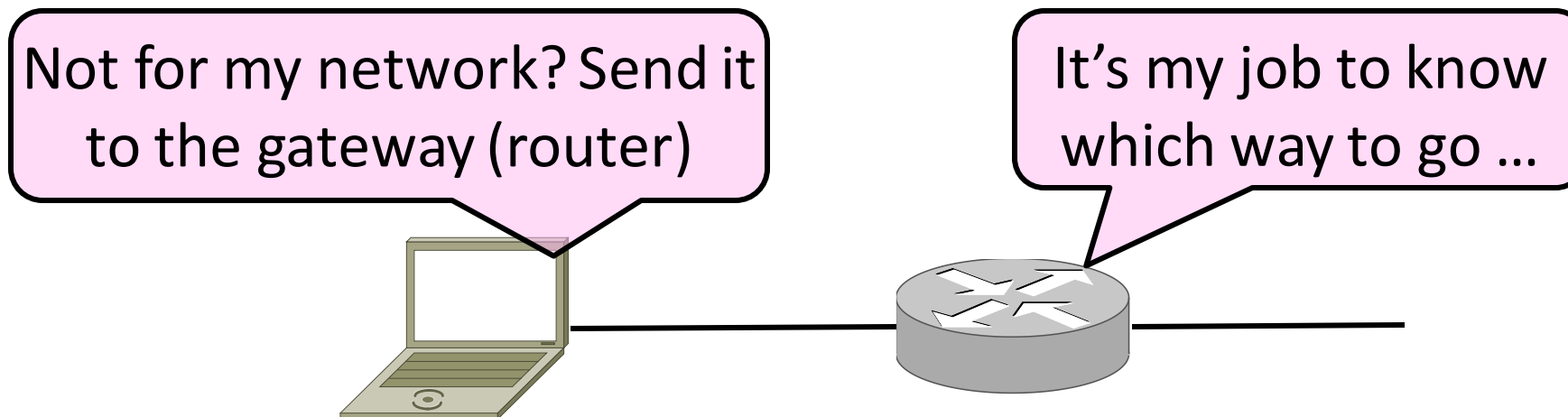
# Longest Matching Prefix (2)





# Host/Router Distinction

- In the Internet:
  - Routers do the routing, know way to all destinations
  - Hosts send remote traffic (out of prefix) to nearest router



# Host Networking

- Consists of 4 pieces of data:
  - IP Address
  - Subnet Mask
    - Defines local addresses
  - Gateway
    - Who (local) to send non-local packets to for routing
  - DNS Server (Later)

# Host Forwarding Table

- Give using longest matching prefix
  - 0.0.0.0/0 is a default route that catches all IP addresses

Prefix	Next Hop
My network prefix	Send to that IP
0.0.0.0/0	Send to my router

# Flexibility of Longest Matching Prefix

- Can provide default behavior, with less specifics
  - Send traffic going outside an organization to a border router (gateway)
- Can special case behavior, with more specifics
  - For performance, economics, security, ...

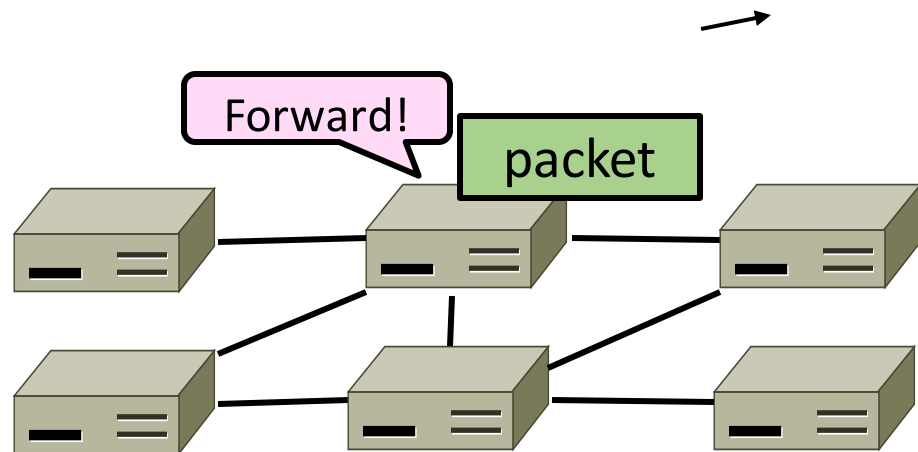
# Performance of Longest Matching Prefix

- Uses hierarchy for a compact table
  - Relies on use of large prefixes
- Lookup more complex than table
  - Used to be a concern for fast routers
  - Not an issue in practice these days

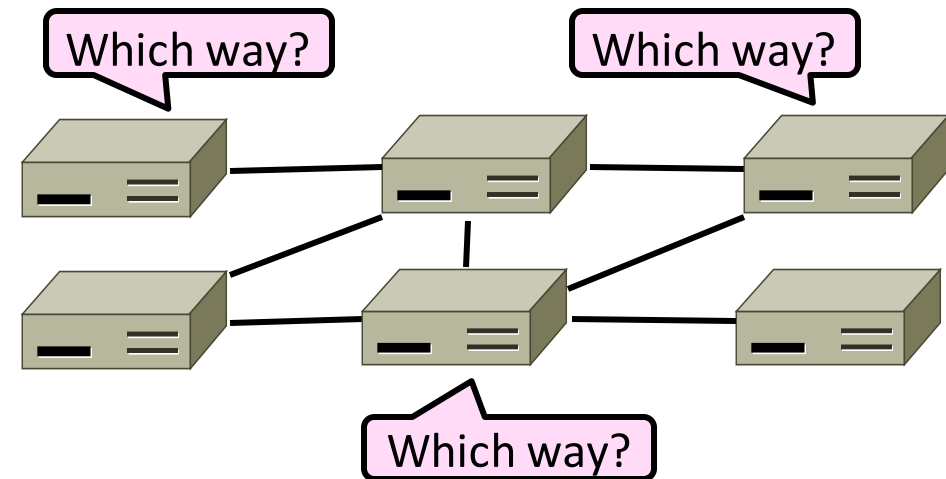
# Routing

# Routing versus Forwarding

- Forwarding is the process of sending a packet on its way

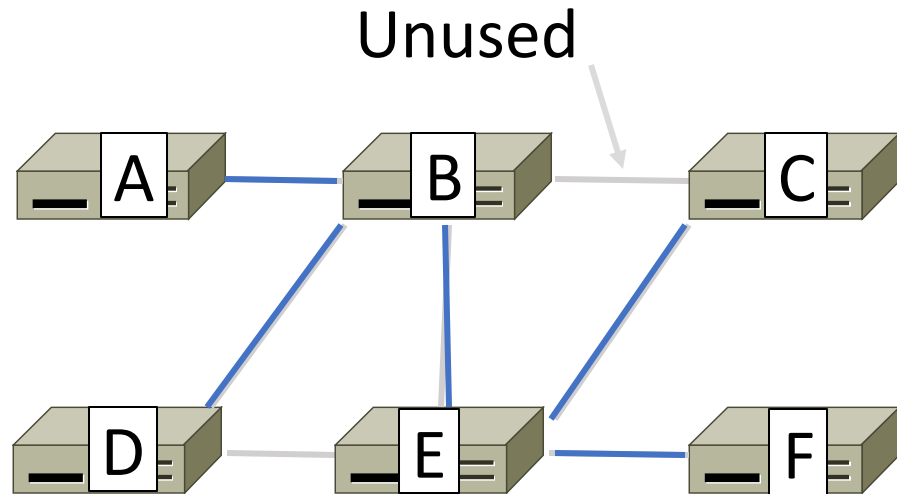


- Routing is the process of deciding in which direction to send traffic

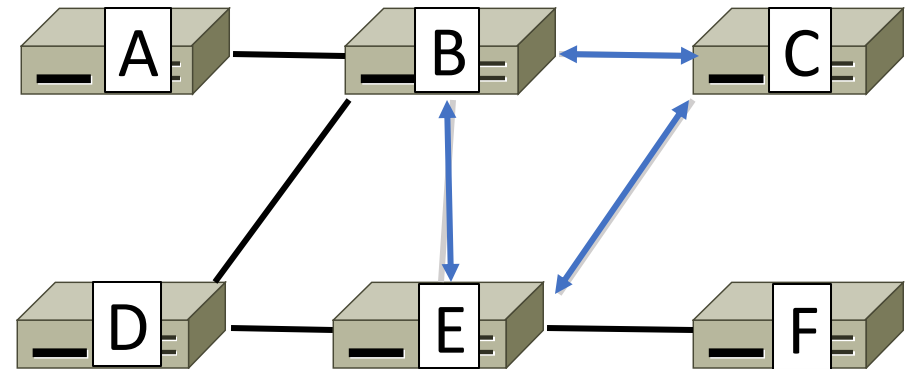


# Improving on the Spanning Tree

- Spanning tree provides basic connectivity
  - e.g., some path  $B \rightarrow C$



- Routing uses all links to find “best” paths
  - e.g., use BC, BE, and CE





# Perspective on Bandwidth Allocation

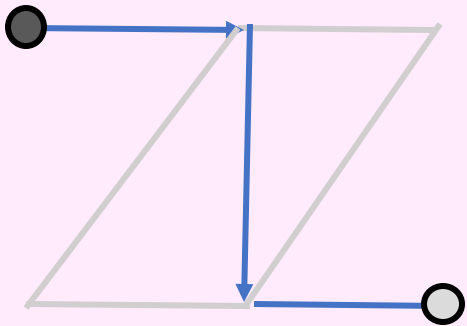
- Routing allocates network bandwidth adapting to failures; other mechanisms used at other timescales

<b>Mechanism</b>	<b>Timescale / Adaptation</b>
Load-sensitive routing	Seconds / Traffic hotspots
Routing	Minutes / Equipment failures
Traffic Engineering	Hours / Network load
Provisioning	Months / Network customers

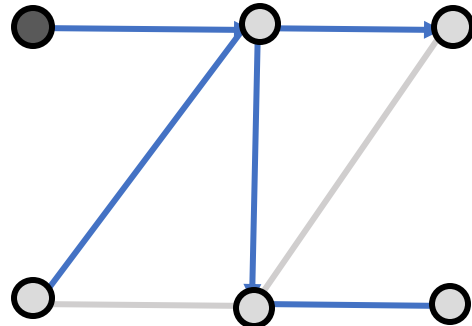
# Delivery Models

- Different routing used for different delivery models

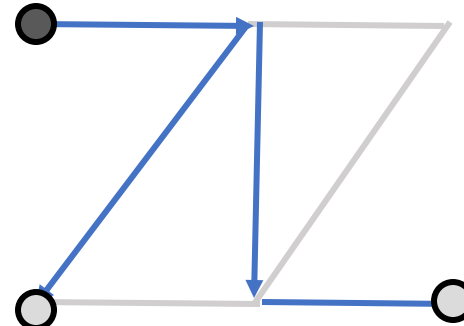
Unicast  
(§5.2)



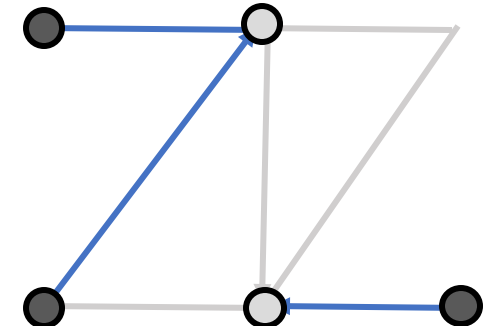
Broadcast  
(§5.2.7)



Multicast  
(§5.2.8)



Anycast  
(§5.2.9)



# 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

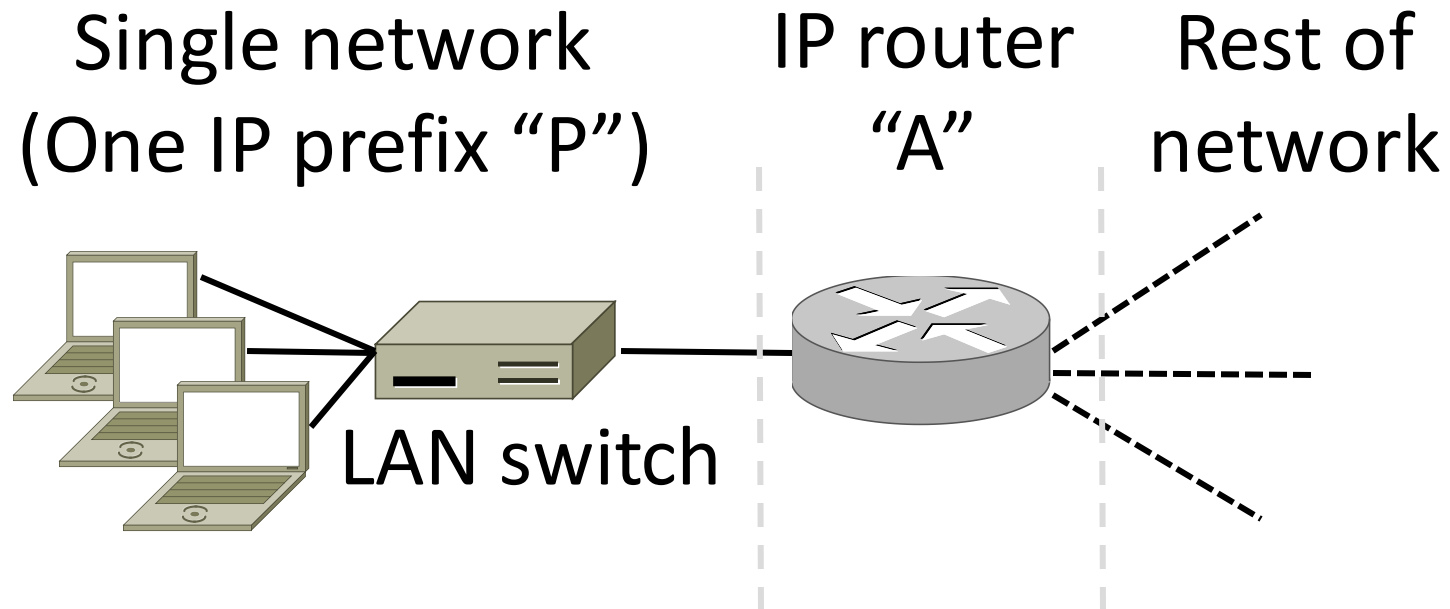
# Rules of Routing Algorithms

- Decentralized, distributed setting
  - All nodes are alike; no controller
  - Nodes only know what they learn by exchanging messages with neighbors
  - Nodes operate concurrently
  - May be node/link/message failures



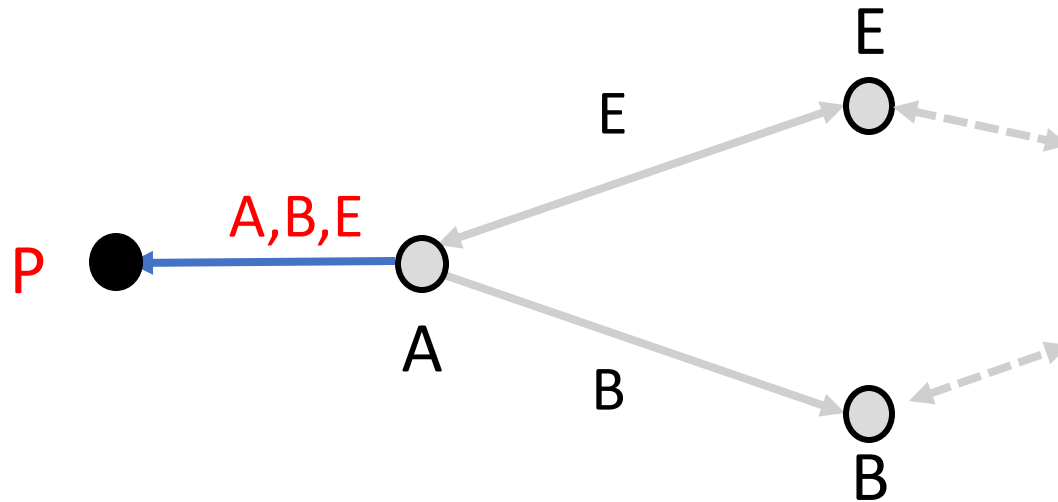
# Host/Router Combination

- Hosts attach to routers as IP prefixes (usually /32)
  - Router needs table to reach all hosts



# Network Topology for Routing

- Send out routes for hosts you have paths to
  - “Advertise” the routes
  - And the routes you’ve received



# Network Topology for Routing (2)

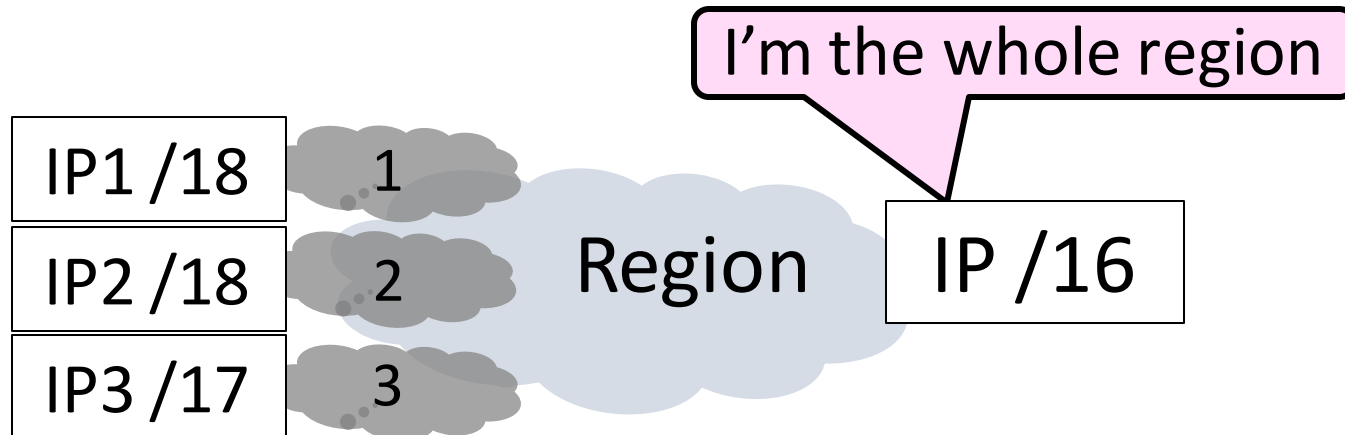
- Routing now works!
  - Routers advertise IP prefixes for hosts
  - Router addresses are “/32” prefixes
  - Lets all routers find a path to hosts
  - Hosts find by sending to their router

# IP Prefix Aggregation and Subnets



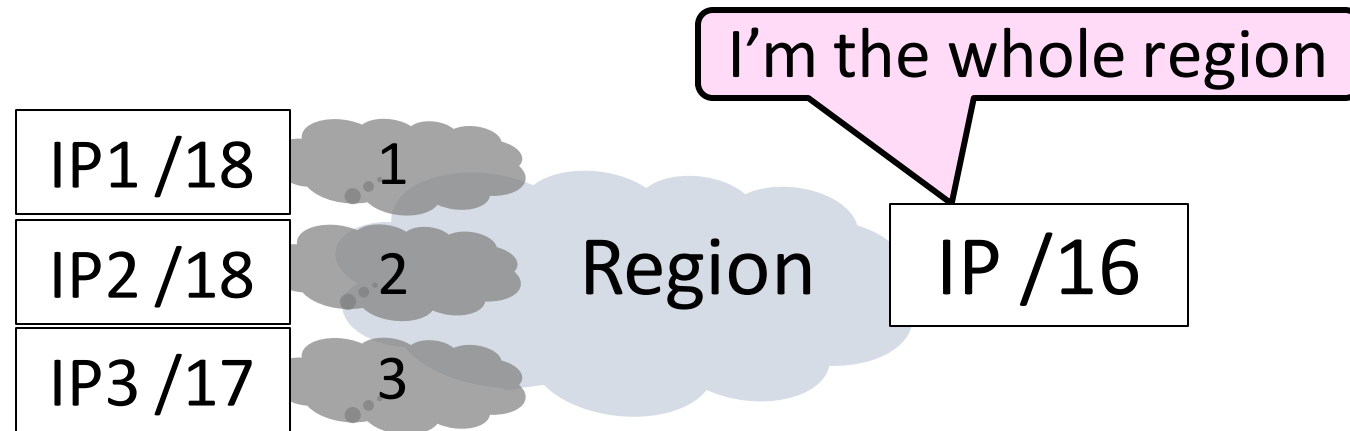
# Idea

- Scale routing by adjusting the size of IP prefixes
  - Split (subnets) and join (aggregation)



# Prefixes and Hierarchy

- IP prefixes help to scale routing, but can go further
  - Use a less specific (larger) IP prefix as a name for a region

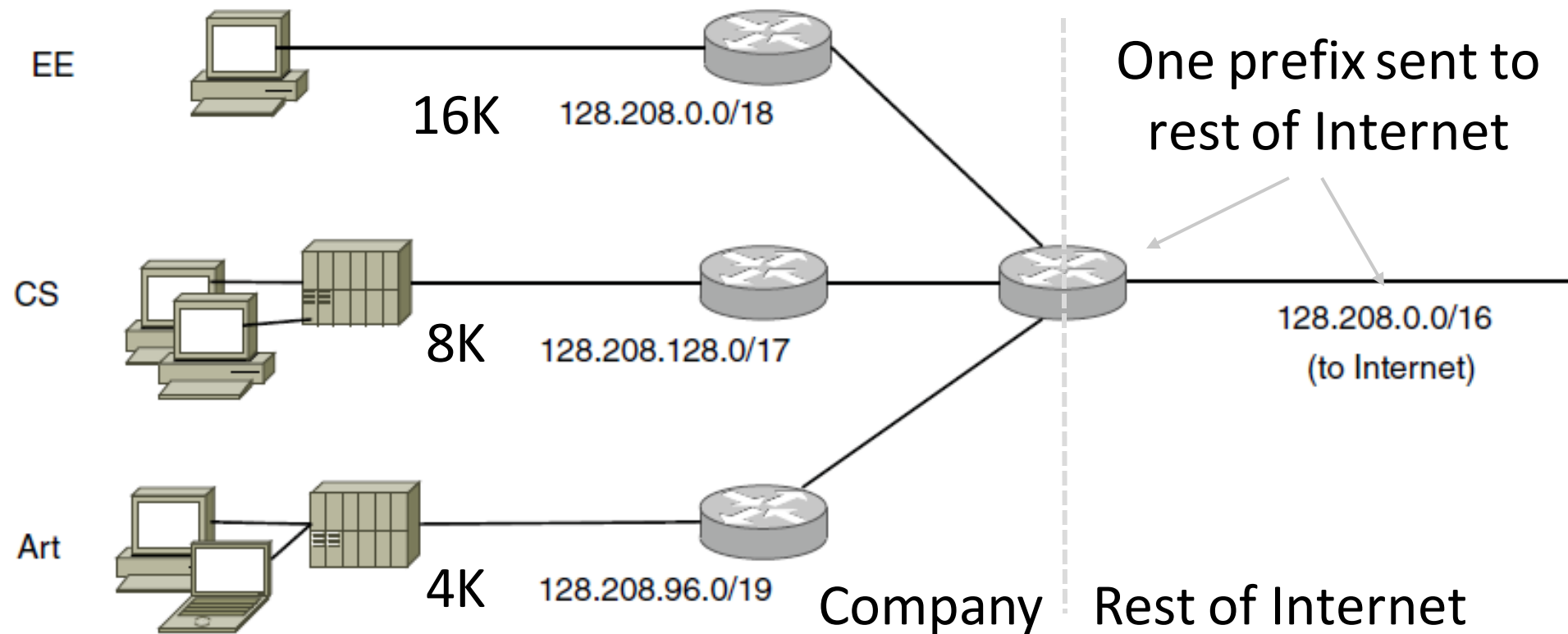


# Subnets and Aggregation

- Two use cases for adjusting the size of IP prefixes; both reduce routing table
  1. Subnets
    - Internally split one large prefix into multiple smaller ones
  2. Aggregation
    - Join multiple smaller prefixes into one large prefix

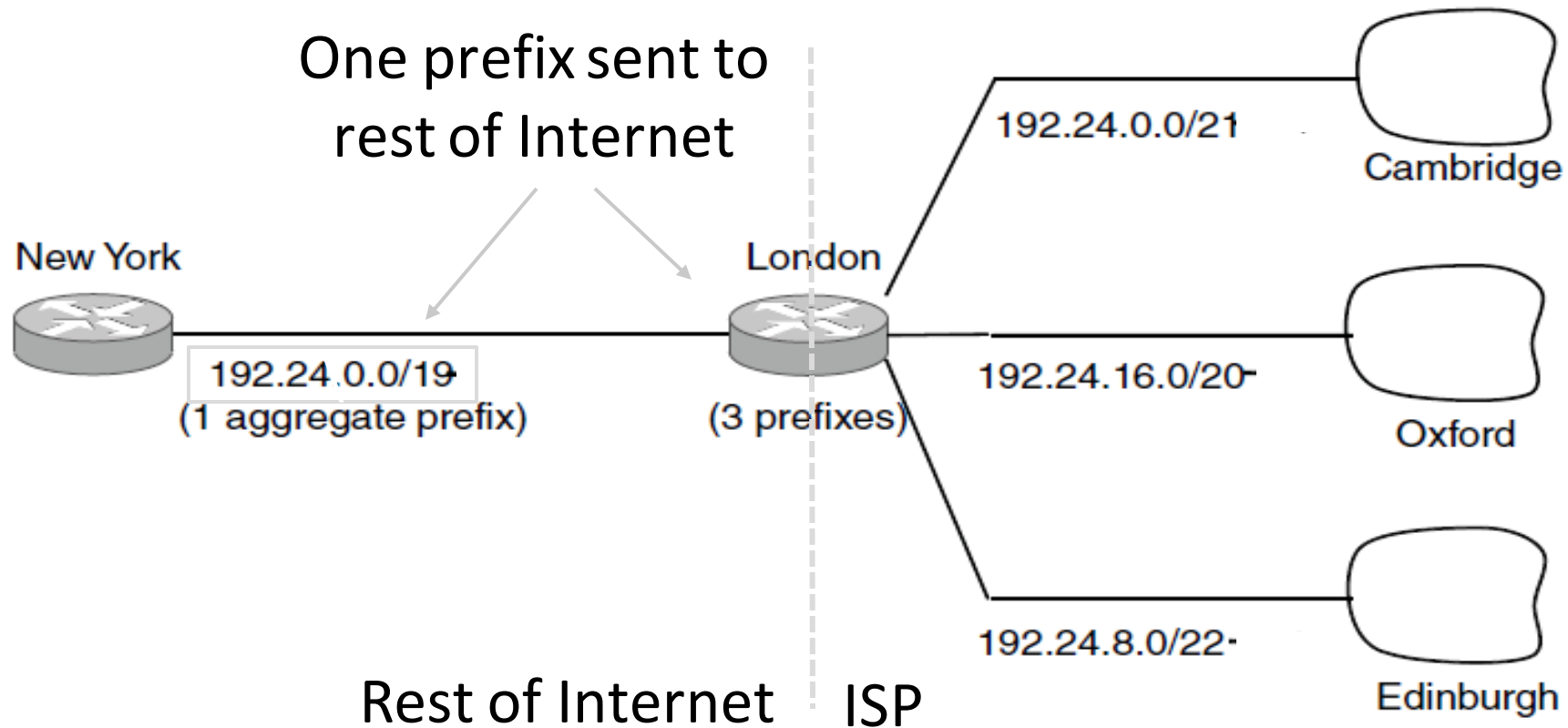
# Subnets

- Internally split up one IP prefix



# Aggregation

- Externally join multiple separate IP prefixes



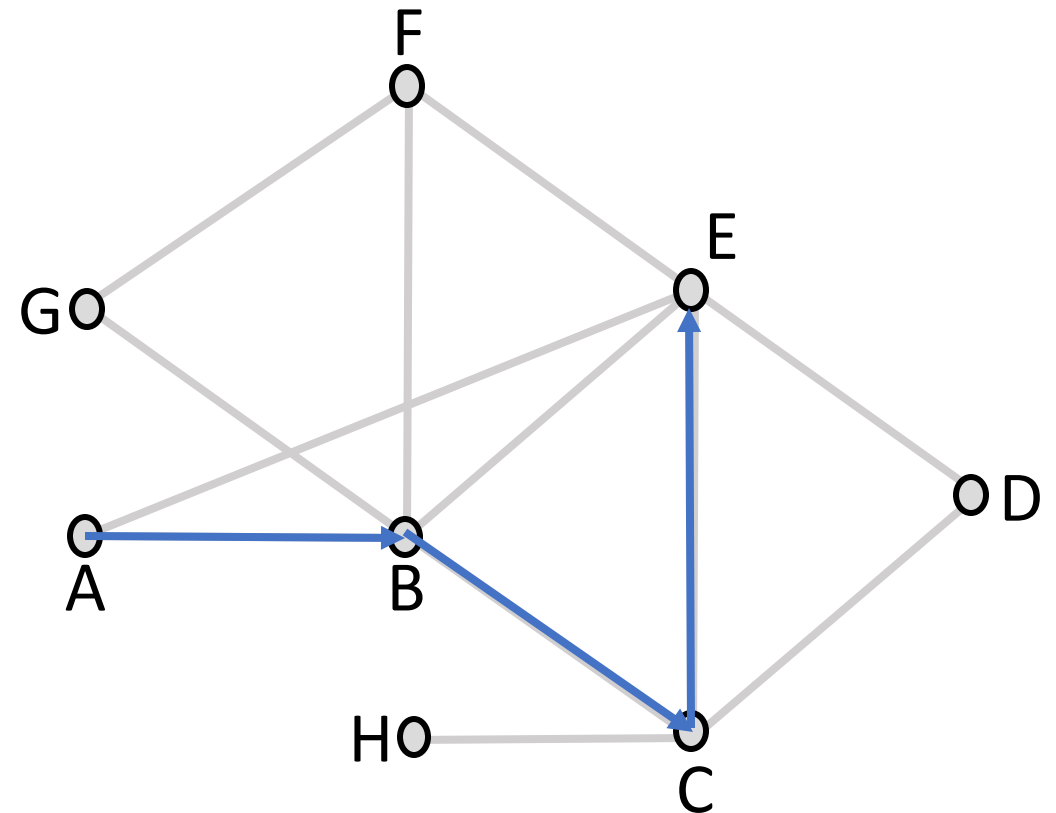
# Routing Process

1. Ship these prefixes or regions around to nearby routers
2. Receive multiple prefixes and the paths of how you got them
3. Build a global routing table

