# Network Layer (IP)

CSE 461

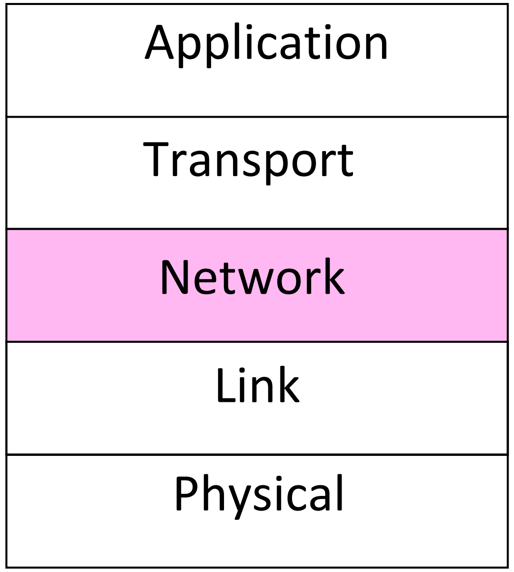
Ratul Mahajan

# Recall the protocol stack

| | |
|---|---|
| Application | – Programs that use network service |
| Transport | – Provides end-to-end data delivery |
| Network | – Send packets over multiple networks |
| Link | – Send frames over one or more links |
| Physical | – Send bits using signals |

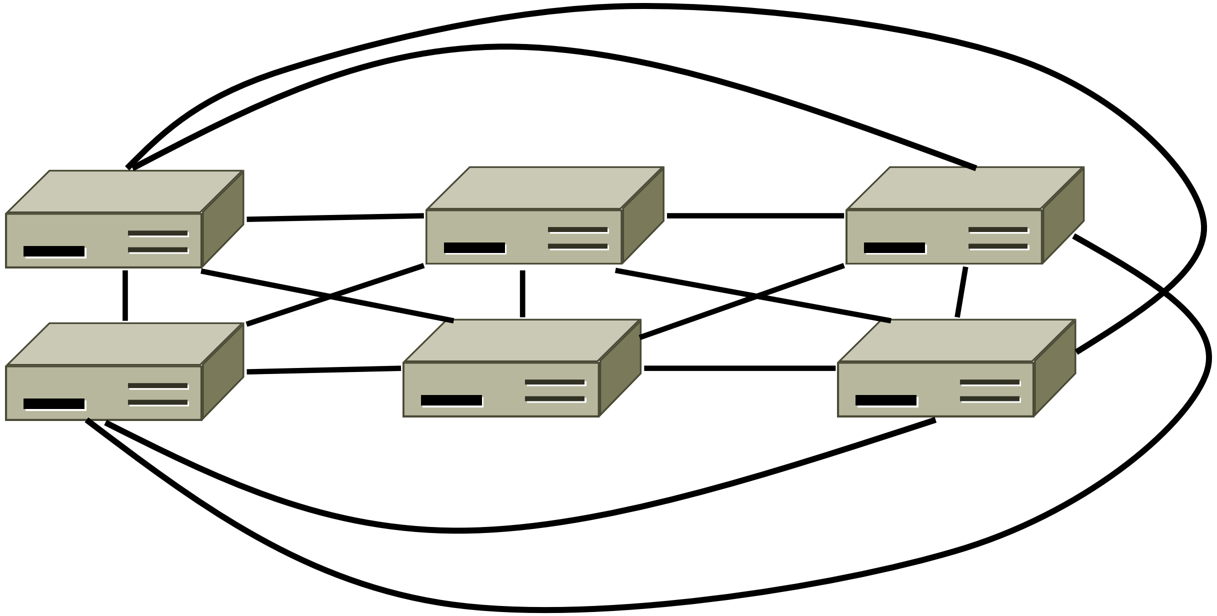
# Network Layer

Goal: Get packets from source to destination, which may be separated by many hops



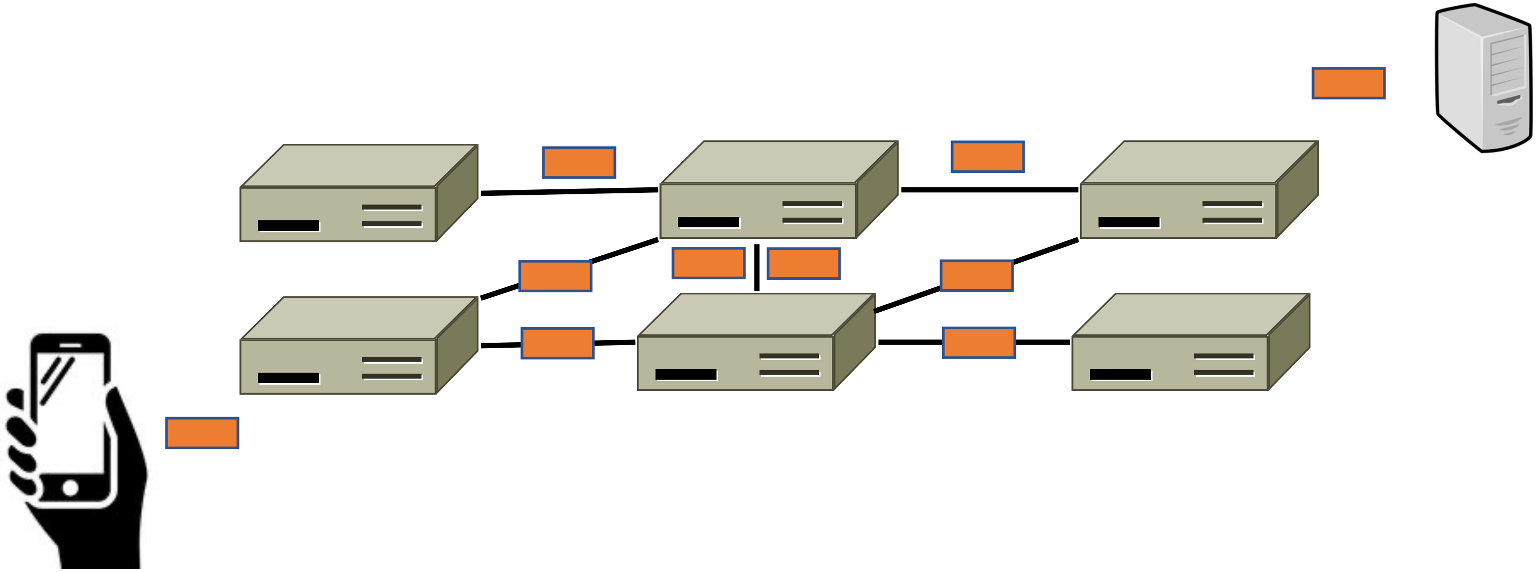
# Why do we need a Network layer?

