Welcome! An aside about addresses…

- Throughout our discussion of the link layer, we kept referring to "addresses", both source and destination
  - Why are link layer addresses important?
    - *What function do they fulfill in the network?*
  - Given out at the factory (so not globally unique). What else could one do?

They Allow Distinguising Hosts on the LAN!

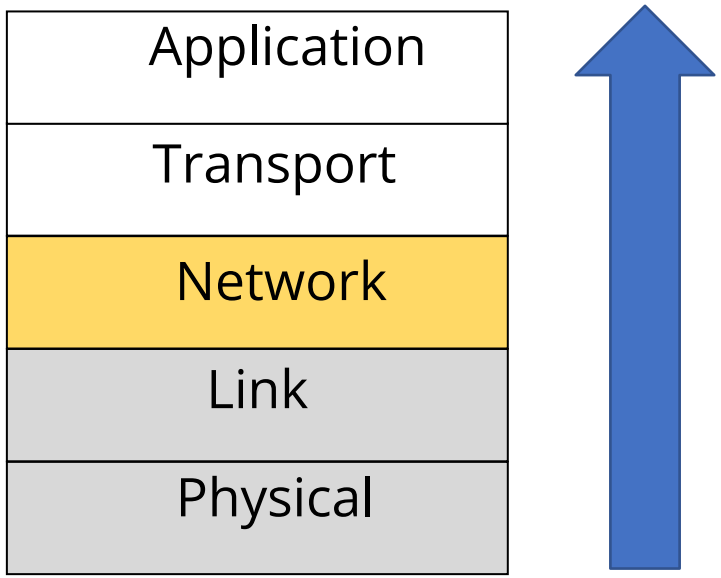
Important that they are *locally* unique!

Can also assign randomly, with some logic to try and detect duplicates. ++privacy, --reliability, works well enough in practice because not many devices on L2 nets so low enough probability of collision

Where we are in the Course

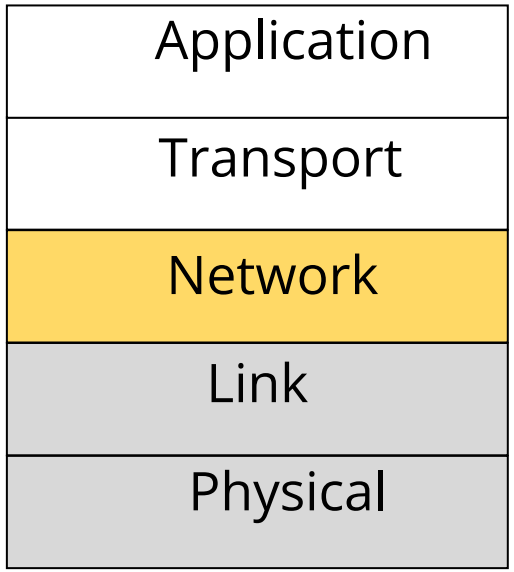
Today: Scaling beyond a single network… internets!

- Moving up the stack to the network layer!



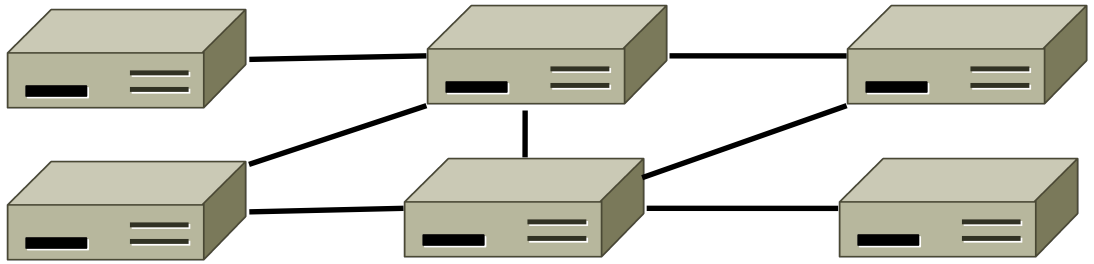
Network Layer

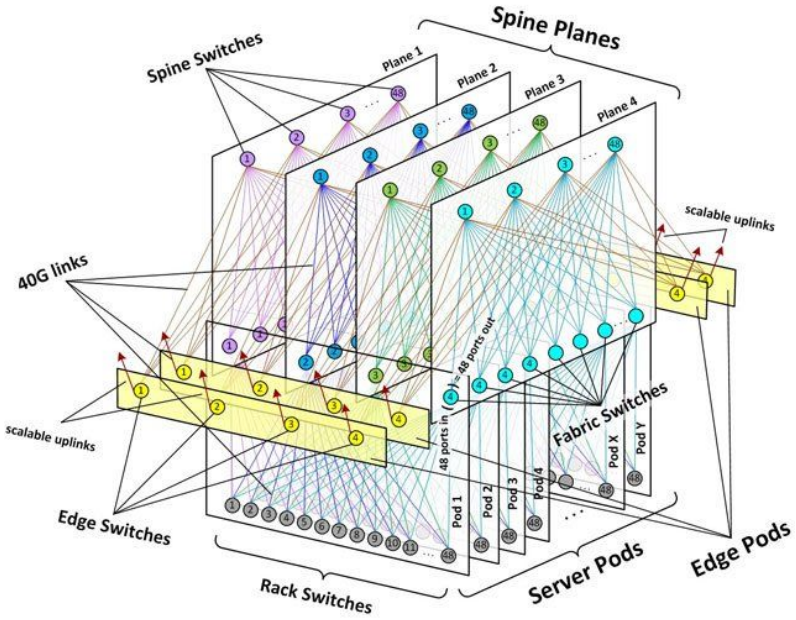
- How to connect different link layer networks
  - Routing as the primary concern



Why do we need a Network layer?

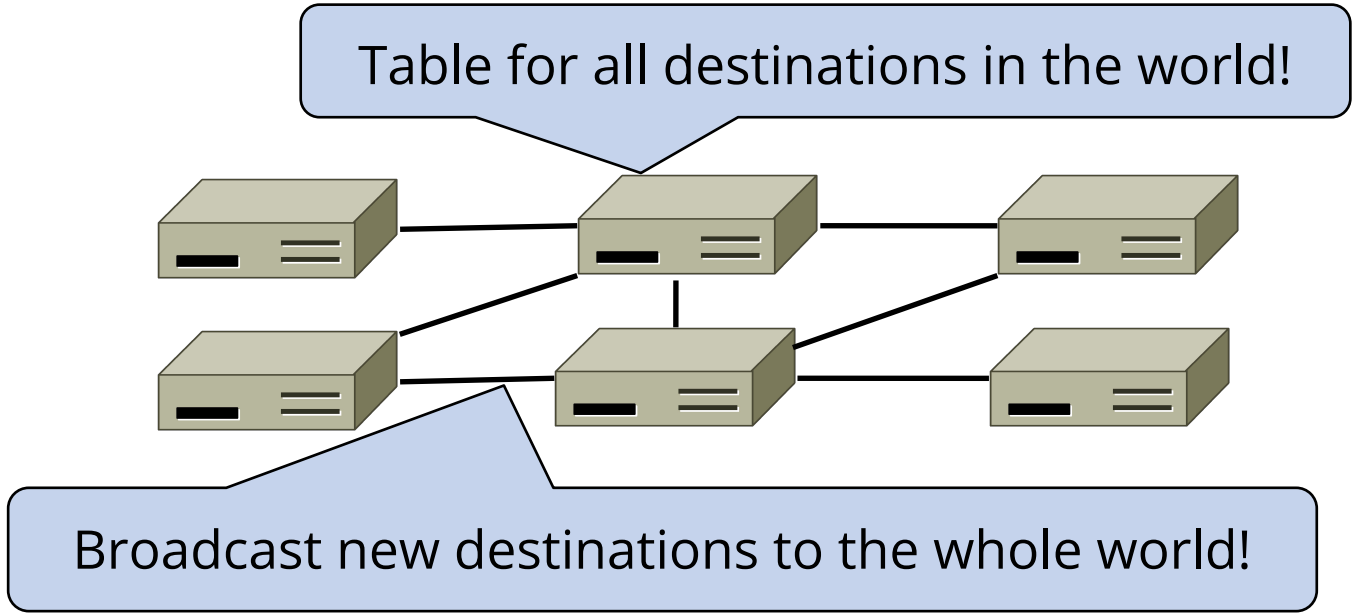
- We can already build networks with links and switches and send frames between hosts …
- And with SDN we can even scale up (kinda)!





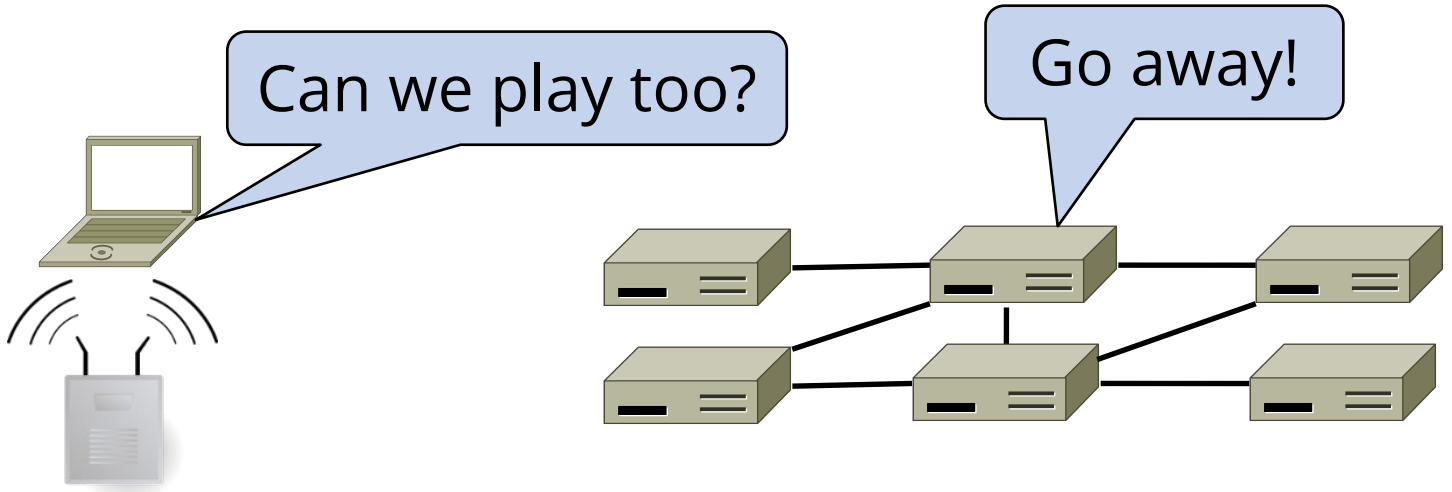
Shortcomings of Switches

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



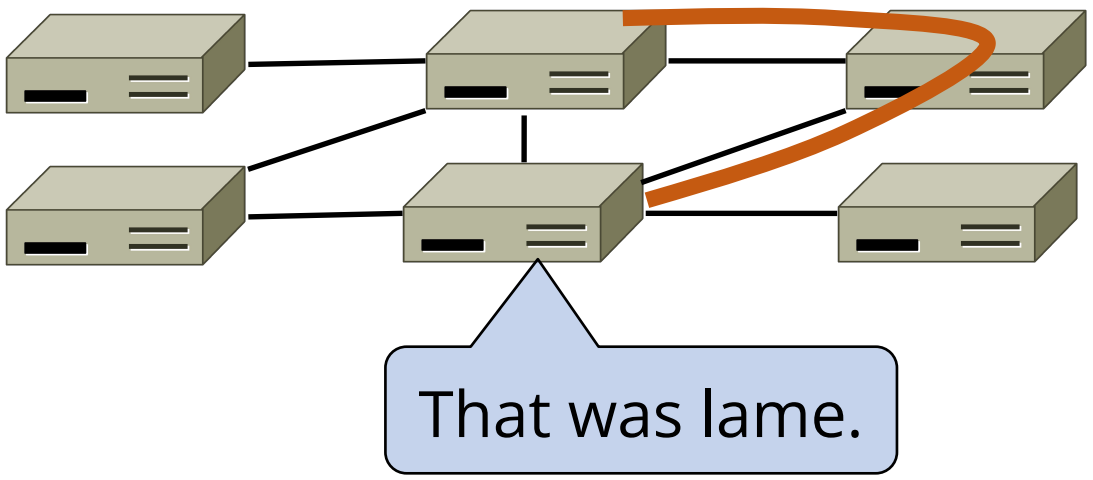
Shortcomings of Switches

2. Don’t work across more than one link layer technology
   - Modern hosts on Ethernet + LTE + 5G + 802.11 + PON + …



Shortcomings of Switches

3. Don’t give much traffic control
   - Want to plan routes / bandwidth
     - (SDN helps with this for networks large enough to justify SDN)



Network Layer Approach

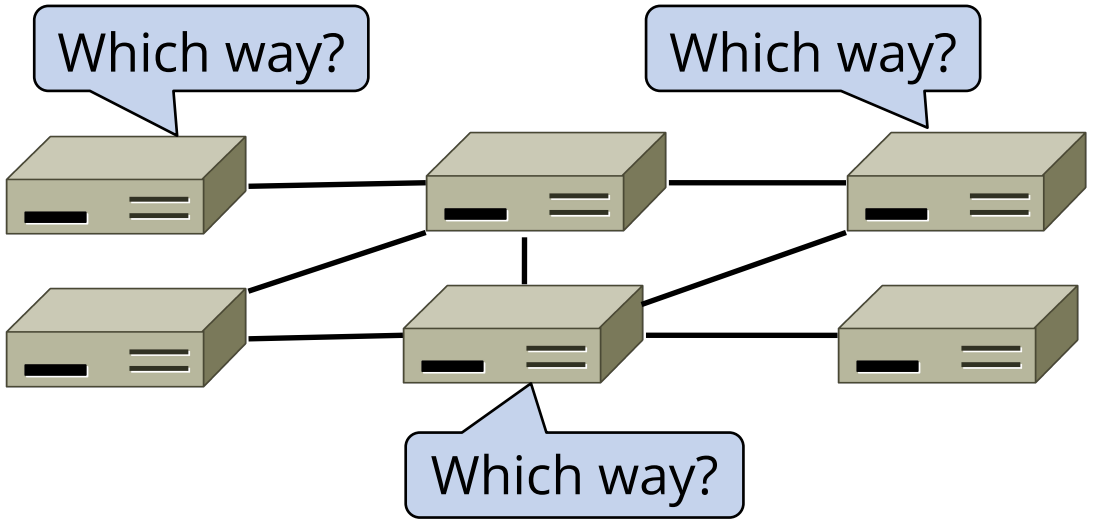
1. Scaling
   - Hierarchy, in the form of prefixes
2. Heterogeneity
   - IP for internetworking over many different links
3. Bandwidth Control
   - Lowest-cost routing
   - Later QOS (Quality of Service)

Learning Objectives

- Network service models
  - Distinguish the pros and cons between datagrams (packets), and virtual circuits
- IP Forwarding + Addressing (Internet Protocol)
  - Explain the rationale for internetworking and how it is different from expanding a layer 2 network
  - Read and interpret IP addresses and subnets for IPv4 and IPv6
  - Examine a forwarding table and understand the forwarding decisions made via longest matching prefix
  - Know the roles of, and how to configure and debug issues with, ARP, DHCP, ICMP, and Fragmentation
  - Enumerate the scaling constraints faced with IPv4, and some solutions for addressing those constraints (IPv6, NAT)
  - Configure and debug NAT systems, and enumerate other types of “middleboxes”
- Routing Algorithms… future :)

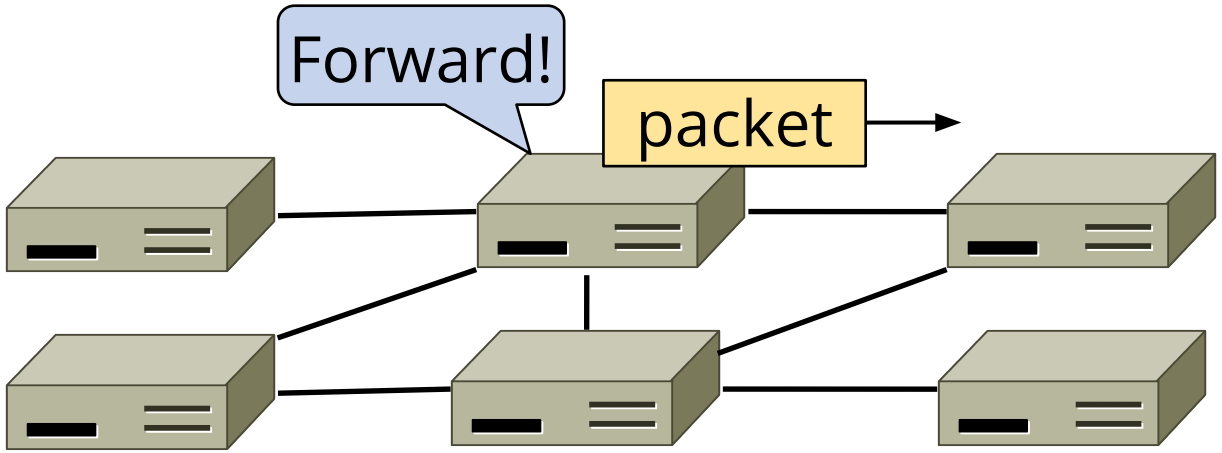
Aside: Routing vs. Forwarding

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



Aside: Routing vs. Forwarding

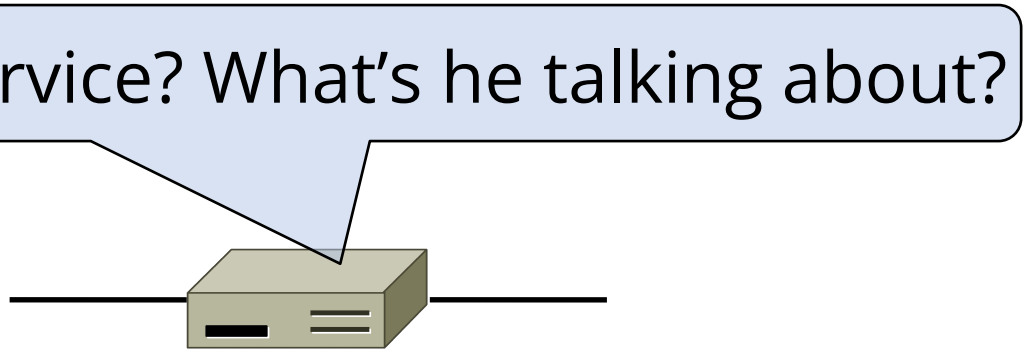
- Forwarding is the process of sending a packet
  - Node process (local) and fast



Networking Service Models

Networking Service Models

- What kind of service does the Network layer provide to the Transport layer?
  - "What's it's API contract?"
- How is it implemented at routers?



Two Network Service Models

- Datagrams, or connectionless service
  - Like postal letters
  - (IP as an example)

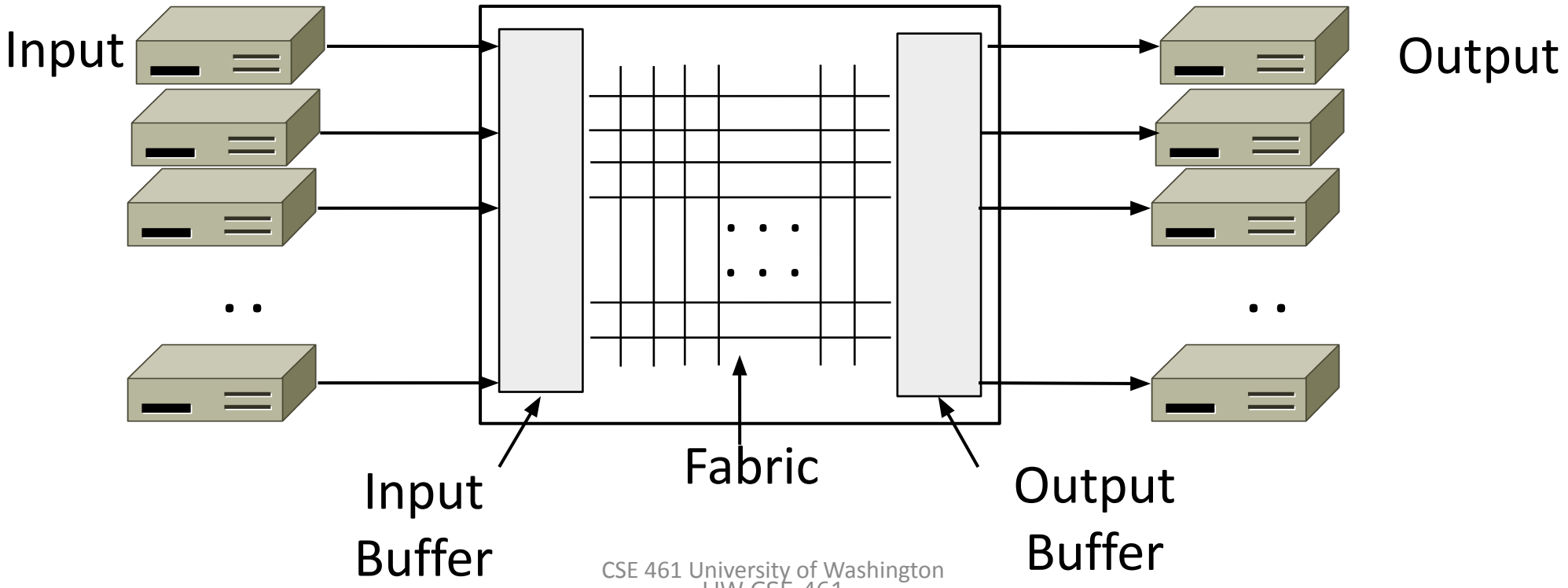
- Virtual circuits, or connection-oriented service
  - Like a telephone call