# Best Path Routing

# What are “Best” paths anyhow?

- Many possibilities:
  - Latency, avoid circuitous paths
  - Bandwidth, avoid slow links
  - Money, avoid expensive links
  - Hops, to reduce switching
- But only consider topology
  - Ignore workload, e.g., hotspots





# Shortest Paths

We'll approximate “best” by a cost function that captures the factors

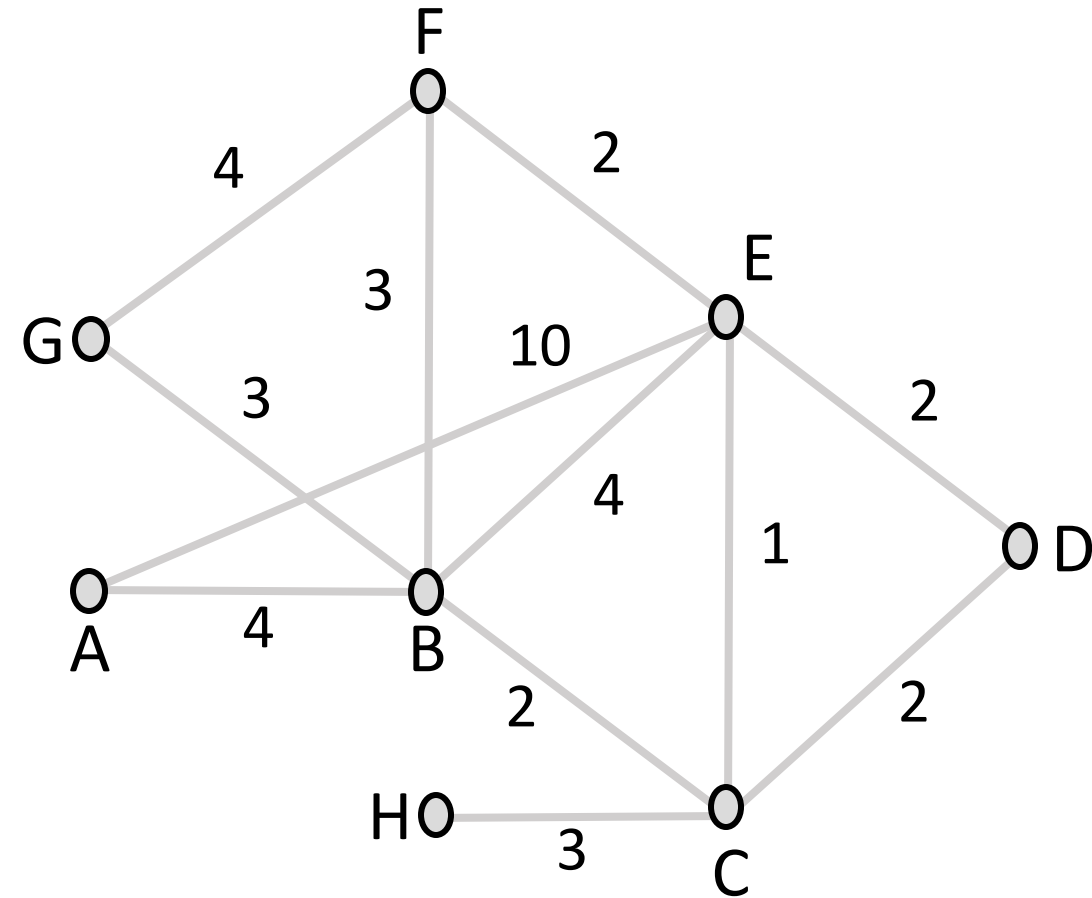
- Often call lowest “shortest”

1. Assign each link a cost (distance)
2. Define best path between each pair of nodes as the path that has the lowest total cost (or is shortest)

3. Pick randomly to any break ties

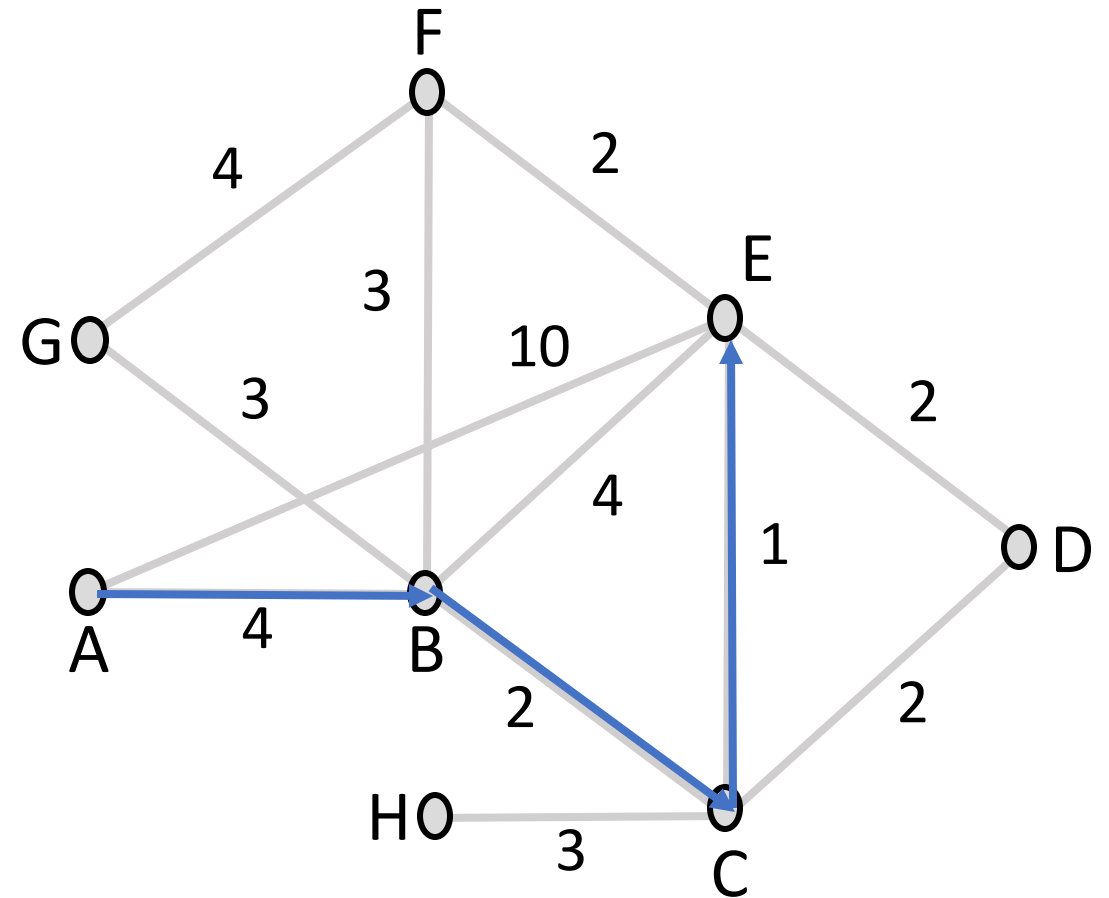
## Shortest Paths (2)

- Find the shortest path  $A \rightarrow E$
- All links are bidirectional, with equal costs in each direction
  - Can extend model to unequal costs if needed



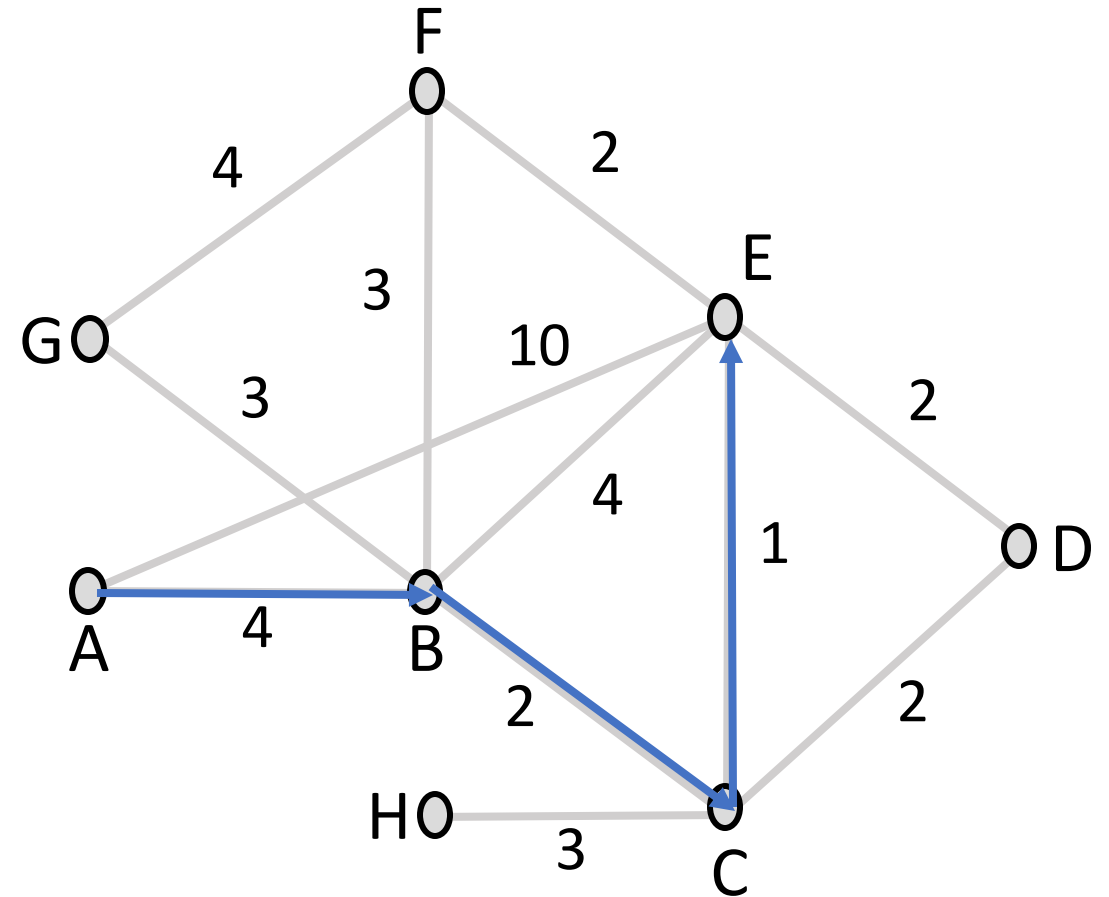
## Shortest Paths (3)

- ABCE is a shortest path
- $\text{dist}(\text{ABCE}) = 4 + 2 + 1 = 7$
- This is less than:
  - $\text{dist}(\text{ABE}) = 8$
  - $\text{dist}(\text{ABFE}) = 9$
  - $\text{dist}(\text{AE}) = 10$
  - $\text{dist}(\text{ABCDE}) = 10$



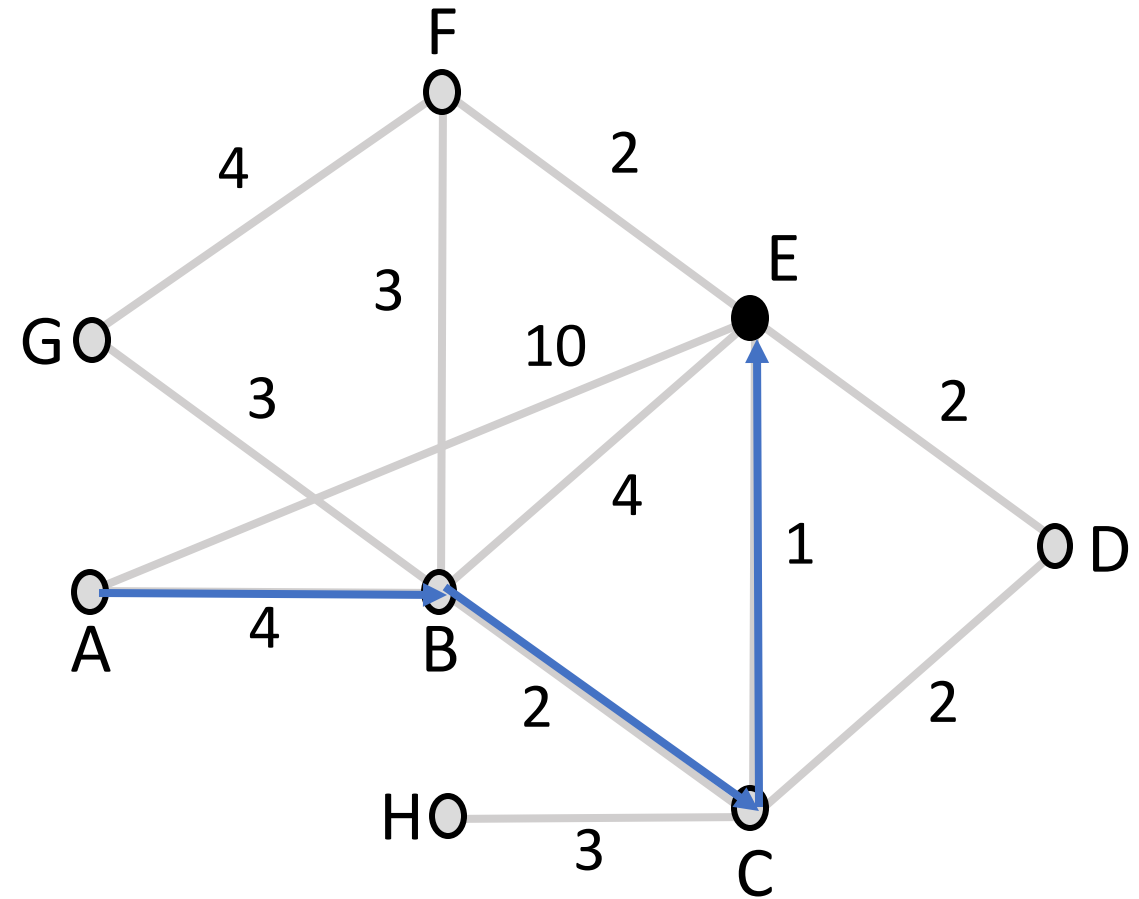
# Shortest Paths (4)

- Optimality property:
  - Subpaths of shortest paths are also shortest paths
- ABCE is a shortest path
  - ⑦ So are ABC, AB, BCE, BC, CE



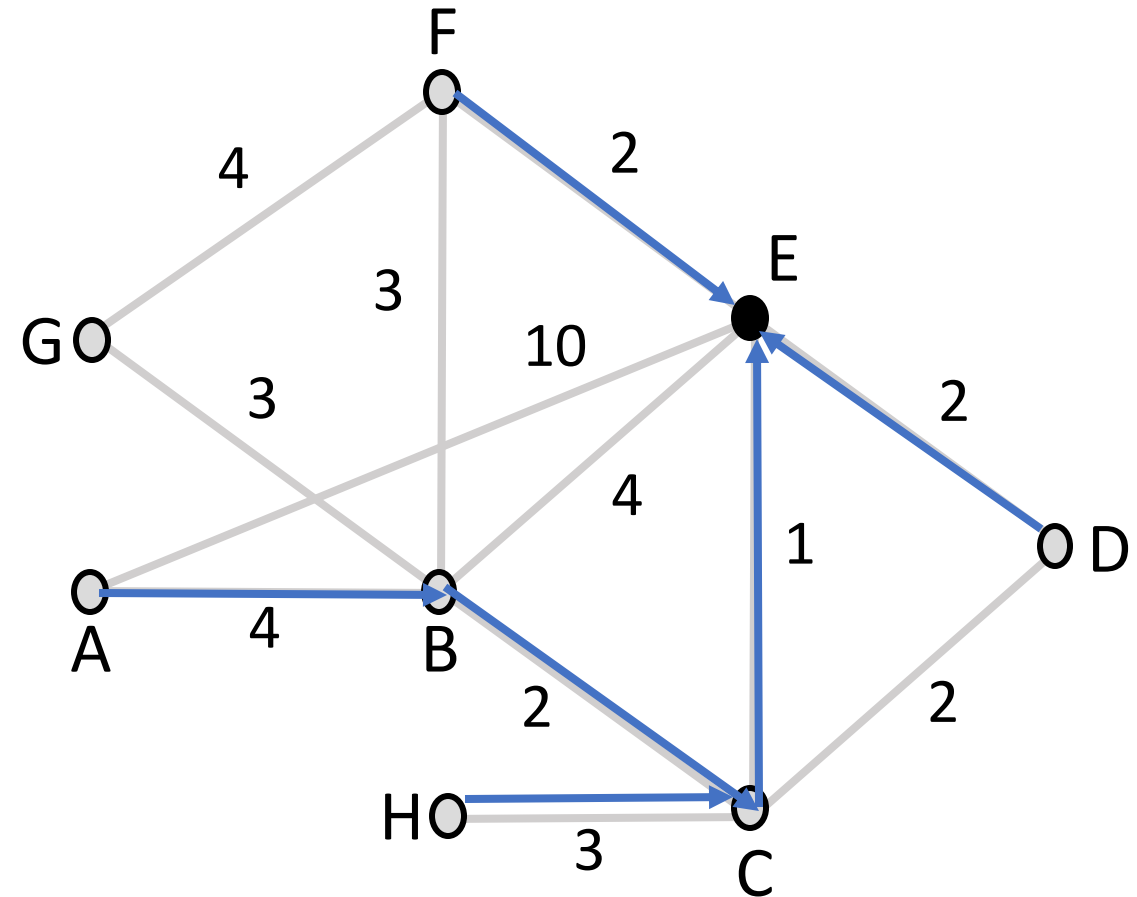
# Sink Trees

- 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 (2)

- 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



# Link-State Routing

# Link-State Routing

- Broad class of routing algorithms
  - Other is distance vector which is used when computation is harder
- Widely used in practice
  - Used in Internet/ARPANET from 1979
  - Modern networks use OSPF (L3) and IS-IS (L2)



# Link-State Setting

Nodes compute their forwarding table in the classic distributed setting:

1. Nodes know only the cost to their neighbors; not topology
2. Nodes can talk only to their neighbors using messages
3. All nodes run the same algorithm concurrently
4. Nodes/links may fail, messages may be lost

# Link-State Algorithm

Proceeds in two phases:

1. Nodes flood topology with link state packets
  - Each node learns full topology
2. Each node computes its own forwarding table
  - By running Dijkstra (or equivalent)

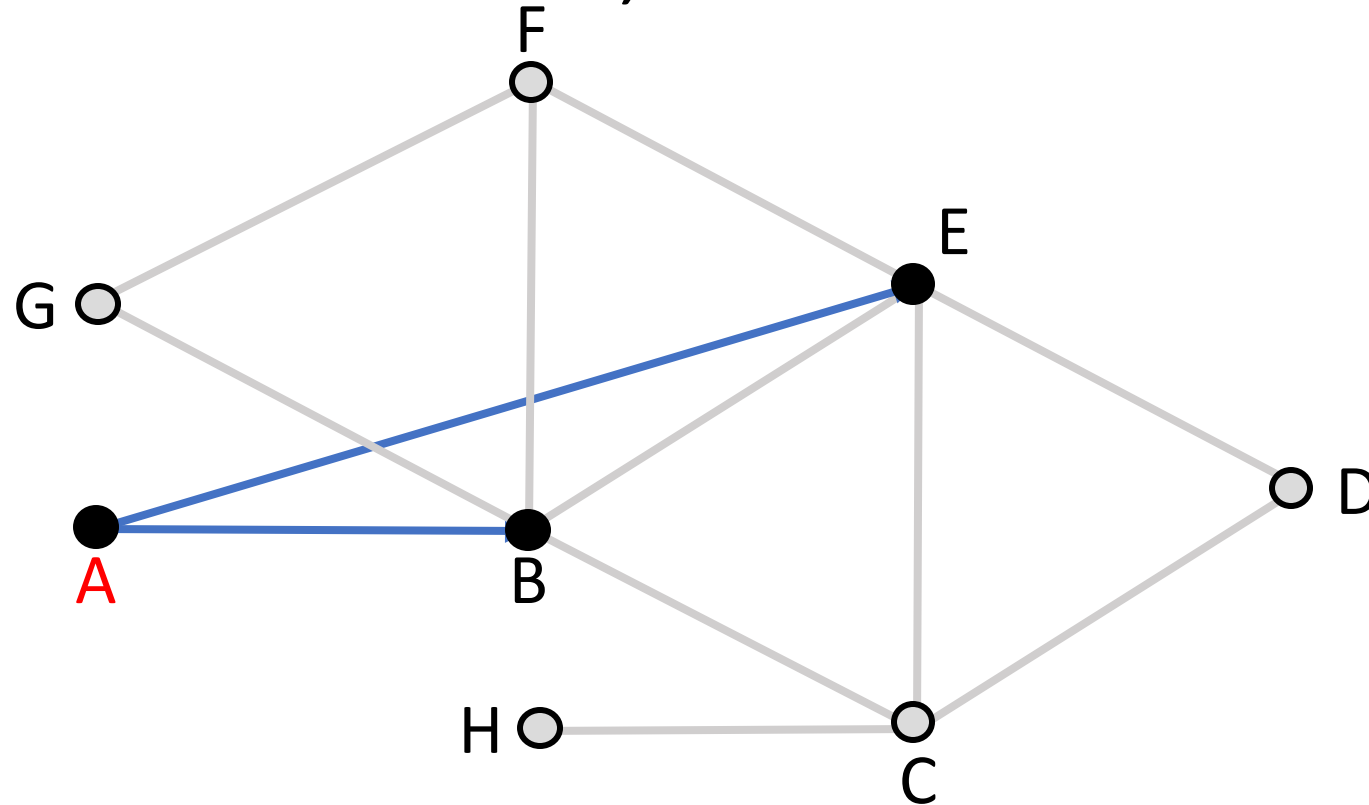
# Part 1: Flood Routing

# Flooding

- Rule used at each node:
  - Sends an incoming message on to all other neighbors
  - Remember the message so that it is only flooded once

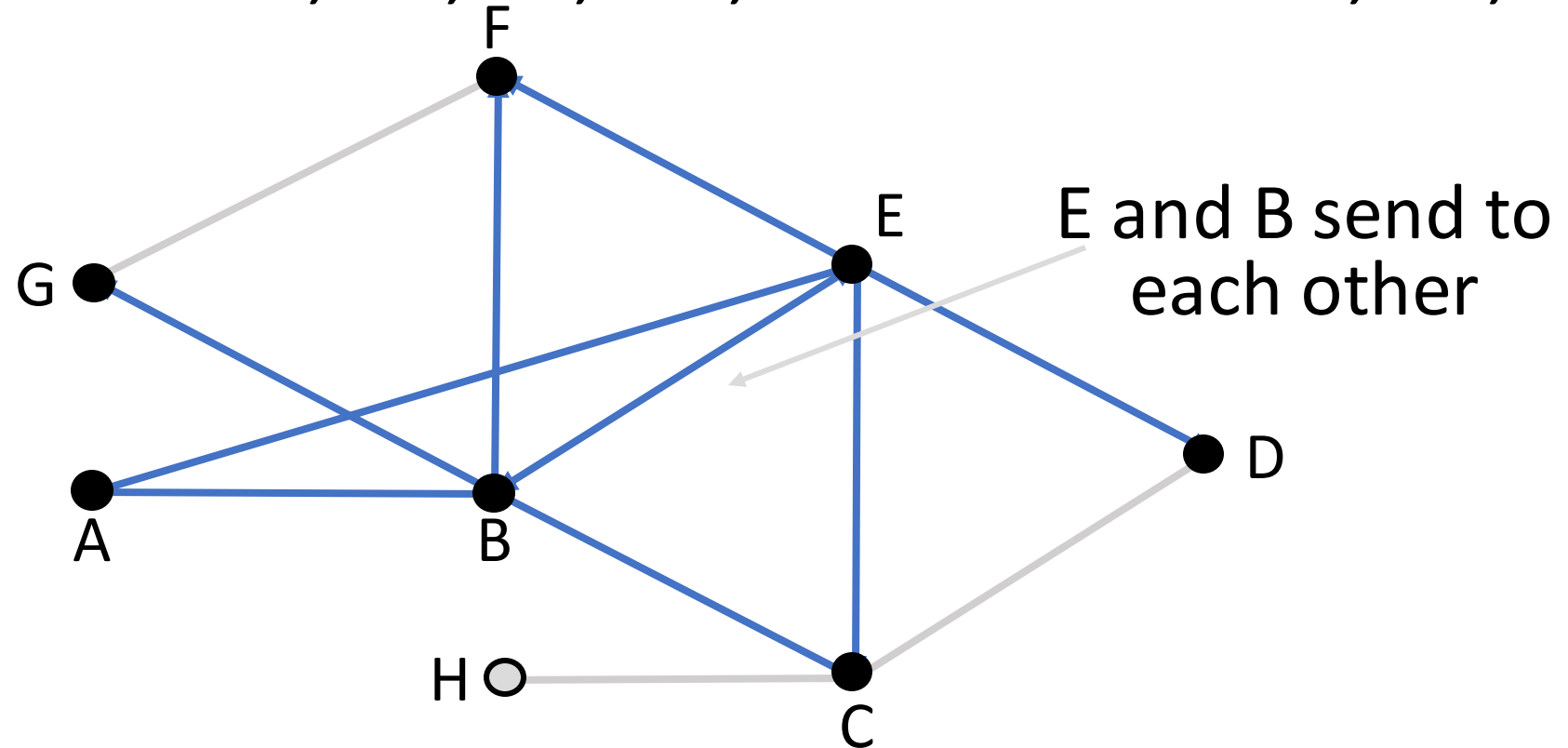
# Flooding (2)