- Cannot afford to directly connect everyone
  - Cost and link layer diversity



# Why do we need a Network layer? (2)

- Cannot broadcast all packets at global scale

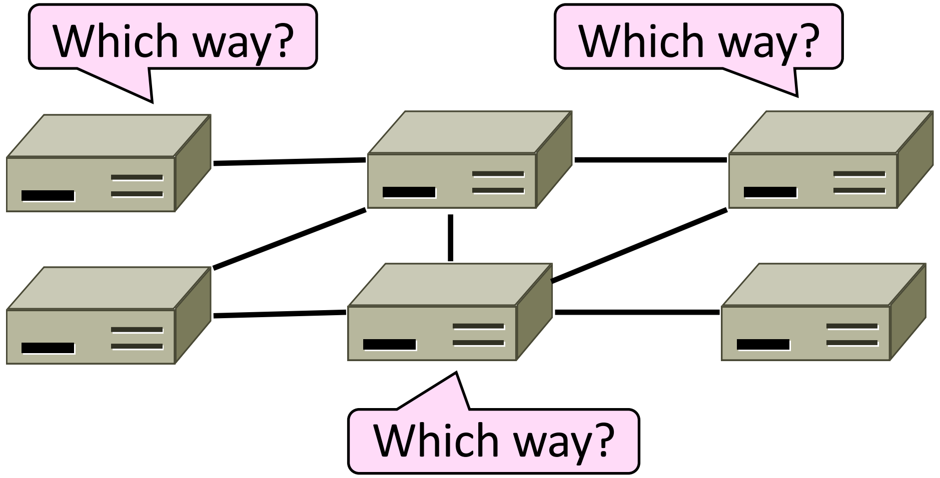


# Why do we need a Network layer? (3)

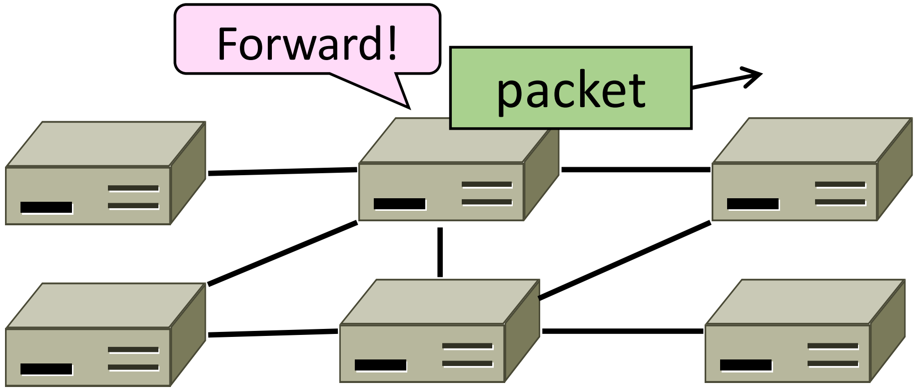
- Internetworking
  - Need to connect different link layer networks
- Addressing
  - Need a globally unique way to “address” hosts
- Routing and forwarding
  - Need to find and traverse paths between hosts