Store-and-Forward Packet Switching

- Both models are commonly 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

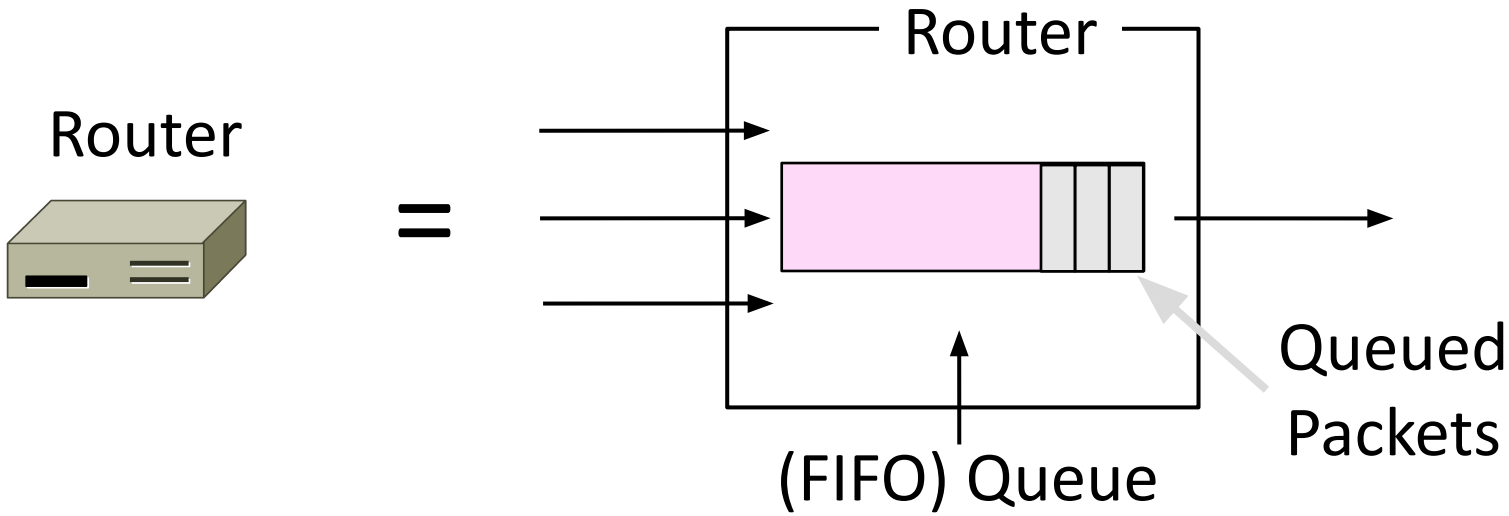
Store-and-Forward

- Switching element has internal buffering for contention



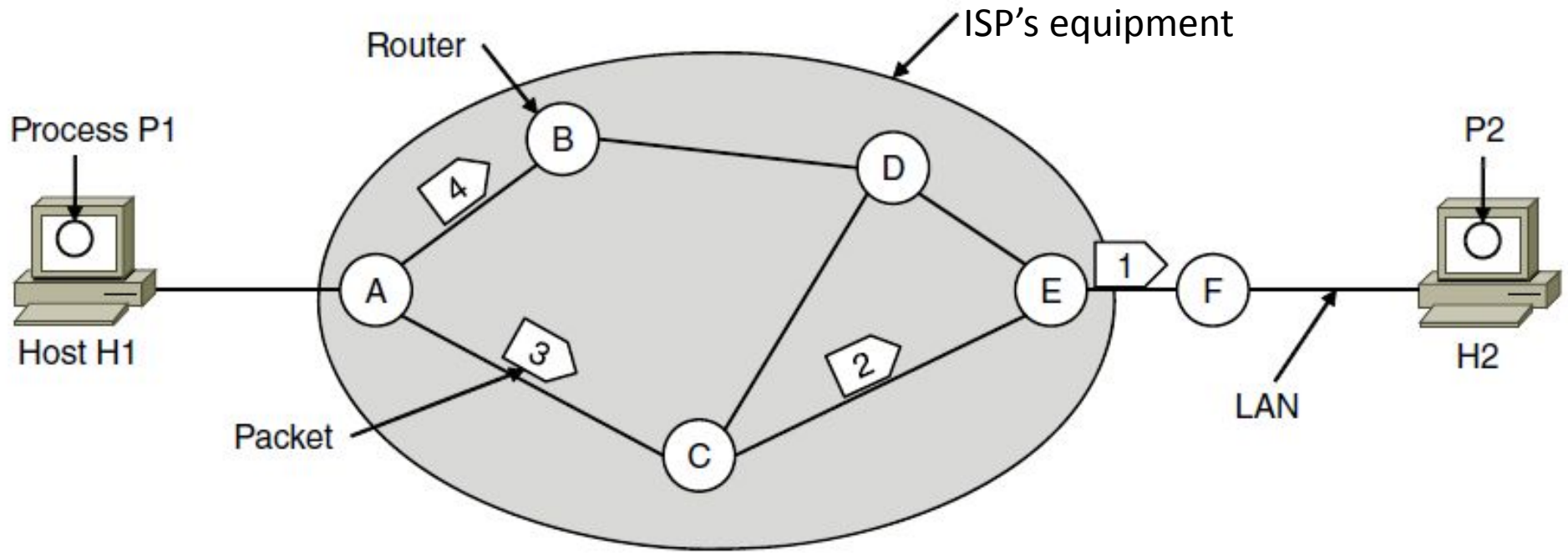
Store-and-Forward

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



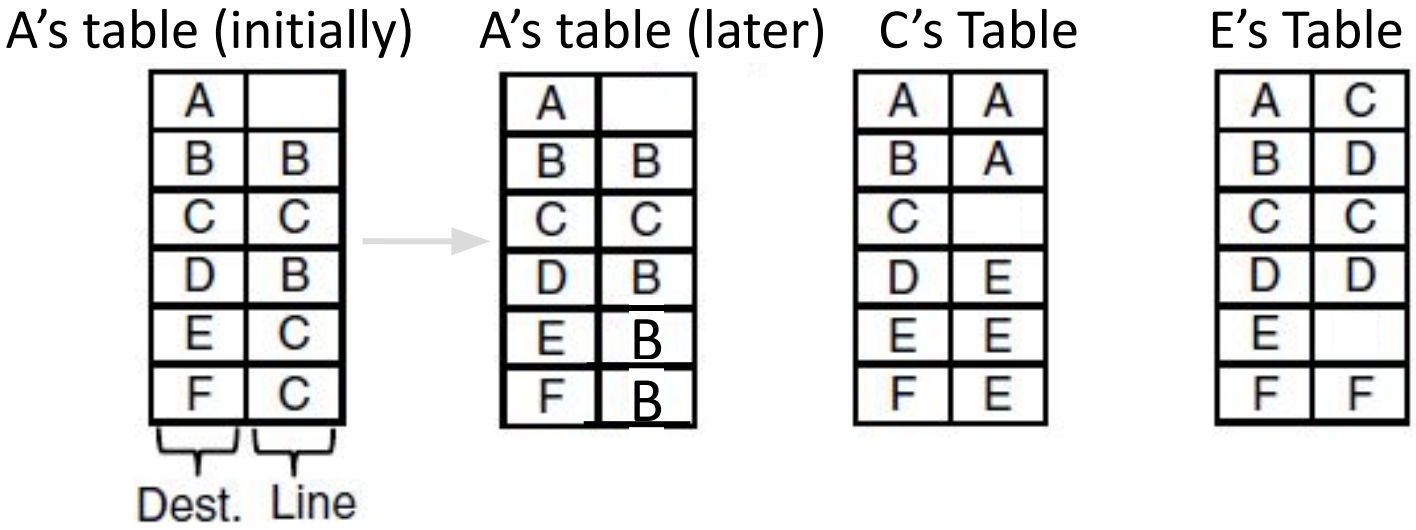
Datagram Model

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



Datagram Model

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



Dest.	Line
A	
B	B
C	C
D	B
E	C
F	C

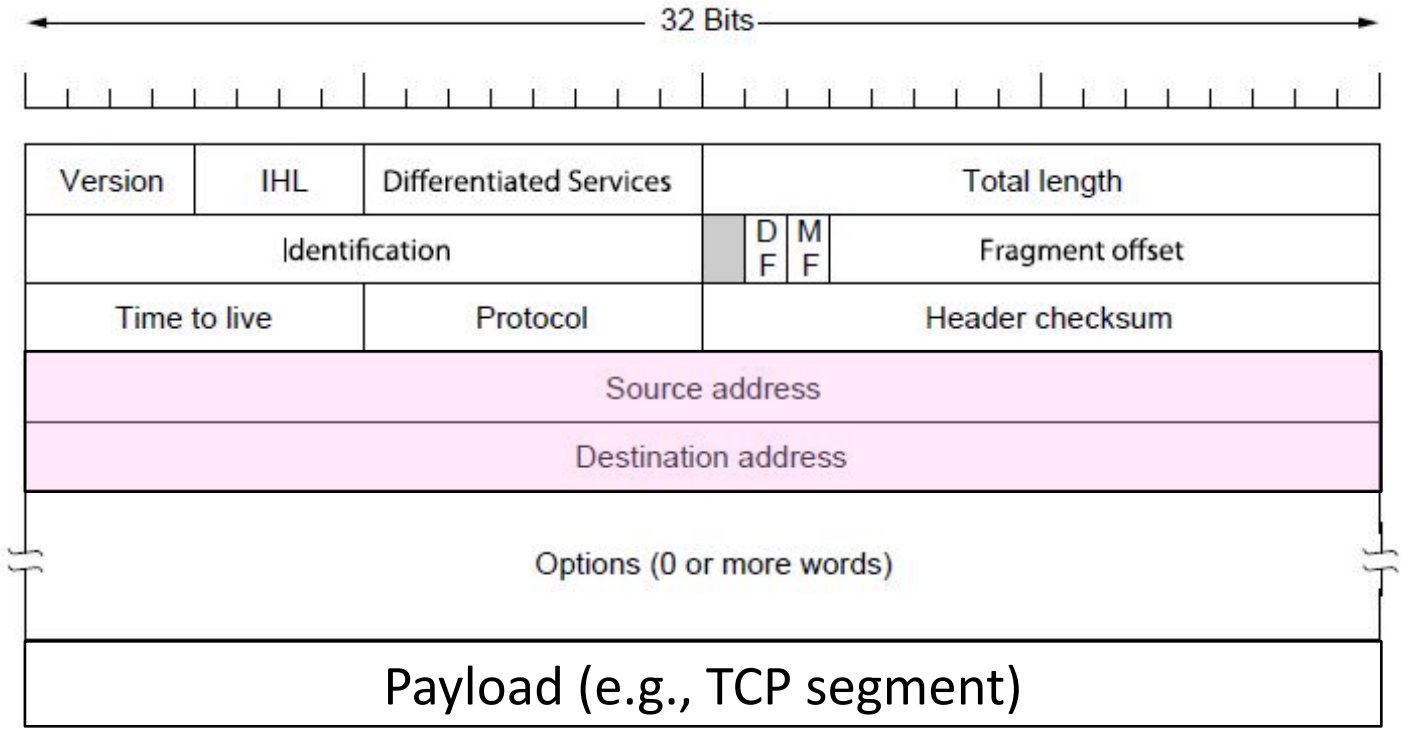
Dest.	Line
A	
B	B
C	C
D	B
E	B
F	B

Dest.	Line
A	A
B	A
C	
D	E
E	E
F	E

Dest.	Line
A	C
B	D
C	C
D	D
E	
F	F

Datagram Example: IP (Internet Protocol)

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

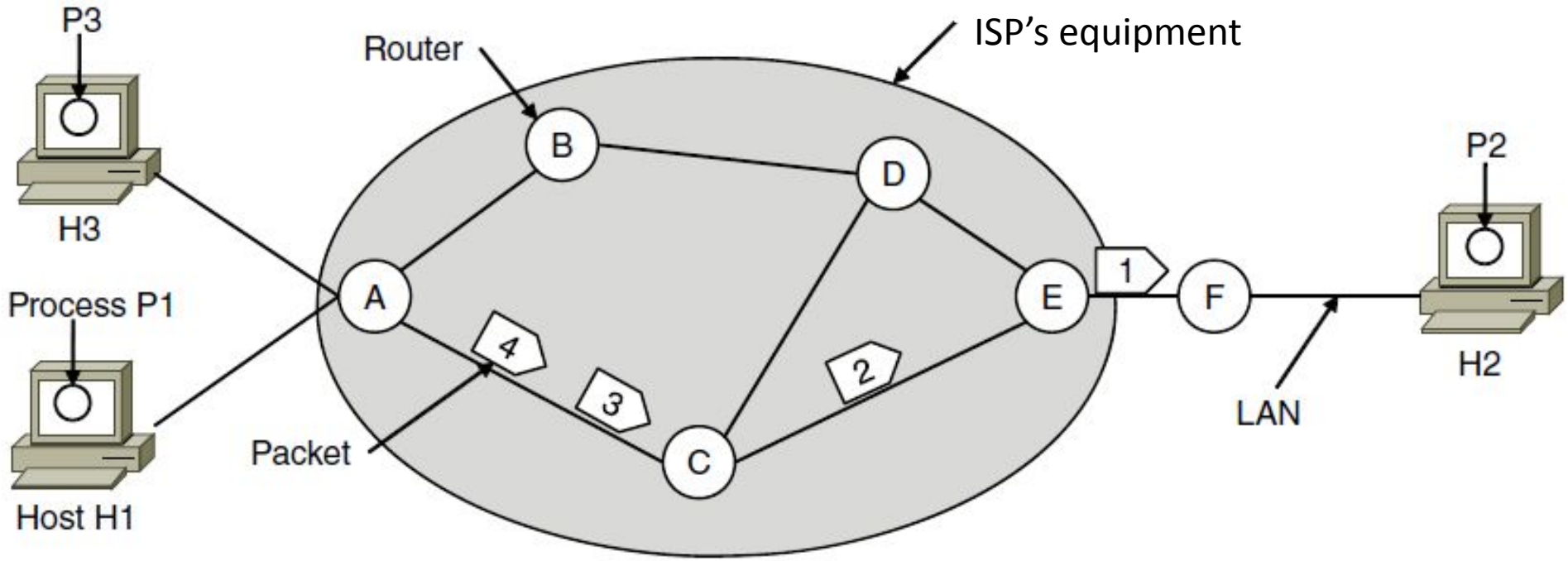


Virtual Circuit Model

- 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 that no bandwidth need be reserved; statistical sharing of links

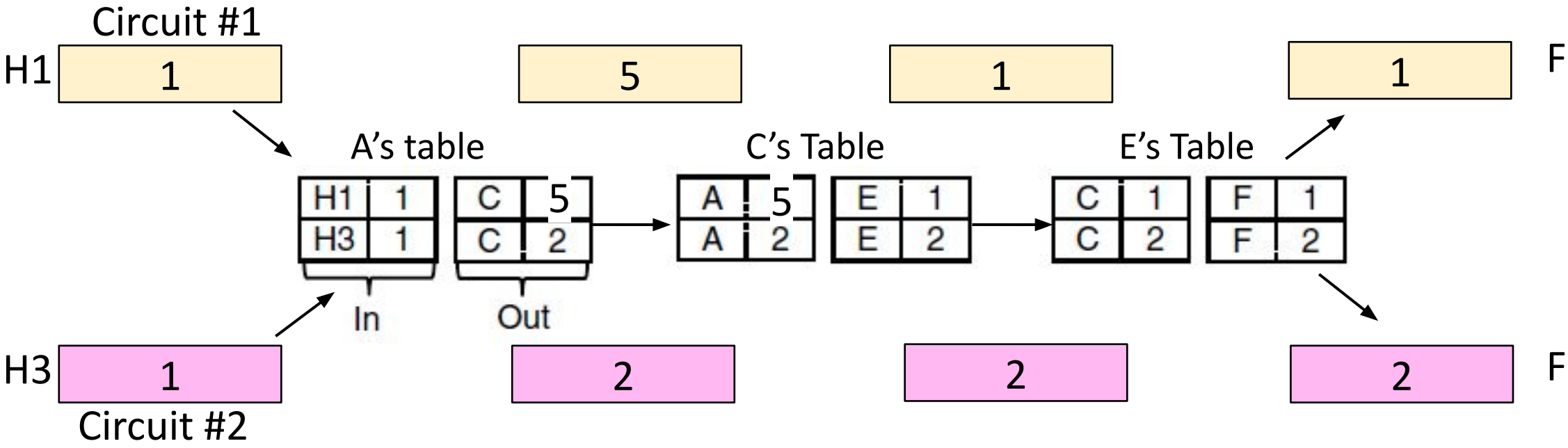
Virtual Circuits (2)

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



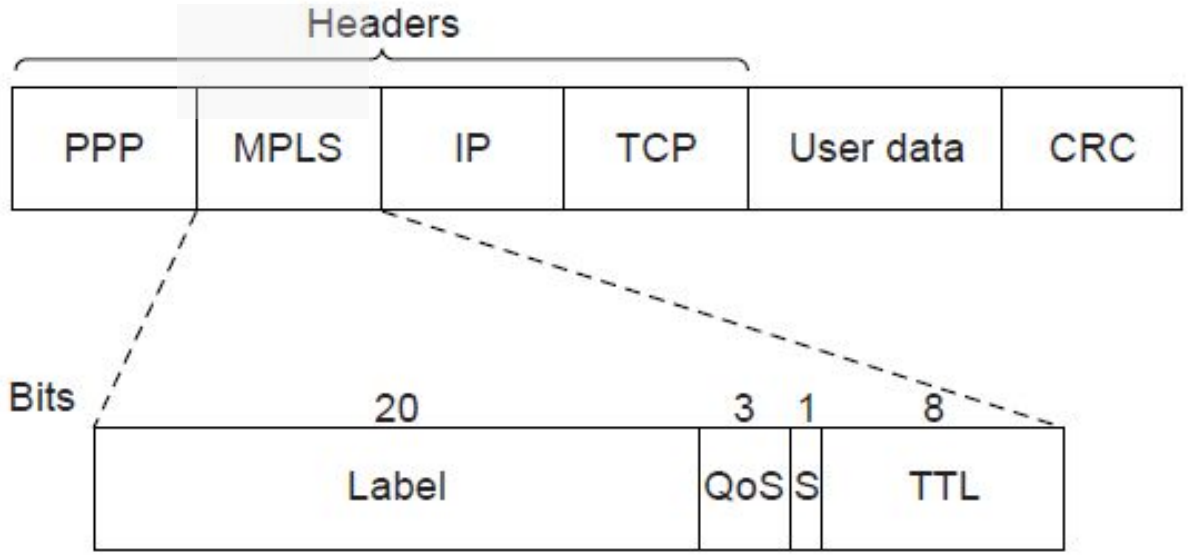
Virtual Circuits (4)

- Each router has a forwarding table keyed by circuit
  - Gives output line and next label to place on packet



MPLS (Multi-Protocol Label Switching, §4.4)

- A virtual-circuit like technology widely used by ISPs
  - ISP sets up circuits inside their backbone ahead of time
  - ISP adds MPLS label to IP packet at ingress, undo at egress



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

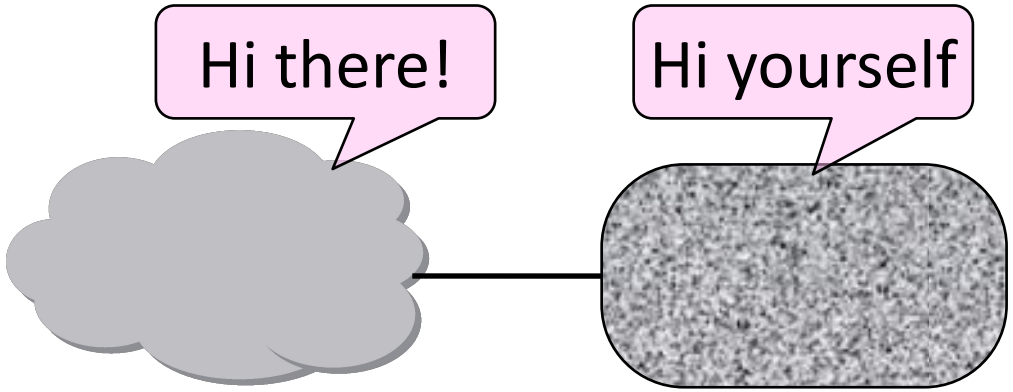
Source Routing (§3.1.3)

- Allow the source to embed the forwarding instructions for each hop in the packet itself
- Similar to virtual circuits in that the path is pre-determined
- Similar to datagrams in that the addresses are not link-specific
- Included as an extension you can use in IPv6!