- Consider a flood from A; first reaches B via AB, E via AE



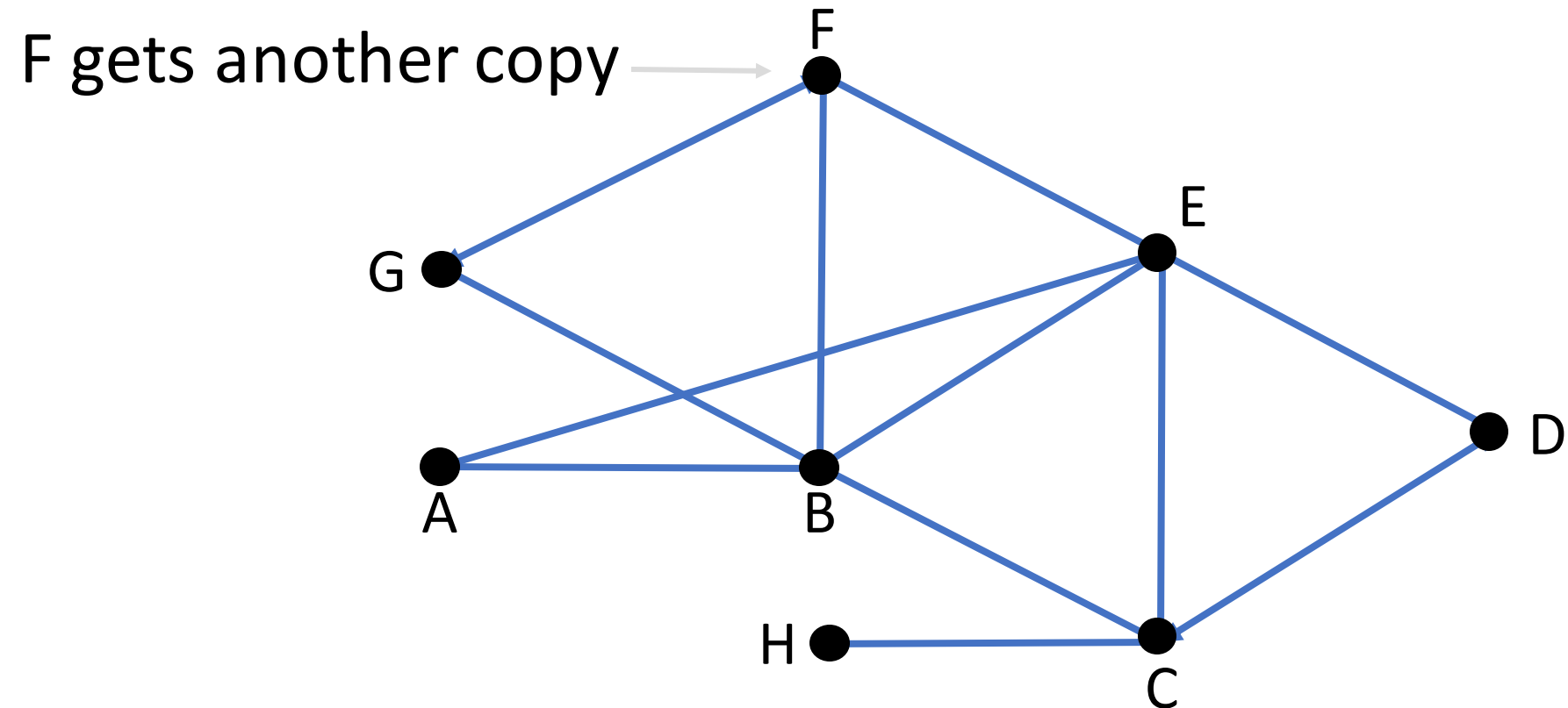
# Flooding (3)

- Next B floods BC, BE, BF, BG, and E floods EB, EC, ED, EF



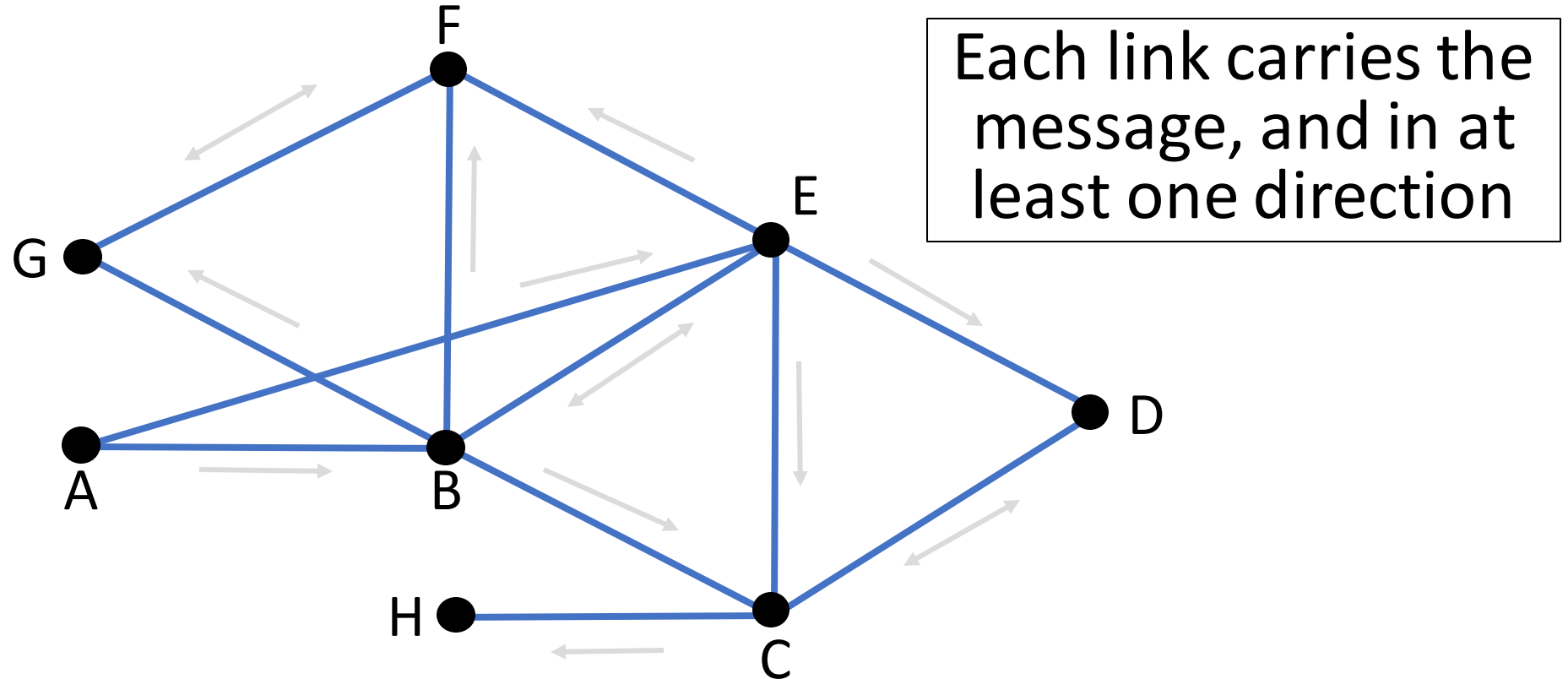
# Flooding (4)

- C floods CD, CH; D floods DC; F floods FG; G floods GF



# Flooding (5)

- H has no-one to flood ... and we're done





# Flooding Details

- Remember message (to stop flood) using source and sequence number
  - So next message (with higher sequence) will go through
- To make flooding reliable, use ARQ
  - So receiver acknowledges, and sender resends if needed

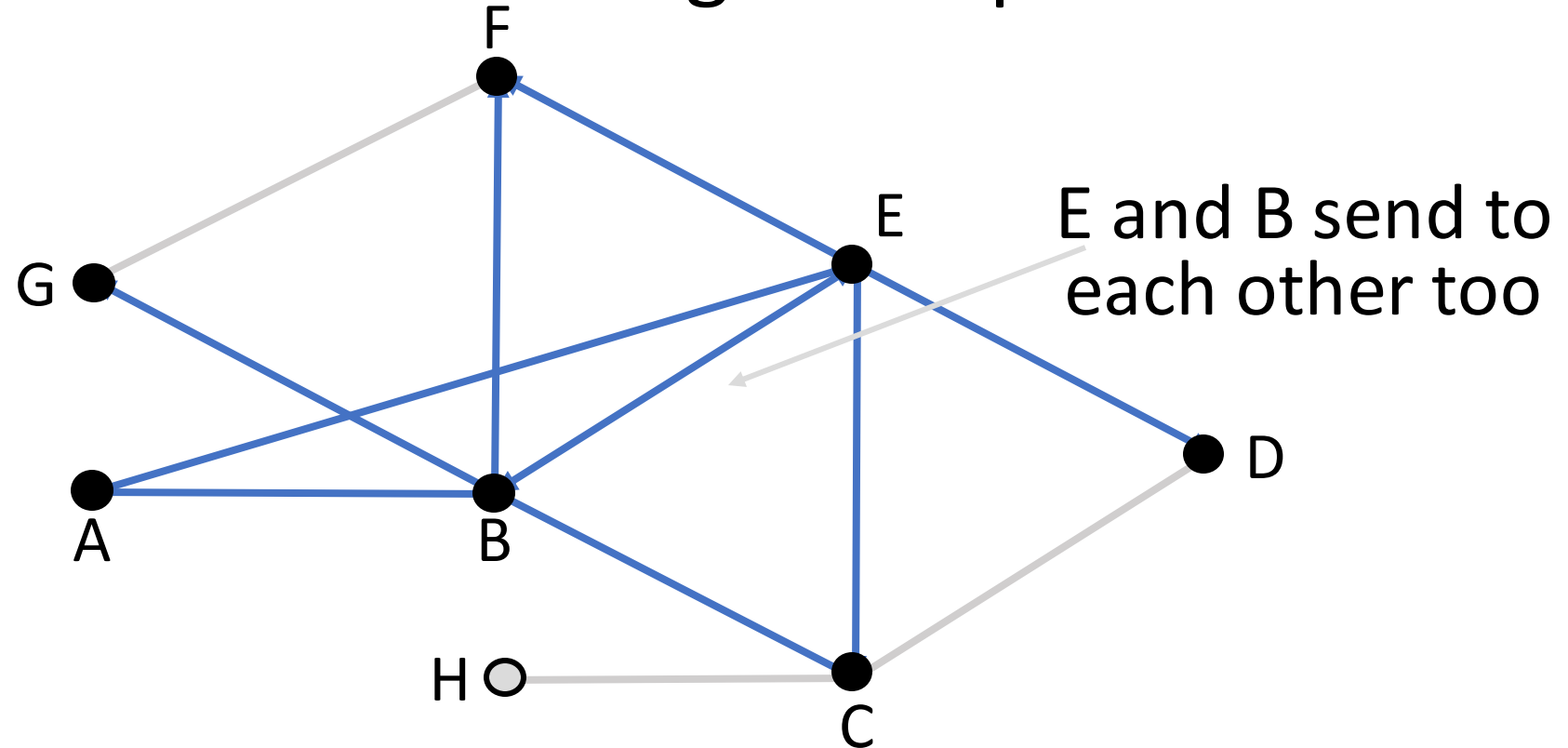
# Flooding Details

- Remember message (to stop flood) using source and sequence number
  - So next message (with higher sequence) will go through
- To make flooding reliable, use ARQ
  - So receiver acknowledges, and sender resends if needed

Problem?

# Flooding Problem

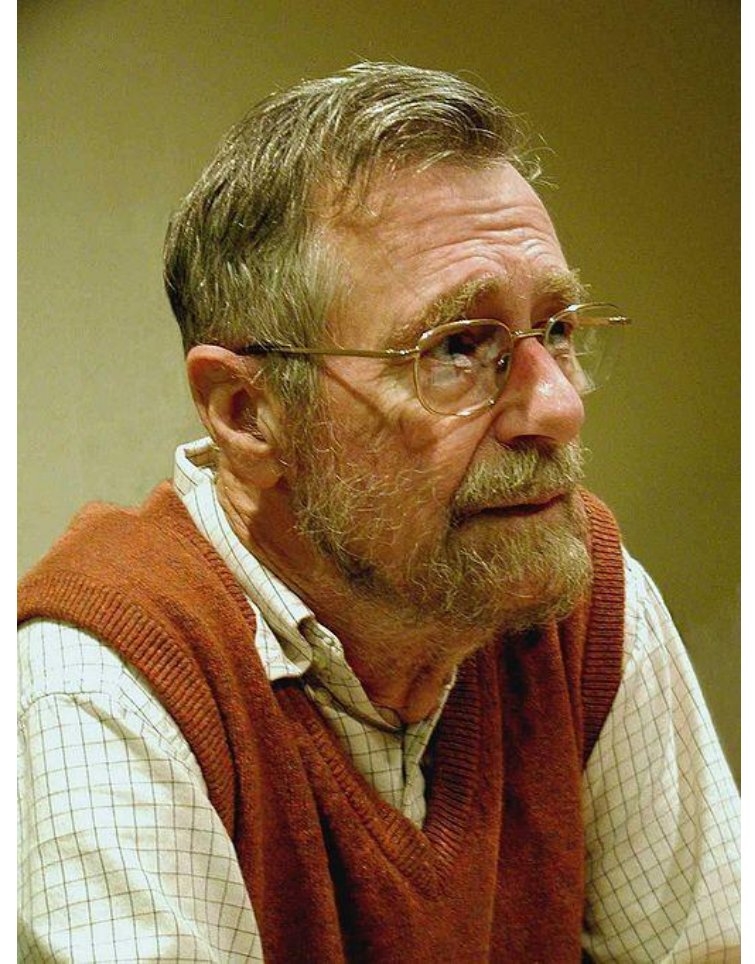
- F receives the same message multiple times



# Part 2: Dijkstra's Algorithm

# Edsger W. Dijkstra (1930-2002)

- Famous computer scientist
  - Programming languages
  - Distributed algorithms
  - Program verification
- Dijkstra's algorithm, 1969
  - Single-source shortest paths, given network with non-negative link costs



By Hamilton Richards, CC-BY-SA-3.0, via Wikimedia Commons

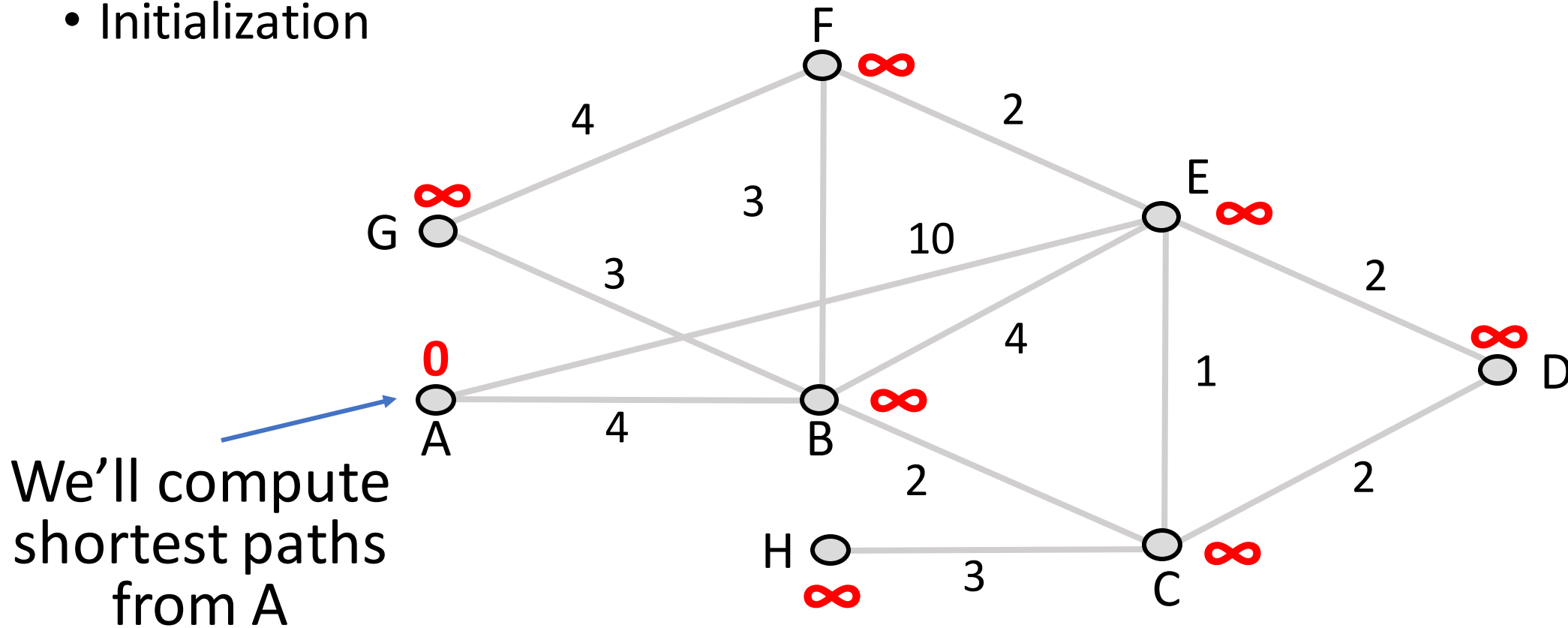
# Dijkstra's Algorithm

## Algorithm:

- Mark all nodes tentative, set distances from source to 0 (zero) for source, and  $\infty$  (infinity) for all other nodes
- While tentative nodes remain:
  - Extract N, a node with lowest distance
  - Add link to N to the shortest path tree
  - Relax the distances of neighbors of N by lowering any better distance estimates

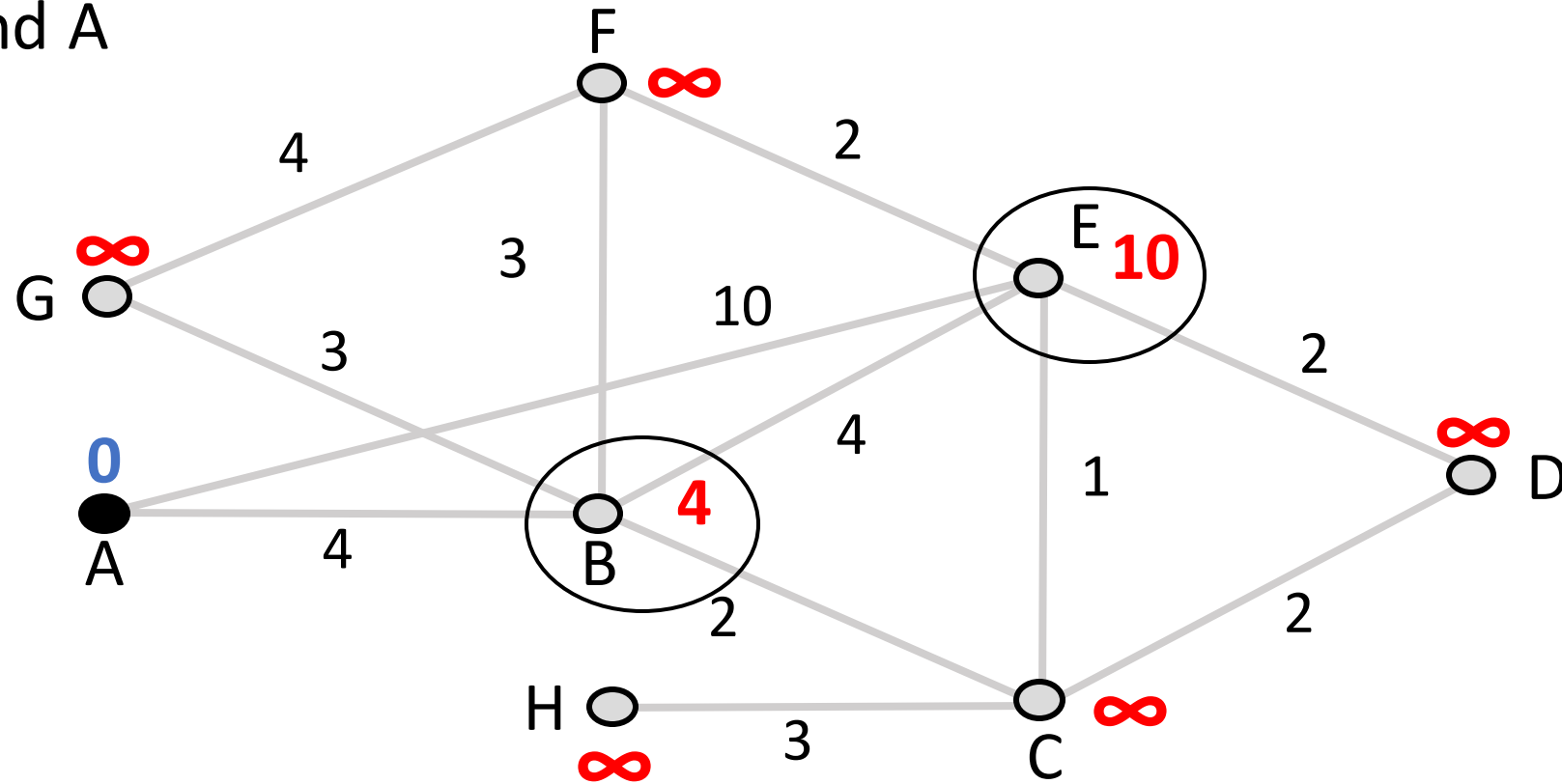
# Dijkstra's Algorithm (2)

- Initialization



# Dijkstra's Algorithm (3)

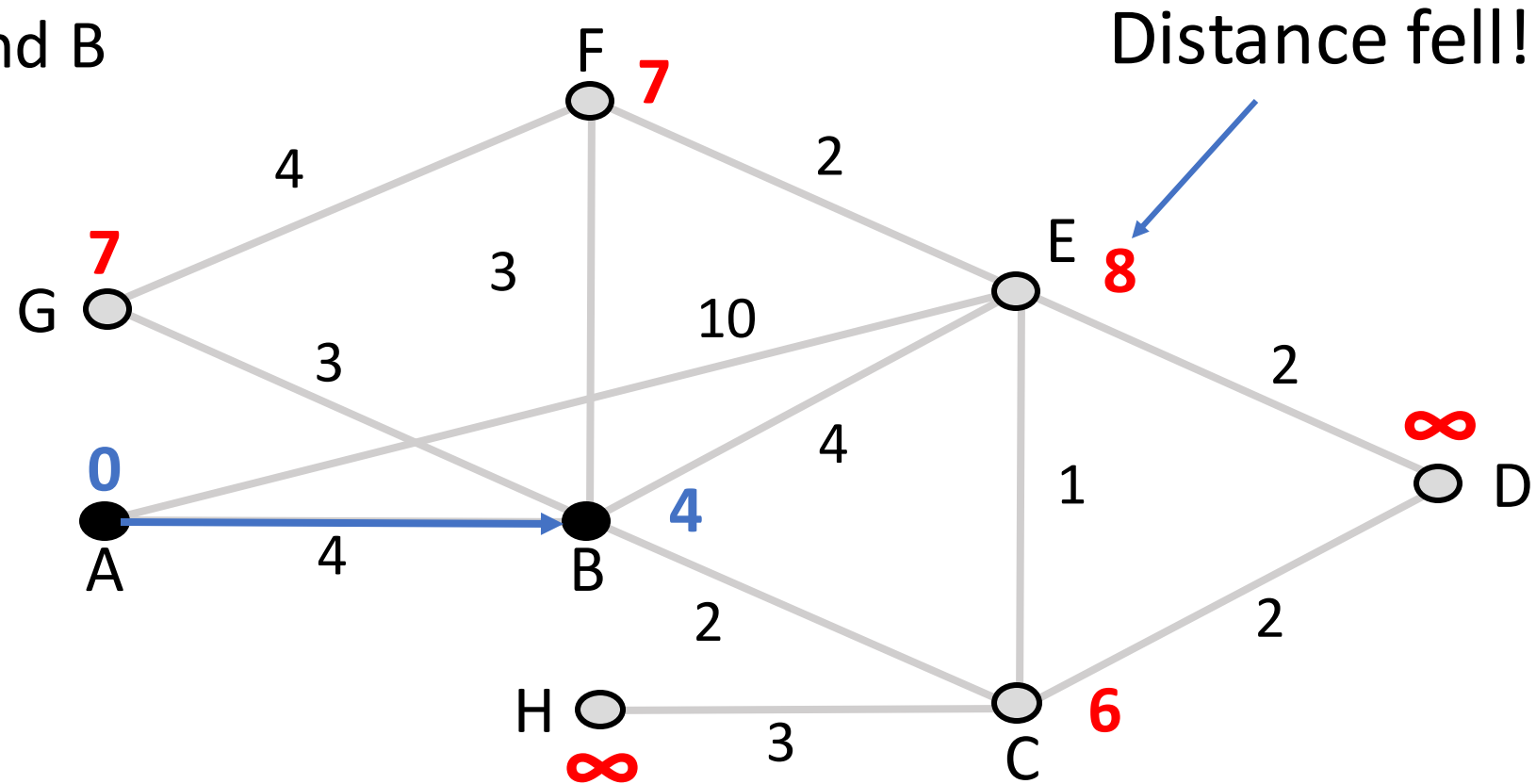
- Relax around A





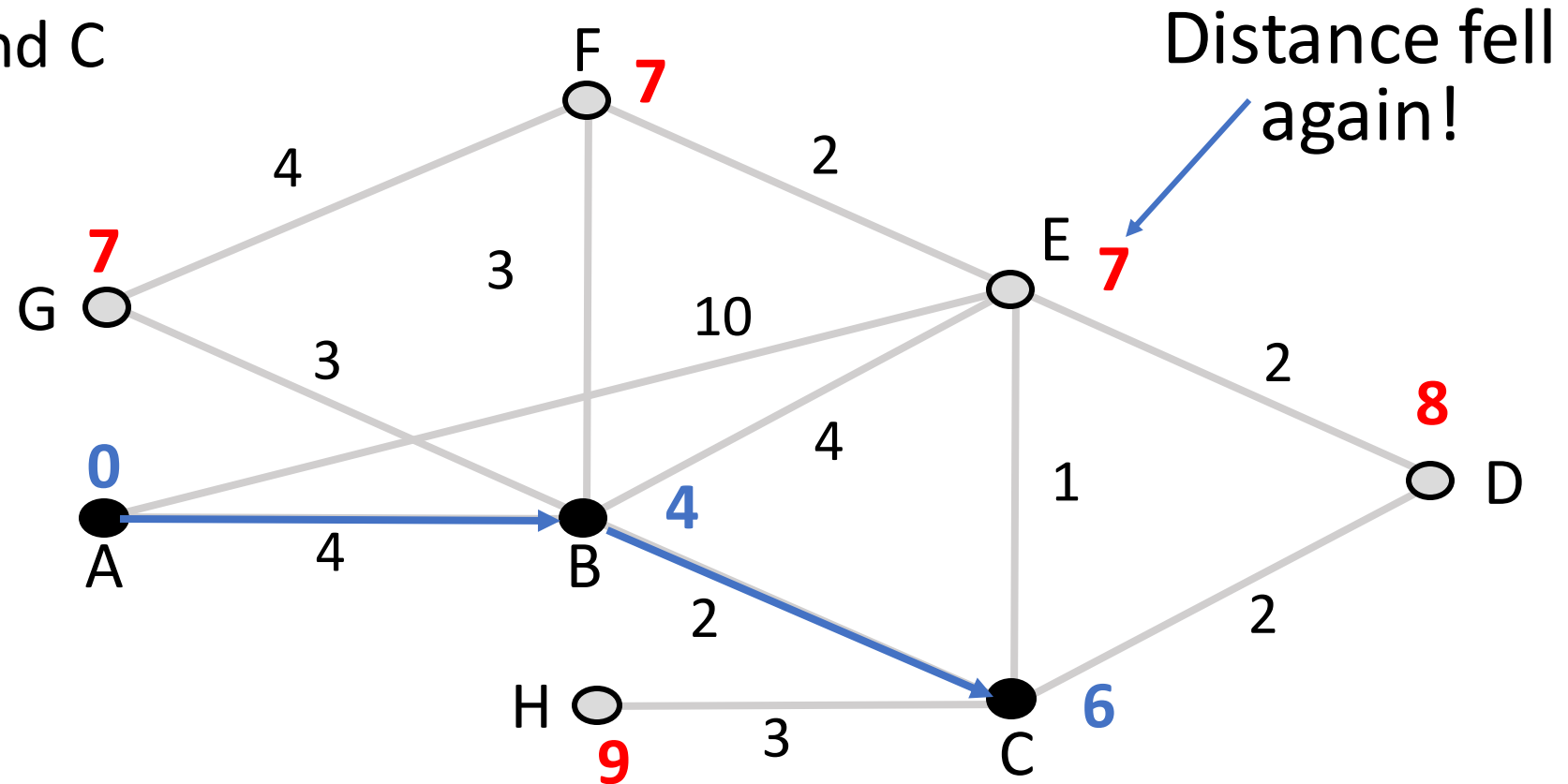
# Dijkstra's Algorithm (4)