# Routing versus Forwarding

- <u>Routing:</u> deciding the direction to send traffic



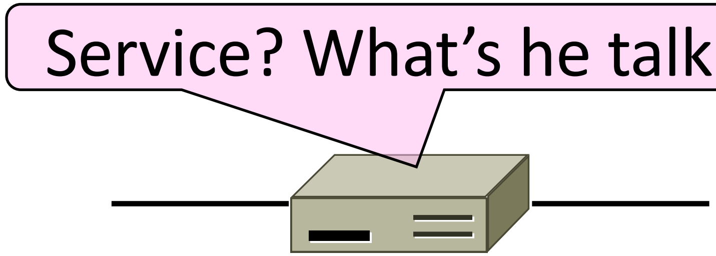

- <u>Forwarding:</u> sending a packet on its way



# Network Service Models

# Network service models

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



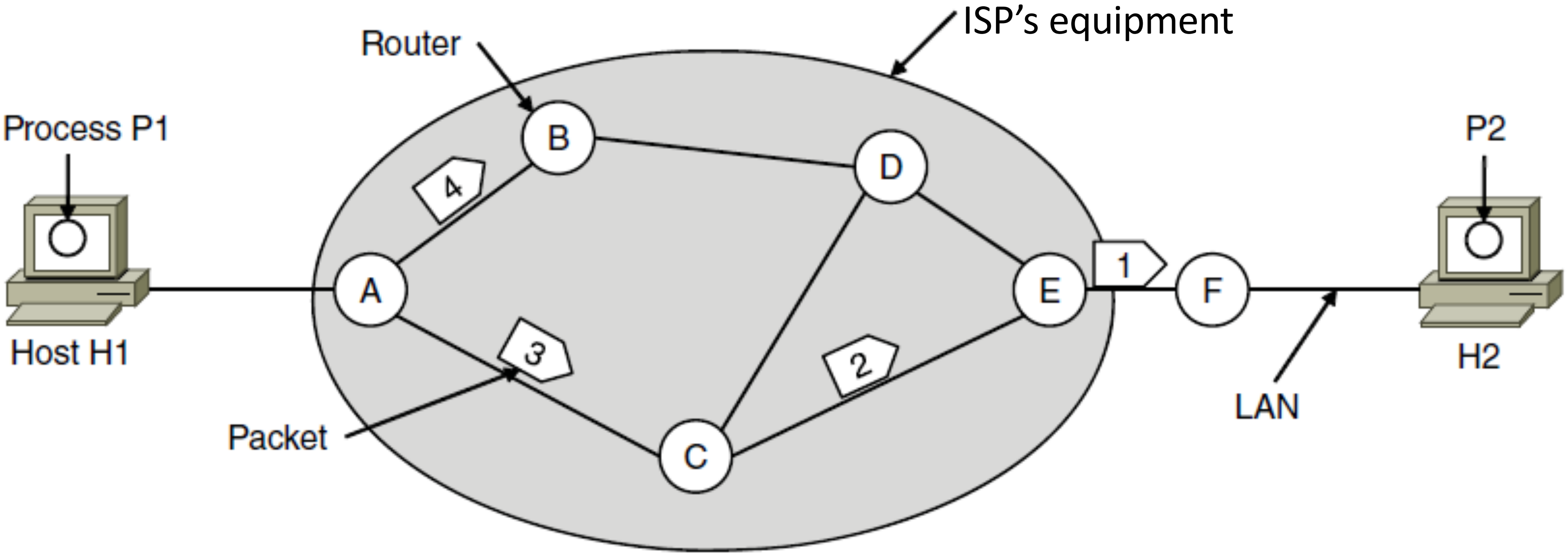
# Two Network Service Models

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

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

# 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

A's table (initially)

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

A's table (later)

| | |
|---|---|
| A | |
| B | B |
| C | C |
| D | B |
| E | B |
| F | B |

C's Table

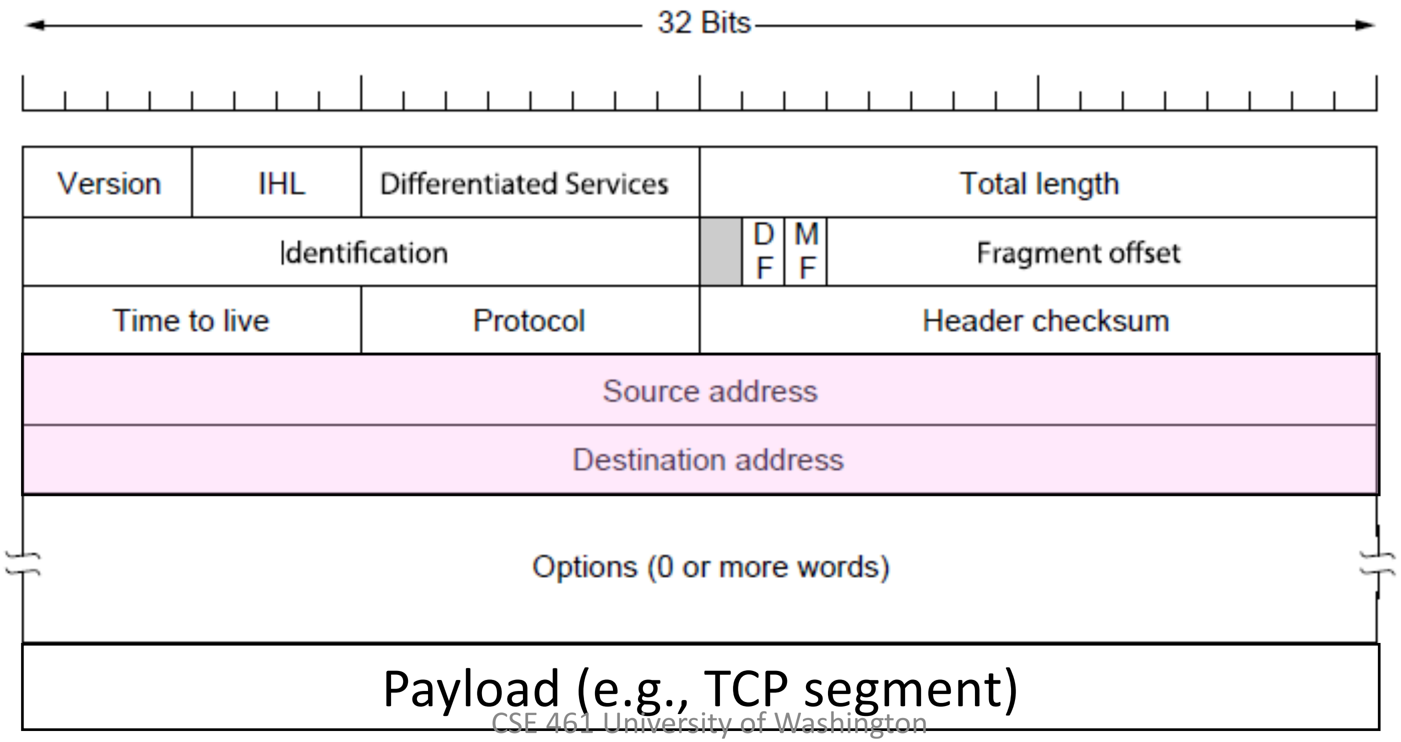
| | |
|---|---|
| A | A |
| B | A |
| C | |
| D | E |
| E | E |
| F | E |