Internetworking (IP)

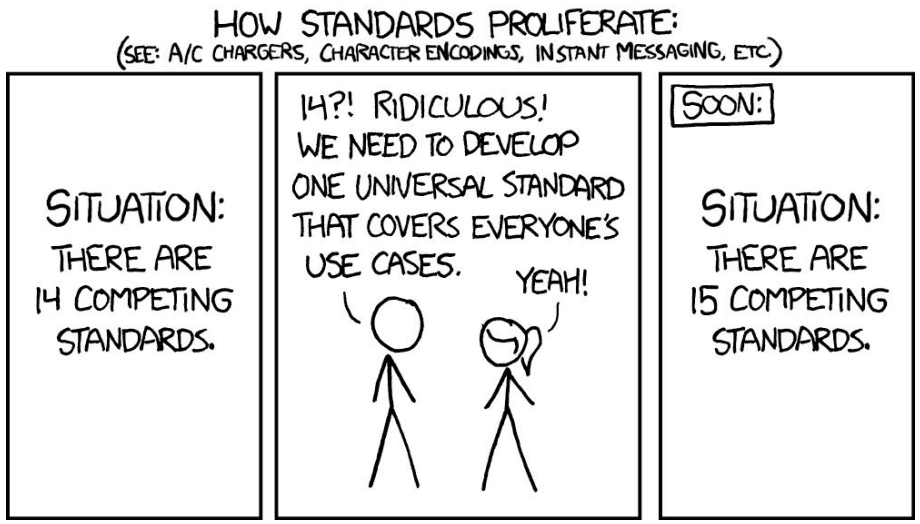
Topic

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



How Networks May Differ

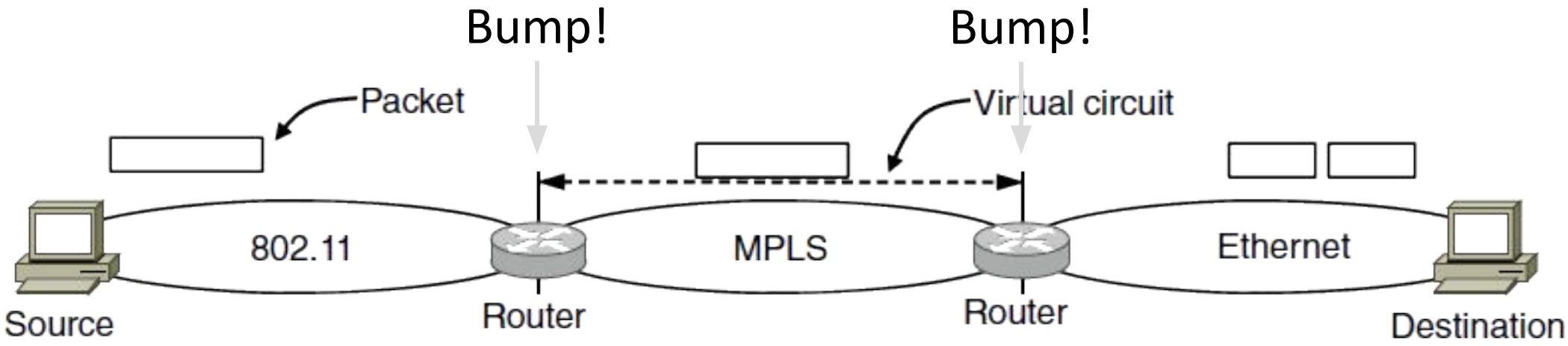
- Basically, in a lot of ways:
  - Service model (datagrams, VCs)
  - Addressing (what kind)
  - QOS (priorities, no priorities)
  - Packet sizes
  - Security (whether encrypted)



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

Connecting Datagram and VC networks

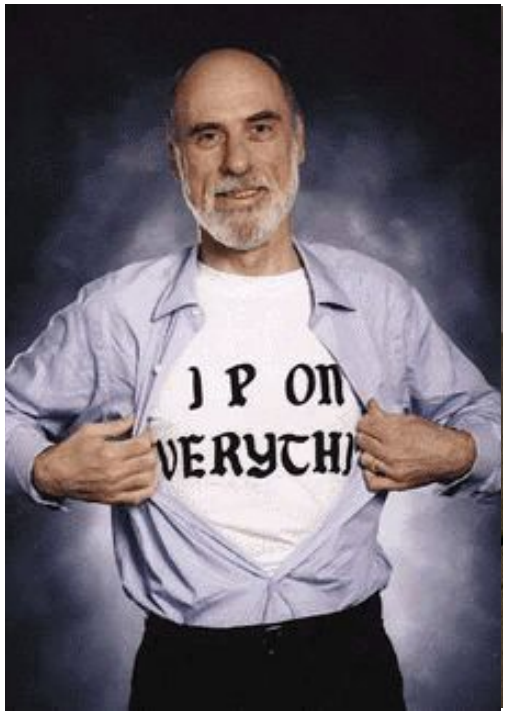
- An example to show that it's not so easy
  - Need to map destination address to a VC and vice-versa
  - A bit of a "road bump", e.g., might have to set up a VC



Internetworking – Cerf and Kahn

- 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

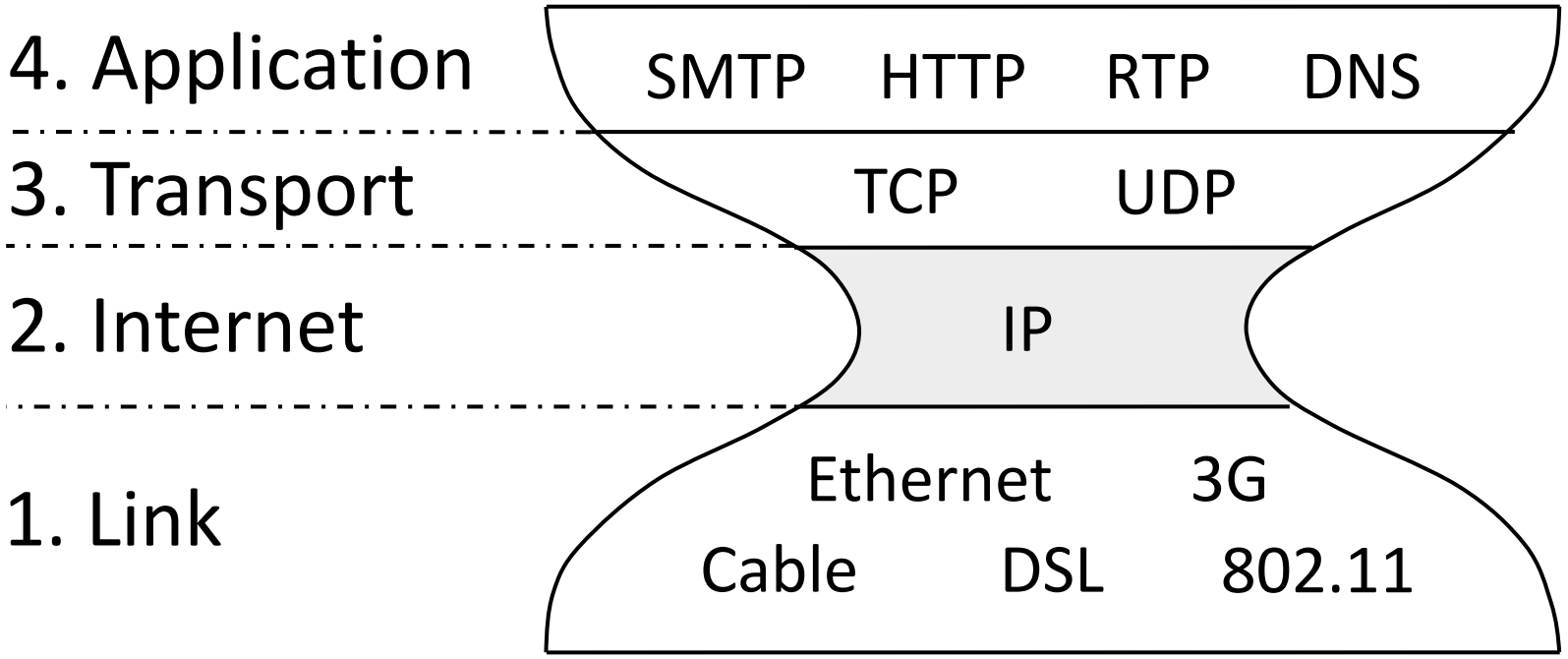
Vint Cerf



Bob Kahn

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 internetwork to support
- Pushes IP to be a "lowest common denominator"
  - Asks little of lower-layer networks
  - Gives little as a higher layer service

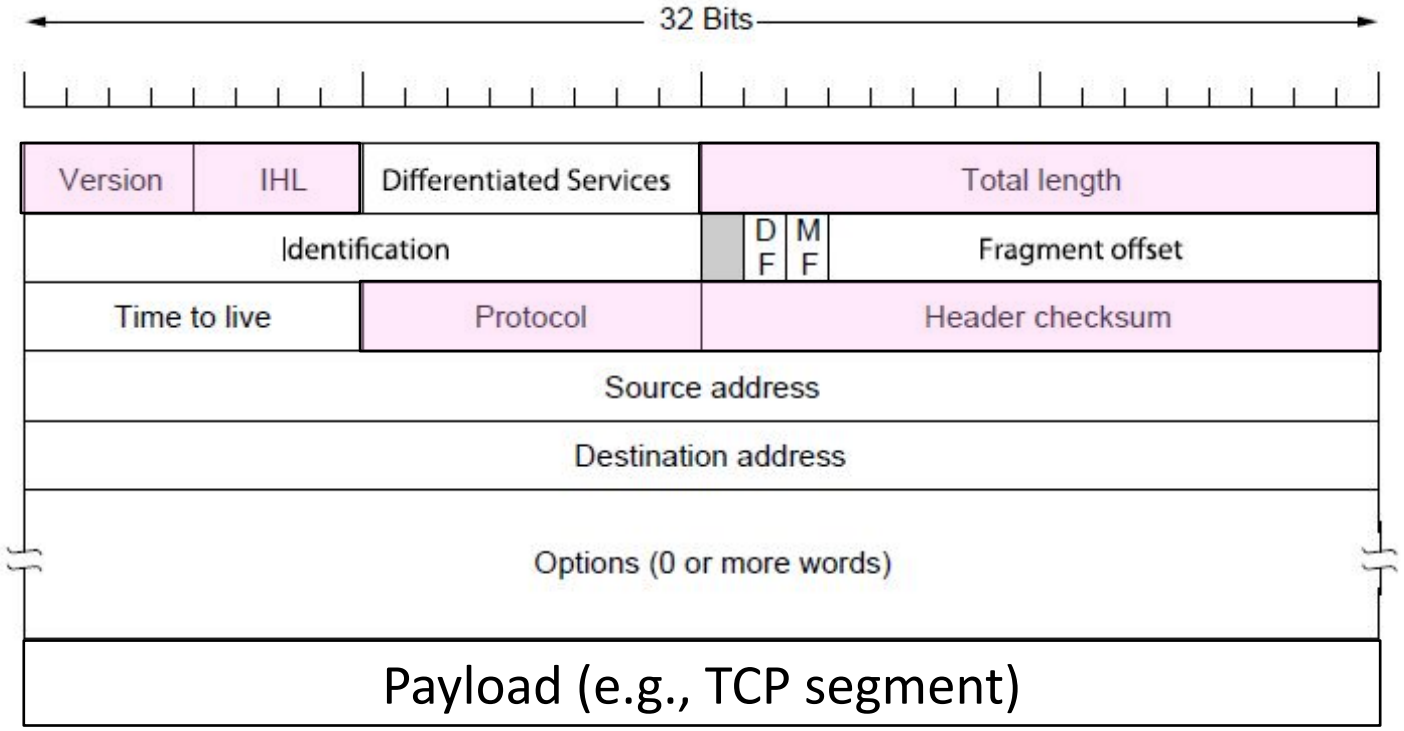
Instead of trying to support every feature of existing networks, took the opposite approach and only supported common features.

IP as a Lowest Common Denominator

- provides “best effort”: no guarantees on delivery
- connectionless: no state

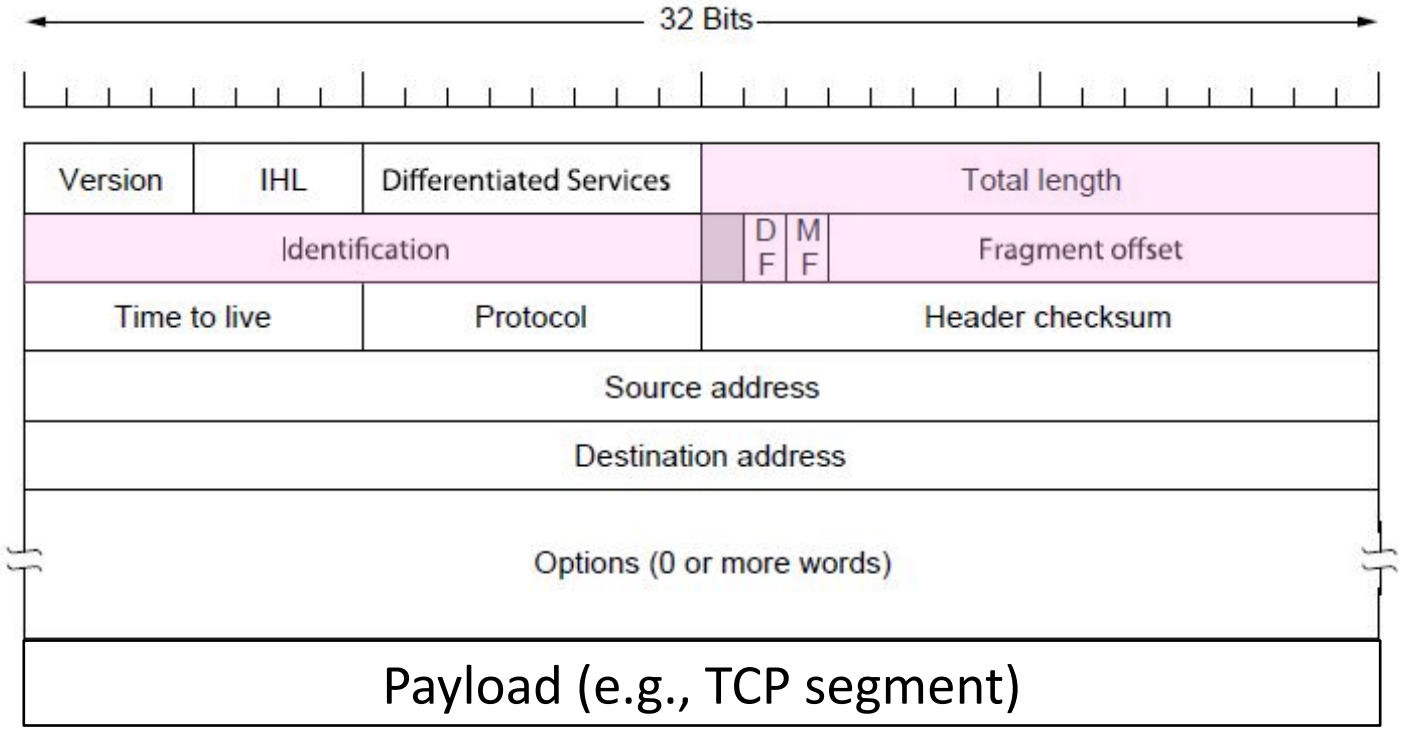
IPv4 (Internet Protocol)

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



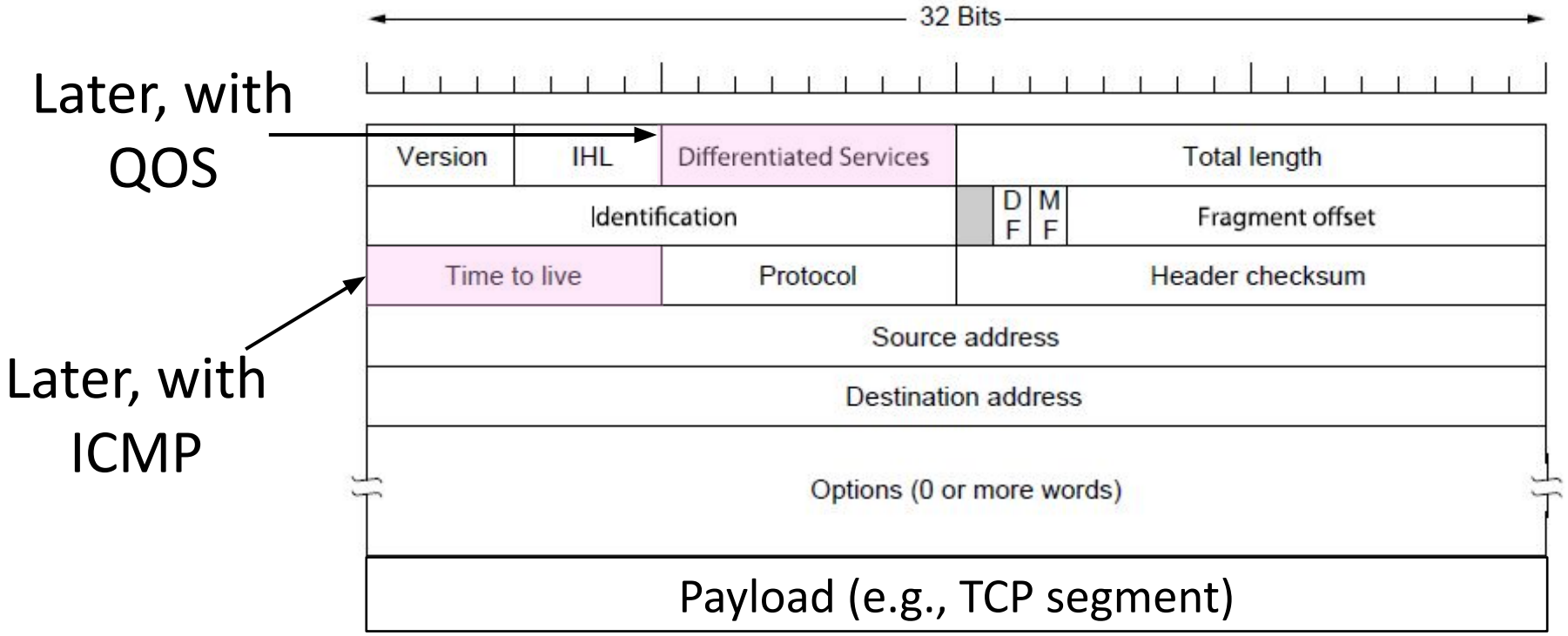
IPv4 (2)

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



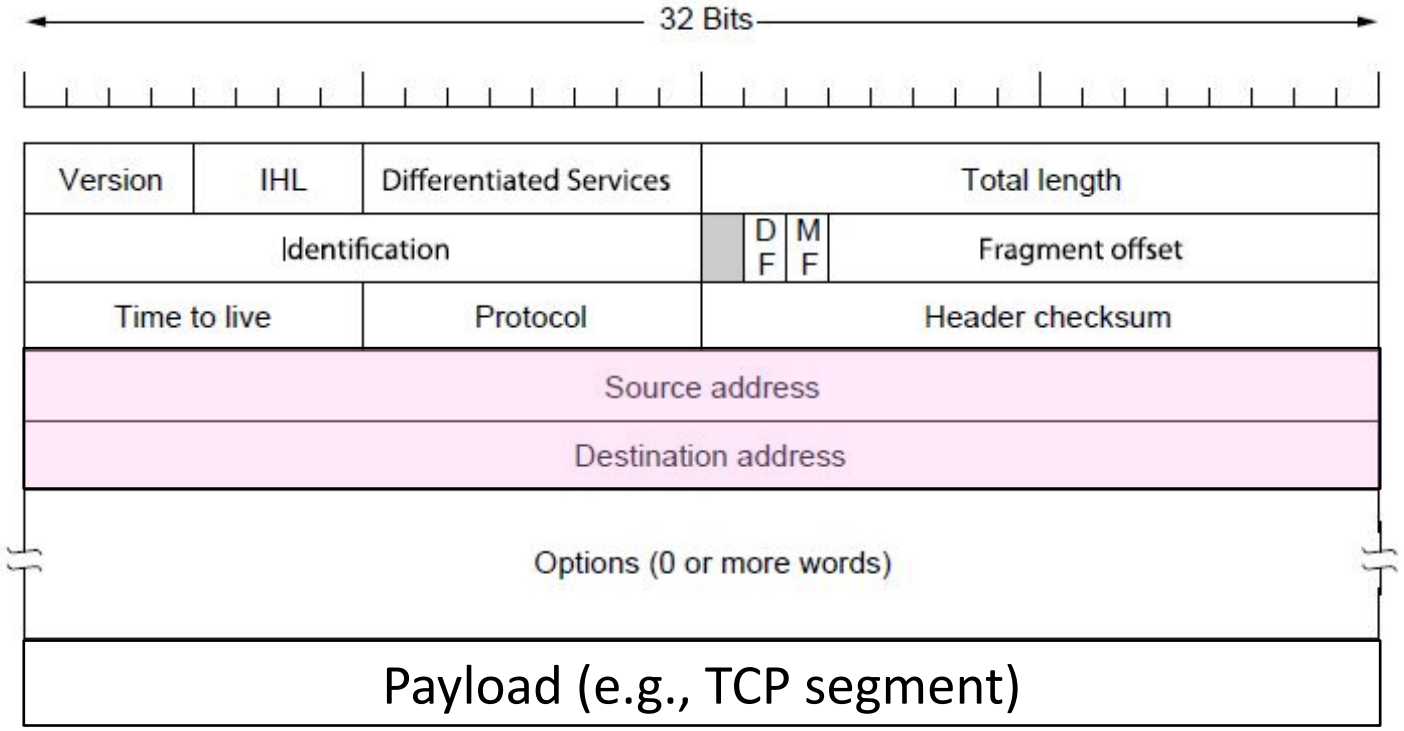
IPv4 (3)

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



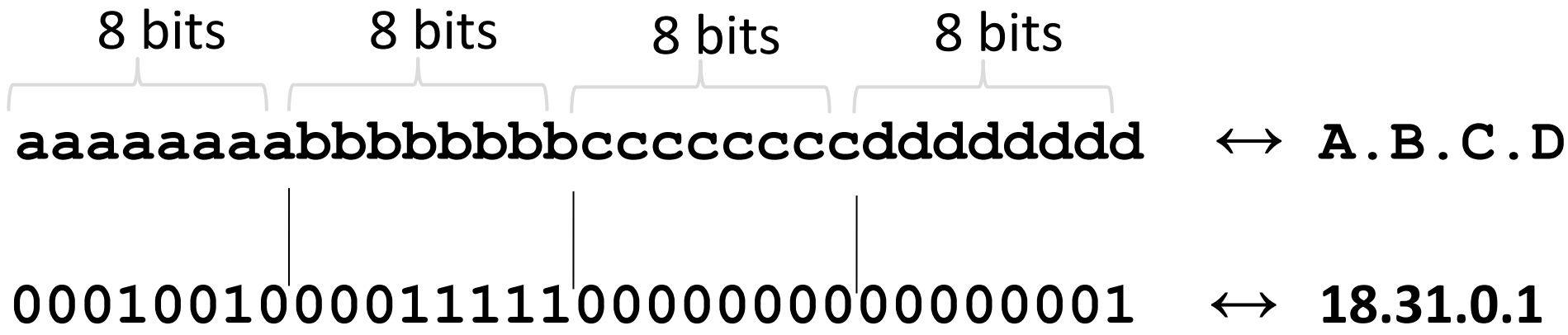
IPv4 (4)