- Relax around B



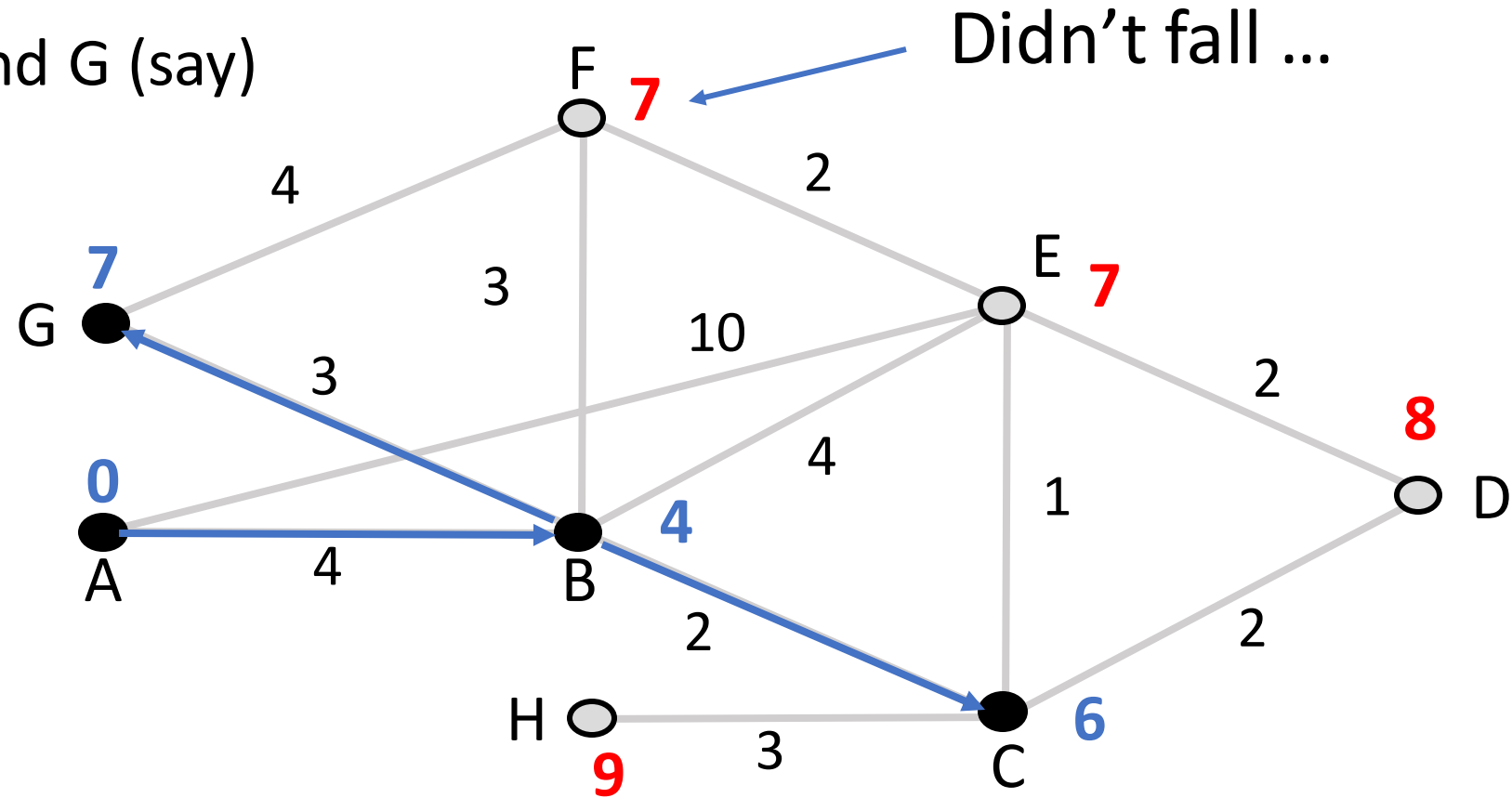
# Dijkstra's Algorithm (5)

- Relax around C



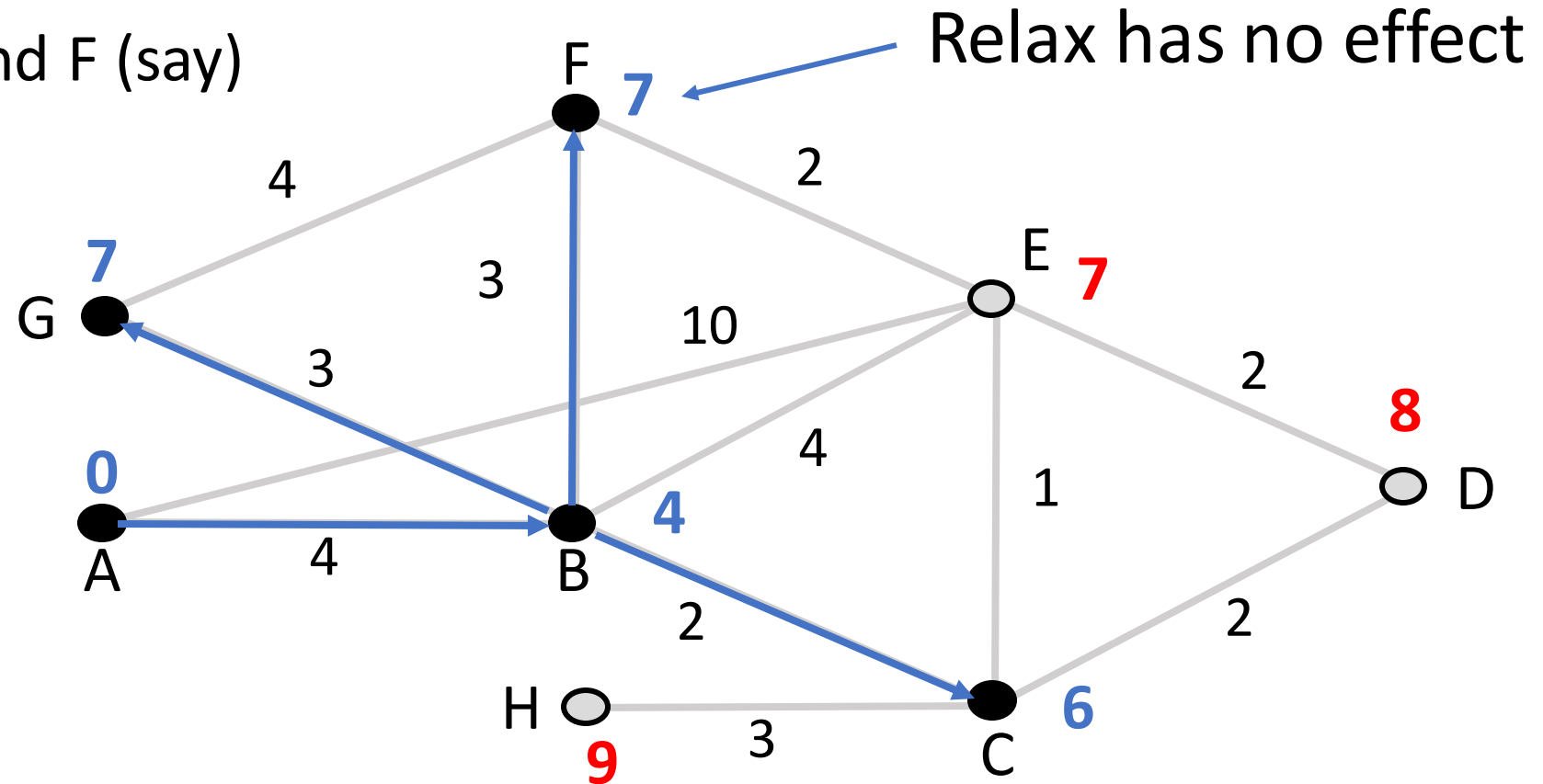
# Dijkstra's Algorithm (6)

- Relax around G (say)



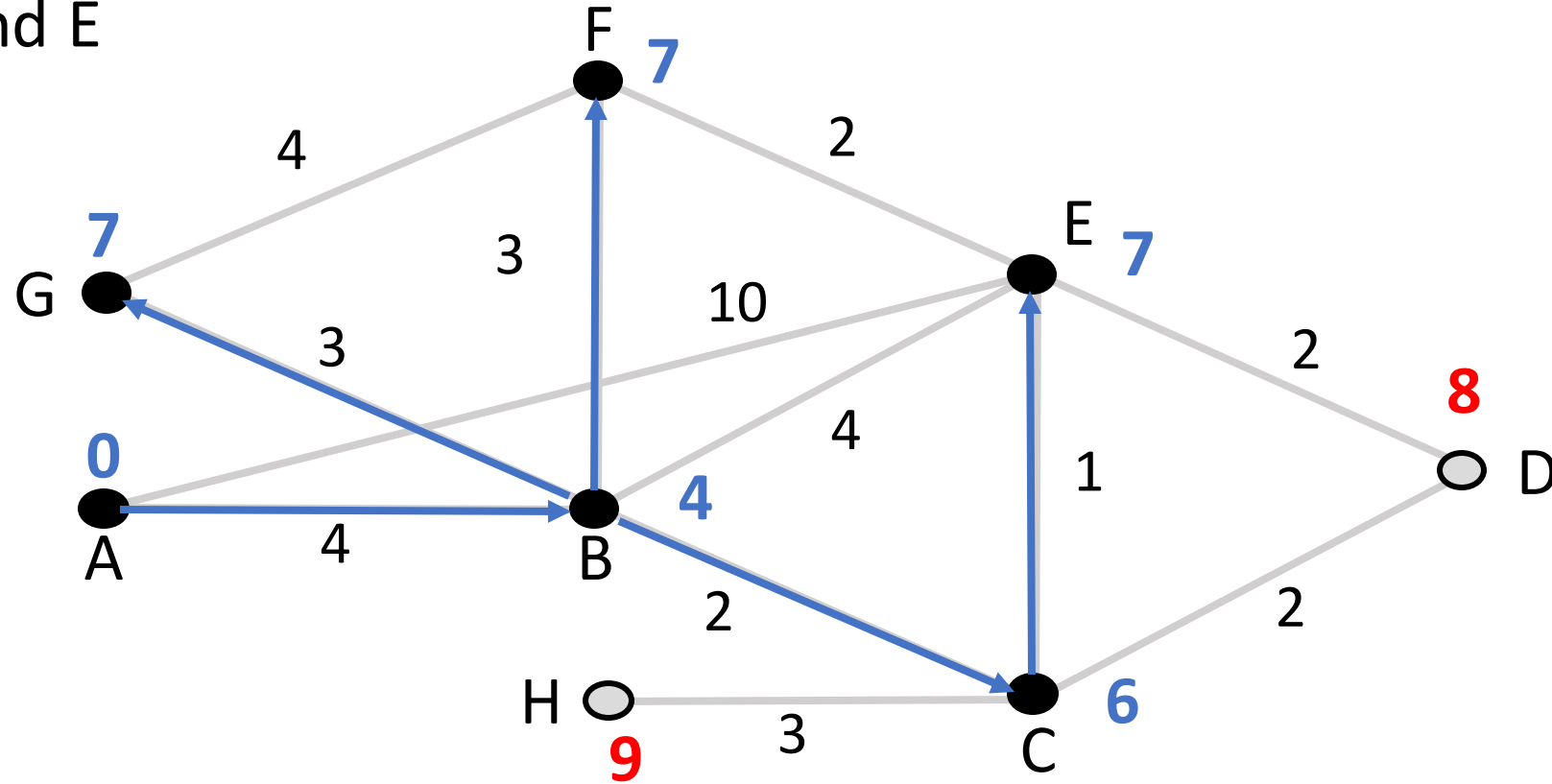
# Dijkstra's Algorithm (7)

- Relax around F (say)



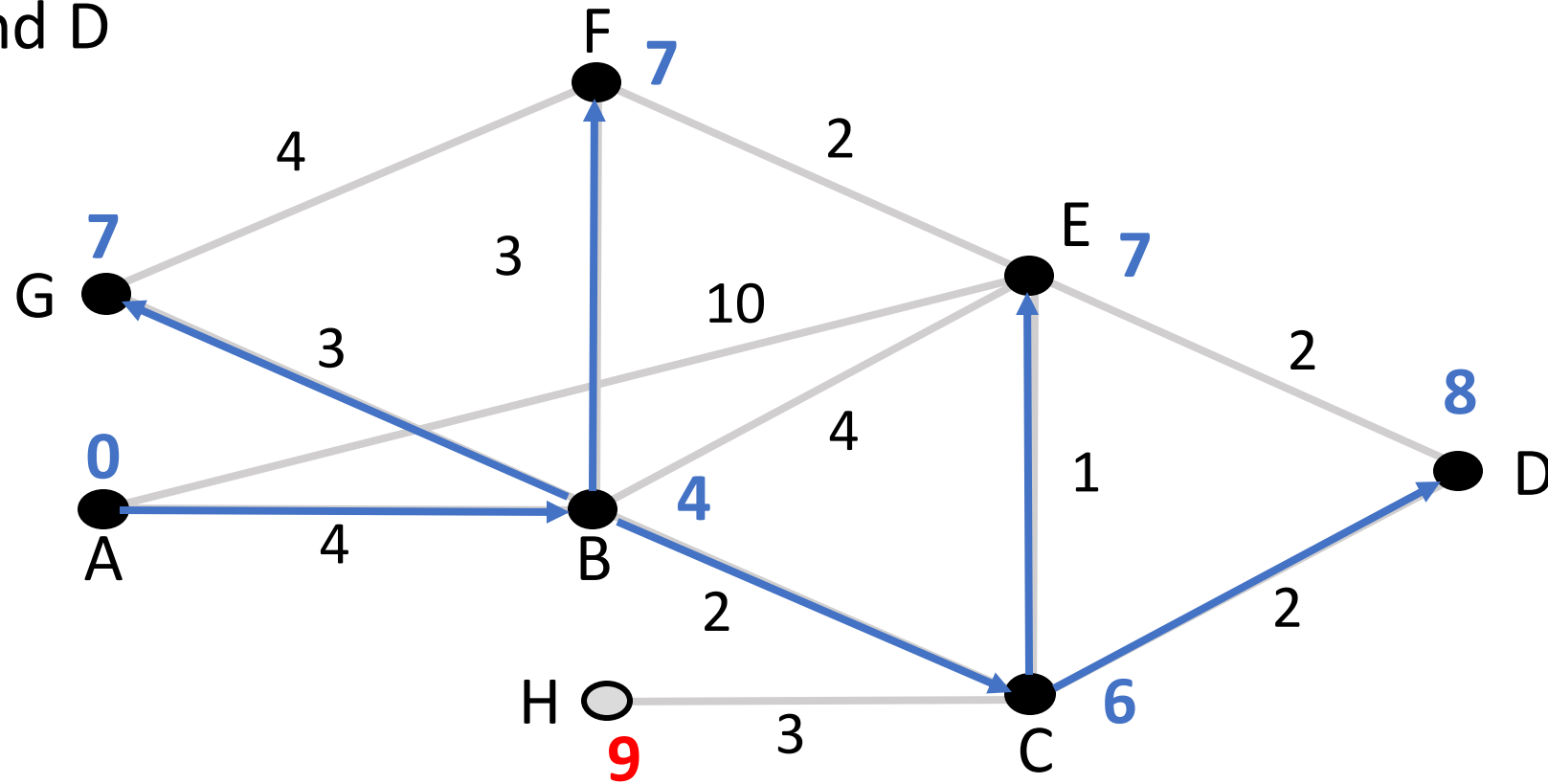
# Dijkstra's Algorithm (8)

- Relax around E



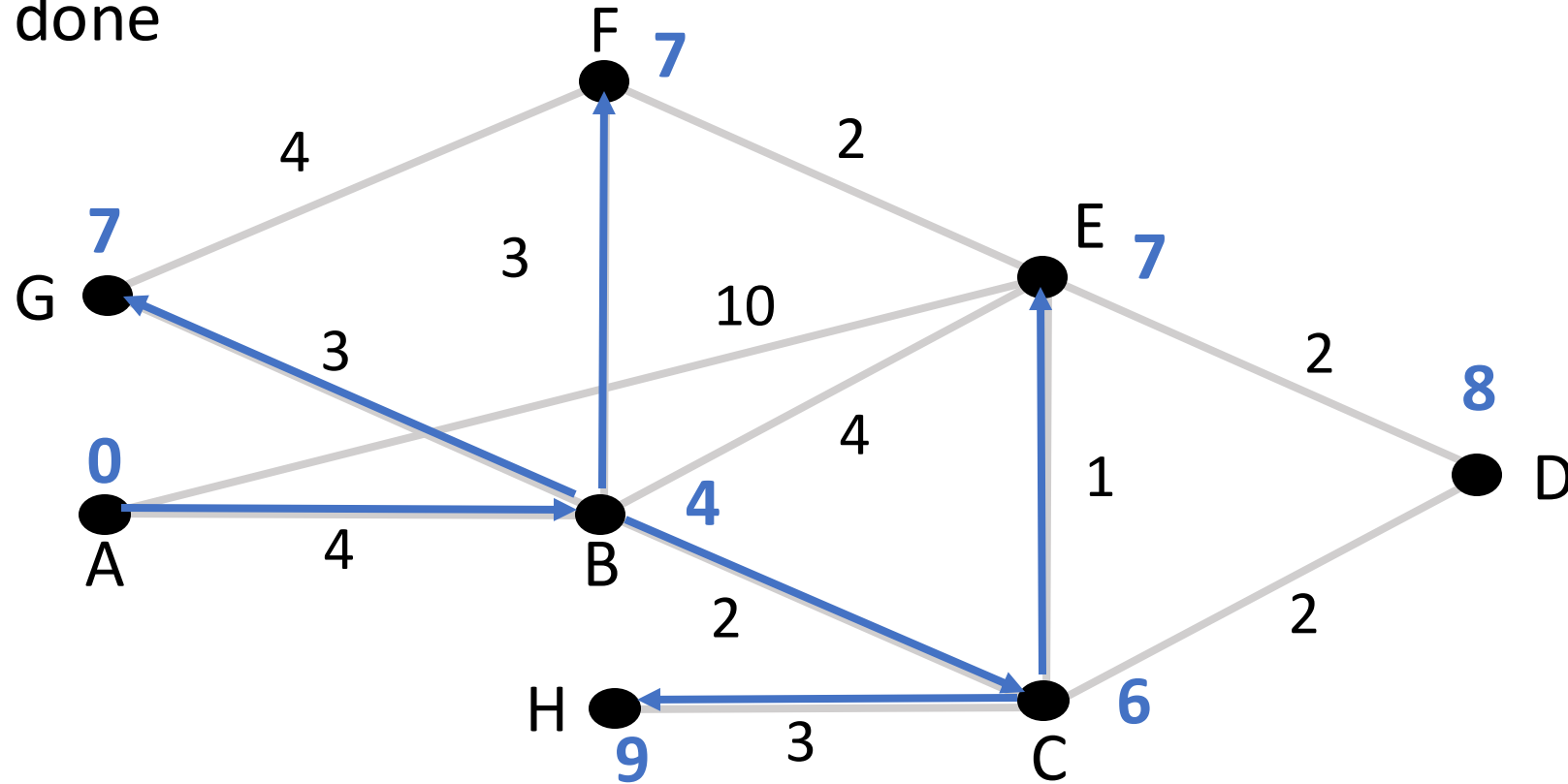
# Dijkstra's Algorithm (9)

- Relax around D



# Dijkstra's Algorithm (10)

- Finally, H ... done



# Dijkstra Comments

- Finds shortest paths in order of increasing distance from source
  - Leverages optimality property
- Runtime depends on cost of extracting min-cost node
  - Superlinear in network size (grows fast)
  - Using Fibonacci Heaps the complexity turns out to be  $O(|E| + |V| \log |V|)$
- Gives complete source/sink tree
  - More than needed for forwarding



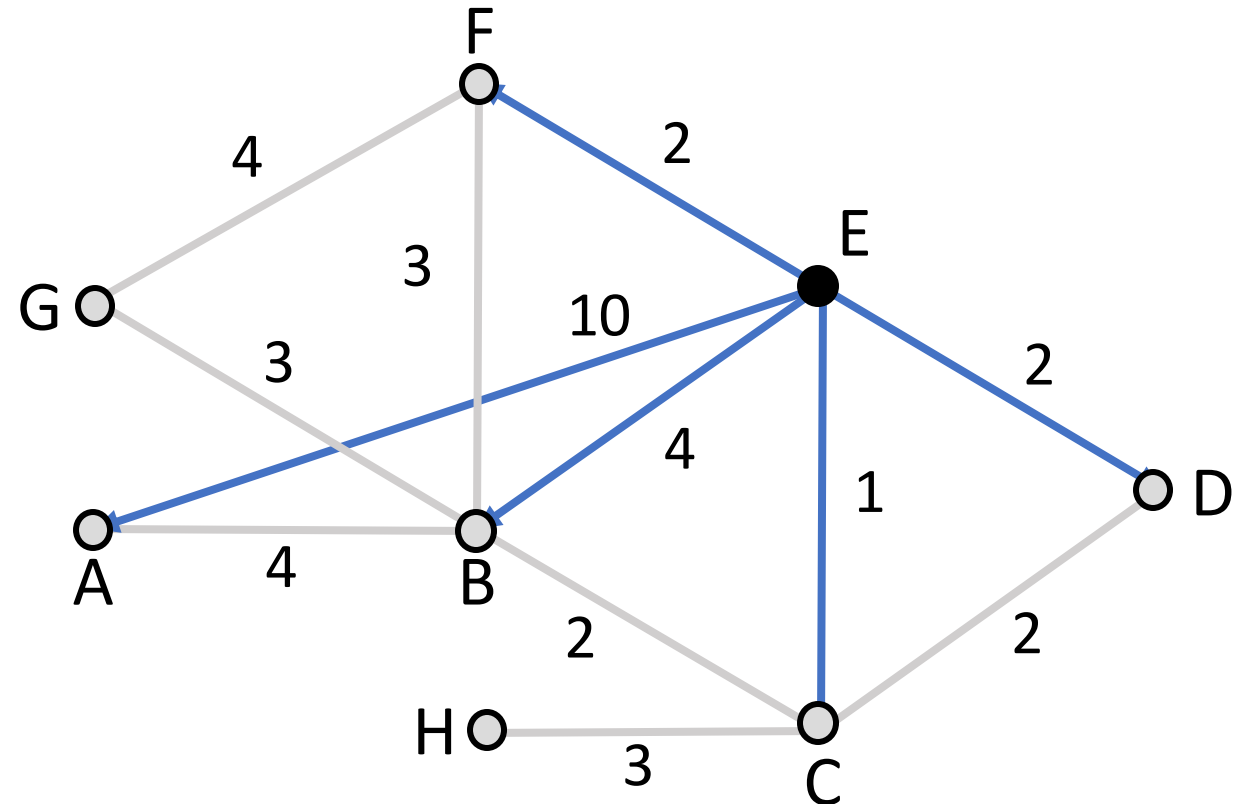
Bringing it all together...

# Phase 1: Topology Dissemination

- Each node floods link state packet (LSP) that describes their portion of the topology

Node E's LSP  
flooded to A, B,  
C, D, and F

Seq. #	
A	10
B	4
C	1
D	2
F	2

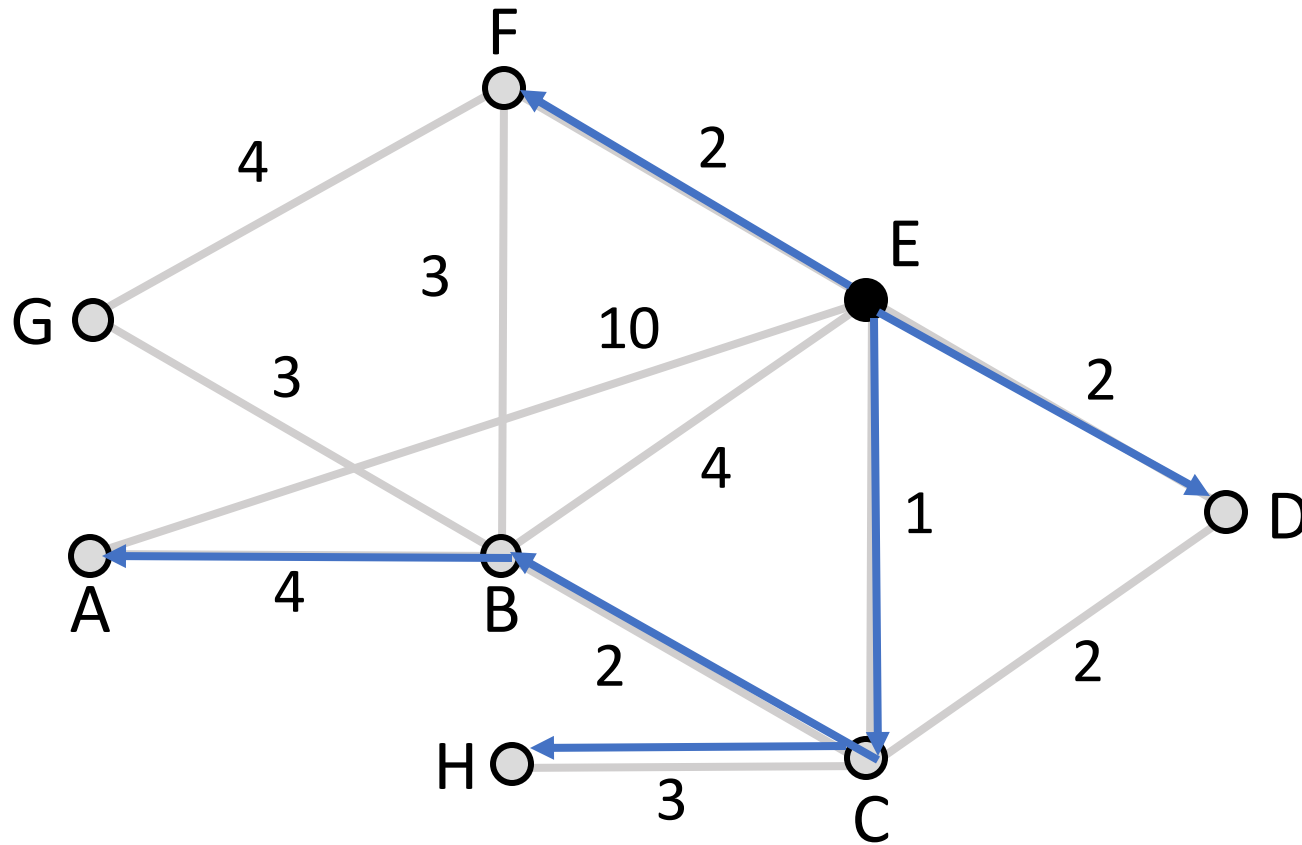


# Phase 2: Route Computation

- Each node has full topology
  - By combining all LSPs
- Each node simply runs Dijkstra
  - Replicated computation, but finds required routes directly
  - Compile forwarding table from sink/source tree
  - That's it folks!

# Forwarding Table

Source Tree for E (from Dijkstra)



E's Forwarding Table

To	Next
A	C
B	C
C	C
D	D
E	-
F	F
G	F
H	C



# Handling Changes (2)

- Link failure
  - Both nodes notice, send updated LSPs
  - Link is removed from topology
- Node failure
  - All neighbors notice a link has failed (link state!)
  - Failed node can't update its own LSP
  - But it is OK: all links to node removed

# Handling Changes (3)

- Addition of a link or node
  - Add LSP of new node to topology
  - Old LSPs are updated with new link
- Additions are the easy case ...