E's Table

| | |
|---|---|
| A | C |
| B | D |
| C | C |
| D | D |
| E | |
| F | F |

# IP (Internet Protocol)

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



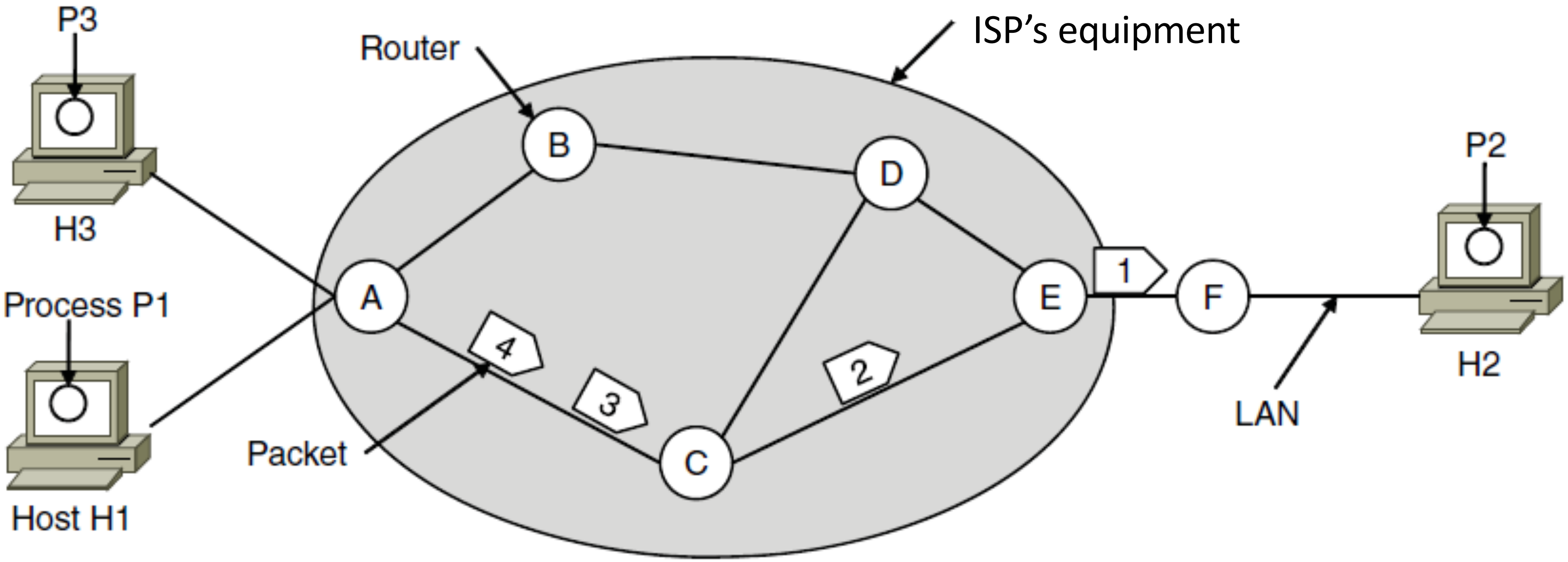
# Virtual Circuit Model

- 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

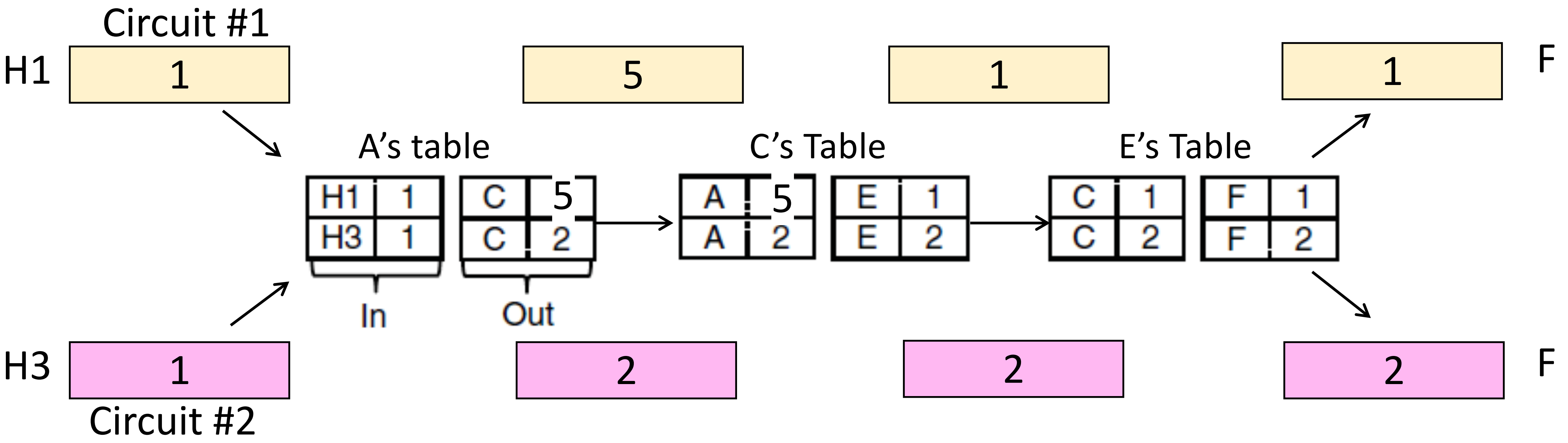
# Virtual Circuits

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



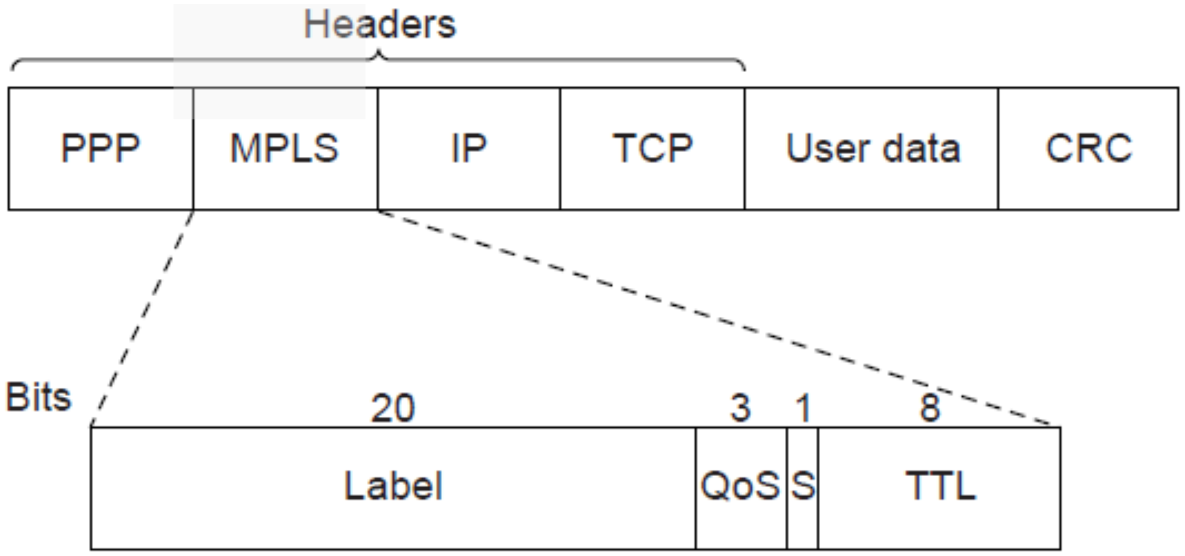
# Virtual Circuits (2)