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



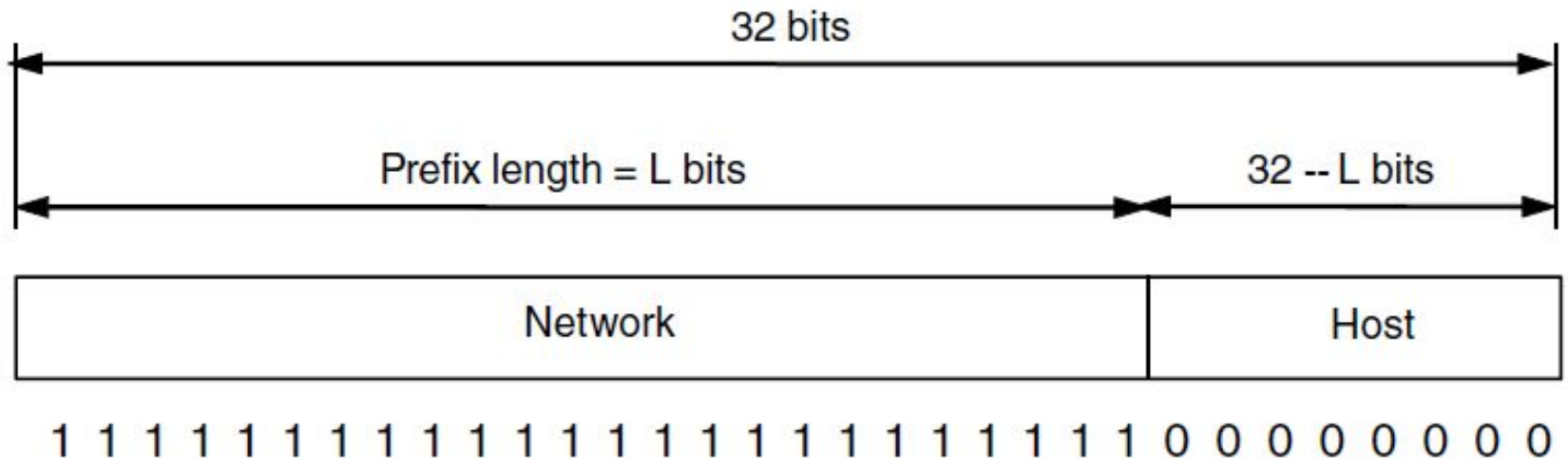
IP Addresses

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



IP Prefixes

- Addresses are allocated in blocks called <u>prefixes</u>
  - 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

`00010010|00011111|00000000|xxxxxxxx` ↔ **18.31.0.0/24**

`10000000|00001101|xxxxxxxx|xxxxxxxx` ↔ `128.13.0.0/16`

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
  - [illegible]ne address

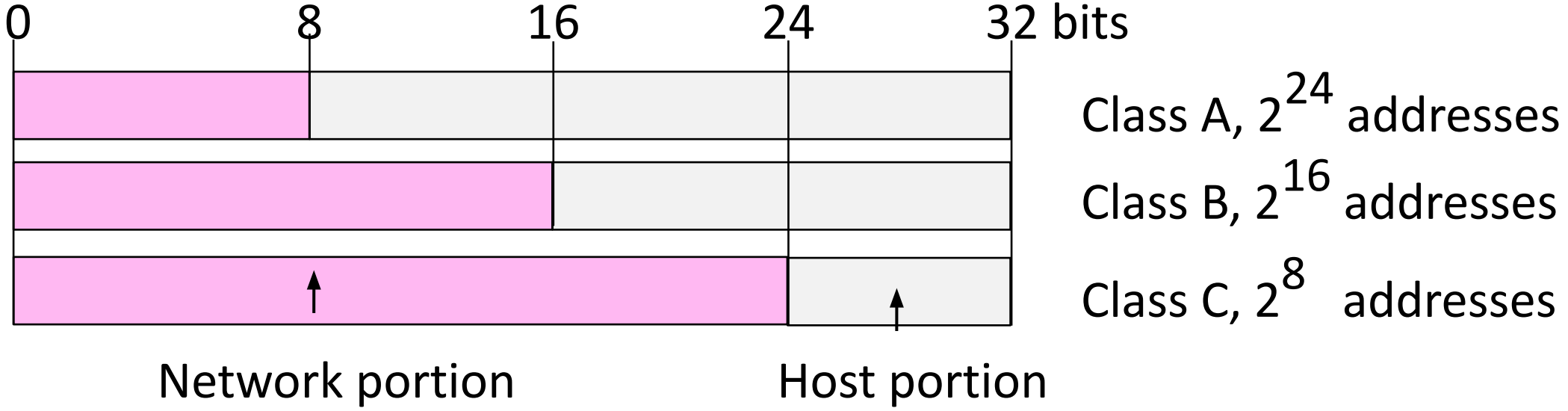
Our Seattle Community Network was allocated 23.146.168.0/24 by ARIN!

00010010|00011111|00000000|xxxxxxxx ↔ **18.31.0.0/24**

10000000|00001101|xxxxxxxx|xxxxxxxx ↔ **128.13.0.0/16**

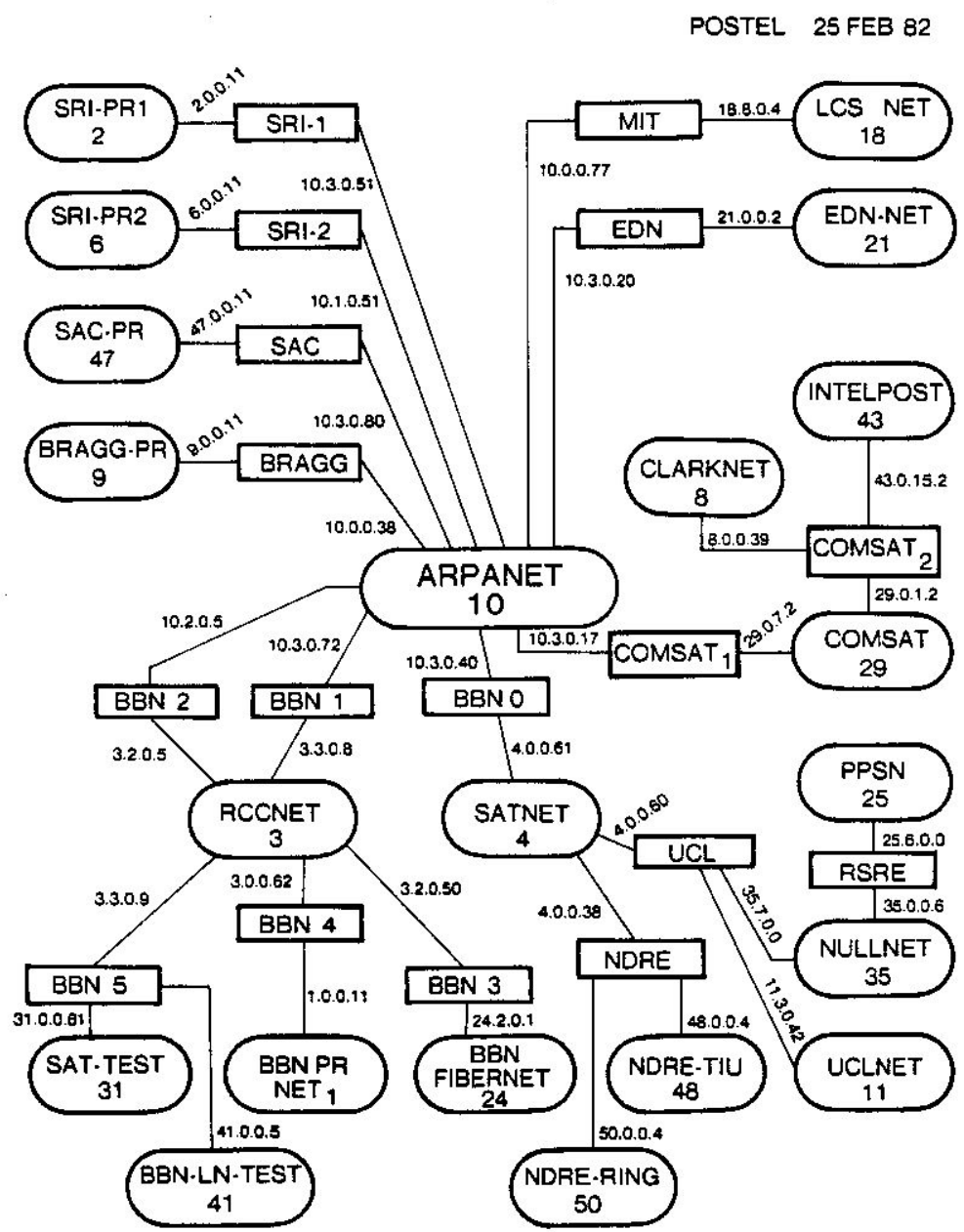
Classful IP Addressing

- 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

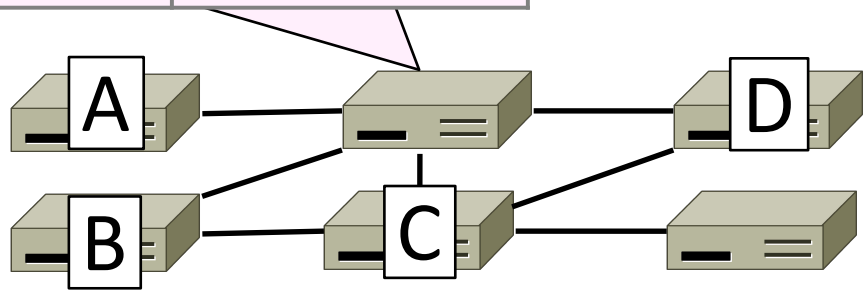
- This is an ARPANet assignment.



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
192.24.0.0/18	D
192.24.12.0/22	B



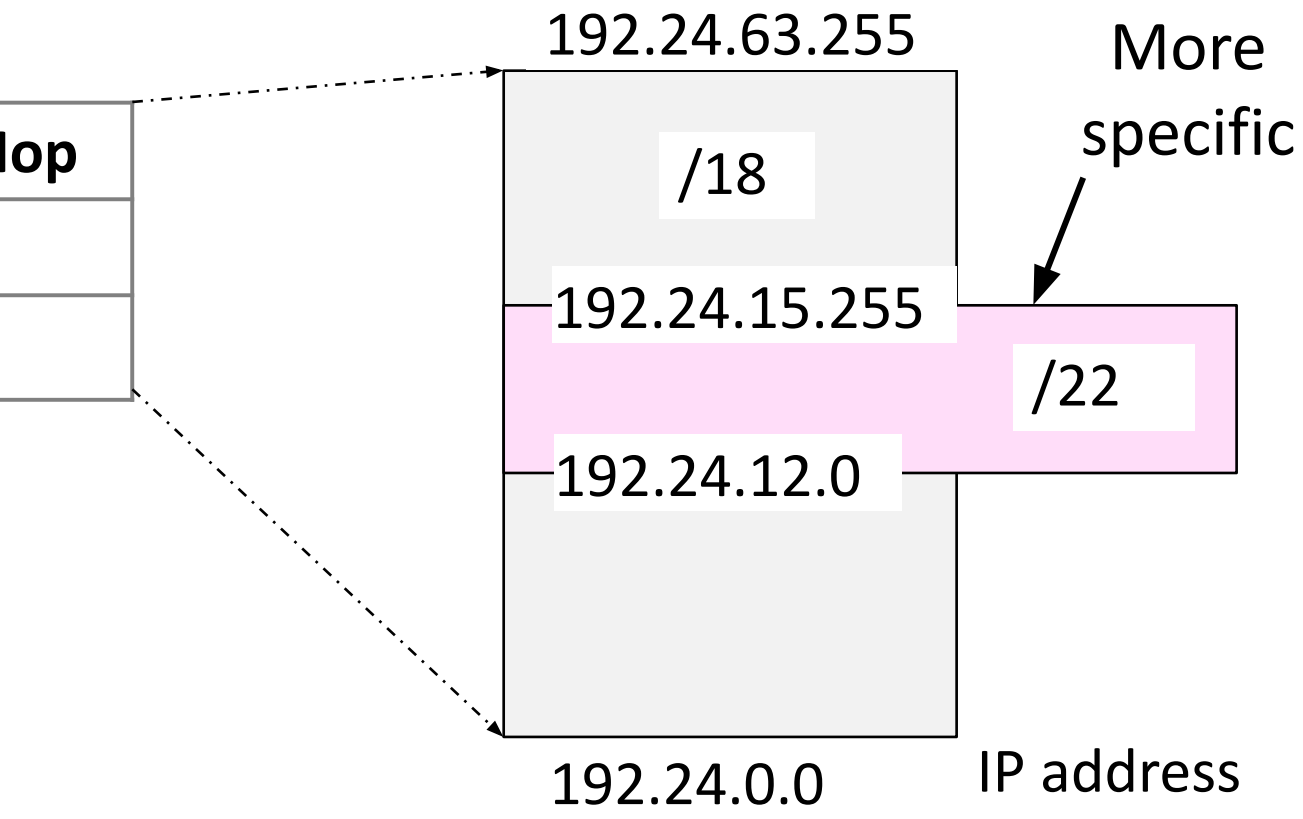
Longest Matching Prefix

- 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

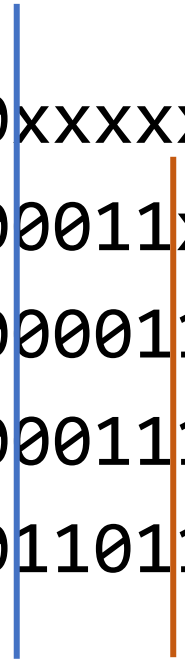
Prefix	Next Hop
192.24.0.0/18	D
192.24.12.0/22	B



192.24.6.0
192.24.14.32
192.24.54.0

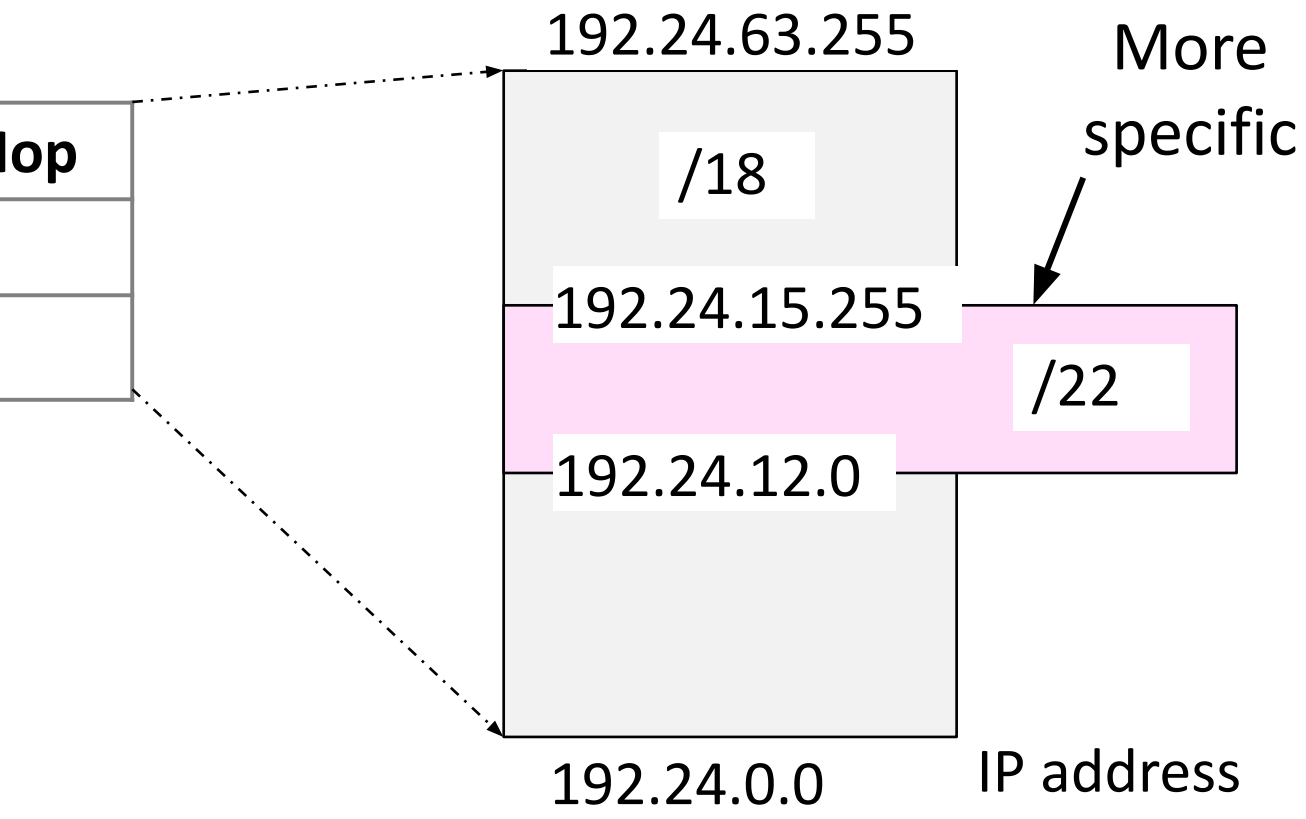
IP Address Work Slide:

- 192.24.0.0/18 (D) = 192.00011000.00xxxxxx.xxxxxxxx
- 192.24.12.0/22 (B)= 192.00011000.000011xx.xxxxxxxx
- 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

Prefix	Next Hop
192.24.0.0/18	D
192.24.12.0/22	B



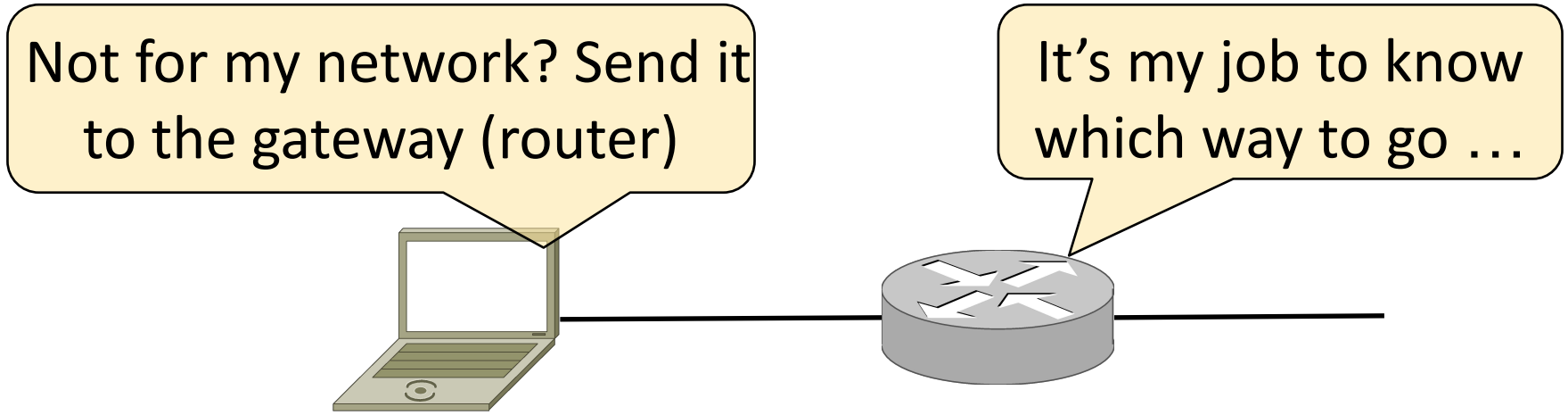
192.24.6.0 → D

192.24.14.32 → B

192.24.54.0 → D

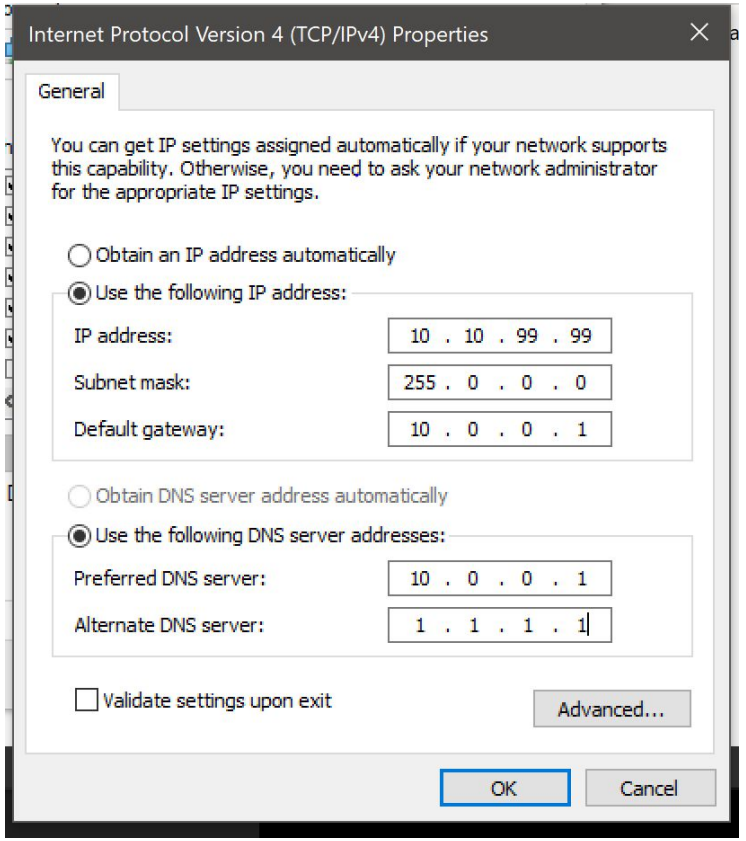
Host/Router Distinction

- 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
  - Prefix / 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 gateway 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
  - "CIDR" Classless Inter-Domain Routing

- Lookup more complex than table
  - Used to be a concern for fast routers
  - Not an issue in practice these days

Classless Addressing and Interface Notation

- With variable length prefixes, need some way to compactly represent the configuration of a given interface on host
  - Could specify the address `192.24.12.17` and prefix `192.24.12.0/24`
  - Or address `192.24.12.17` and subnet mask `255.255.255.0`
- Since in any real network the address needs to be in the local prefix, can blend the notations into one compact form
  - Addr `192.24.12.17` + Prefix `192.24.12.0/24` □ **`192.24.12.17/24`**
- Known as “Interface Notation”
  - Commonly seen in practice
  - Technically neither an address nor a prefix :)

Some tips

1. AND the subnet mask to get the network id
2. OR the subnet mask to get the broadcast address
3. Question:
    a. What is the network id for this network: 192.168.4.12/20
        i. What is the subnet mask?
        ii. What is the broadcast address?
        iii. What are the useable IPs?
    b. What is the network id for this network: 192.168.4.12/31

Issues?

- What's still not solved?

Issues?

- Where does this break down?