# Link-State Complications

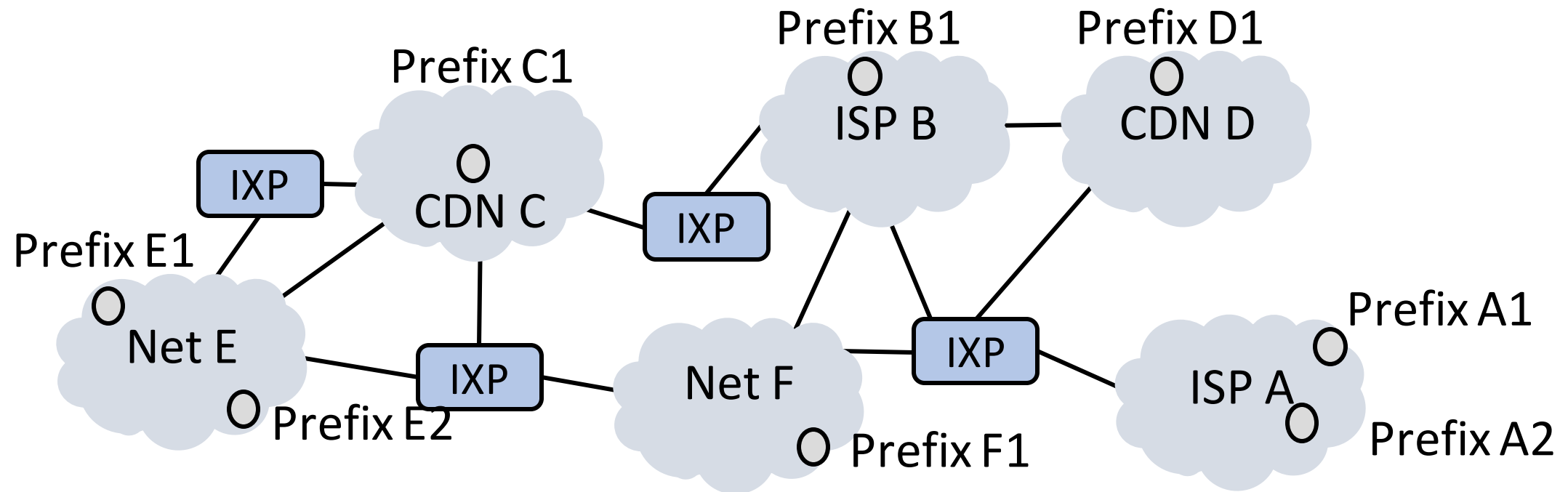
- Things that can go wrong:
  - Seq. number reaches max, or is corrupted
  - Node crashes and loses seq. number
  - Network partitions then heals
- Strategy:
  - Include age on LSPs and forget old information that is not refreshed
- Much of the complexity is due to handling corner cases



# Border Gateway Protocol (BGP)

# Structure of the Internet

- Networks (ISPs, CDNs, etc.) group with IP prefixes
- Networks are richly interconnected, often using IXPs



# Internet-wide Routing Issues

- Two problems beyond routing within a network

1. Scaling to very large networks

- Techniques of IP prefixes, hierarchy, prefix aggregation

2. Incorporating policy decisions

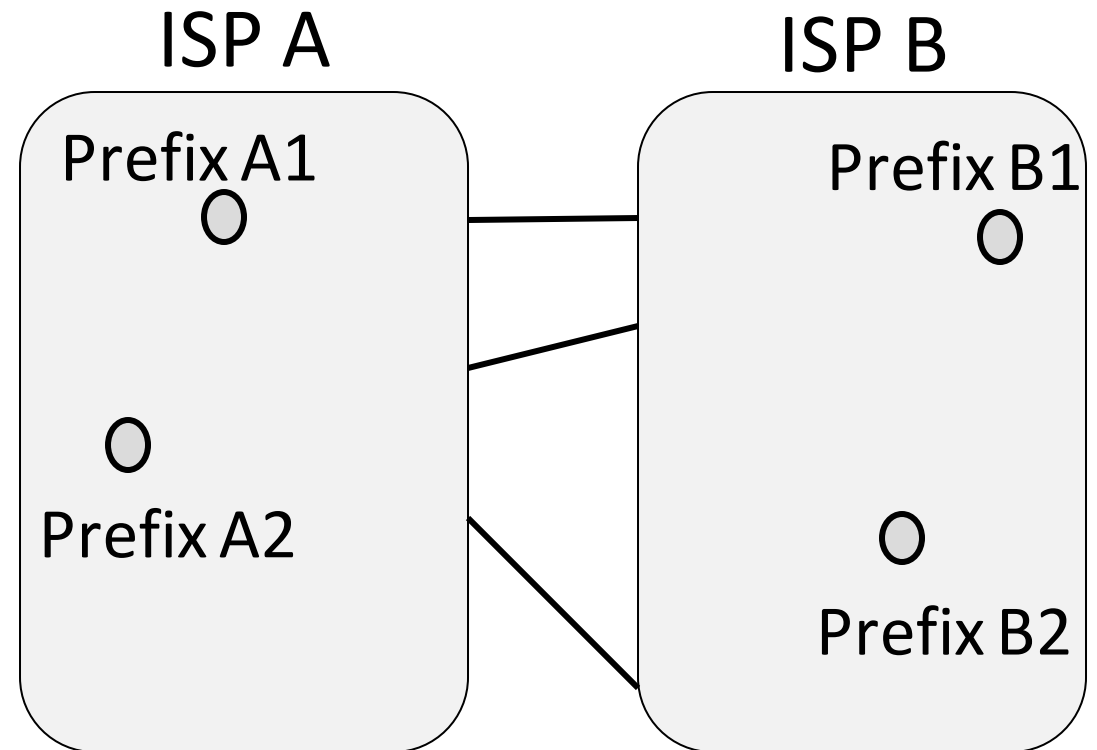
- Letting different parties choose their routes to suit their own needs



Yikes!

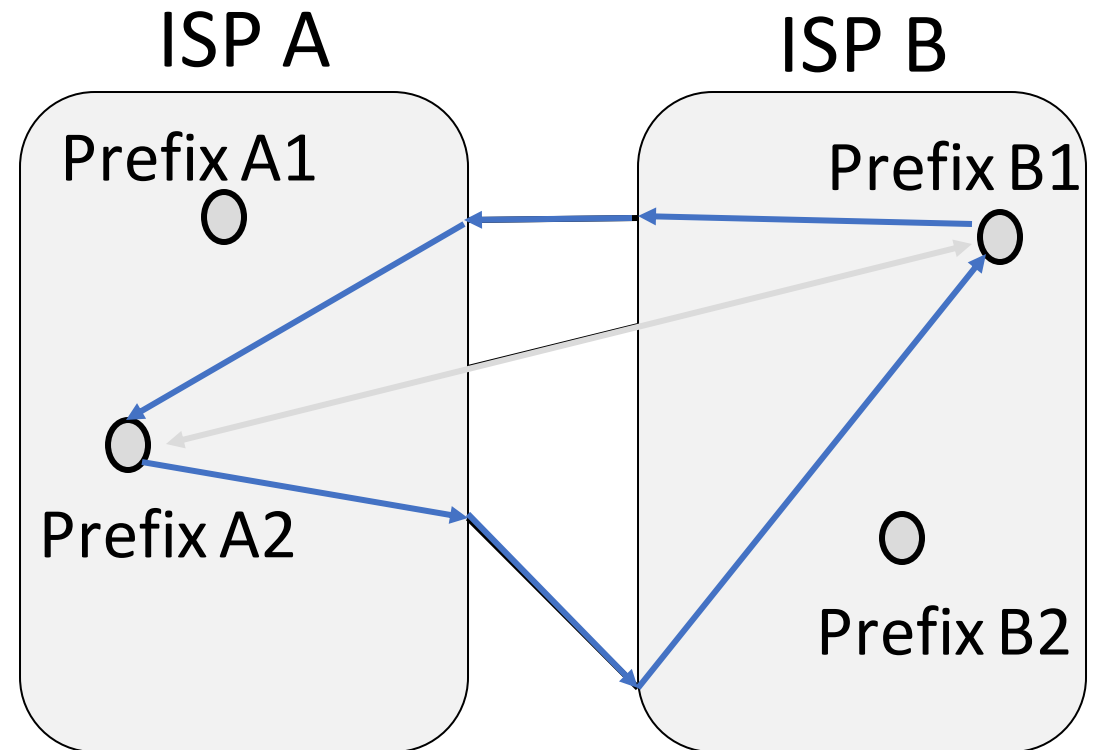
# Effects of Independent Parties

- Each party selects routes to suit its own interests
  - e.g, shortest path in ISP
- What path will be chosen for  $A2 \rightarrow B1$  and  $B1 \rightarrow A2$ ?
  - What is the best path?



## Effects of Independent Parties (2)

- Selected paths are longer than overall shortest path
  - And symmetric too!
- This is a consequence of independent goals and decisions, not hierarchy

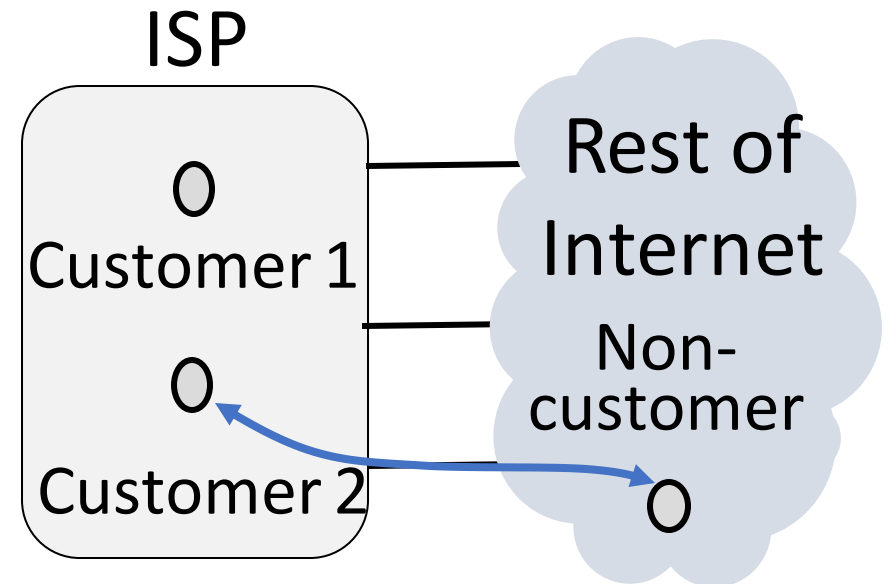


# Routing Policies

- Capture the goals of different parties
  - Could be anything
  - E.g., Internet2 only carries non-commercial traffic
- Common policies we'll look at:
  - ISPs give TRANSIT service to customers
  - ISPs give PEER service to each other

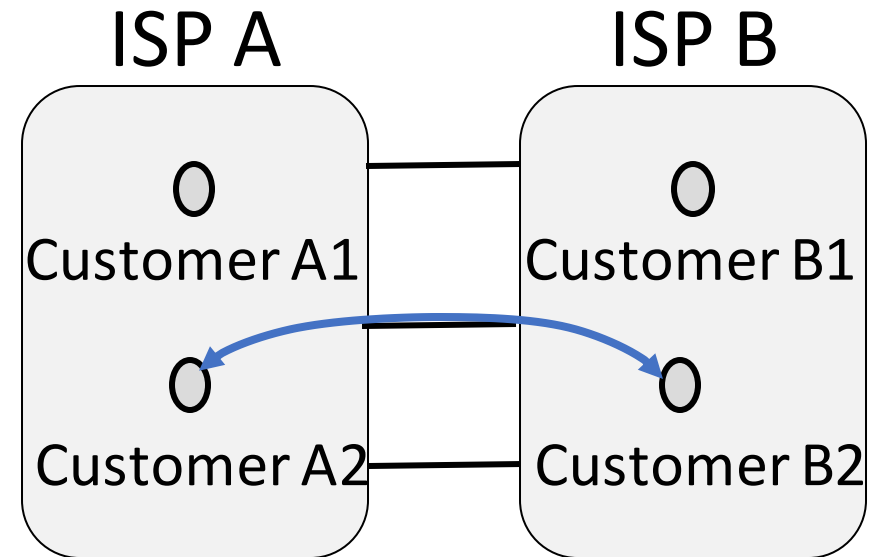
# Routing Policies – Transit

- One party (customer) gets TRANSIT service from another party (ISP)
  - ISP accepts traffic for customer from the rest of Internet
  - ISP sends traffic from customer to the rest of Internet
  - Customer pays ISP for the privilege



# Routing Policies – Peer

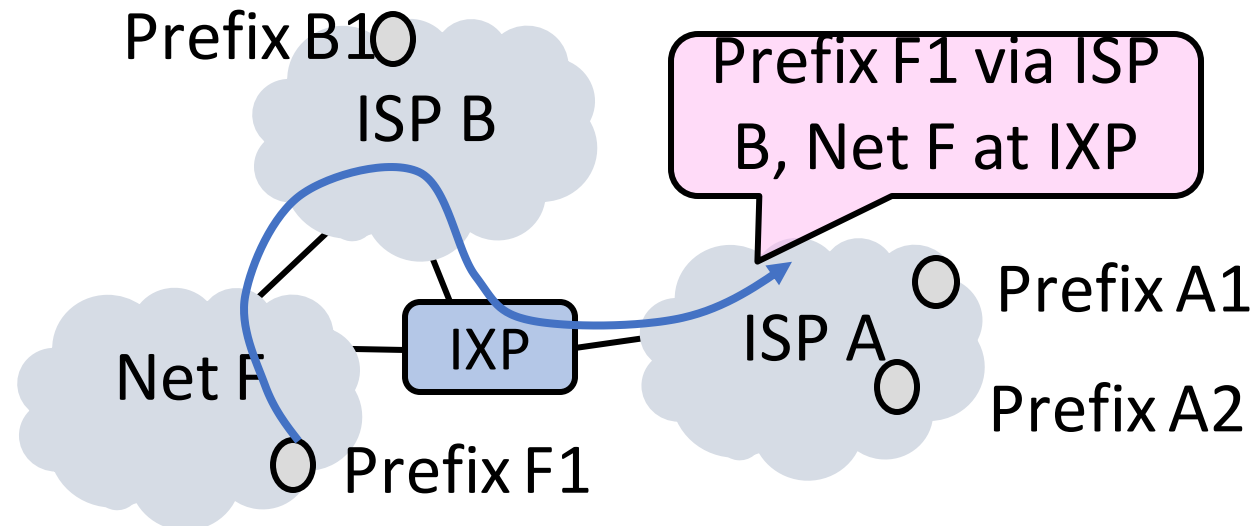
- Both party (ISPs in example) get PEER service from each other
  - Each ISP accepts traffic from the other ISP only for their customers
  - ISPs do not carry traffic to the rest of the Internet for each other
  - ISPs don't pay each other





# Routing with BGP (Border Gateway Protocol)

- iBGP is for internal routing
- eBGP is interdomain routing for the Internet
  - Path vector, a kind of distance vector



## Routing with BGP (2)

- Parties like ISPs are called AS (Autonomous Systems)
  - AS numbers assigned by regional Internet Assigned Numbers Authority (IANA) like APNIC
- AS's **MANUALLY** configure their internal BGP routes/advertisements
- External routes go through complicated filters for forwarding/filtering
- AS BGP routers communicate with each other to

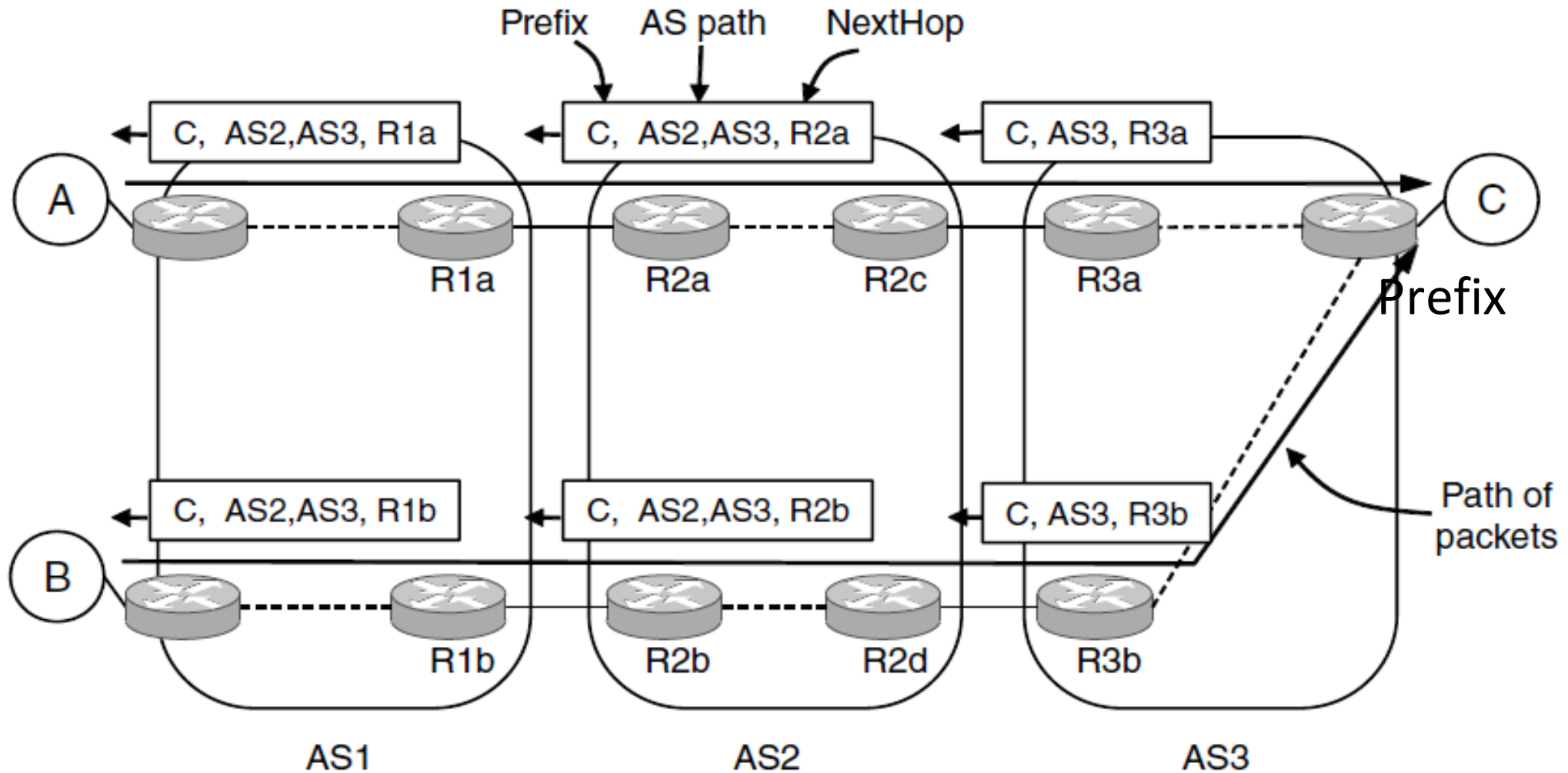
## Routing with BGP (2)

- Border routers of ASes announce BGP routes
- Route announcements have IP prefix, path vector, next hop
  - Path vector is list of ASes on the way to the prefix
  - List is to find loops
- Route announcements move in the opposite direction to traffic

# Routing with BGP (3)

- Application-layer protocol (uses TCP)
- Types of BGP Messages
  - Open: Create a relationship
  - Keepalive: Still here (reset timeouts)
  - Update: A route changed
  - Notification: Error message
  - Route Refresh: Please send me the route again

# Routing with BGP (5)



# Routing with BGP (5)

## Border Gateway Protocol - UPDATE Message

- Marker: ffffffffffffffffffffffffffffffffff
- Length: 56
- Type: UPDATE Message (2)
- Withdrawn Routes Length: 0
- Total Path Attribute Length: 28
- Path attributes
  - Path Attribute - ORIGIN: IGP
  - Path Attribute - AS\_PATH: empty
  - Path Attribute - NEXT\_HOP: 192.168.12.1
  - Path Attribute - MULTI\_EXIT\_DISC: 0
  - Path Attribute - LOCAL\_PREF: 100
- Network Layer Reachability Information (NLRI)
  - 1.1.1.1/32
    - NLRI prefix length: 32
    - NLRI prefix: 1.1.1.1 (1.1.1.1)

## Border Gateway Protocol - UPDATE Message

- Marker: ffffffffffffffffffffffffffffffffff
- Length: 28
- Type: UPDATE Message (2)
- Withdrawn Routes Length: 5
- Withdrawn Routes
  - 1.1.1.1/32
    - Withdrawn route prefix length: 32
    - Withdrawn prefix: 1.1.1.1 (1.1.1.1)
- Total Path Attribute Length: 0

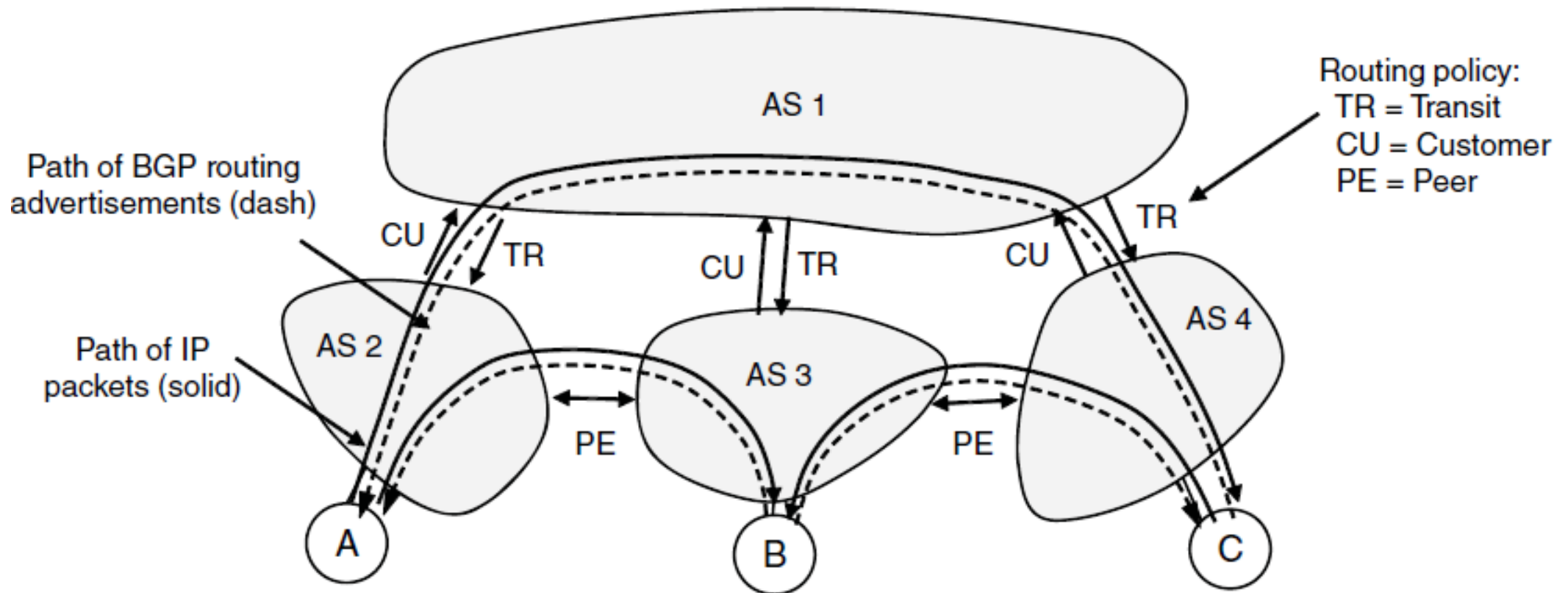
# Routing with BGP (6)

Policy is implemented in two ways:

1. Border routers of ISP announce paths only to other parties who may use those paths
  - Filter out paths others can't use
2. Border routers of ISP select the best path of the ones they hear in any, non-shortest way

# Routing with BGP (7)

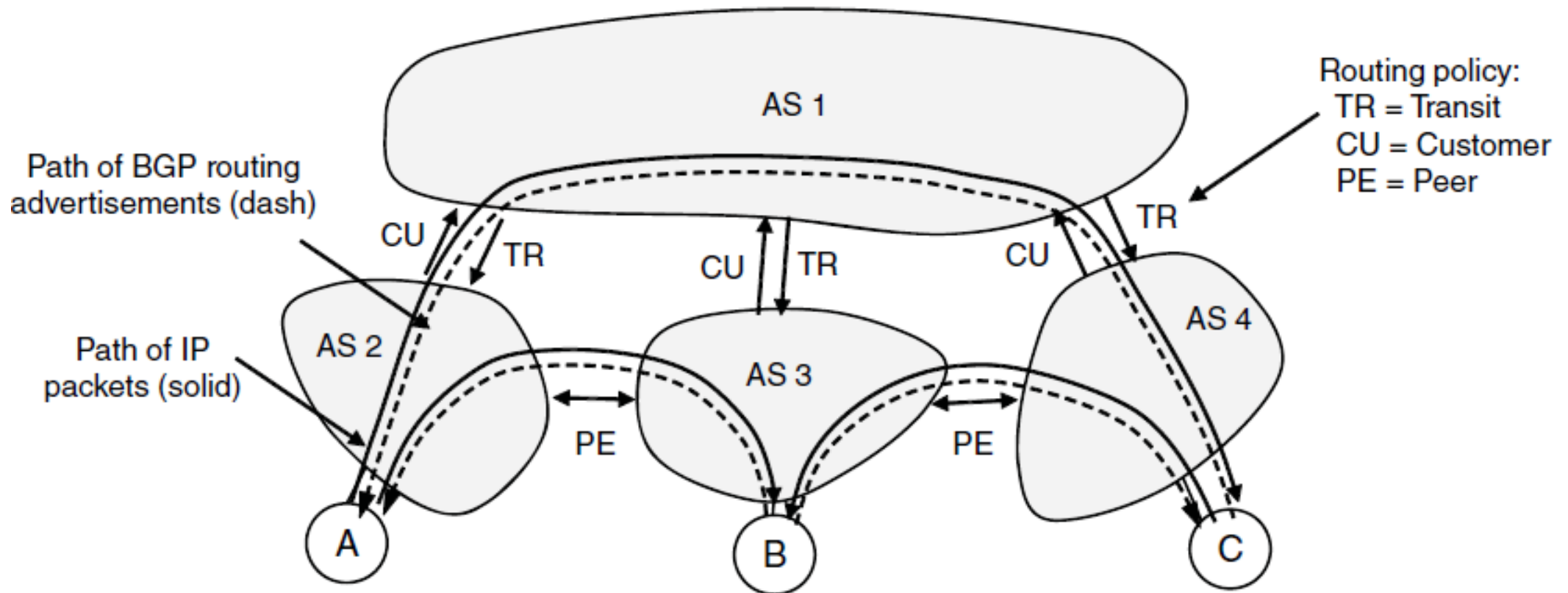
- TRANSIT: AS1 says [B, (AS1, AS3)], [C, (AS1, AS4)] to AS2





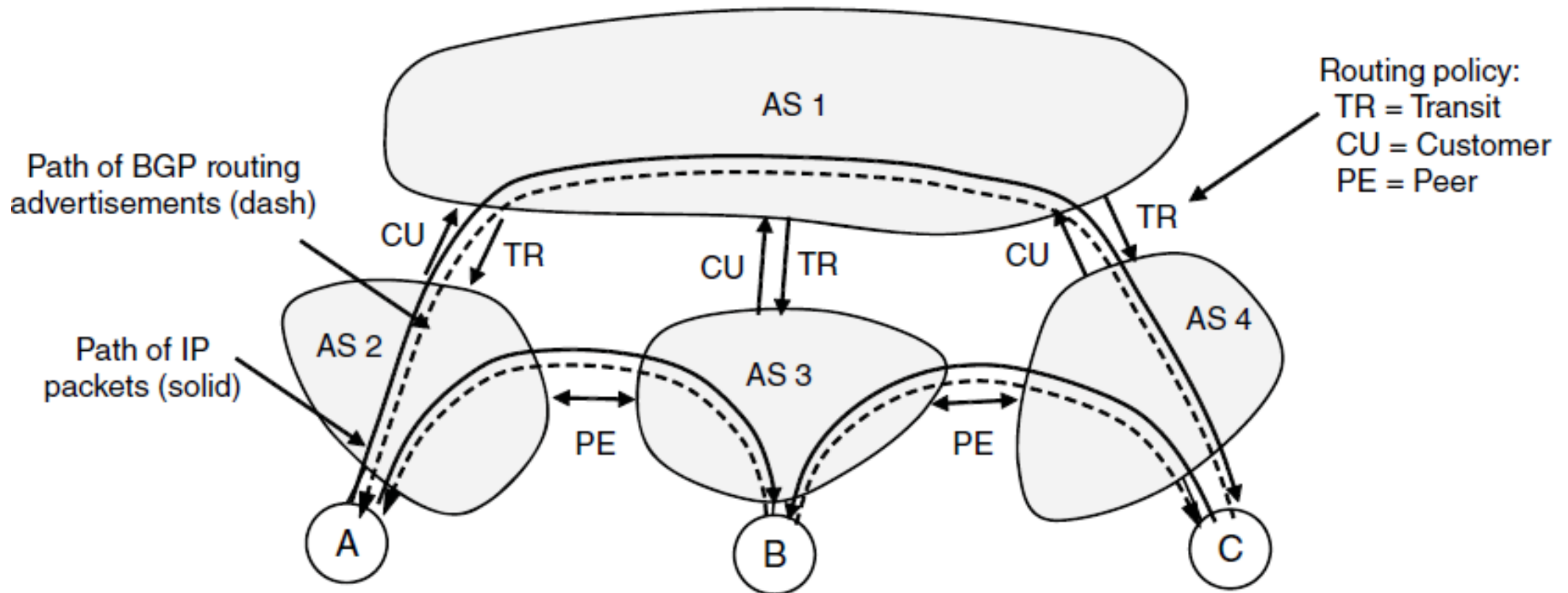
# Routing with BGP (8)