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



# MPLS (Multi-Protocol Label Switching, §5.6.5)

- 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

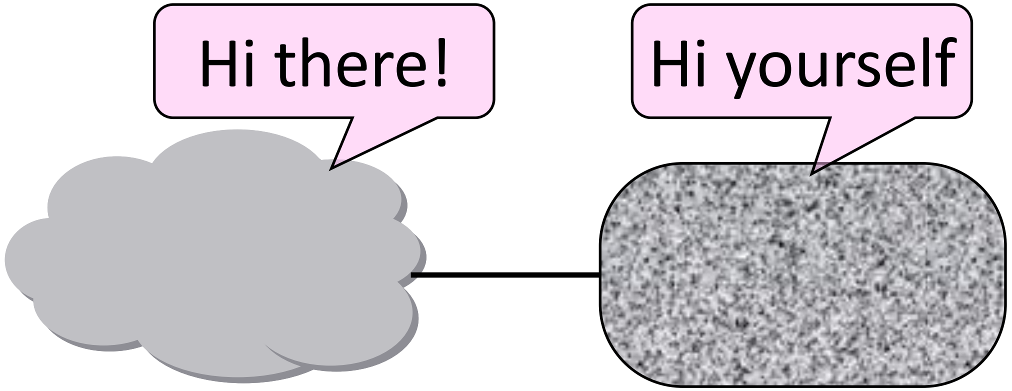
- Complementary strengths

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

# Internetworking (IP)

# Topic

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



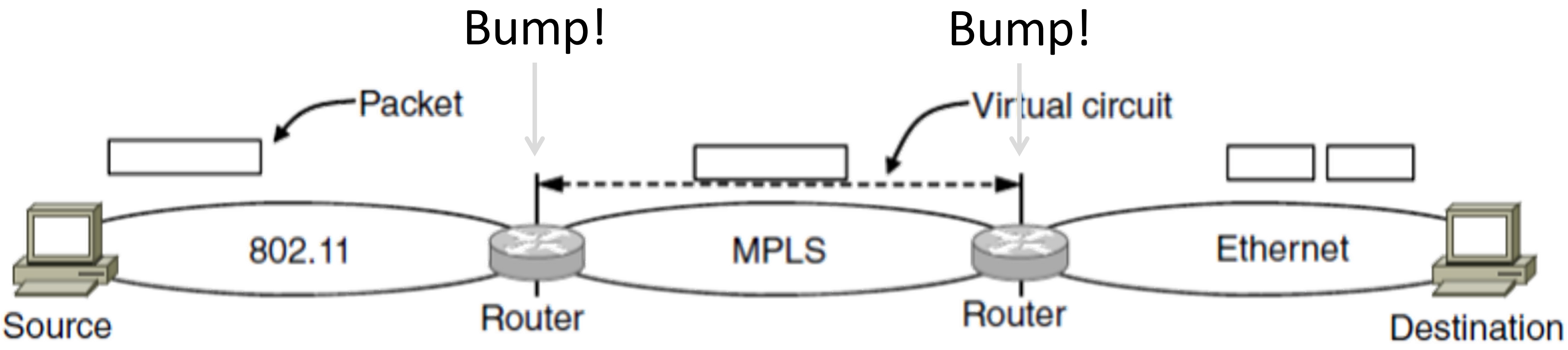
# How Networks May Differ

- 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

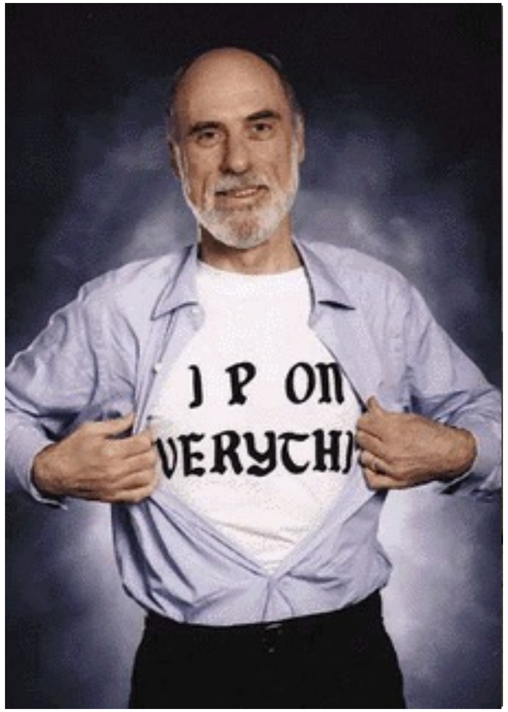
- An example to show that it's not so easy
  - Need to map destination address to a VC and vice-versa
  - A bit of a "road bump", e.g., might have to set up a VC