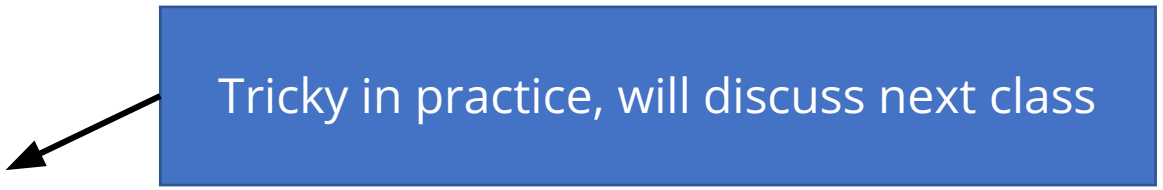
Finding Link nodes (ARP)

Bootstrapping (DHCP)

~~Really big packets (Fragmentation)~~

Errors in the network (ICMP)

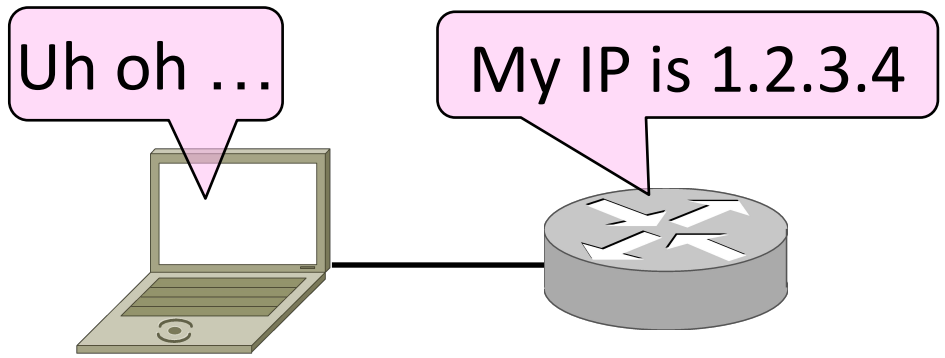
Running out of addresses before the IPv6 transition (NAT, Overlay)



Address Resolution Protocol (ARP)

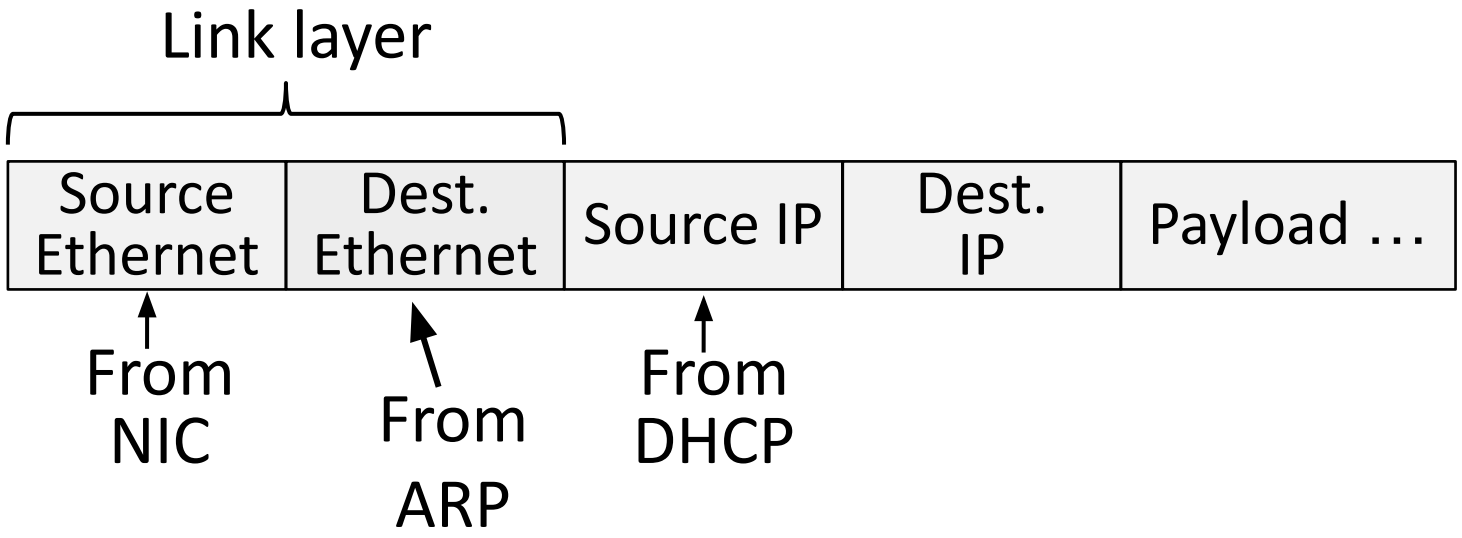
Sending an IP Packet

- 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



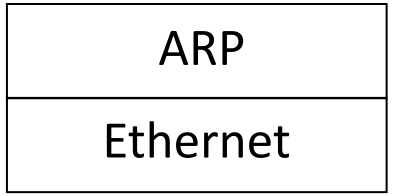
👉"NIC" = "Network Interface Controller/Card", refers to the actual interface hardware (may not be a "card" in many cases :))

ARP (Address Resolution Protocol)

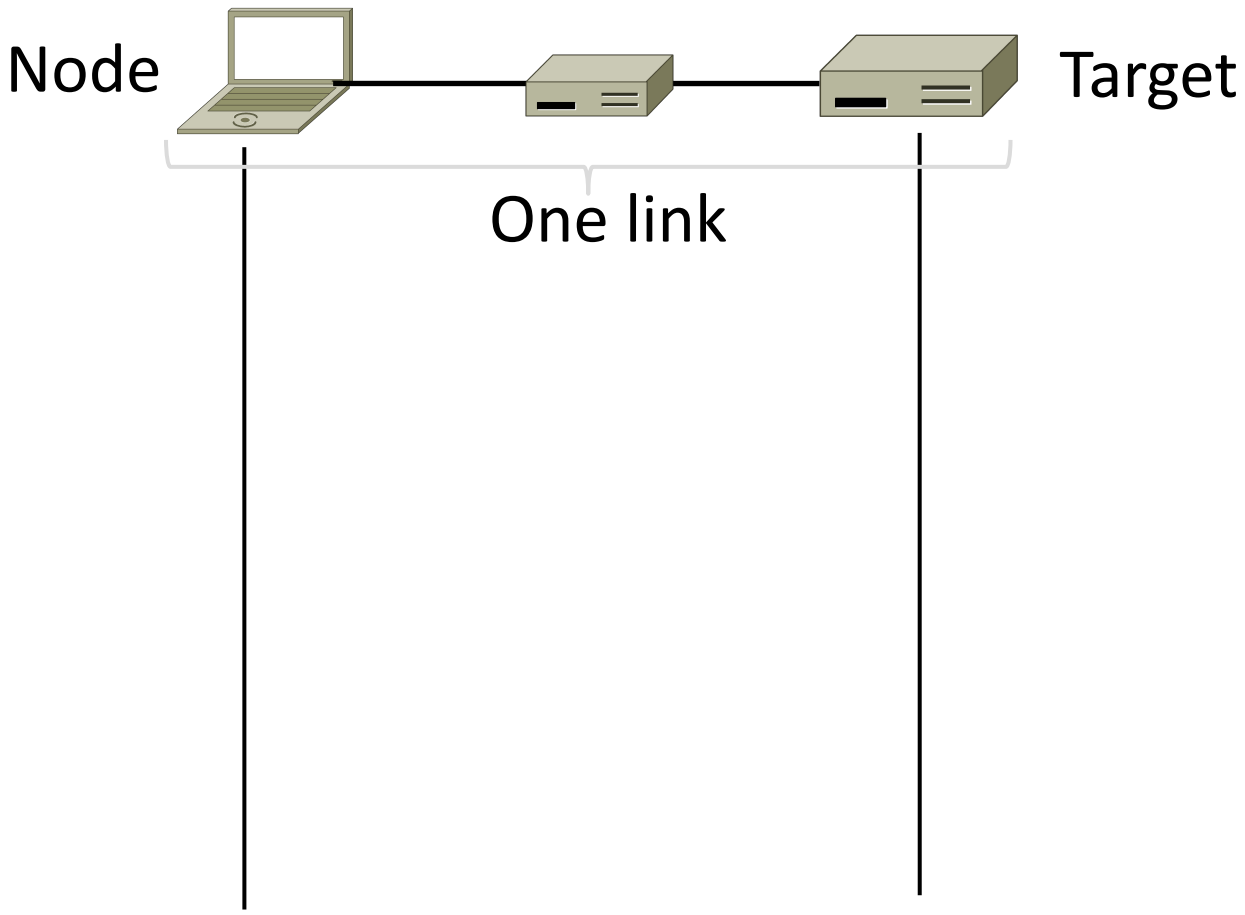
- process of discovering mapping from *logical* to *physical* address
- operates on system on the same IP subnet

ARP Protocol Stack

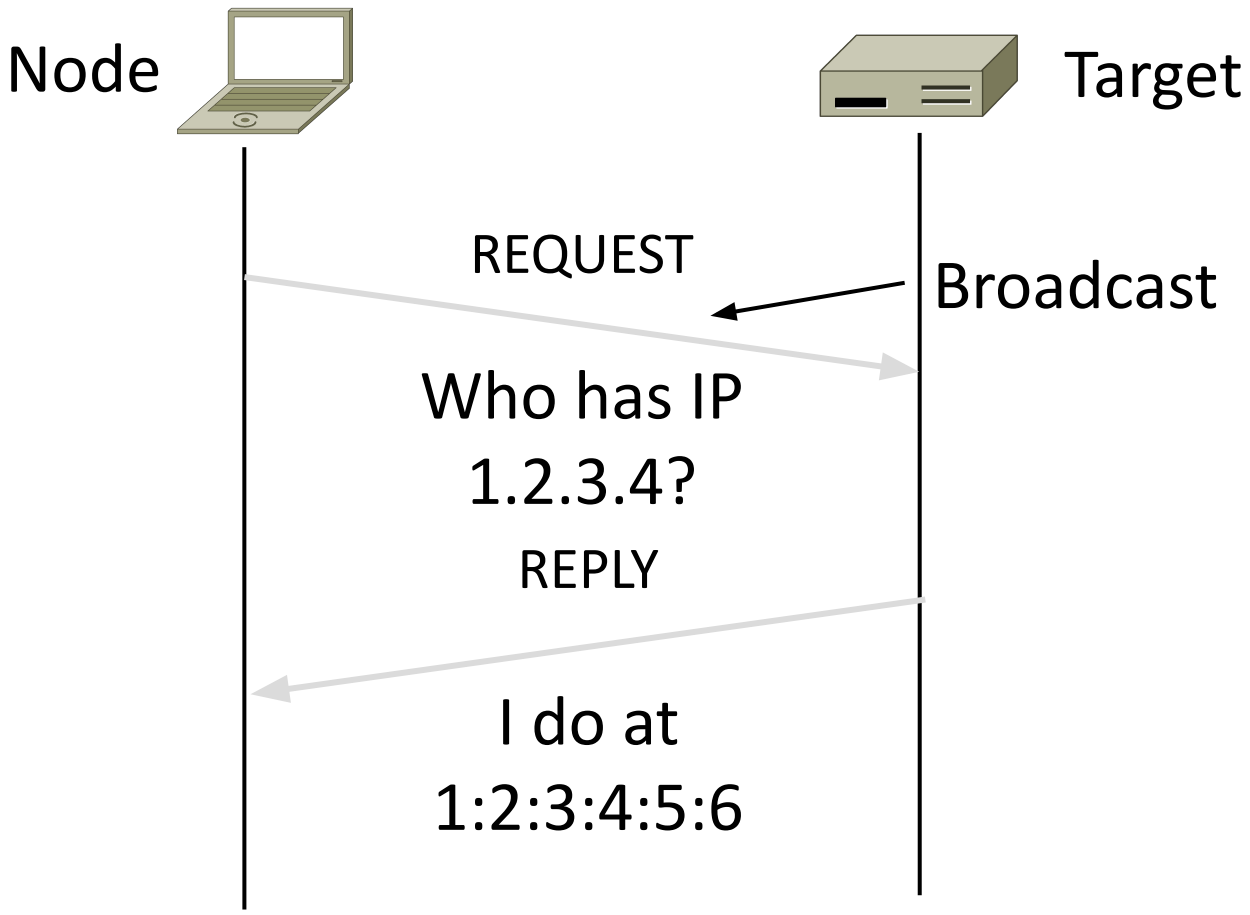
- ARP sits right on top of link layer
  - No servers, just asks node with target IP to identify itself
  - Uses broadcast to reach all nodes



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

Similar to backwards learning from bridges…
some implementations will do passive learning

```
# 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
[...]
```

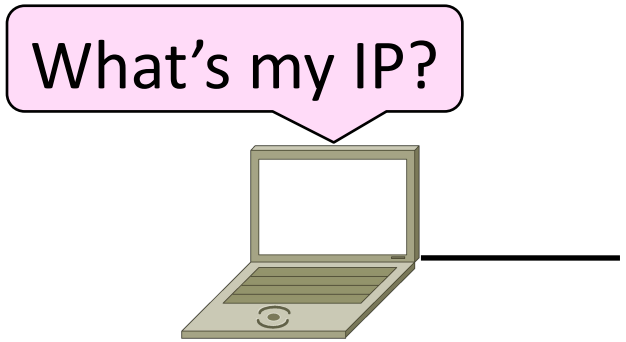
Discovery Protocols

- 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

Dynamic Host Configuration Protocol (DHCP)

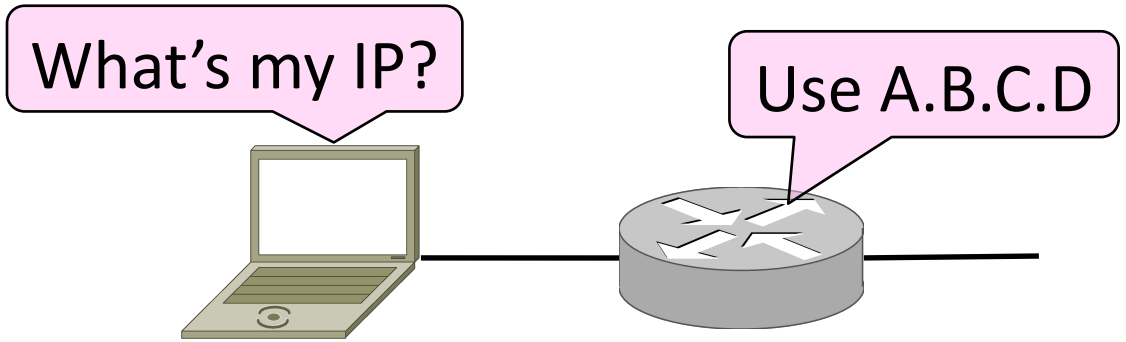
Bootstrapping

- 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

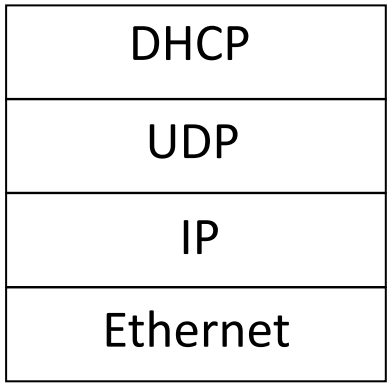


DHCP

- 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

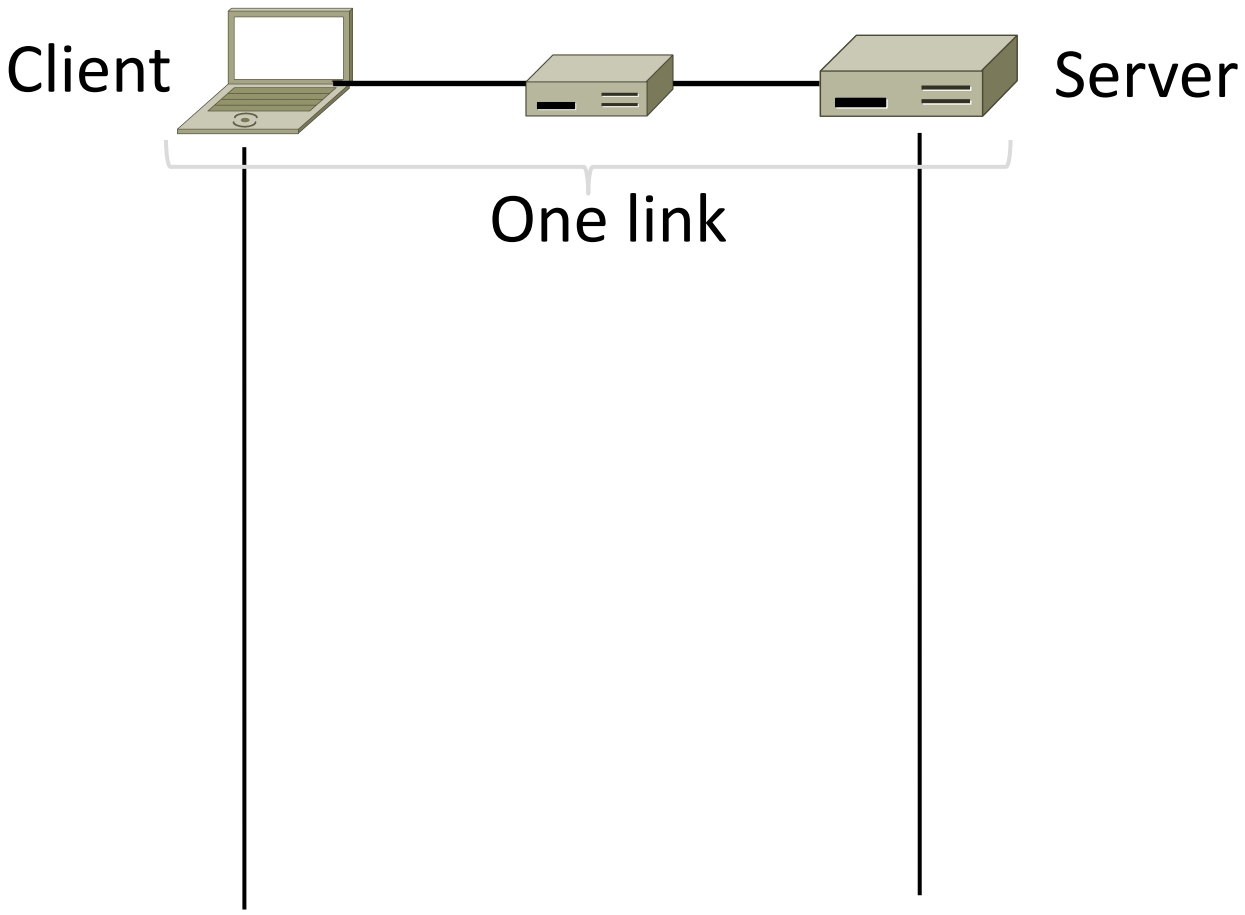
- DHCP is a client-server application
  - Uses UDP ports 67, 68



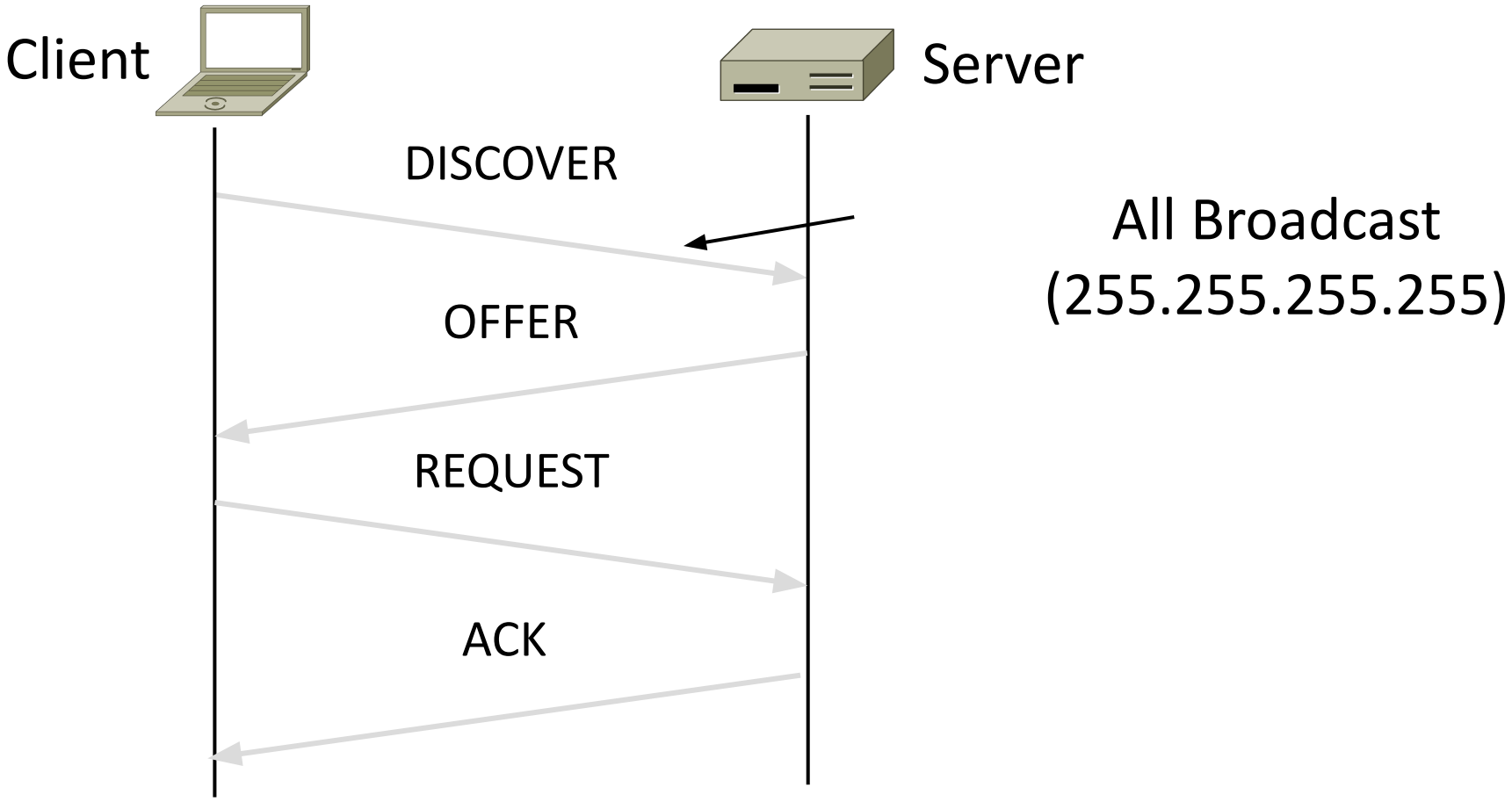
DHCP Addressing

- 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)