- CUSTOMER (other side of TRANSIT): AS2 says [A, (AS2)] to AS1



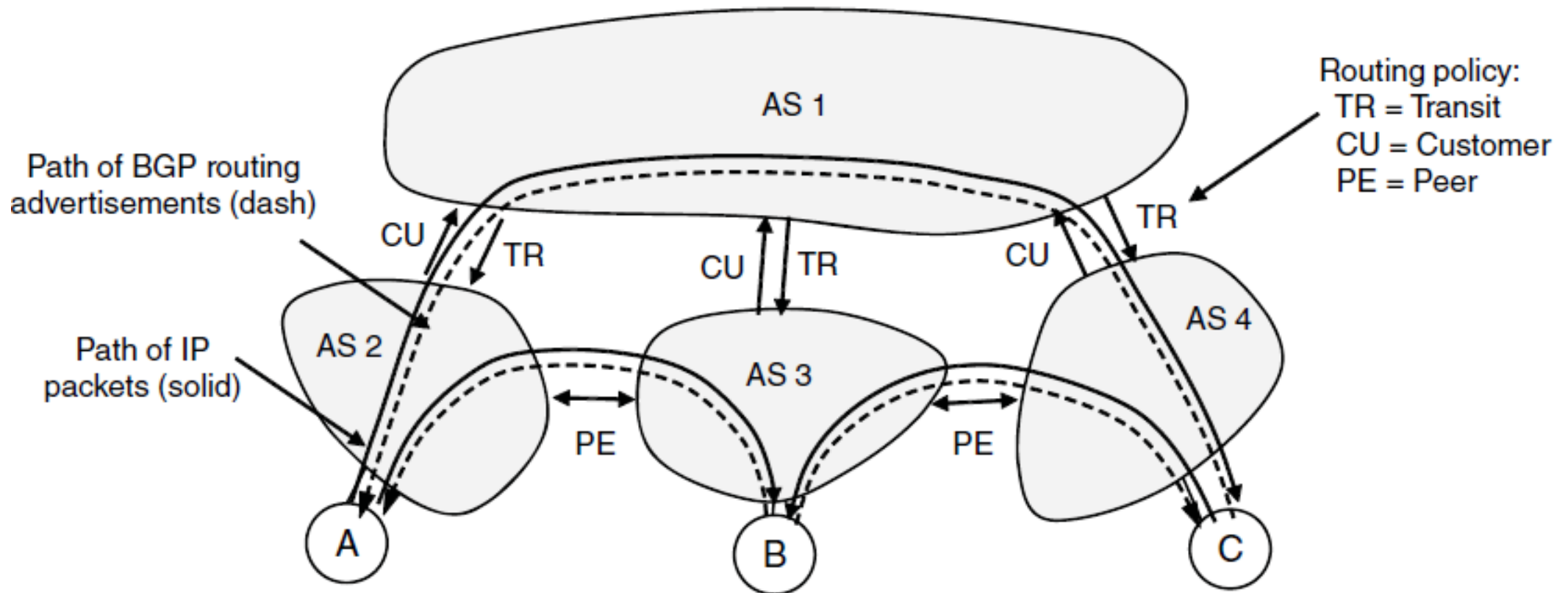
# Routing with BGP (9)

- PEER: AS2 says [A, (AS2)] to AS3, AS3 says [B, (AS3)] to AS2



# Routing with BGP (10)

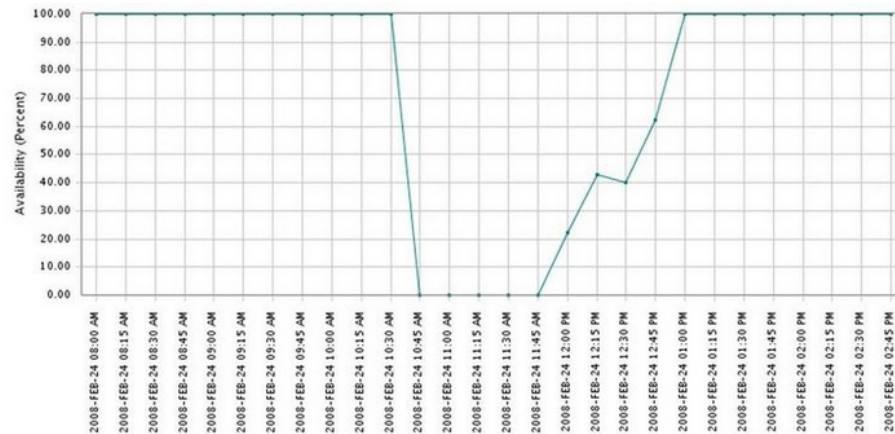
- AS2 has two routes to B (AS1, AS3) and chooses AS3 (Free!)



# How Pakistan knocked YouTube offline (and how to make sure it never happens again)

YouTube becoming unreachable isn't the first time that Internet addresses were hijacked. But if it spurs interest in better security, it may be the last.

BY DECLAN MCCULLAGH | FEBRUARY 25, 2008 4:28 PM PST



This graph that network-monitoring firm Keynote Systems provided to us shows the worldwide availability of YouTube.com dropping dramatically from 100 percent to 0 percent for over an hour. It didn't recover completely until two hours had elapsed.

Keynote Systems

A high-profile incident this weekend in which Pakistan's state-owned telecommunications company managed to cut YouTube off the global Web highlights a long-standing security weakness in the way the Internet is managed.

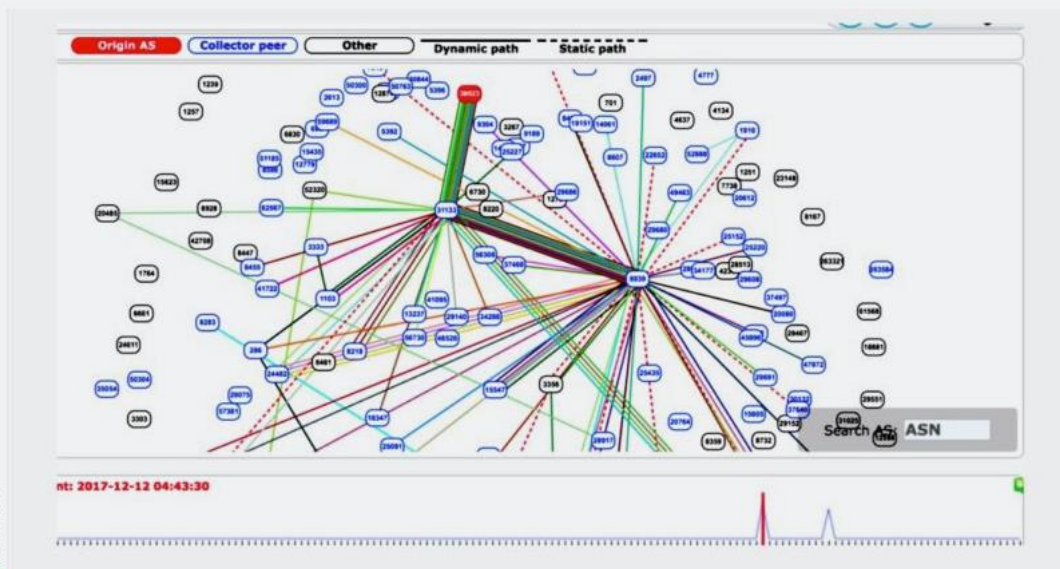
After receiving a censorship order from the telecommunications ministry directing that YouTube.com be blocked, Pakistan Telecom went even further. By accident or

BIZ &amp; IT—

# “Suspicious” event routes traffic for big-name sites through Russia

Google, Facebook, Apple, and Microsoft all affected by “intentional” BGP mishap.

DAN GOODIN - 12/13/2017, 2:43 PM



BGP Mon

[Enlarge](#)

104

Traffic sent to and from Google, Facebook, Apple, and Microsoft was briefly routed through a previously unknown Russian Internet provider Wednesday under circumstances researchers said was suspicious and intentional.



The unexplained incident involving the Internet's **Border Gateway Protocol** is the latest to raise troubling questions about the trust and reliability of communications sent over the global network. BGP routes large-scale amounts of traffic among Internet backbones, ISPs, and other large networks. But despite the sensitivity and amount of data it controls, BGP's security is often based on trust and word of mouth. Wednesday's event comes eight months after large chunks of network traffic belonging to **MasterCard, Visa, and more than two dozen other financial services** were briefly routed through a Russian government-



#### FURTHER READING

Russian-controlled telecom hijacks financial services' Internet traffic

# BGP Thoughts

- Much more beyond basics to explore!
- Policy is a substantial factor
  - Can independent decisions be sensible overall?
- Other important factors:
  - Convergence effects
  - How well it scales
  - Integration with intradomain routing
  - And more ...

# IP Services

# Issues?

- What else is missing?



# Issues?

- Where does this break down?

Bootstrapping (DHCP)

Finding Link nodes (ARP)

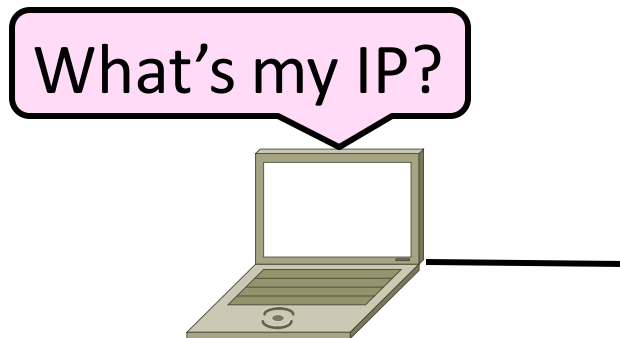
Errors in the network (ICMP)

Running out of addresses (IPv6, NAT)

# Dynamic Host Configuration Protocol (DHCP)

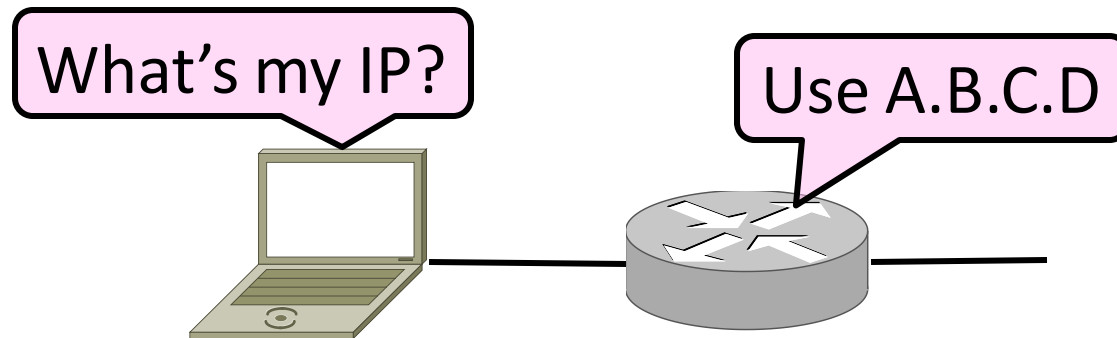
# Bootstrapping

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



# Bootstrapping (2)

1. Manual configuration (old days)
  - Can't be factory set, depends on use
2. DHCP: Automatically configure addresses
  - Shifts burden from users to IT folk

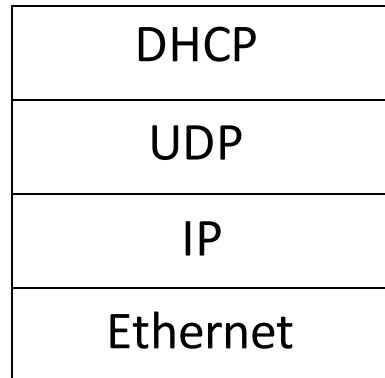


# DHCP

- 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

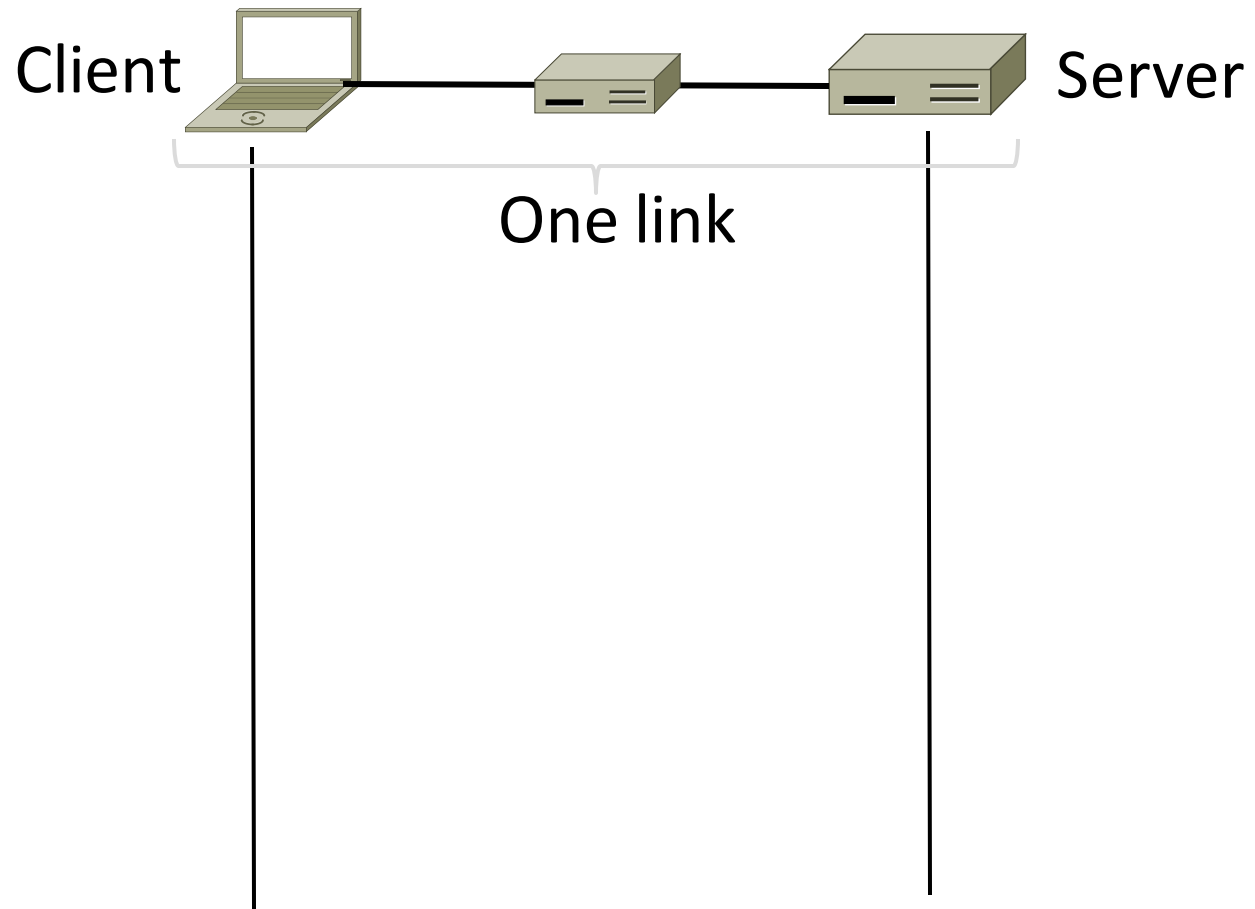
- DHCP is a client-server application
  - Uses UDP ports 67, 68



# DHCP Addressing

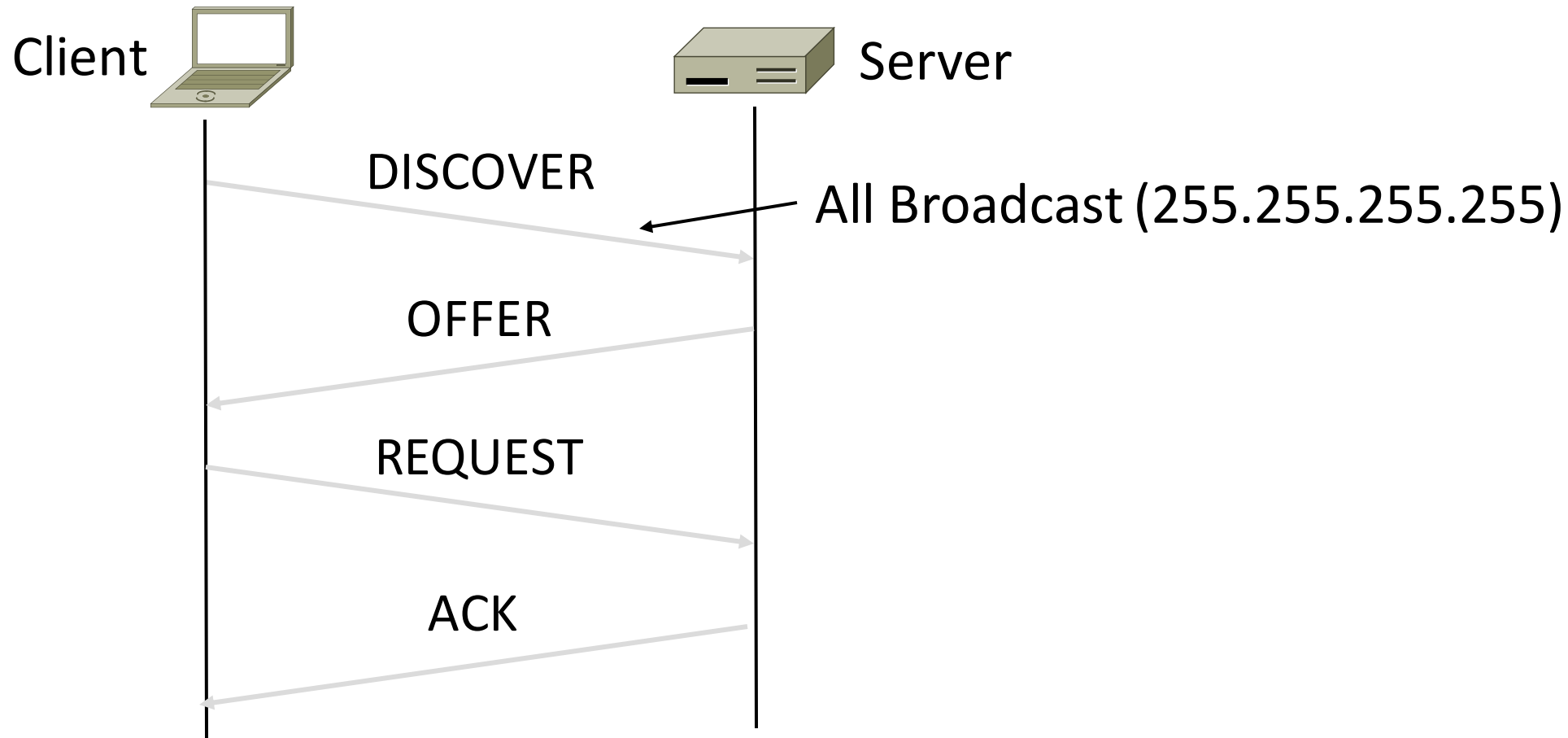
- Bootstrap issue:
  - How does node send a message to DHCP server before it is configured?
- Answer:
  - Node sends broadcast messages that delivered to all nodes on the network
  - Broadcast address is all 1s
  - IP (32 bit): 255.255.255.255
  - Ethernet (48 bit): ff:ff:ff:ff:ff:ff

# DHCP Messages





# DHCP Messages (2)



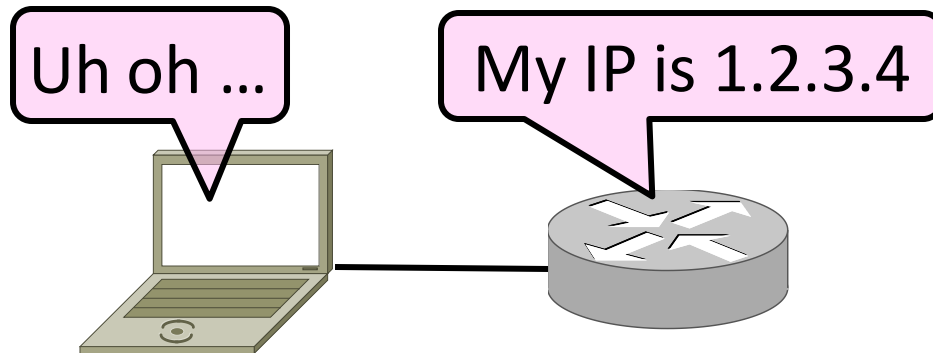
# DHCP Messages (3)

- To renew an existing lease, an abbreviated sequence is used:
  - REQUEST, followed by ACK
- Protocol also supports replicated servers for reliability

# Address Resolution Protocol (ARP)

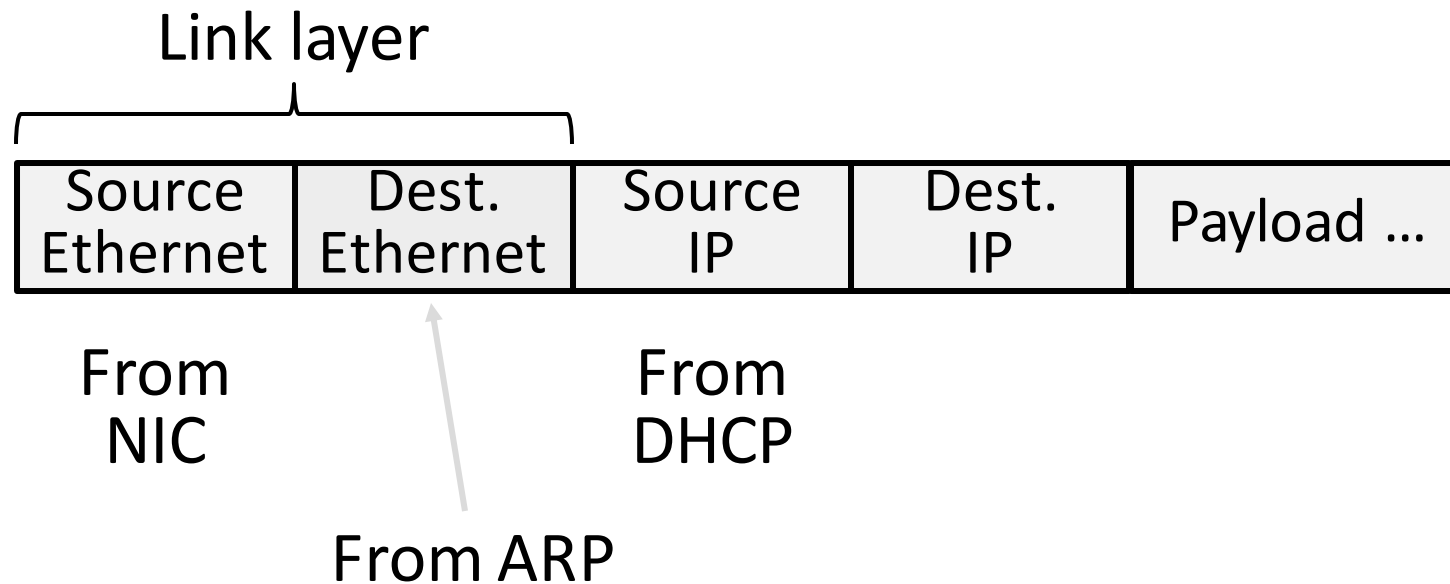
# Sending an IP Packet

- 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?



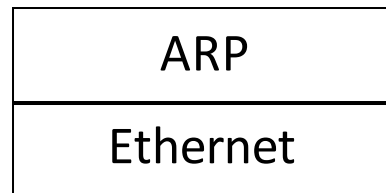
# ARP (Address Resolution Protocol)

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

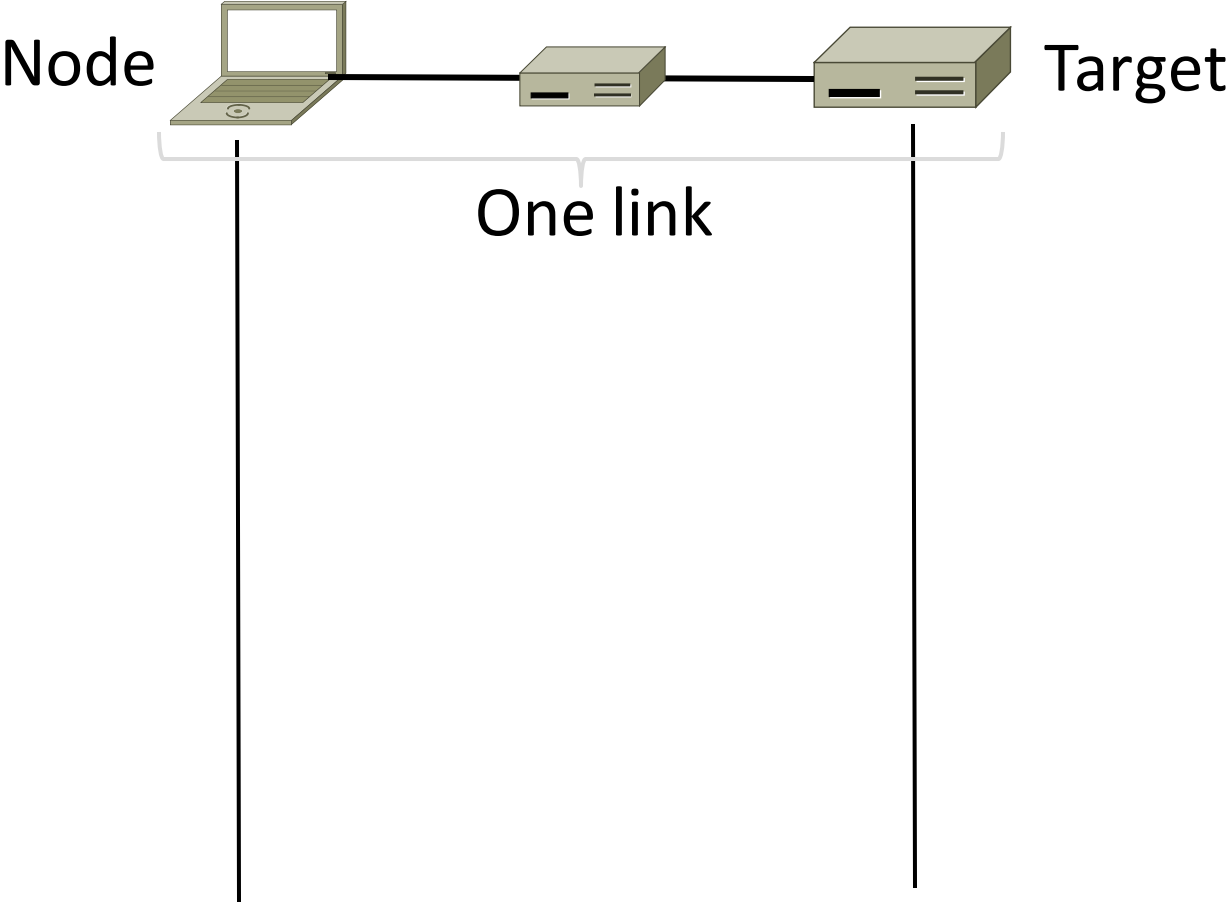


# ARP Protocol Stack

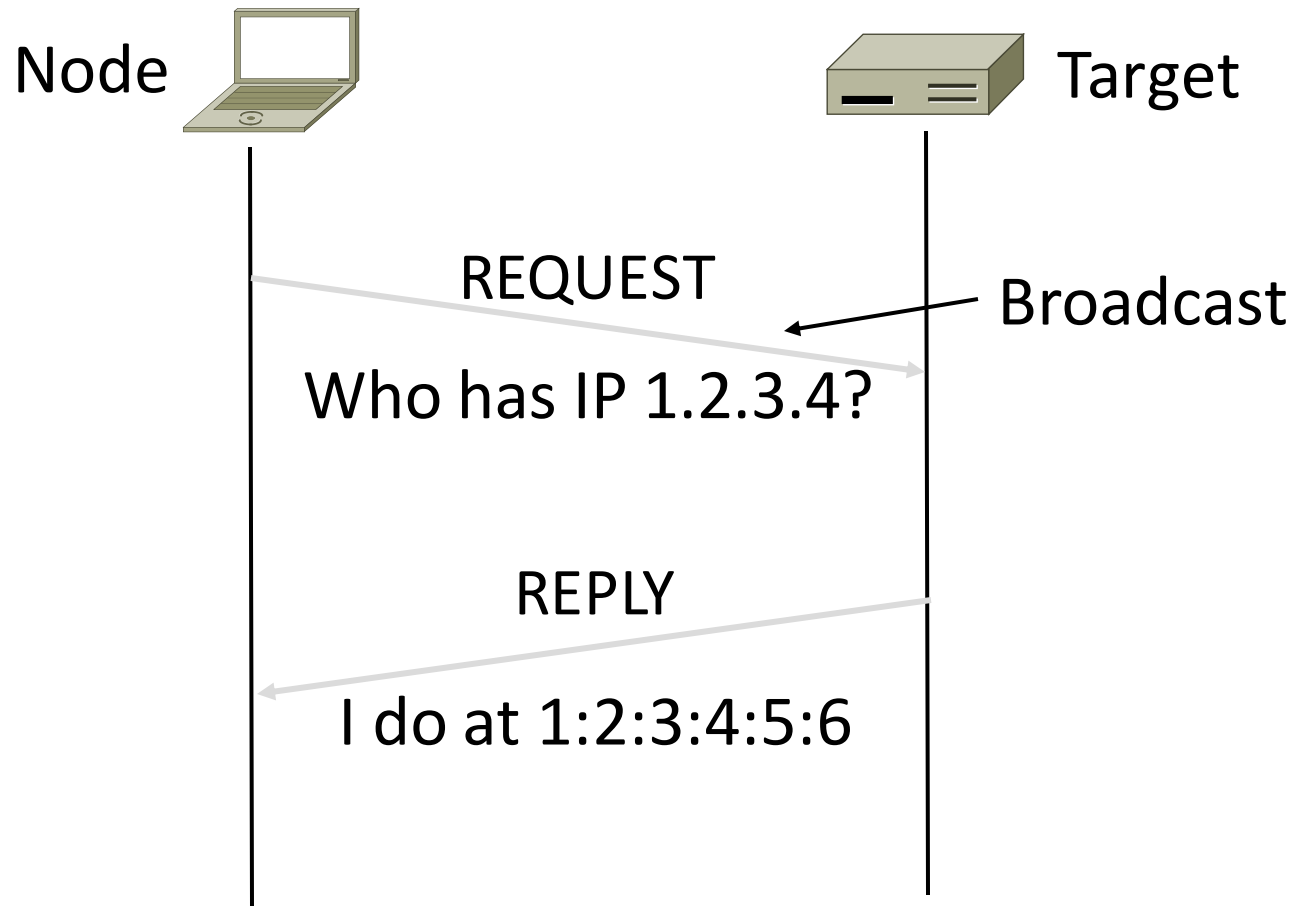
- ARP sits right on top of link layer
  - No servers, just asks node with target IP to identify itself
  - Uses broadcast to reach all nodes