# 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 technology

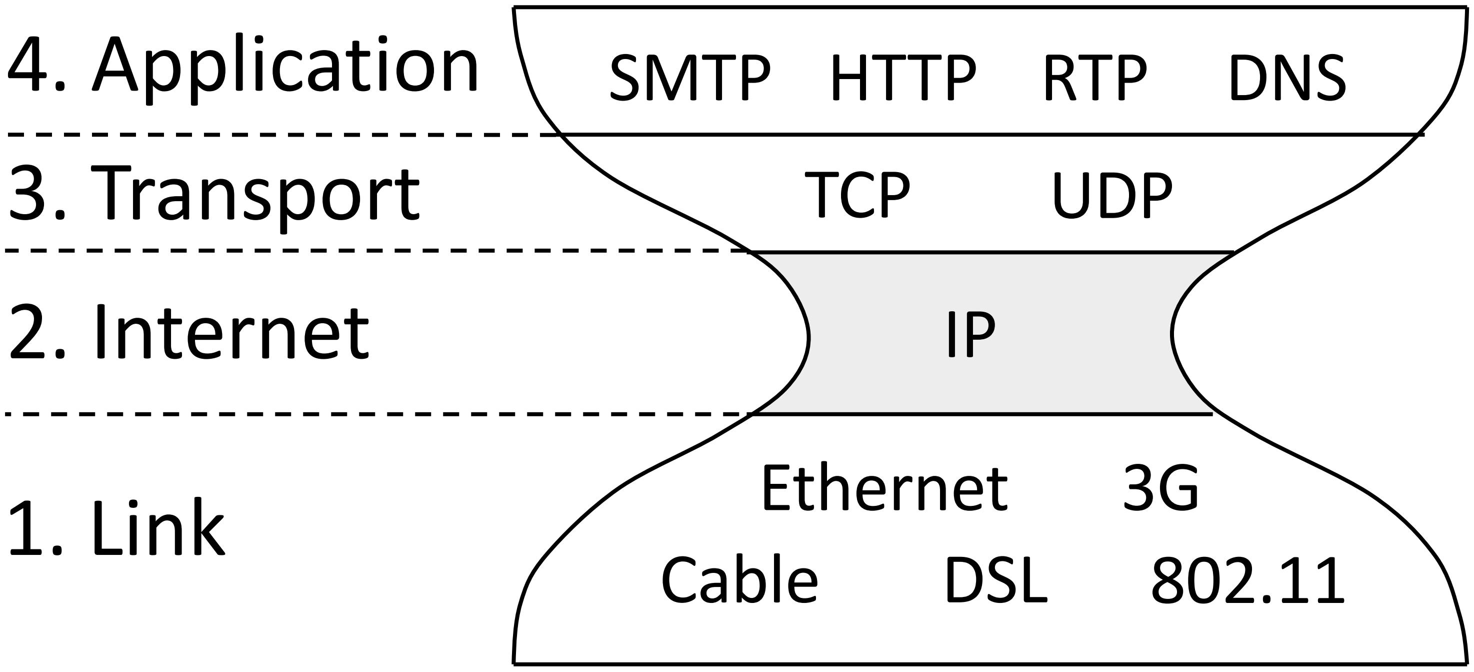
Vint Cerf



Bob Kahn

# Internet Reference Model

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

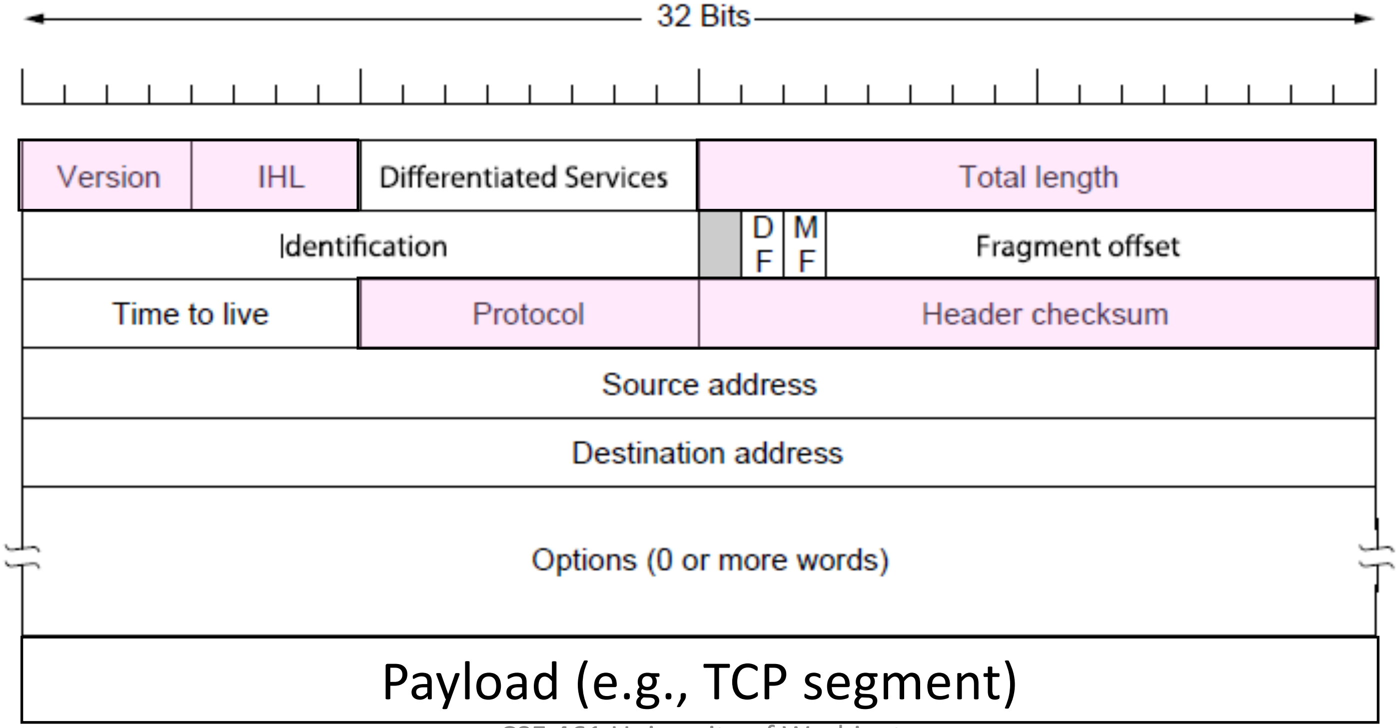