All Broadcast
(255.255.255.255)

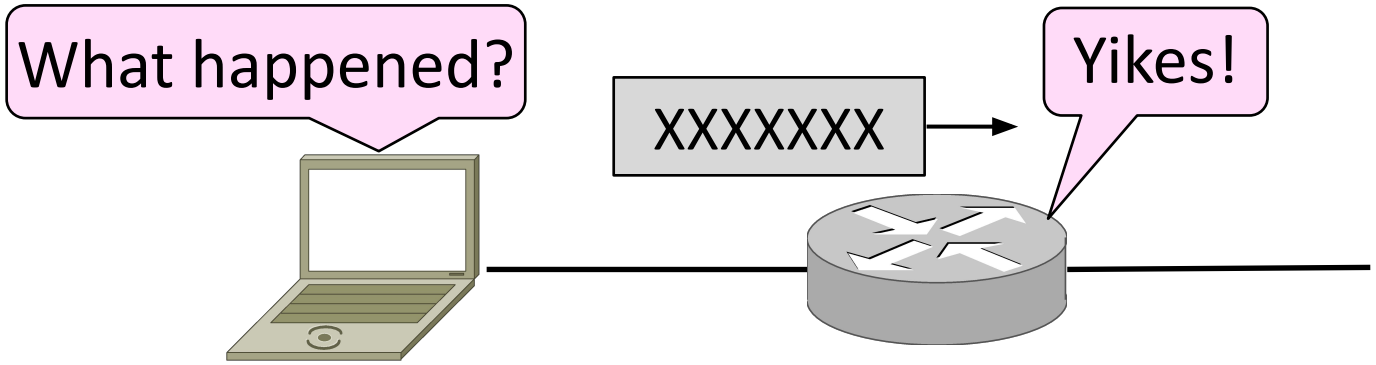
DHCP Messages (3)

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

Internet Control Message Protocol (ICMP)

Topic

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



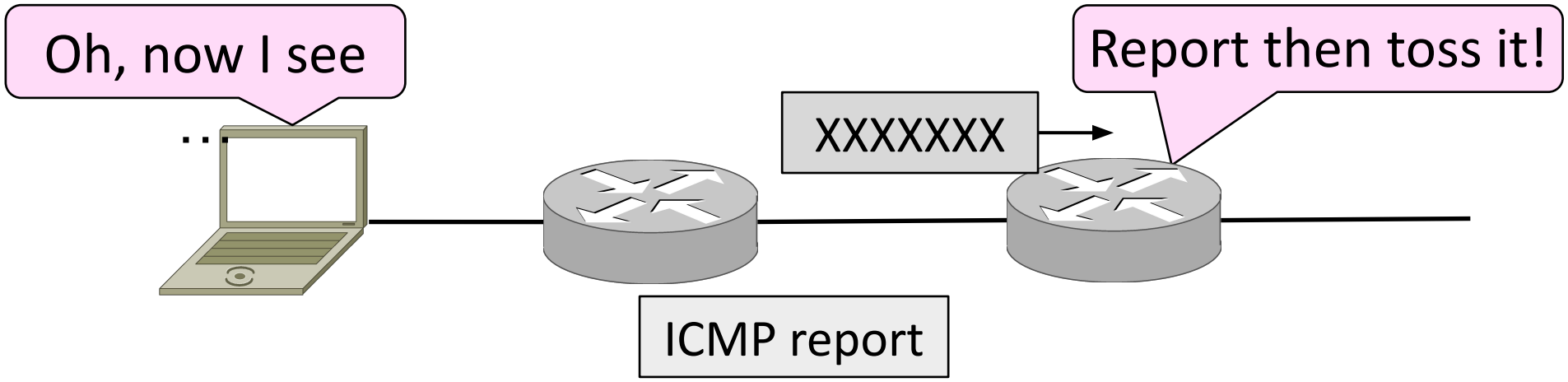
Internet Control Message Protocol

- 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

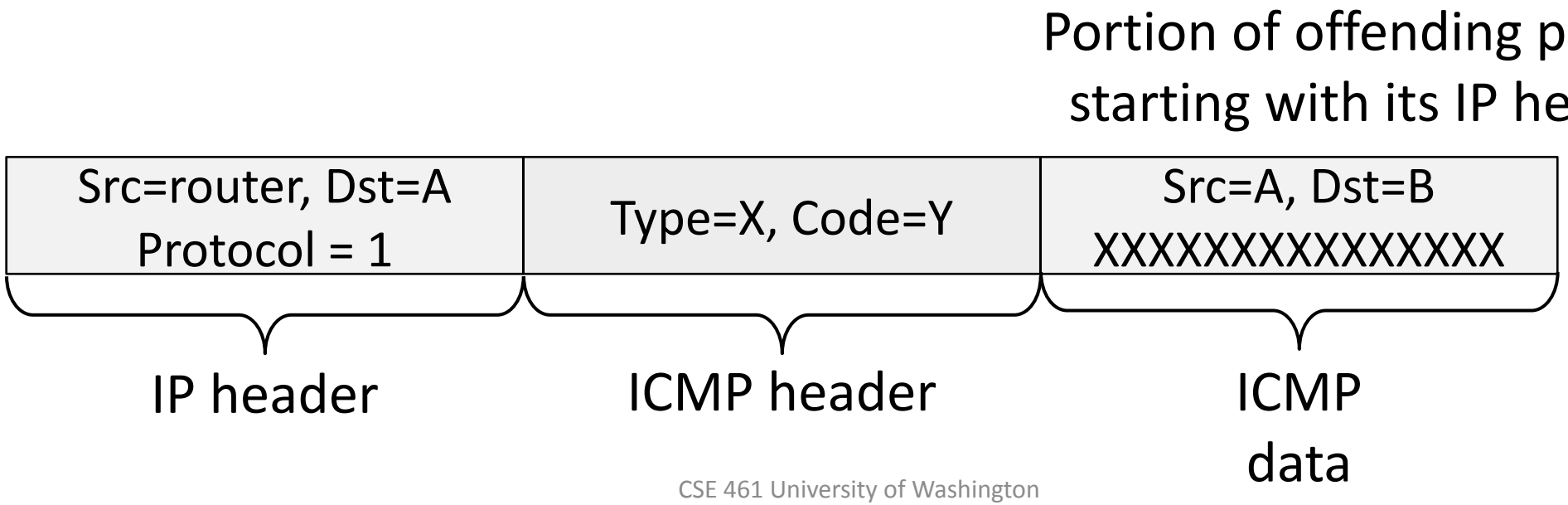
ICMP Errors

- 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



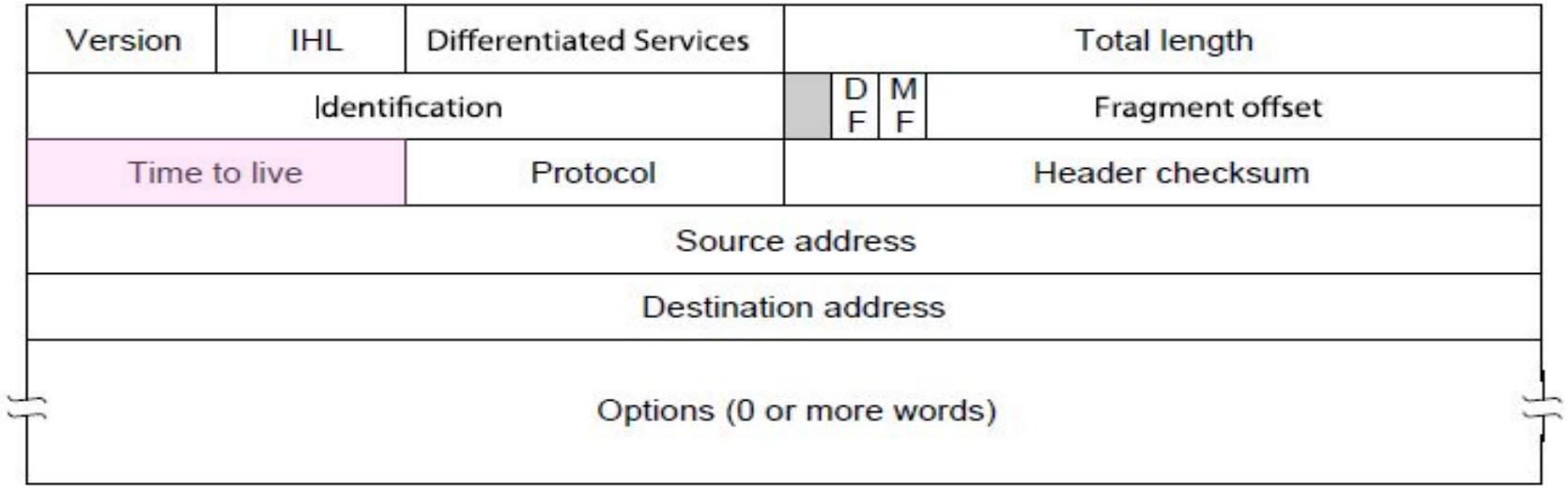
Example ICMP Messages

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

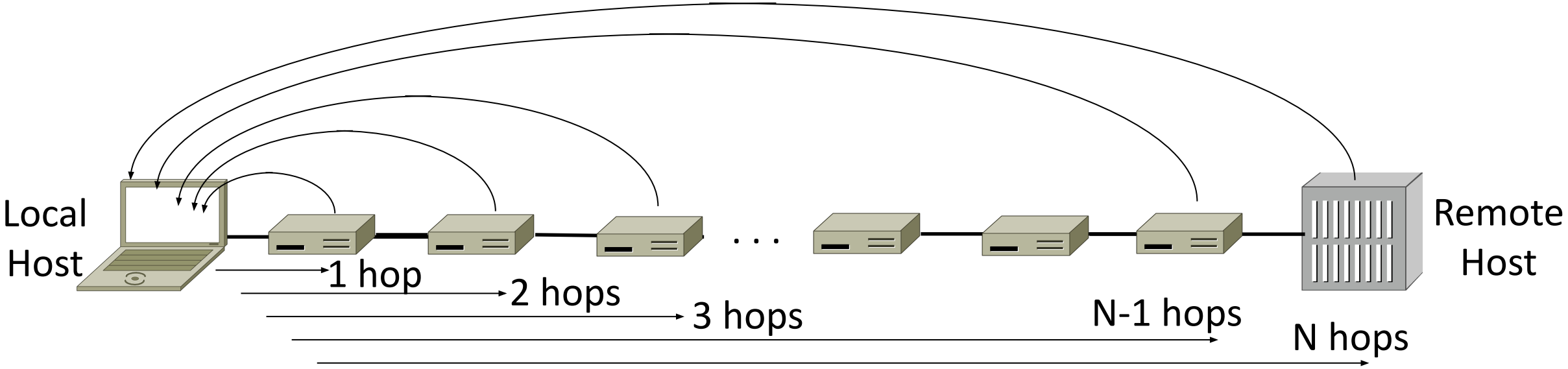
Traceroute

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



Traceroute (2)

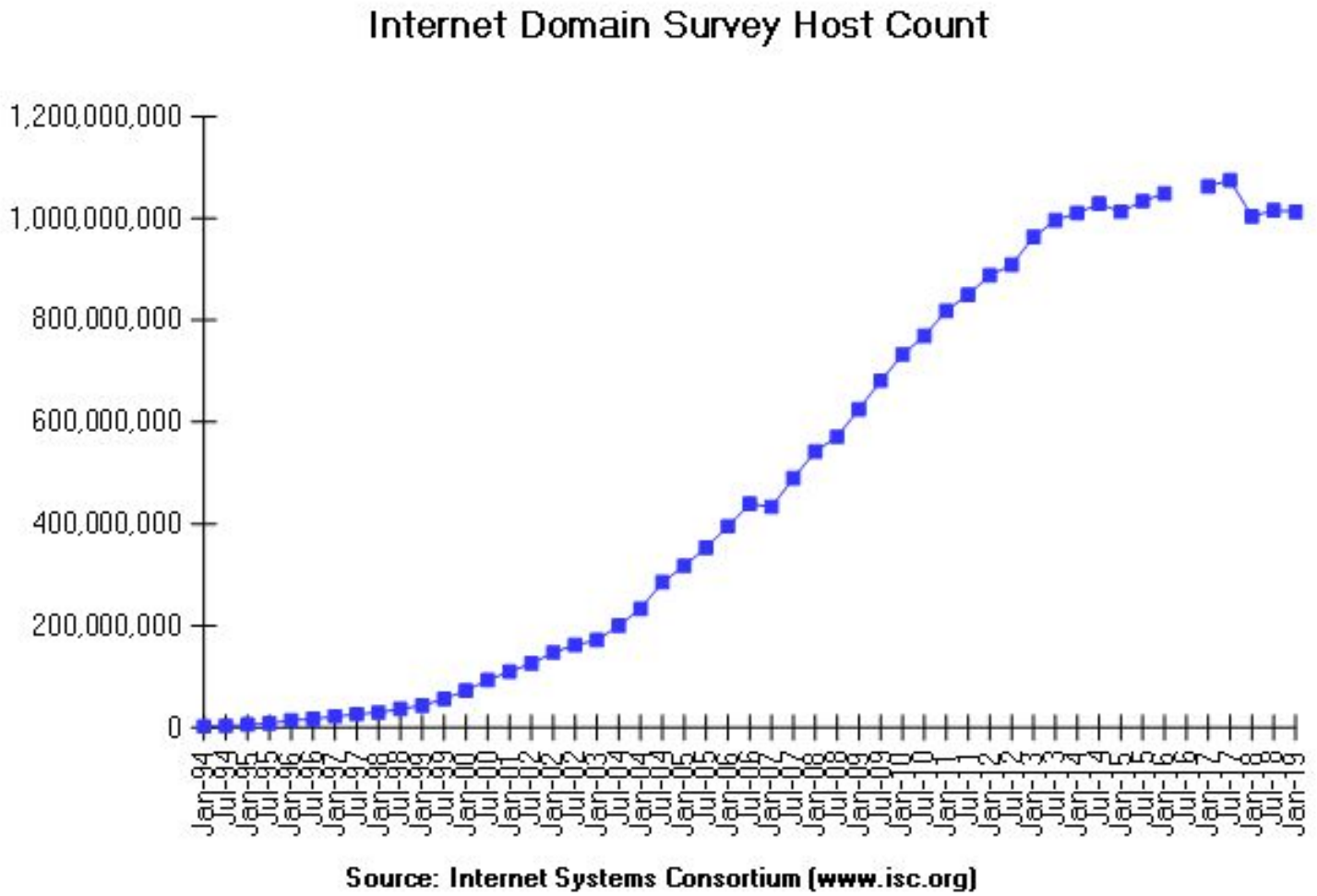
- Traceroute repurposes TTL and ICMP functionality
  - Sends probe packets increasing TTL starting from 1
  - ICMP errors identify routers on the path



Network Address Translation (NAT)

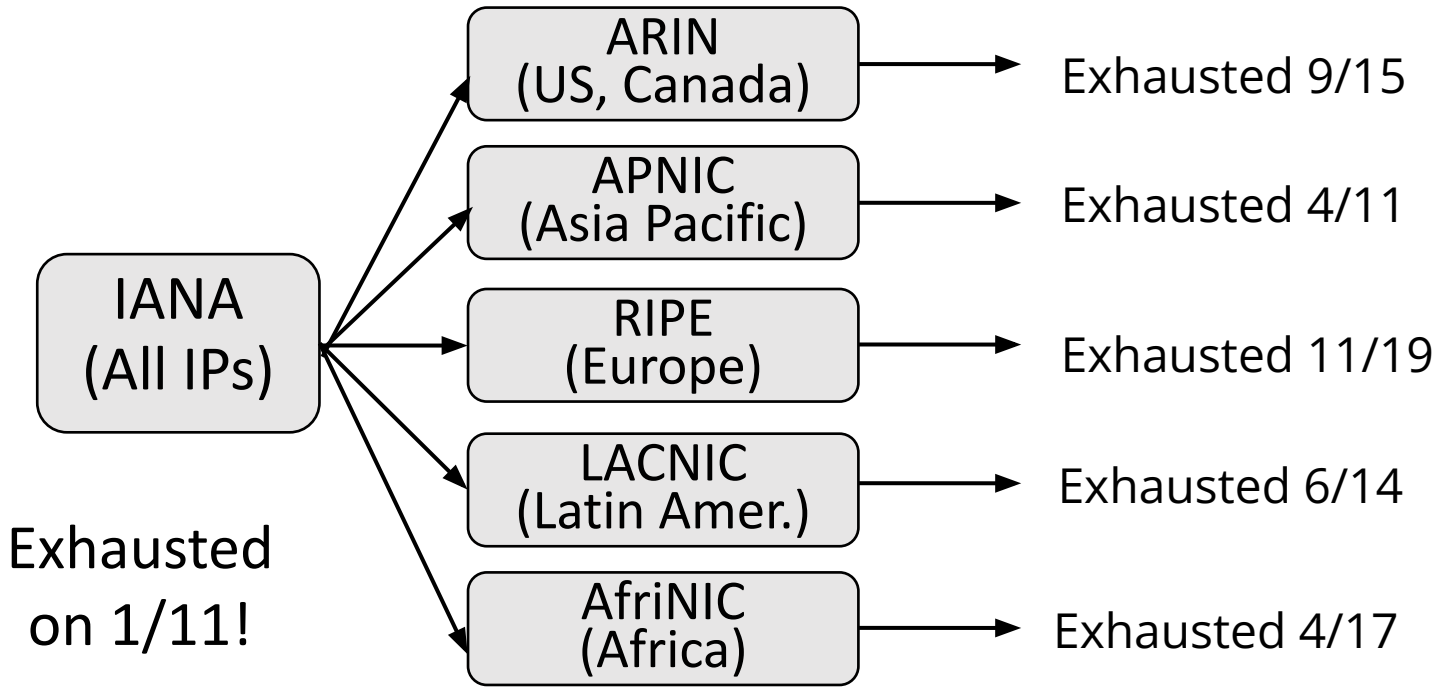
Problem: Internet Growth

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



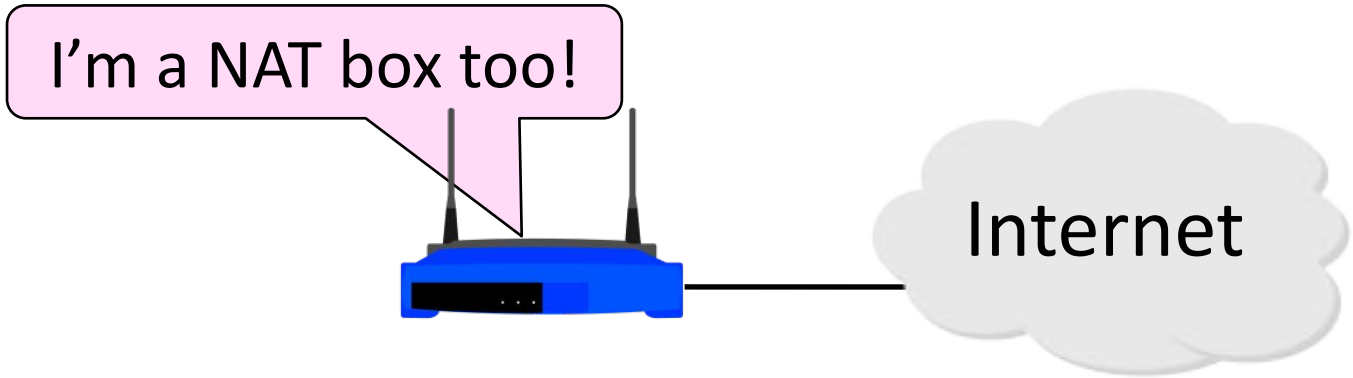
The End of New IPv4 Addresses

- All gone in 2019!