# ARP Messages



# ARP Messages (2)



```
[root@host ~]# tcpdump -lni any arp &  
( sleep 1; arp -d 10.0.0.254; ping -c1 -n  
10.0.0.254 )
```

```
listening on any, link-type LINUX_SLL  
(Linux cooked), capture size 96 bytes
```

```
17:58:02.155495 arp who-has  
10.2.1.224 tell 10.2.1.253
```

```
17:58:02.317444 arp who-has 10.0.0.96  
tell 10.0.0.253
```

```
17:58:02.370446 arp who-has 10.3.1.12  
tell 10.3.1.61
```



# ARP Table

```
# arp -an | grep 10
```

```
? (10.241.1.114) at 00:25:90:3e:dc:fc [ether] on vlan241
```

```
? (10.252.1.8) at 00:c0:b7:76:ac:19 [ether] on vlan244
```

```
? (10.252.1.9) at 00:c0:b7:76:ae:56 [ether] on vlan244
```

```
? (10.241.1.111) at 00:30:48:f2:23:fd [ether] on vlan241
```

```
? (10.252.1.6) at 00:c0:b7:74:fb:9a [ether] on vlan244
```

```
? (10.241.1.121) at 00:25:90:2c:d4:f7 [ether] on vlan241
```

```
[...]
```

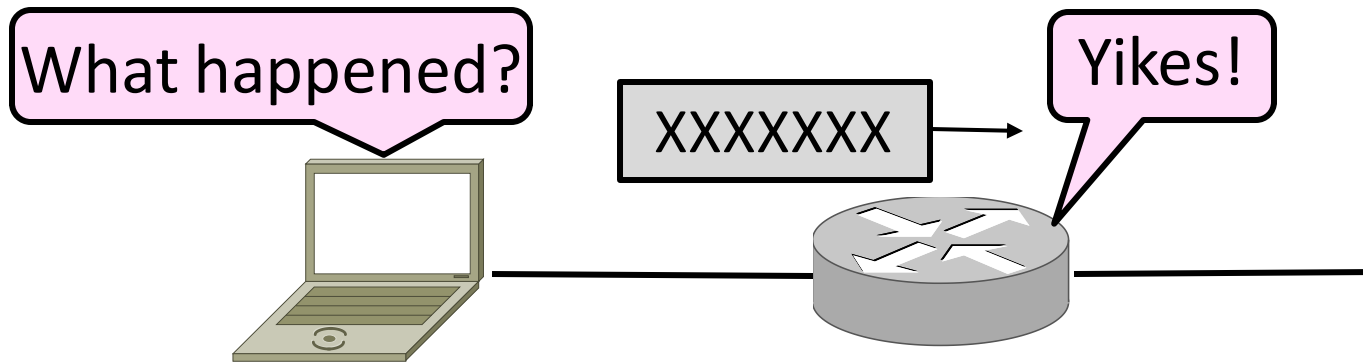
# Discovery Protocols

- 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

# Internet Control Message Protocol (ICMP)

# Topic

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

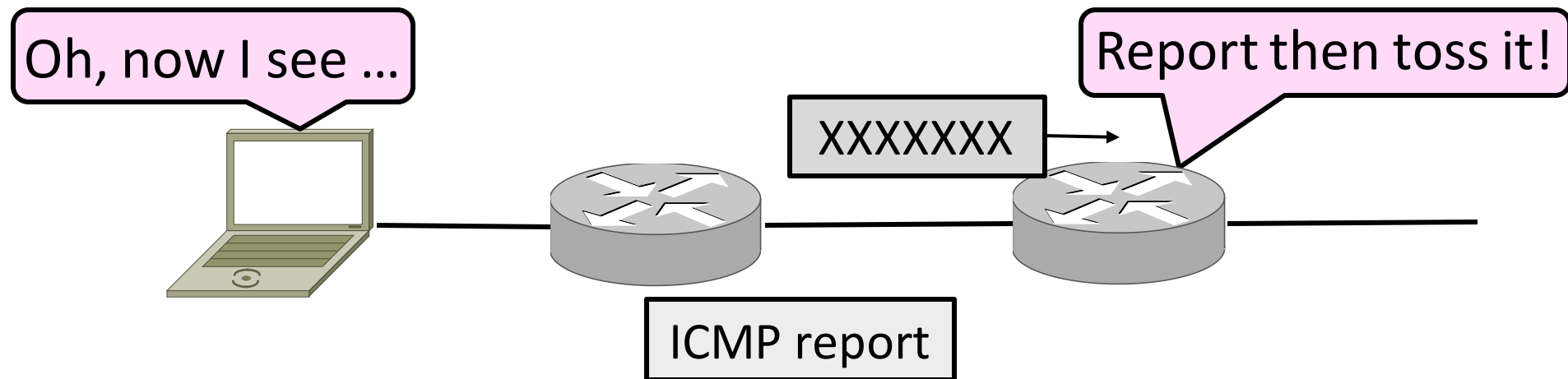


# Internet Control Message Protocol

- ICMP is a companion protocol to IP
  - They are implemented together
  - Sits on top of IP (IP Protocol=1)
- Provides error report and testing
  - Error is at router while forwarding
  - Also testing that hosts can use

# ICMP Errors

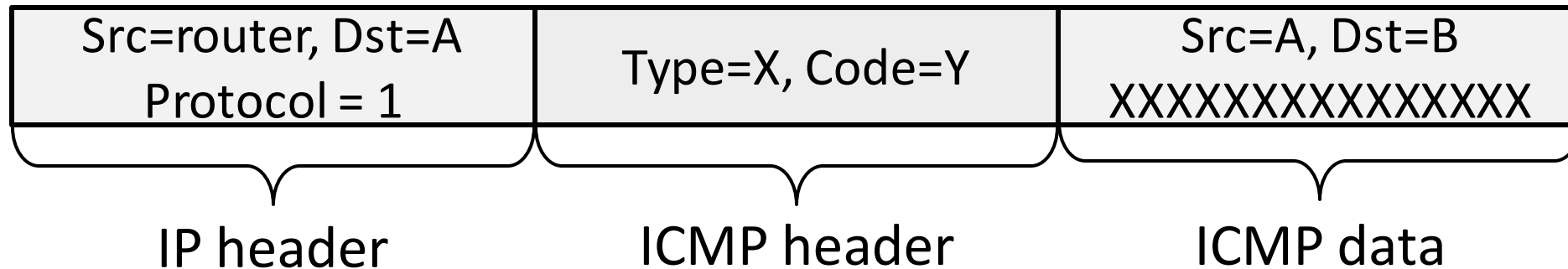
- When router encounters an error while forwarding:
  - It sends an ICMP error report back to the IP source
  - It discards the problematic packet; host needs to rectify



# ICMP Message Format (2)

- 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


Portion of offending packet,  
starting with its IP header



# Example ICMP Messages

<b>Name</b>	<b>Type / Code</b>	<b>Usage</b>
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

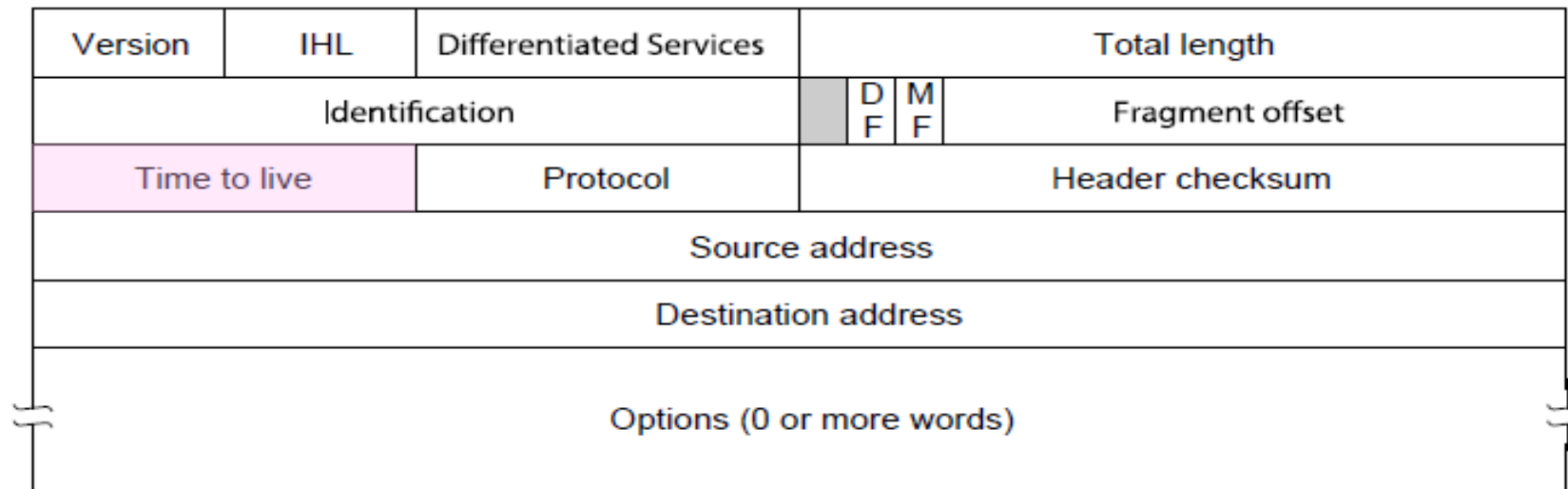
Testing, not a forwarding error: Host sends Echo Request, and destination responds with an Echo Reply





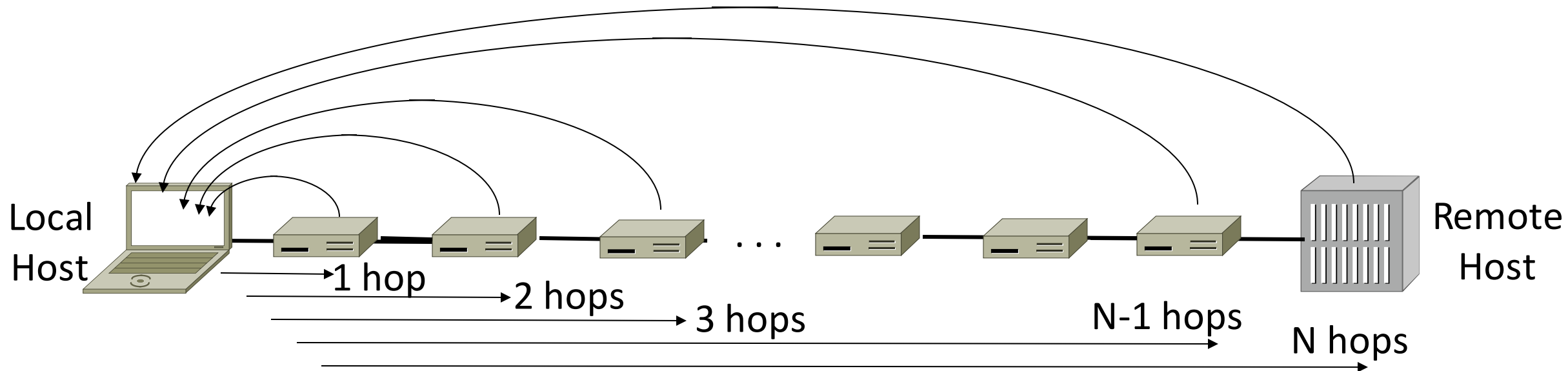
# Traceroute

- IP header contains TTL (Time to live) field
  - Decrement every router hop, with ICMP error at zero
  - Protects against forwarding loops



# Traceroute (2)

- Traceroute repurposes TTL and ICMP functionality
  - Sends probe packets increasing TTL starting from 1
  - ICMP errors identify routers on the path

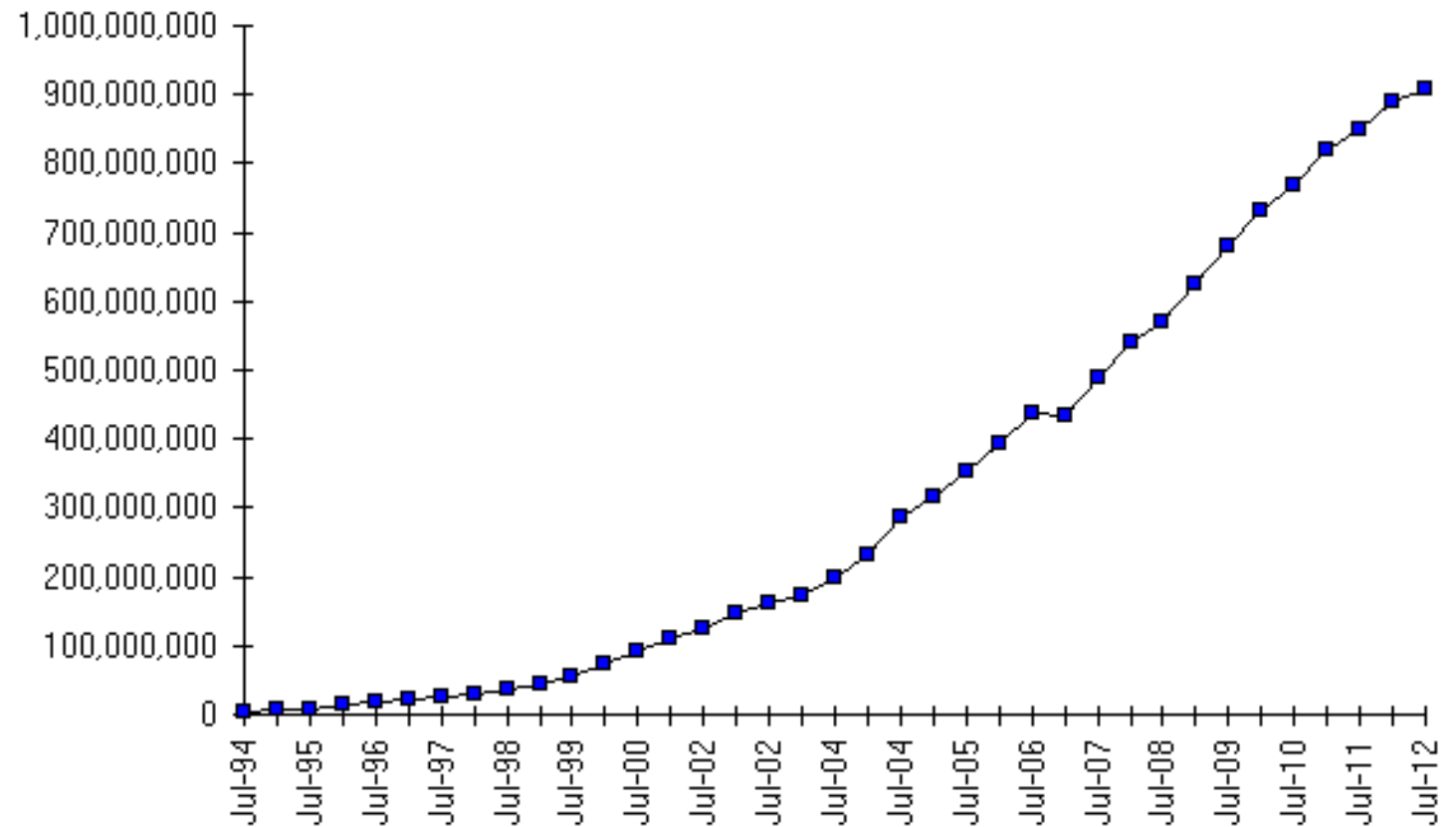


# Network Address Translation (NAT)

# Problem: Internet Growth

- Many billions of hosts
- And we're using 32-bit addresses!

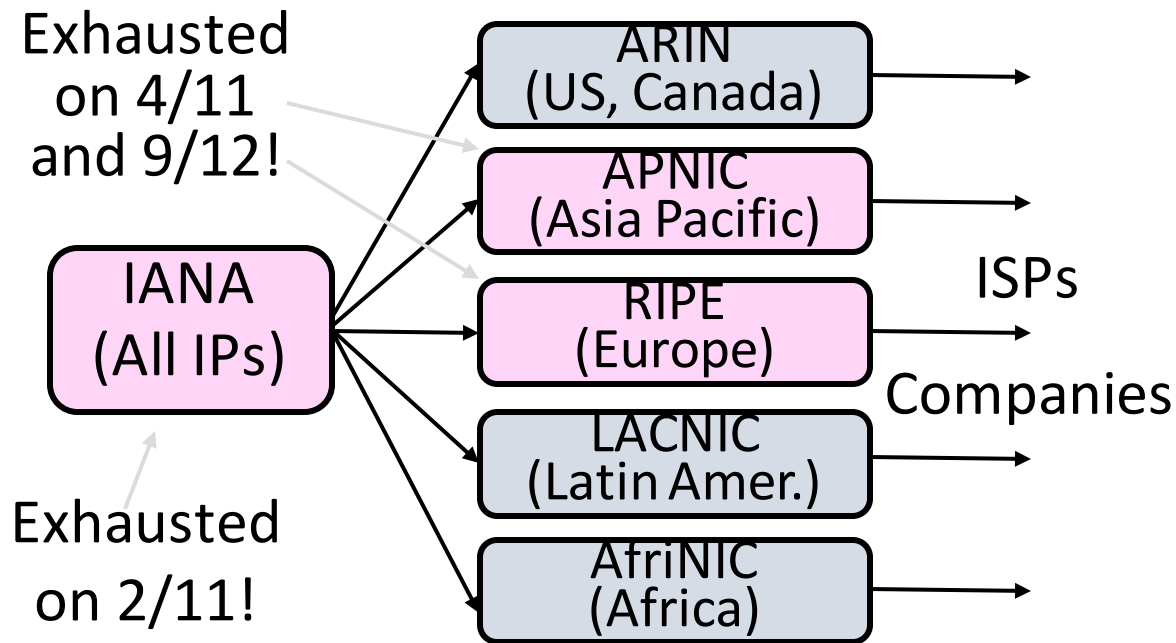
Internet Domain Survey Host Count



Source: Internet Systems Consortium ([www.isc.org](http://www.isc.org))

# The End of New IPv4 Addresses

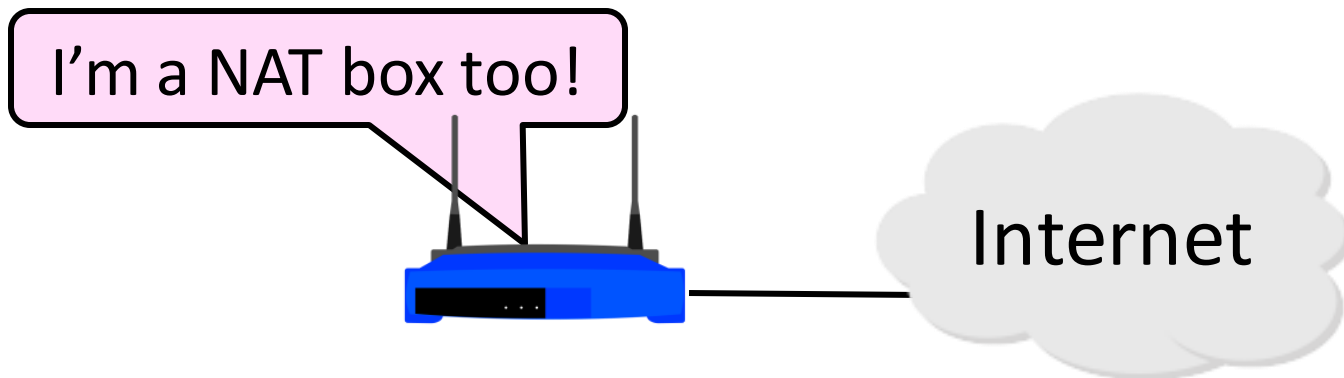
- Now running on leftover blocks held by the regional registries; much tighter allocation policies



End of the world ? 12/21/12?

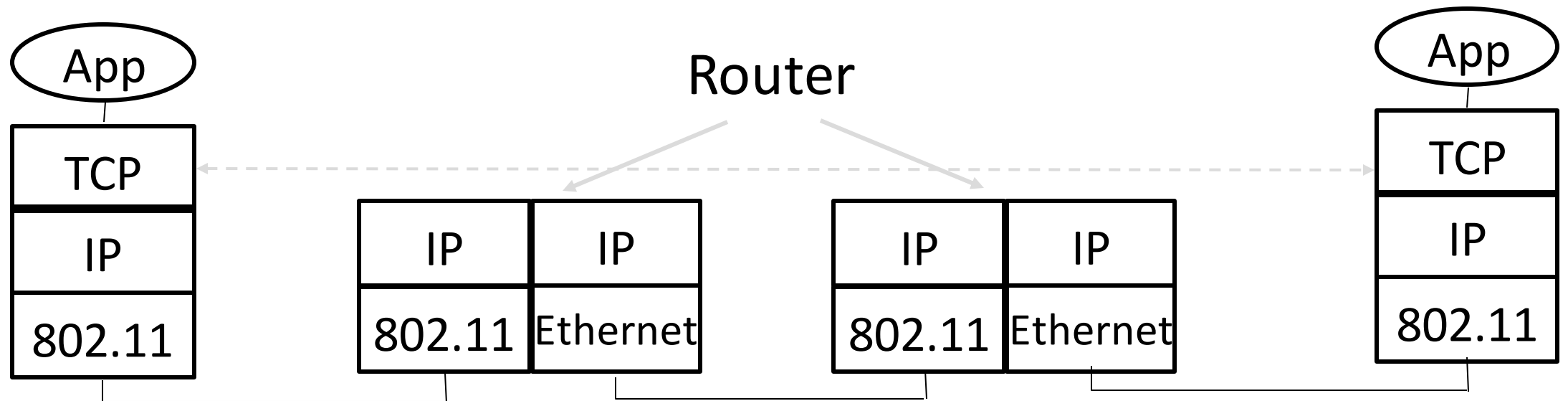
# Solution 1: Network Address Translation (NAT)

- Basic idea: Map many “Private” IP addresses to one “Public” IP.
- Allocate IPs for private use (192.168.x, 10.x)



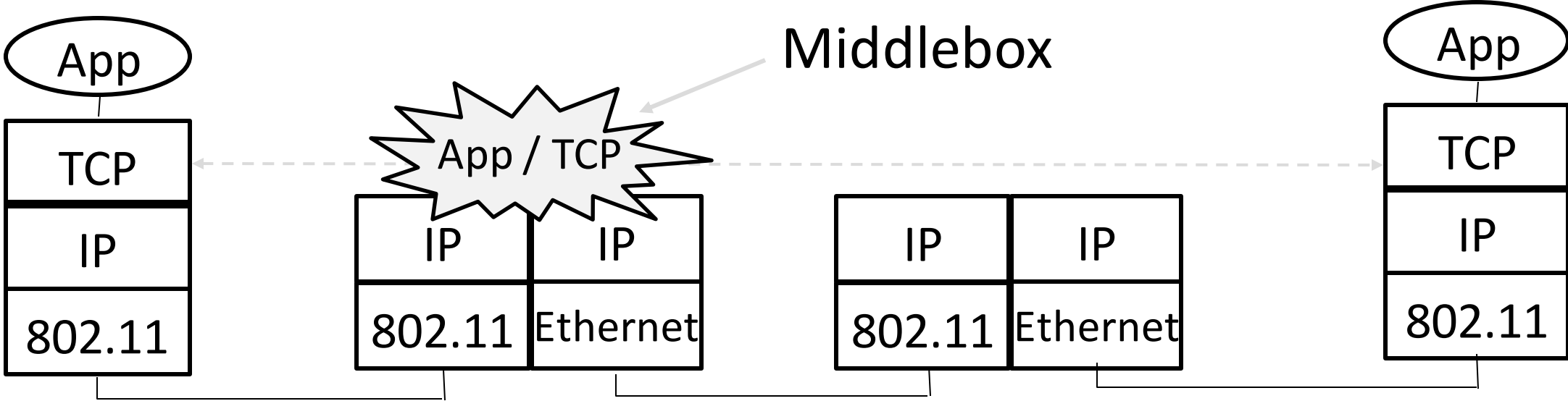
# Layering Review

- Remember how layering is meant to work?
  - “Routers don’t look beyond the IP header.” Well ...