# IP as a Lowest Common Denominator

- 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

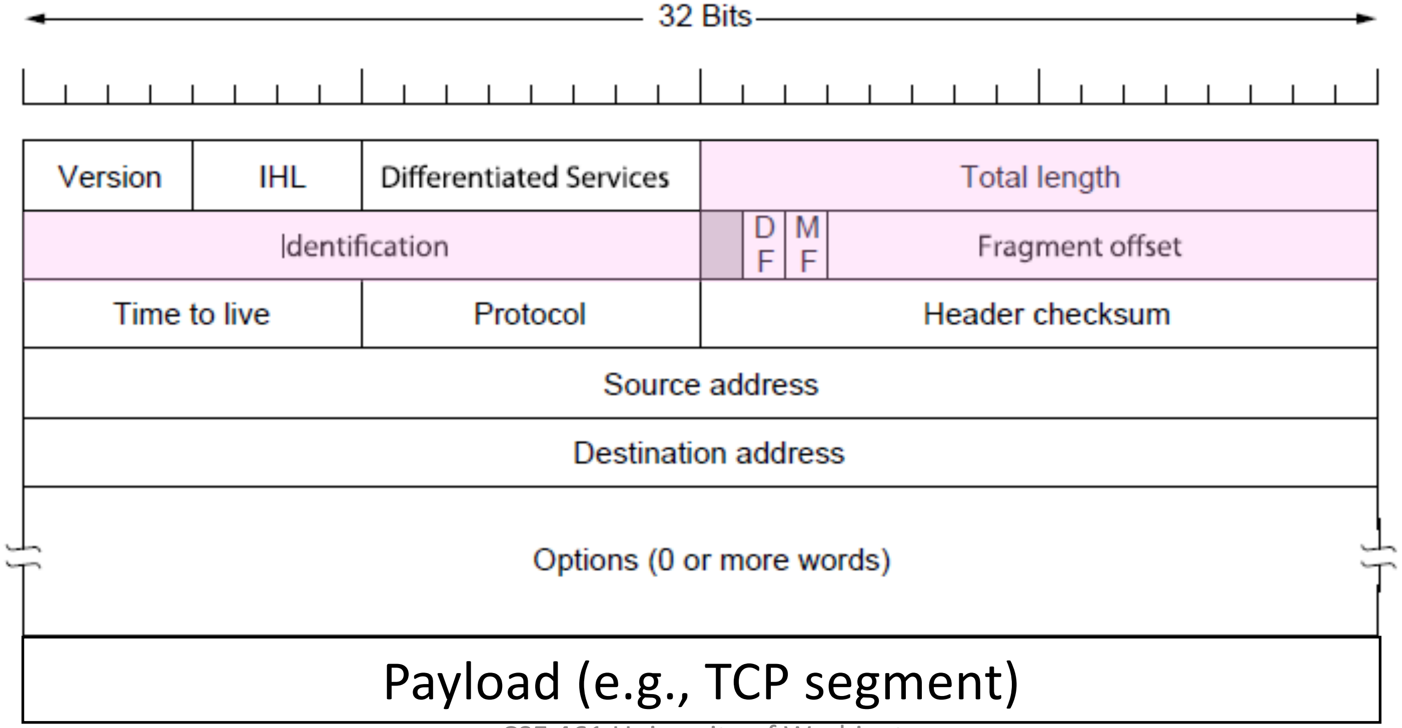
# IPv4 (Internet Protocol)

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



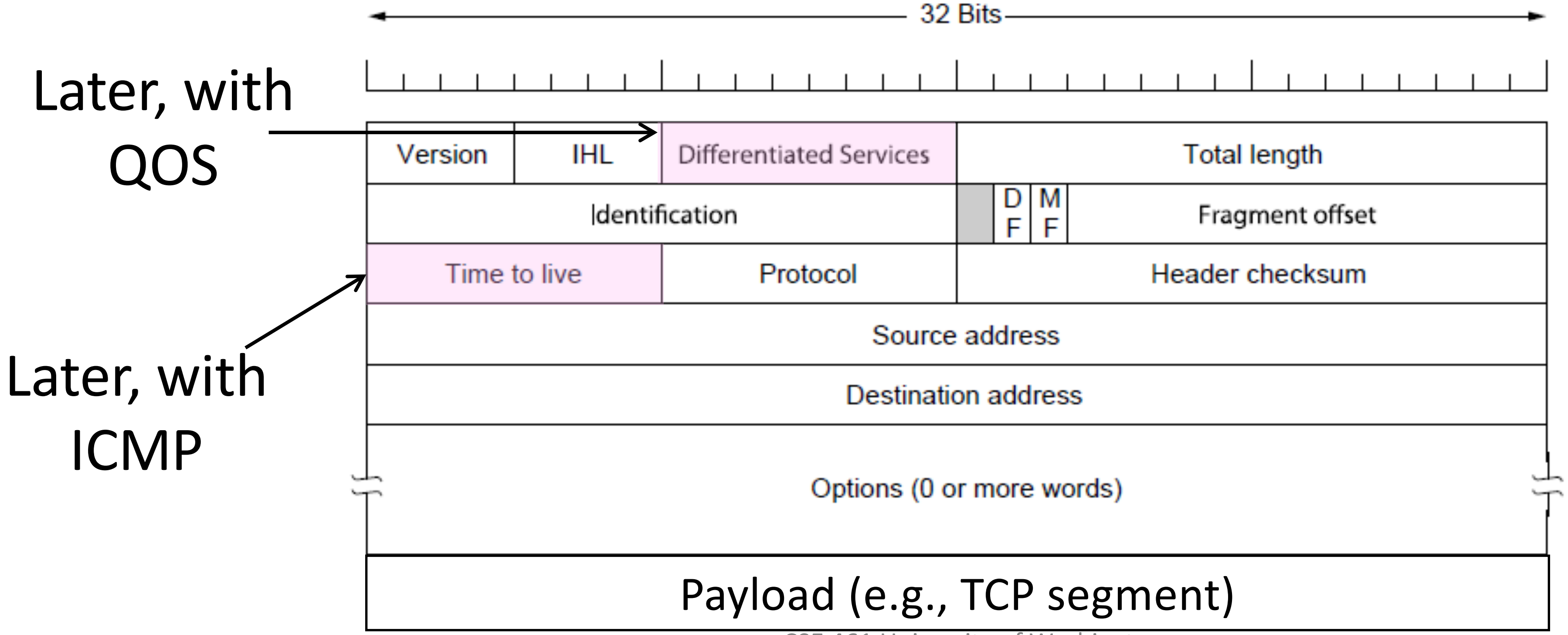
# IPv4 (2)