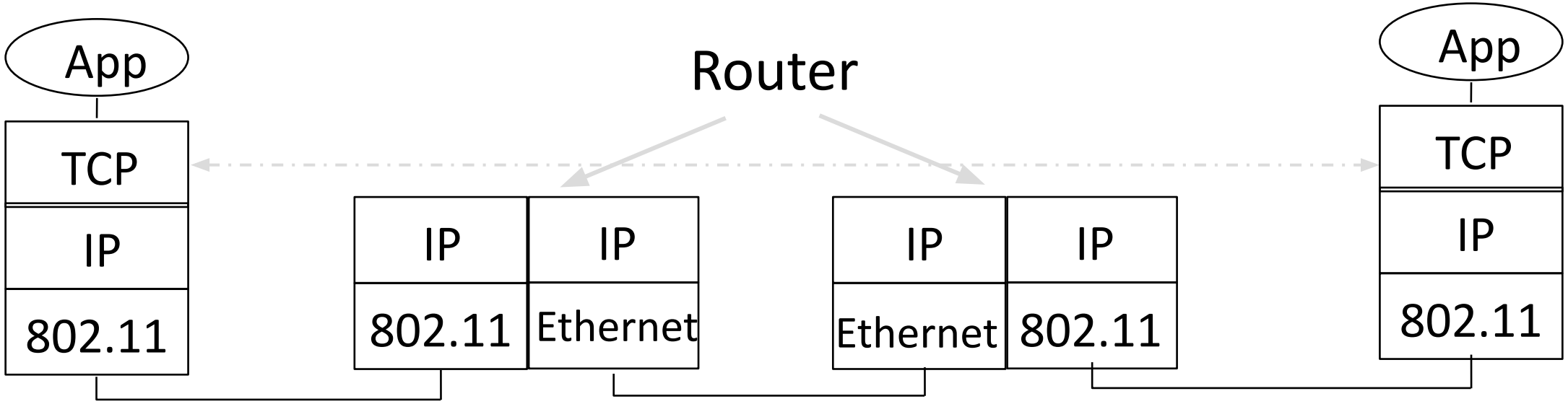
Stopgap Solution: Network Address Translation (NAT)

- Basic idea: Map many “Private” addresses to one “Public” IP.
- Allocated IP ranges for private use
  - `192.168.0.0/16 , 172.16.0.0/12 , 10.0.0.0/8`
- An additional range just for ISP NAP (“CGNAT”)
  - `100.64.0.0/10`



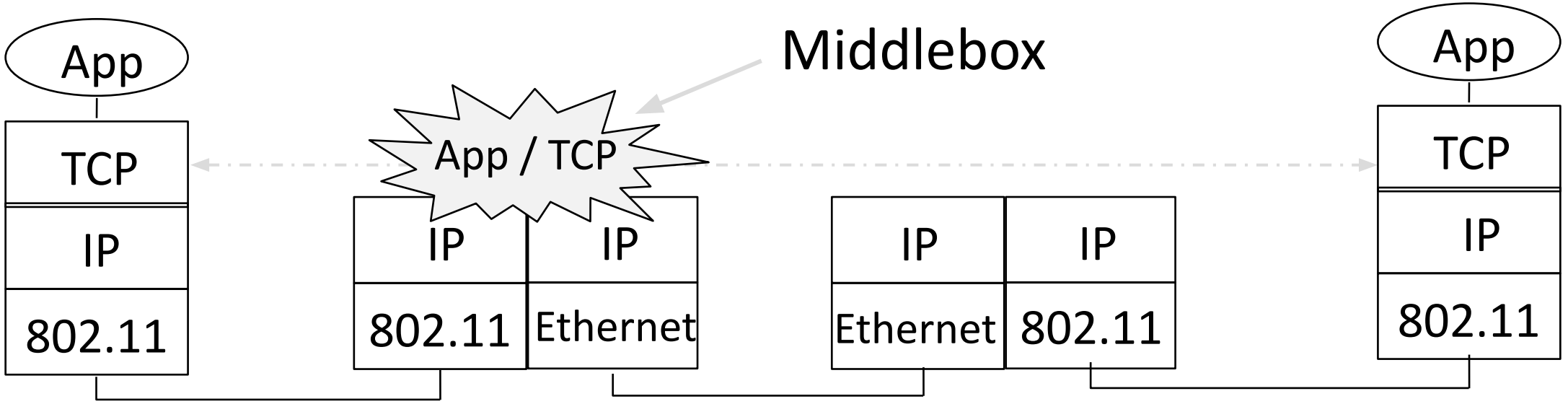
Layering Review

- 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)

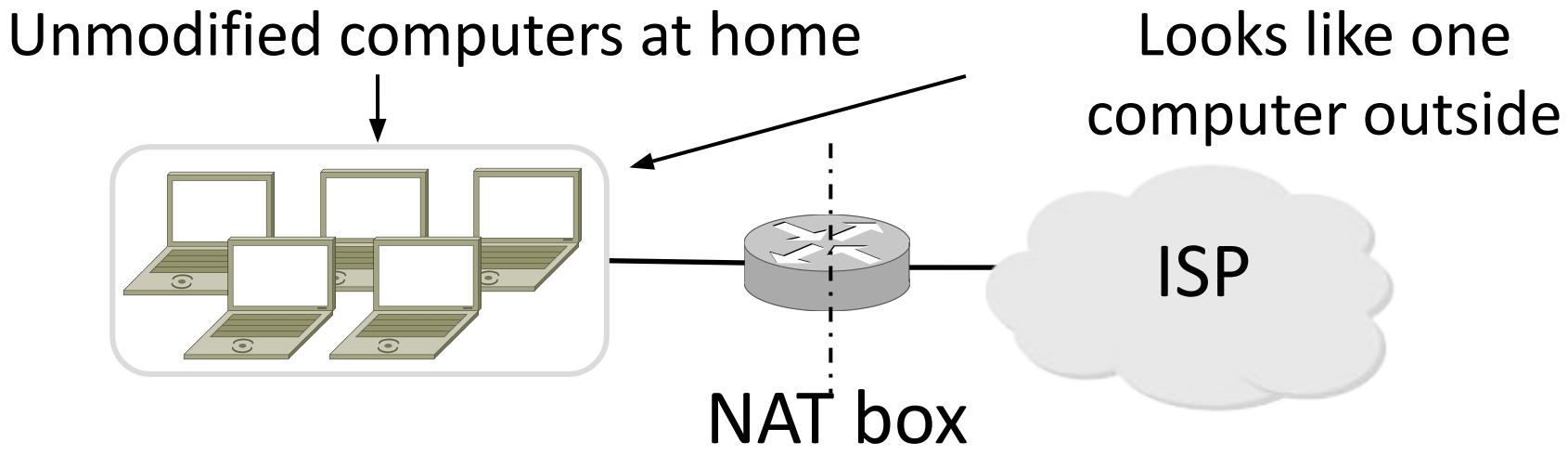
- Advantages
  - A possible rapid deployment path when no other option
  - Control over many hosts (IT)
- Disadvantages
  - Breaking layering interferes with connectivity
    - strange side effects
  - Poor vantage point for many tasks

NAT (Network Address Translation) Box

- 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
  - NAT (in AP/firewall) connects home to ISP using a single external IP address



How NAT Works

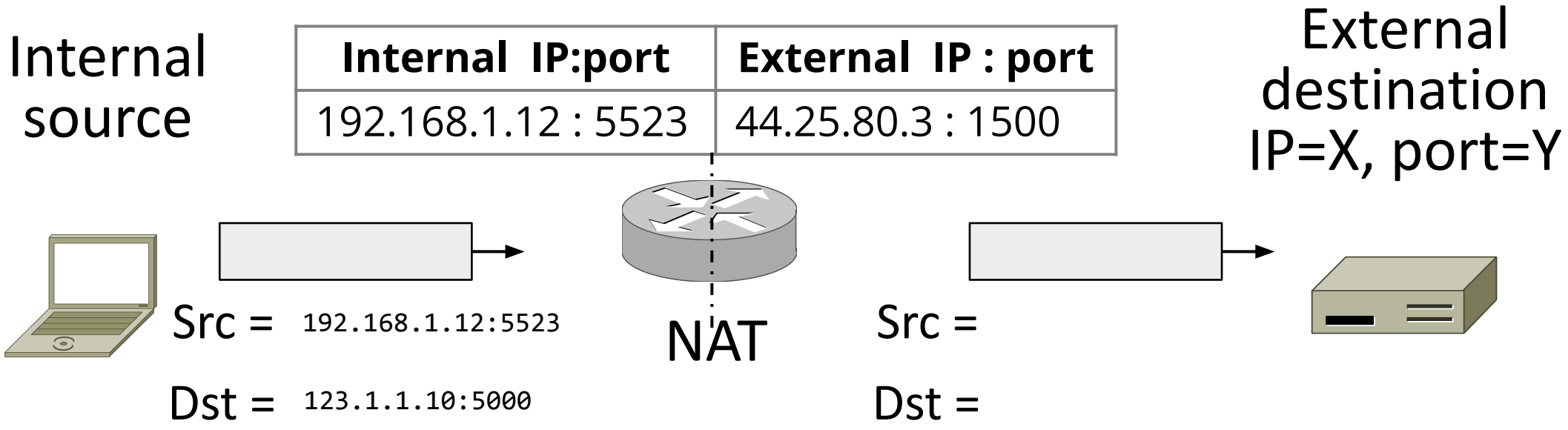
- Keeps an internal/external translation table
  - Typically uses IP address + TCP/UDP port
  - This is address and port translation

What host thinks	What ISP thinks
Internal IP:port	**External IP : port**
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

- Need ports to make mapping 1-1 since there are fewer external IPs

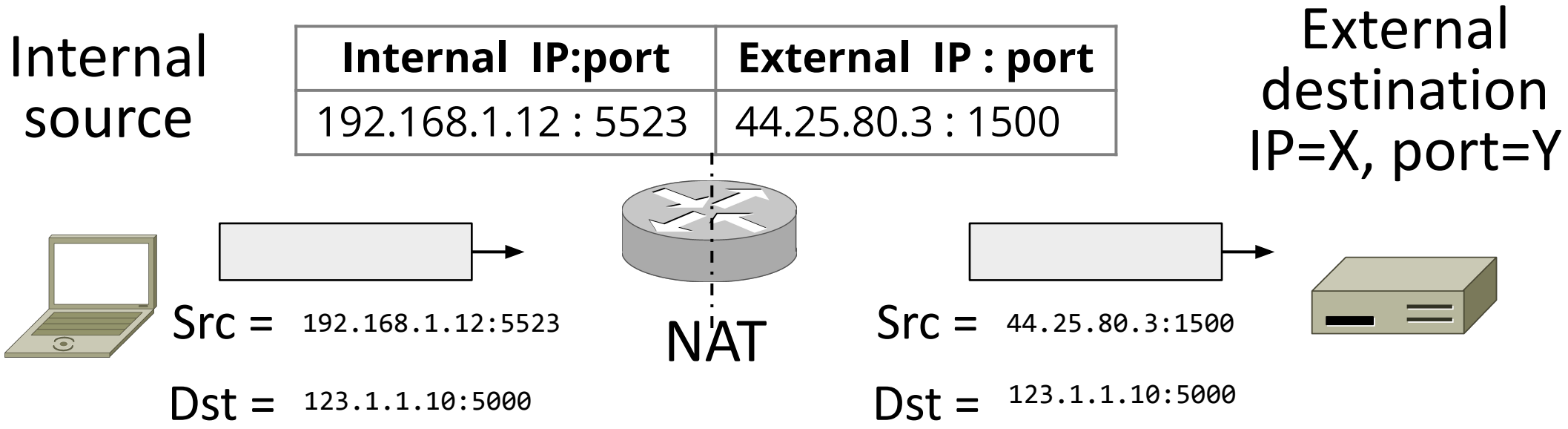
How NAT Works

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



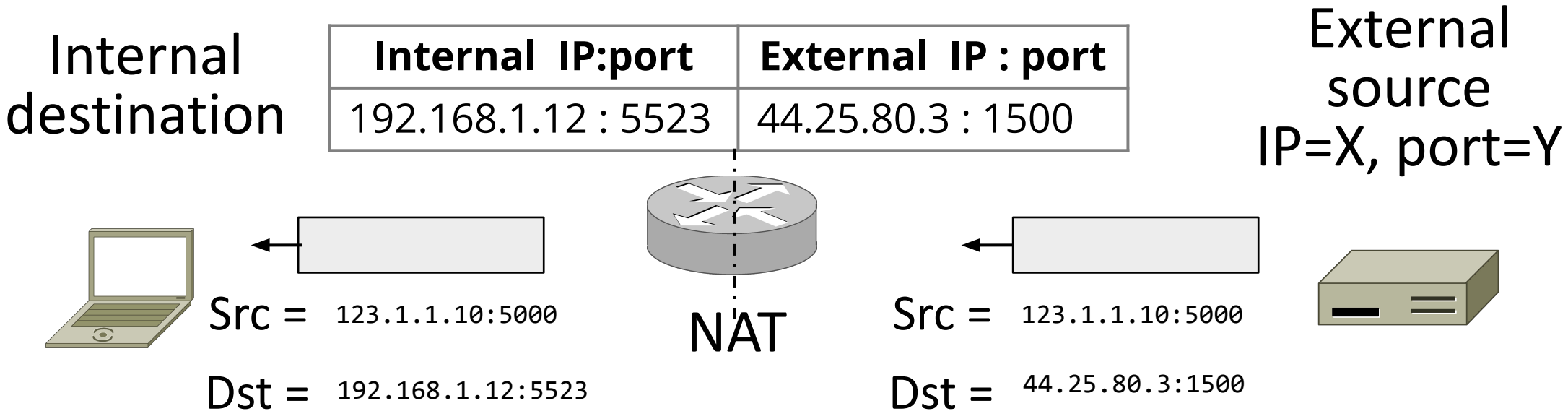
How NAT Works

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



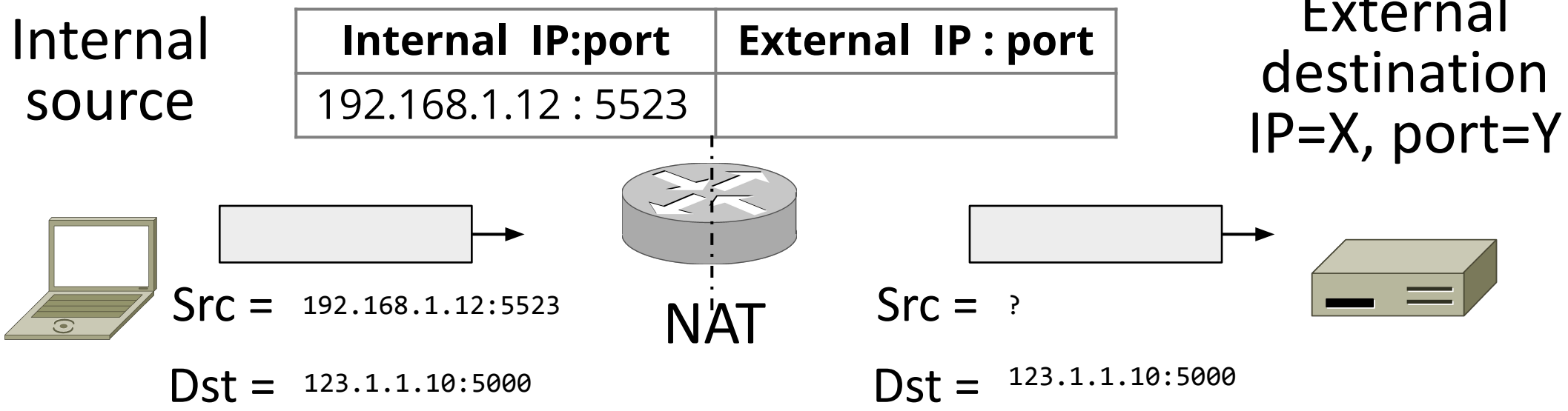
How NAT Works

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



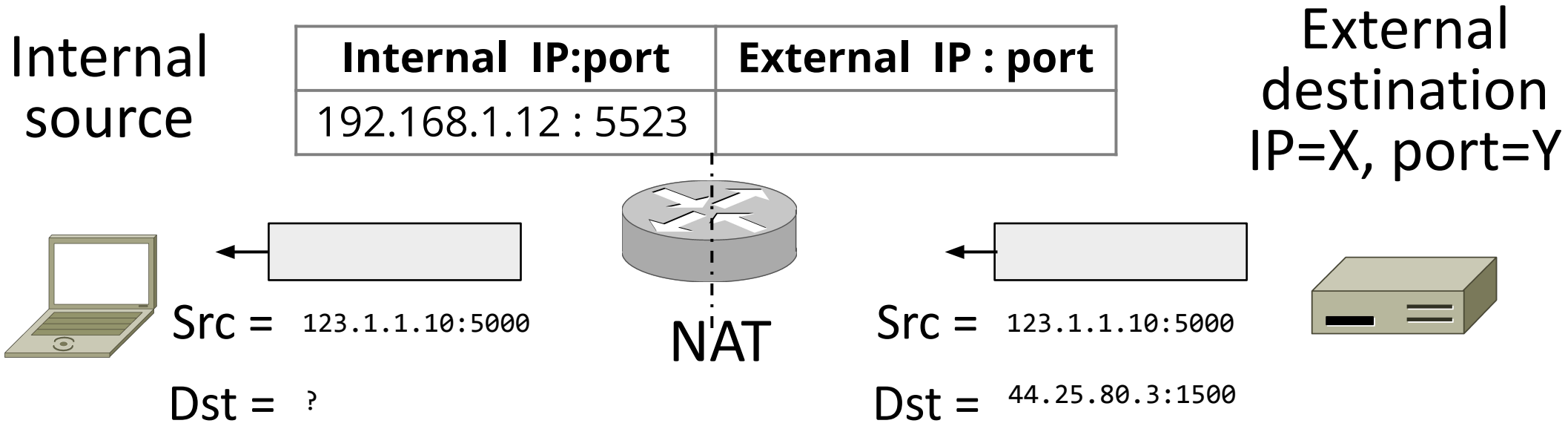
How NAT Works

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



How NAT Works

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



NAT Proliferation

- Like tunnels, NAT might not be elegant, but it
  - Solves The Problem™ (while creating many more :/)
- NAT is *extremely ubiquitous* in IPv4 ca. 2025
  - Your home network: very likely using NAT
  - Your WiFi here at UW: it uses NAT
  - Your wired computer here at UW: likely uses NAT
    - unless you ask very nicely for one of those $$ precious global addresses
  - Your WiFi at the airport: it uses NAT
  - Etc…

Small groups

- What are two upsides of NAT?
- What are two downsides of NAT?

NAT Upsides

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

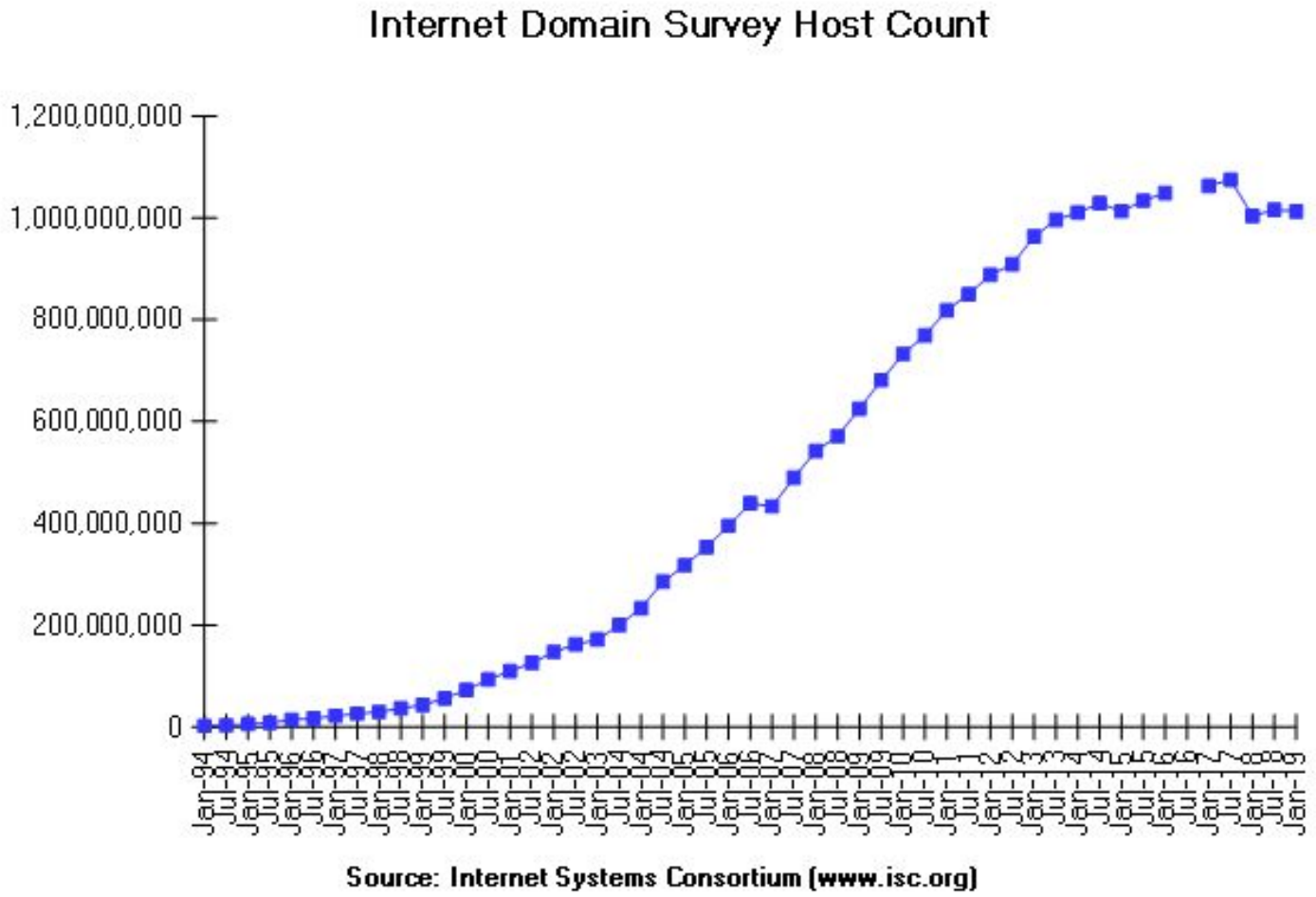
NAT Downsides

- 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 (Zoom???)
    - Requires an external rendezvous server (STUN/TURN)
- Doesn't work well when there are no connections (UDP)
- It's a middlebox, and middleboxes break abstraction layers
  - Cannot use new transports without also updating middleboxs
- Breaks apps that expose their IP addresses (FTP)

IPv6

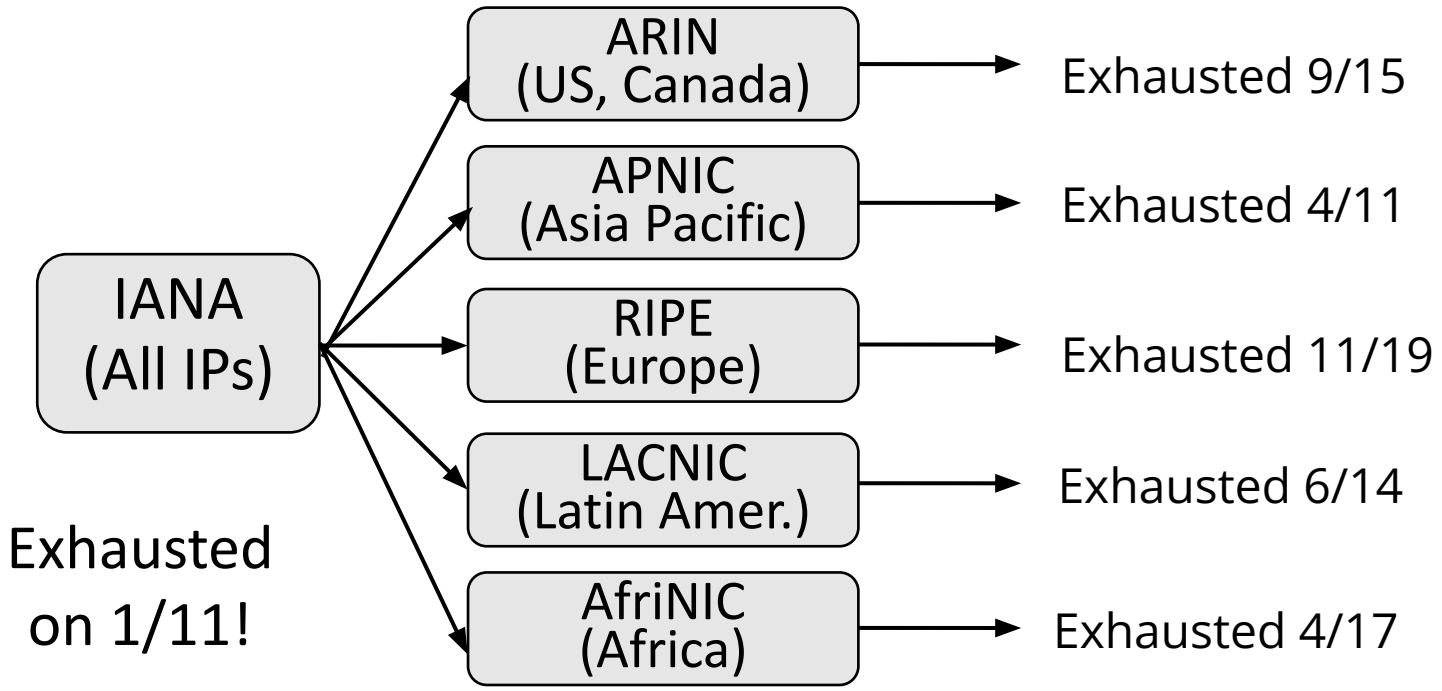
Problem: Internet Growth

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



The End of New IPv4 Addresses

- All gone in 2019!