# Middleboxes

- Sit “inside the network” but perform “more than IP” processing on packets to add new functionality
  - NAT box, Firewall / Intrusion Detection System





# Middleboxes (2)

- Advantages

- A possible rapid deployment path when no other option
- Control over many hosts (IT)

- Disadvantages

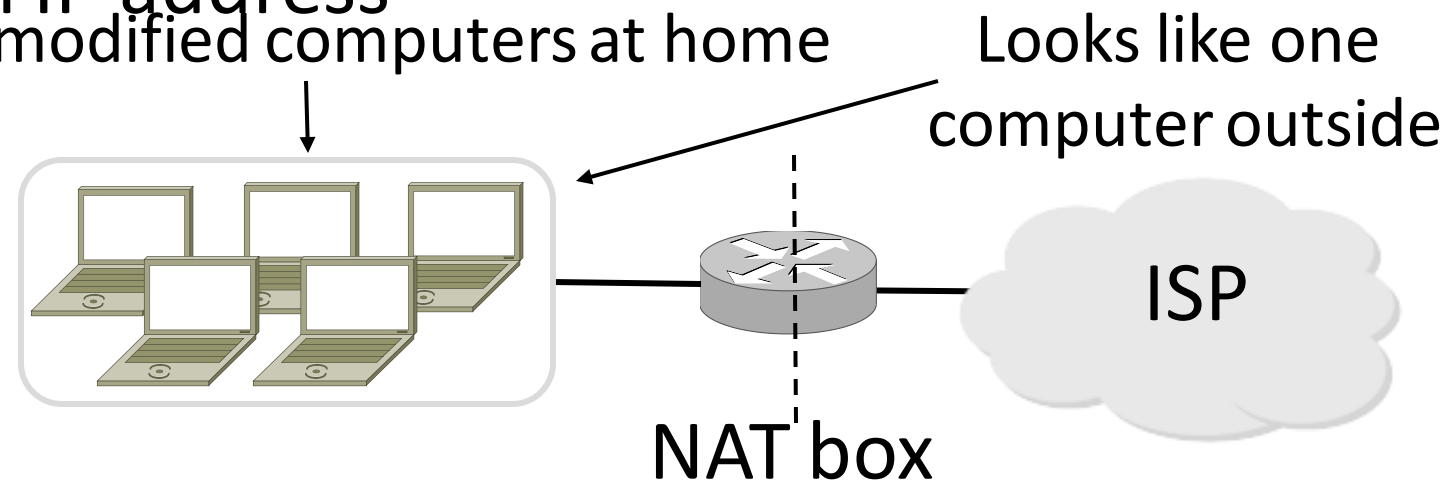
- Breaking layering interferes with connectivity
  - strange side effects
- Poor vantage point for many tasks

# NAT (Network Address Translation)

- NAT box maps an internal IP to an external IP
  - Many internal hosts connected using few external addresses
  - Middlebox that “translates addresses”
- Motivated by IP address scarcity
  - Controversial at first, now accepted

# NAT (2)

- Common scenario:
  - Home computers use “private” IP addresses
    - 10.0.0.0/8, 172.16.0.0/12, 192.168.0.0/16
  - NAT (in AP/firewall) connects home to ISP using a single external IP address



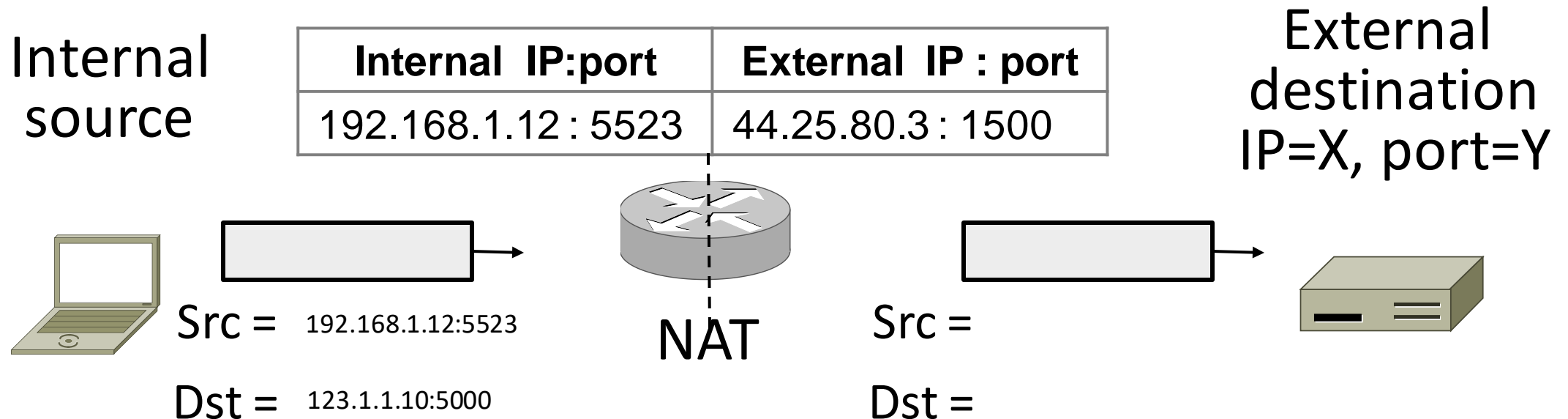
# How NAT Works

- Keeps an internal/external translation table
  - Typically uses IP address + TCP port
  - This is address and port translation

What host thinks	What ISP thinks
<b>Internal IP : port</b>	<b>External IP : port</b>
192.168.1.12 : 5523	44.25.80.3 : 1500
192.168.1.13 : 1234	44.25.80.3 : 1501
192.168.2.20 : 1234	44.25.80.3 : 1502

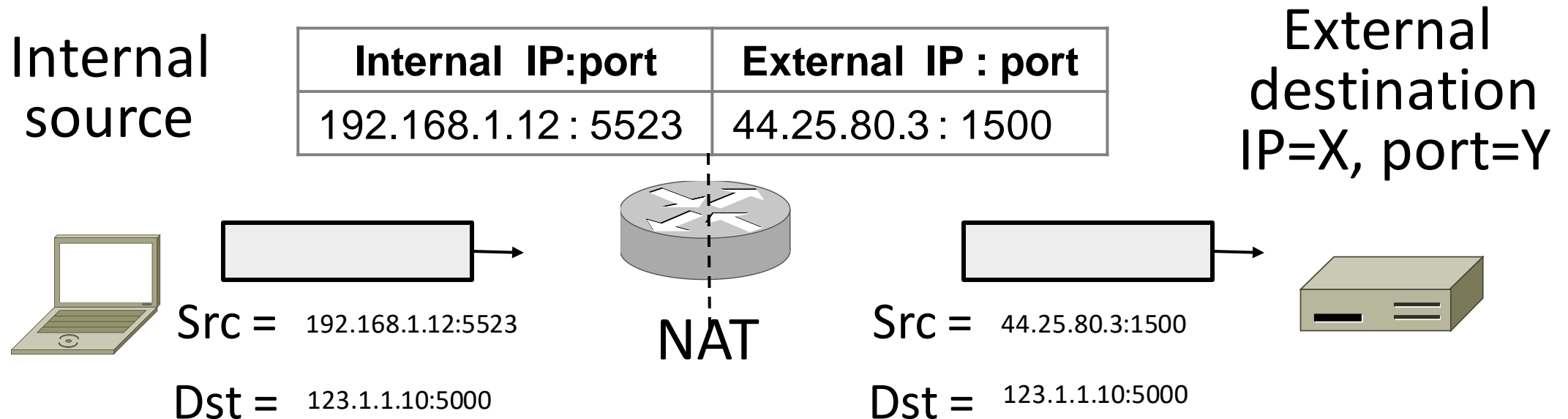
# How NAT Works (2)

- Internal → External:
  - Look up and rewrite Source IP/port



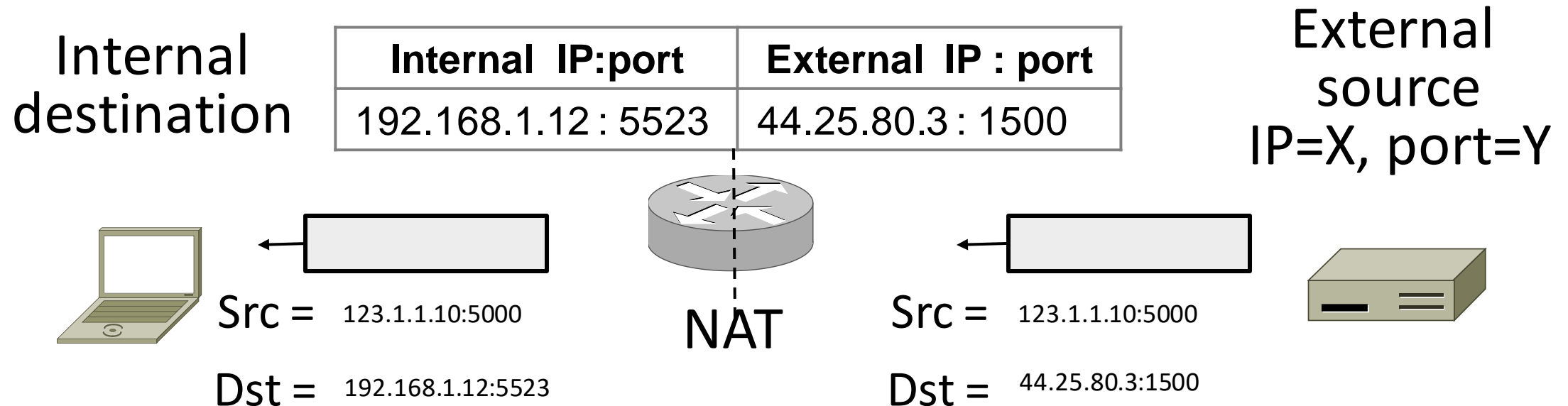
# How NAT Works (2)

- Internal → External:
  - Look up and rewrite Source IP/port



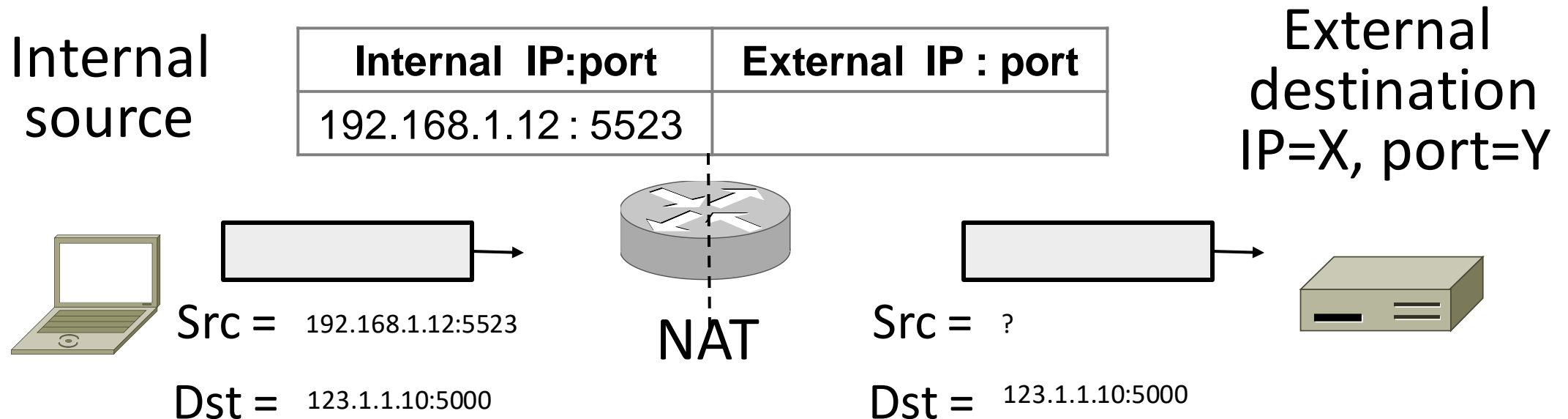
# How NAT Works (3)

- External ← Internal
  - Look up and rewrite Destination IP/port



# How NAT Works (4)

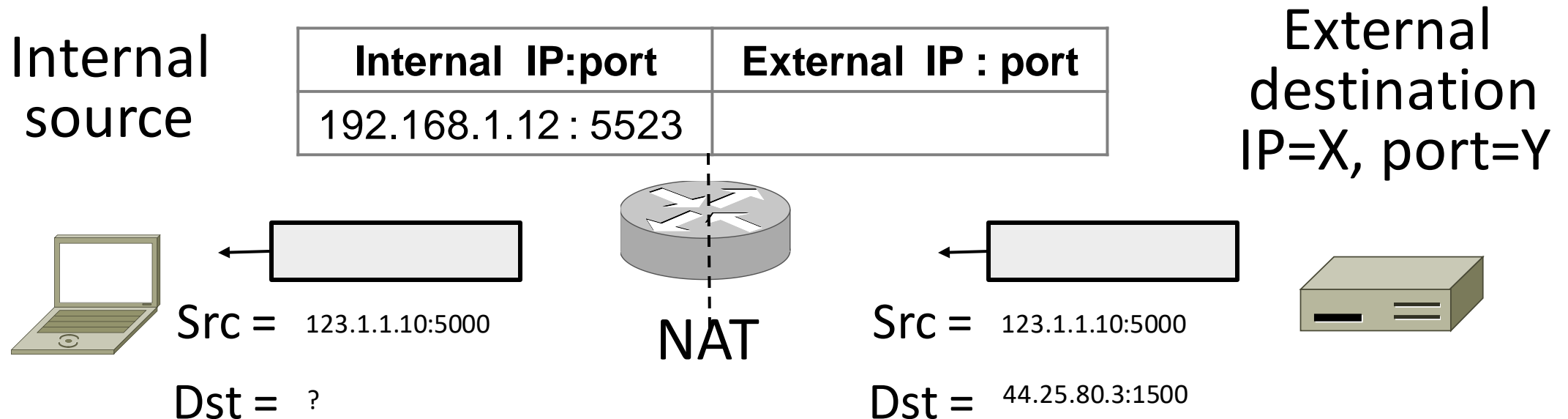
- Need to enter translations in the table for it to work
  - Create external name when host makes a TCP connection





# How NAT Works (5)

- What happens when a message arrives for an internal source without a table entry?



# NAT Downsides

- Connectivity has been broken!
  - Can only send incoming packets after an outgoing connection is set up
  - Difficult to run servers or peer-to-peer apps (Skype)
- Doesn't work when there are no connections (UDP)
- Breaks apps that expose their IP addresses (FTP)

# NAT Upsides

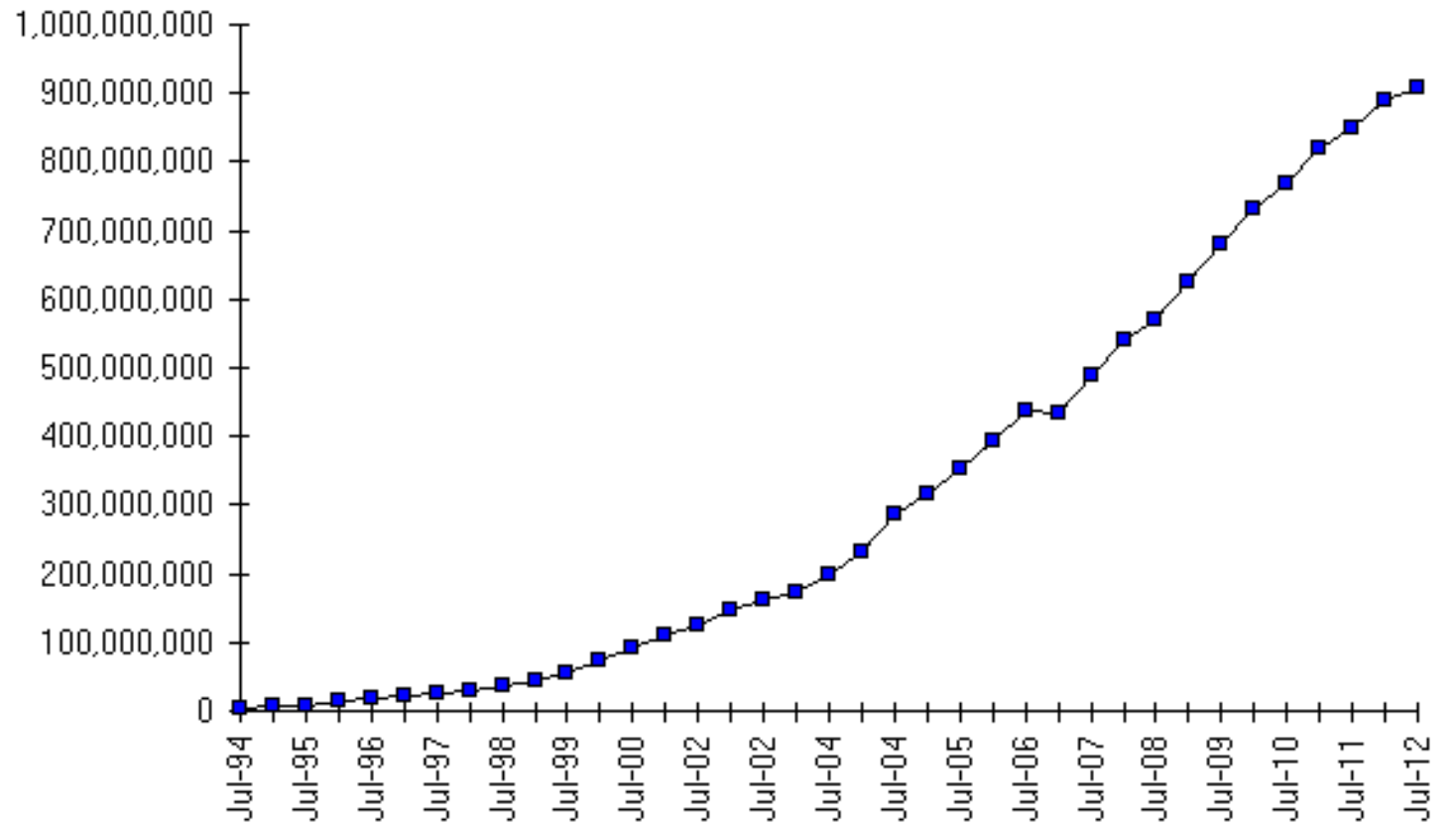
- Relieves much IP address pressure
  - Many home hosts behind NATs
- Easy to deploy
  - Rapidly, and by you alone
- Useful functionality
  - Firewall, helps with privacy
- Kinks will get worked out eventually
  - “NAT Punching/Traversal” for incoming traffic

IPv6

# Problem: Internet Growth

- Many billions of hosts
- And we're using 32-bit addresses!

Internet Domain Survey Host Count



Source: Internet Systems Consortium ([www.isc.org](http://www.isc.org))

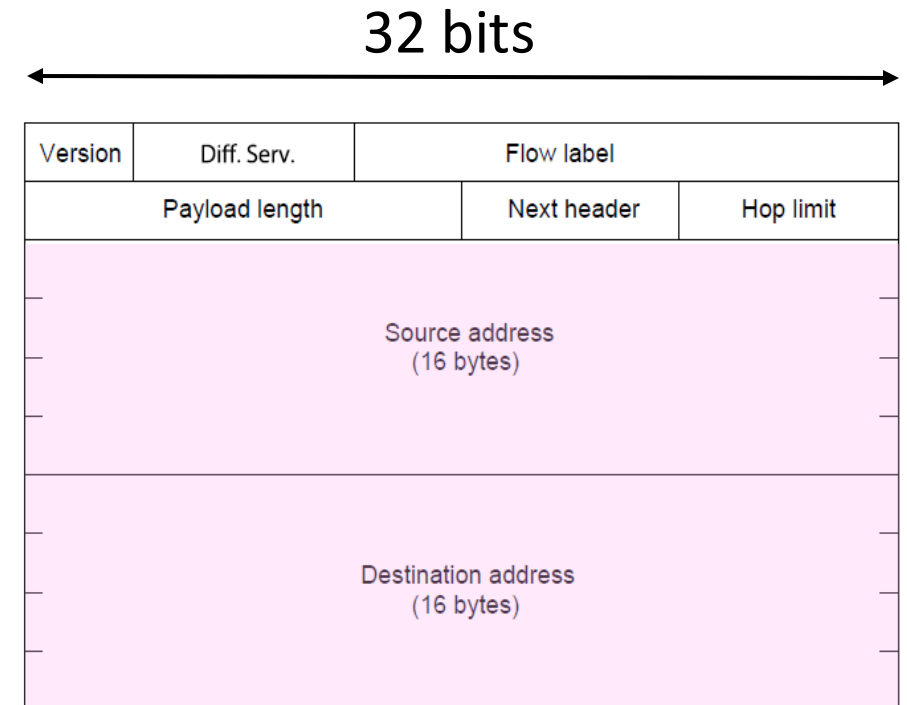
# IP Version 6 to the Rescue

- Effort started by the IETF in 1994
  - Much larger addresses (128 bits)
  - Many sundry improvements
- Became an IETF standard in 1998
  - Nothing much happened for a decade
  - Hampered by deployment issues, and a lack of adoption incentives
  - Big push ~2011 as exhaustion loomed

# IPv6

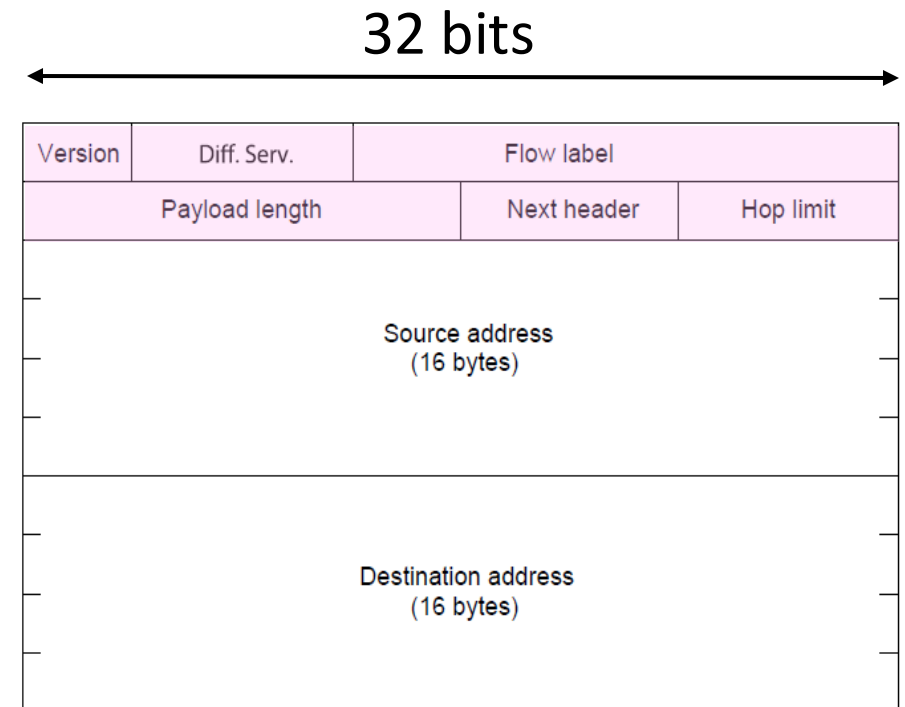
- Features large addresses
  - 128 bits, most of header
- New notation
  - 8 groups of 4 hex digits (16 bits)
  - Omit leading zeros, groups of zeros

Ex: 2001:0db8:0000:0000:0000:ff00:0042:8329  
→ 2001:db8::ff00:42:8329



# IPv6 (2)

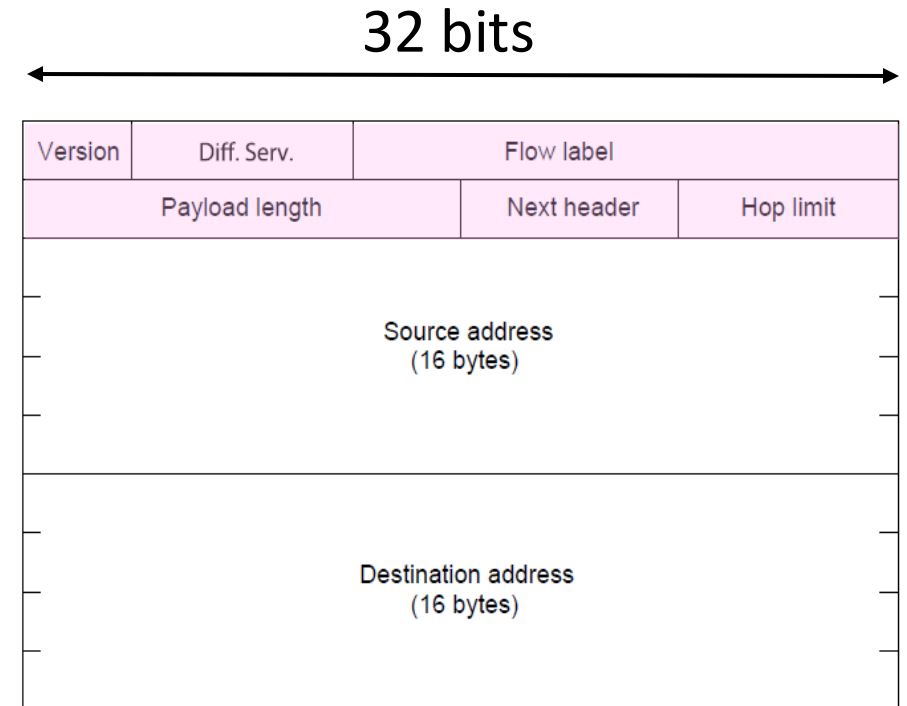
- Lots of other changes
  - Only public addresses
    - No more NAT!
  - Streamlined header processing
  - Flow label to group of packets
  - IPSec by default
  - Better fit with “advanced” features (mobility, multicasting, security)





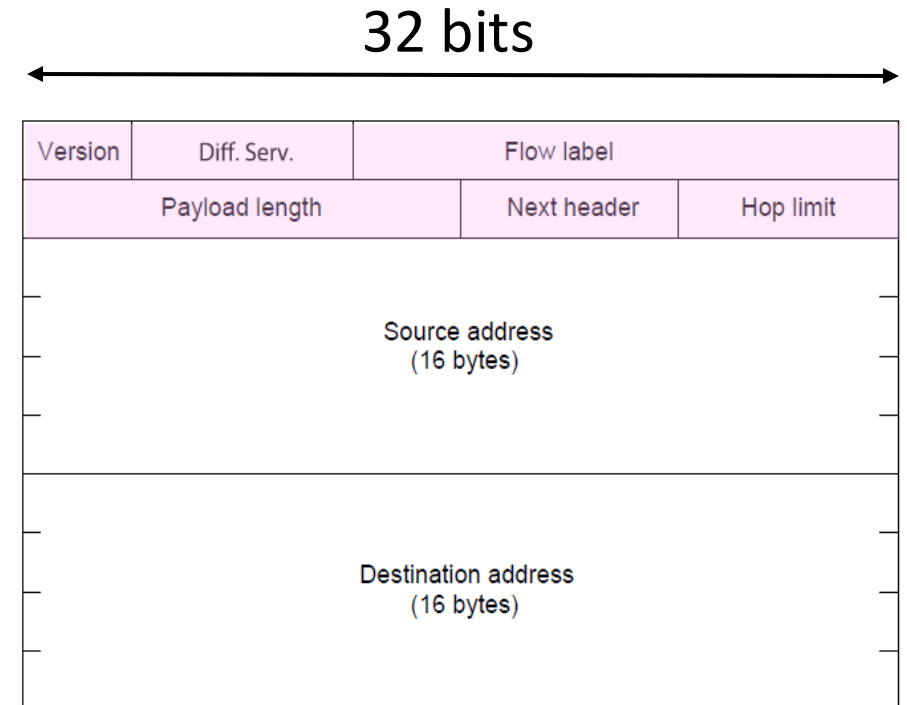
# IPv6 (3)

- Does this fix ARP?
- Does this fix DHCP?
- Does this fix NAT?



# IPv6 (3)

- Does this fix ARP? No: NDP
- Does this fix DHCP? No: SLAAC
- Does this fix NAT? Yes!



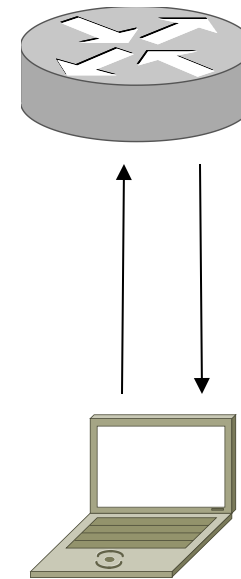
# Neighbor Discovery Protocol

- Uses ICMPv6
- DHCP Functions:
  - Router discovery (133)/advertisement (134)
- ARP Functions:
  - Neighbor discovery (135)/advertisement (136)

# Stateless Autoconfiguration (SLAAC)

- Replaces DHCP (sorta...)
- Uses ICMPv6
- Process:
  - Send broadcast message
  - Get prefix from router
  - Attach MAC to router Prefix /w some math
  - 48 bit → EUI-64 format

Address: 2000:1234:5678::1001/64  
Prefix: 2000:1234:5678::/64



MAC: 0200:1234:5678 → 0000:12FF:FE34:5678  
Address: 2000:1234:5678::12FF:FE34:5678

# IPv6 Transition

- The Big Problem:
  - How to deploy IPv6?
  - Fundamentally incompatible with IPv4
- Dozens of approaches proposed
  - Dual stack (speak IPv4 and IPv6)
  - Translators (convert packets)
  - Tunnels (carry IPv6 over IPv4)