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

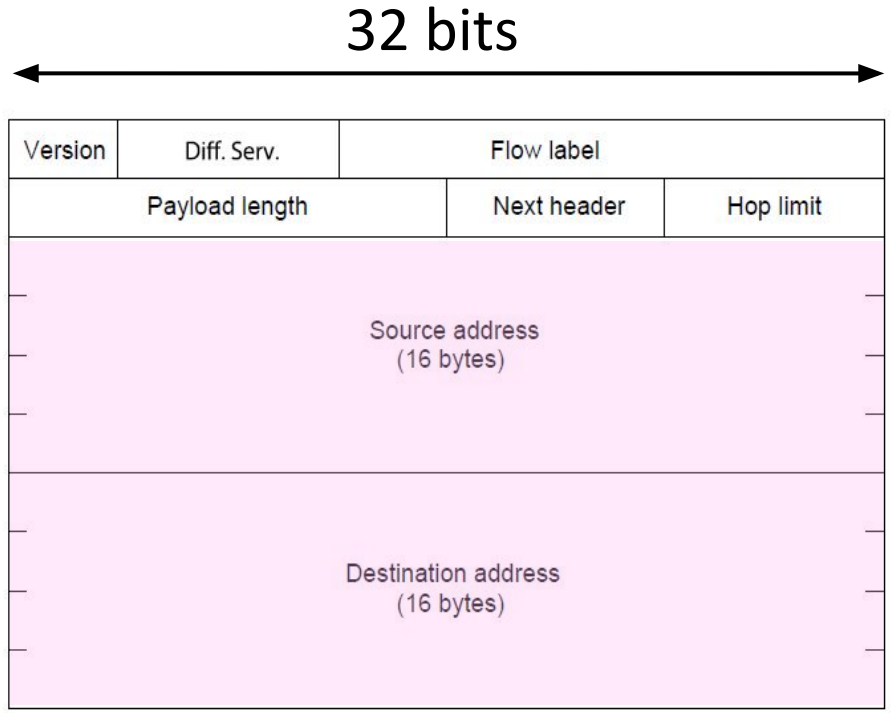
IPv6

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

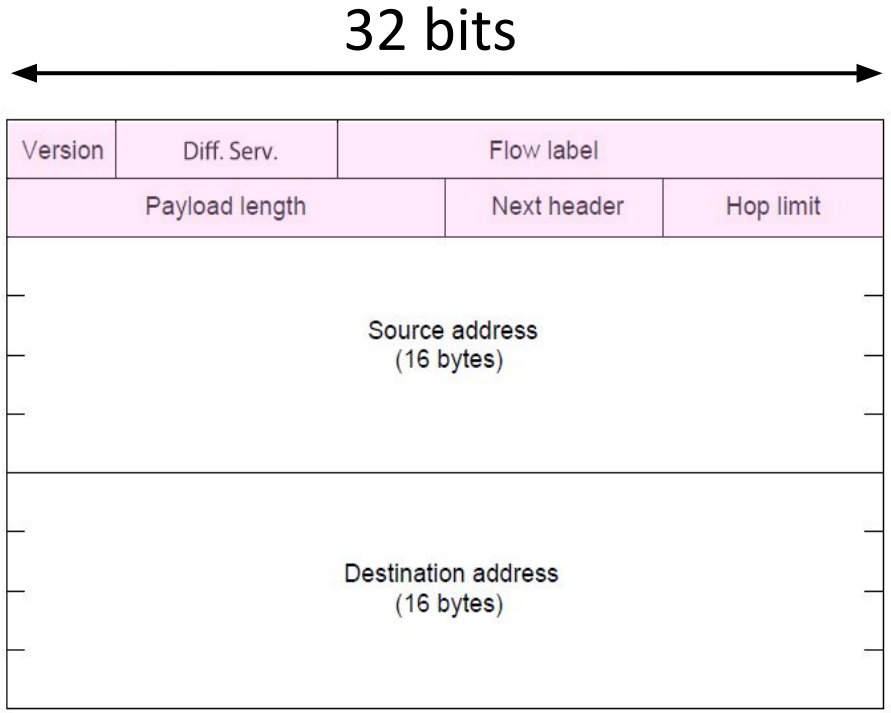
Ex:

2001:0db8:0000:0000:0000:ff00:0042:8329

→ 2001:db8::ff00:42:8329



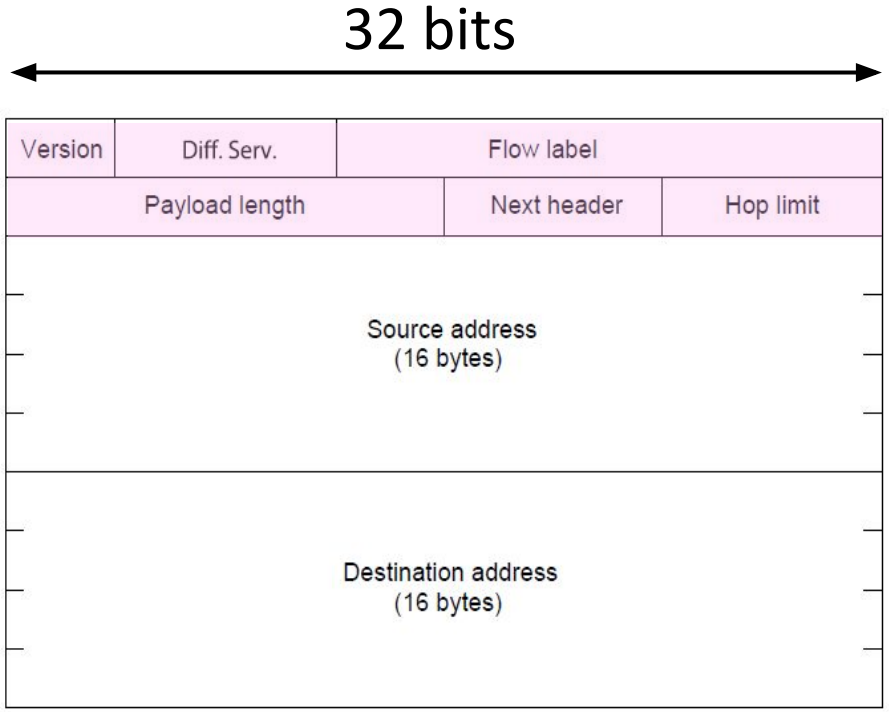
IPv6 (2)



- Lots of other changes
  - Link-local and unique-local addresses
  - Streamlined header processing
    - No checksum (why's that faster?)
  - Flow label to group of packets
  - IPSec by default
  - Better fit with "advanced" features (mobility, multicasting, security)

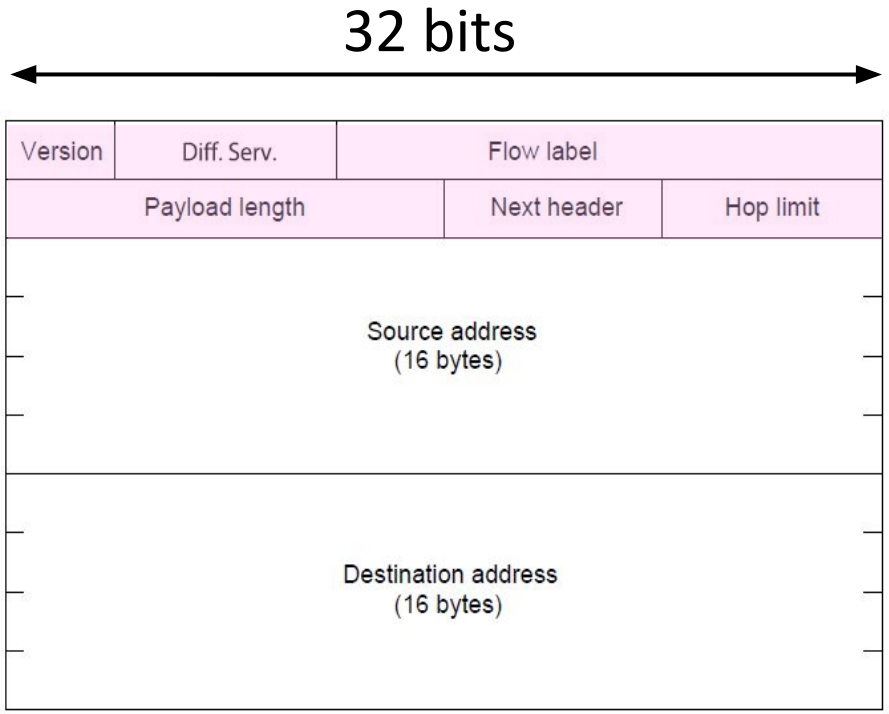
IPv6

- Does IPv6 fix the need for ARP?
- Does IPv6 fix the need for DHCP?
- Does IPv6 fix the need for ICMP?
- Does IPv6 fix the need for NAT?



IPv6

- Does IPv6 fix the need for ARP?
- Does IPv6 fix the need for DHCP?
- Does IPv6 fix the need for ICMP?
- Does IPv6 fix the need for NAT?



No, no, no, yes!

Neighbor Discovery Protocol

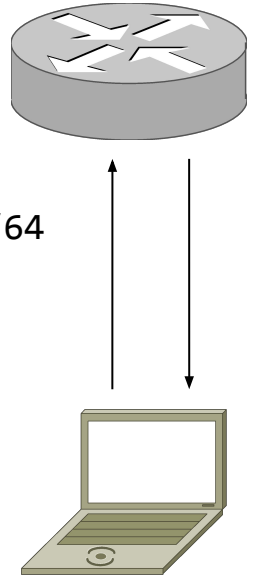
- Uses ICMPv6
- DHCP Functions:
  - Router discovery (133)/advertisement (134)
- ARP Functions:
  - Neighbor discovery (135)/advertisement (136)

Lots of fun NDP functionality

- Informed by tens of years of IPv4 operation
- Duplicate address detection (DAD)
  - nodes can check whether an address is already in use.
- Next hop determination
- Neighbor unreachability detection (NUD)
  - determine that a neighbor is no longer reachable on the link.
- Etc.

Stateless Autoconfiguration (SLAAC)

- With NDP, replaces common DHCP cases
  - Drops less frequently used DHCP features like NTP servers, netboot image locations, etc.
- Uses ICMPv6 + Link-Local Multicast
- Process:
  - Send local multicast “All Routers” message
  - Get prefix(es) from responding router(s)
  - Attach MAC to router Prefix /w some math
    - 48 bit → EUI-64 format
    - Or random for privacy preservation



MAC: 0200:1234:5678 → 0000:12FF:FE34:5678
Address: 2000:1234:5678::12FF:FE34:5678/64

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)

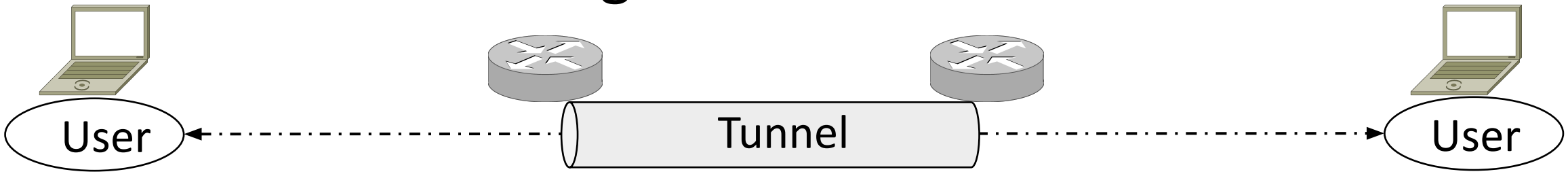
IPv6 Transition

We're breaking the one-protocol abstraction!!! A big part of why rollout was/is slow 😬

- 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)

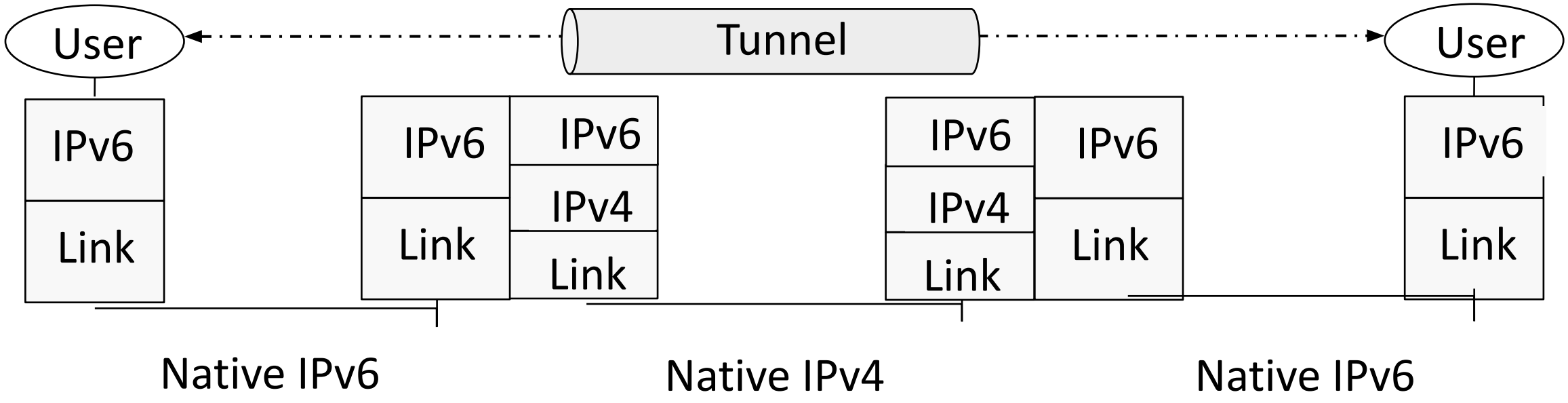
Tunneling

- Tunnel acts as a single link across IPv4 network



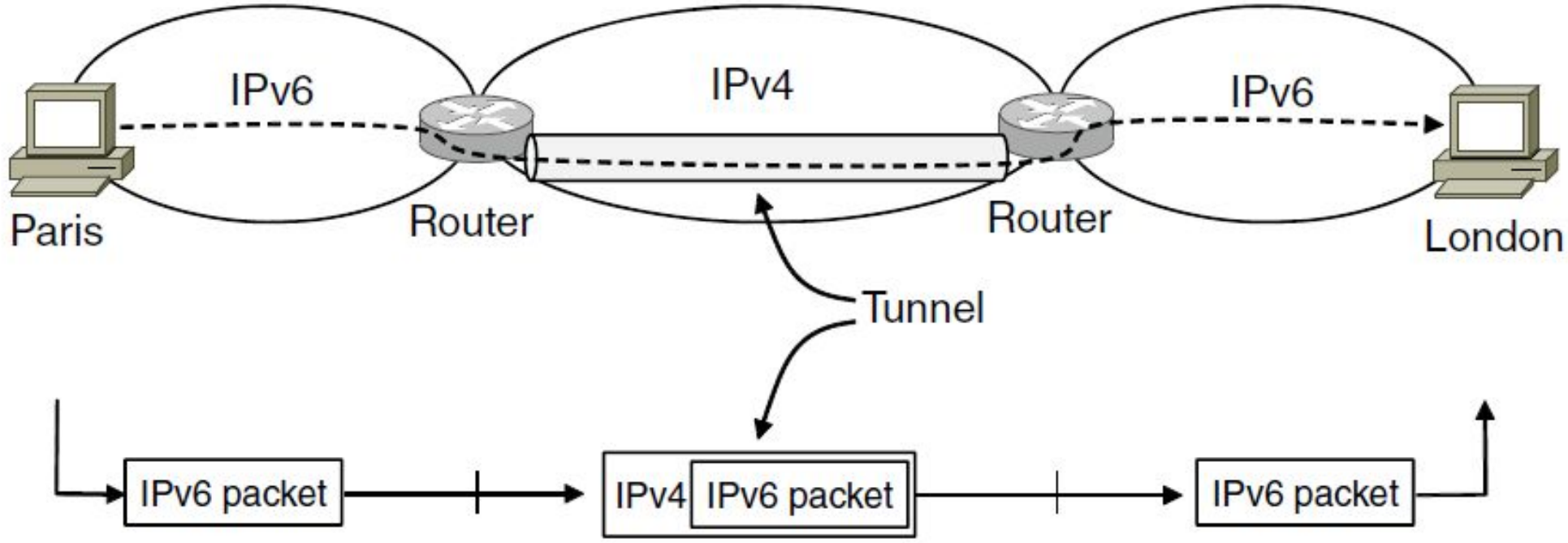
Tunneling

- Tunnel acts as a single link across IPv4 network
  - Difficulty is to set up tunnel endpoints and routing



Tunneling

- Native IPv6 islands connected via IPv4
  - Tunnel carries IPv6 packets across IPv4 network



Tunneling Proliferation

- Tunneling has become extremely common
- Widely used in datacenters + containers for virtual networks
- Allows adding support for new things transparently across existing infrastructure, layer of security w/ encrypted tunnels
- What are some possible disadvantages of tunnels?

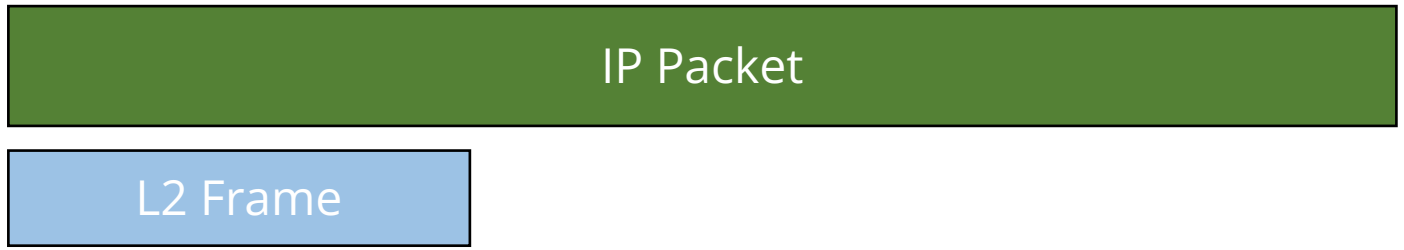
Tunneling Proliferation

- Tunneling has become extremely common
- Widely used in datacenters + containers for virtual networks
- Allows adding support for new things transparently across existing infrastructure, layer of security w/ encrypted tunnels
- What are some possible disadvantages of tunnels?
  - Setup complexity
  - Increased packet size (next section)
  - Opacity prevents routing optimization

Packet Sizing & Fragmentation

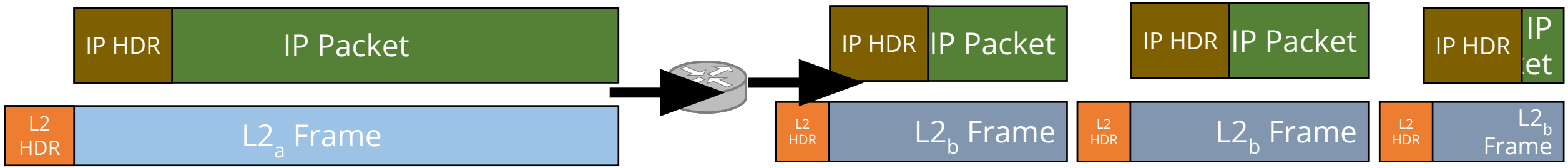
L3 Abstracts Over Many L2 Technologies

- Recall one of the goals of the network layer was to allow compatibility between many different kinds of links!
- But at the end of the day *the packets still have to be transported over the L2 frame abstraction…*
- What happens when packets are too big for the underlying frame?
  - The max supported size is the MTU (Maximum Transmission Unit)



One solution: fragmentation

- Supported *in the network* in IPv4
- Routers along the path can fragment packets as needed



- Pretty Clever™ Solution…
- Any issues???

(Many) Fragmentation Problems

Fragmentation can cause many practical issues and kill performance *invisibly*, good to be aware that it can happen!

- Multiple packets now instead of one
  - Increased processing overhead!
  - Duplication of headers
  - Can get really unlucky and get fragments of fragments 😱
- Need to reassemble on reception…
  - Buffers, reordering, etc.
- If there is an error, there's no way to signal to the source that you only need one of the fragments!
  - Have to re-send everything
  - Fragmenting happens in-network, the source never knows!
- Made worse with tunnels!
  - On-the-wire packet sizes opaquely increase!

Fragmentation in IPv6

- With the benefit of hindsight, IPv6 goes out of its way to avoid fragmentation!
- Mandated minimum MTU for L2 technologies supporting IPv6
  - 1280Bytes … Large enough that a simple implementation always using 1280 as the MTU doesn't have too much overhead
- Uses "path MTU discovery" to find the minimum (most restrictive) MTU along the path
  - The host is then responsible for telling the transport layer to decrease packet sizes, or fragmenting at the end-host
  - No in-network fragmentation! Too big packets are dropped and an ICMPv6 error is returned