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



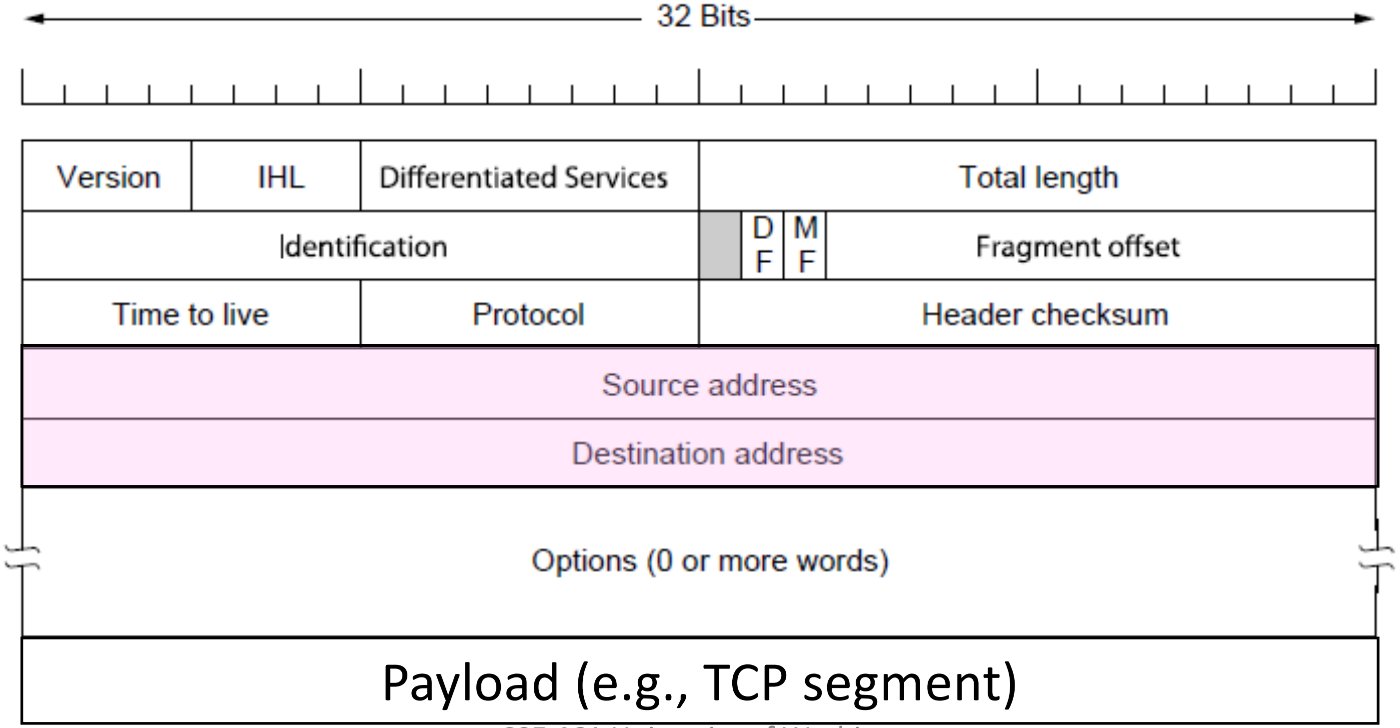
# IPv4 (3)

- 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)



# Recap of network layer

Needed to deliver packets to their destination

Solves three problems

- Internetworking
- Addressing
- Routing and forwarding

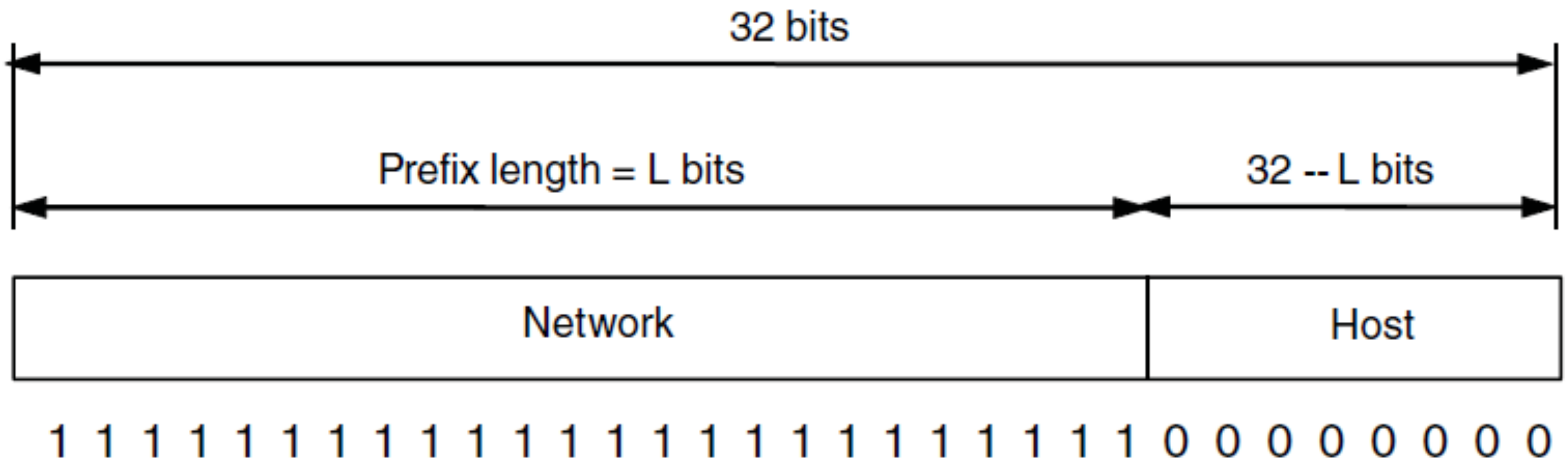
# IP Addresses

- 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

| 8 bits | 8 bits | 8 bits | 8 bits | | |
|---|---|---|---|---|---|
| aaaaaaaa | bbbbbbbb | cccccccc | dddddddd | ↔ | A.B.C.D |
| 00010010 | 00011111 | 00000000 | 00000001 | ↔ | ?? |

# 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



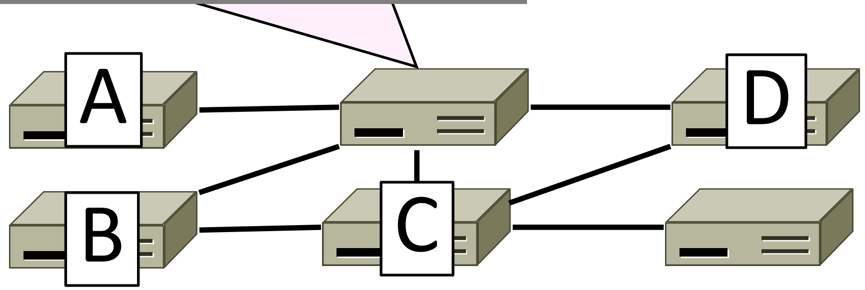
# IP Prefixes (2)

- 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 /32 is 1 address

# IP Forwarding

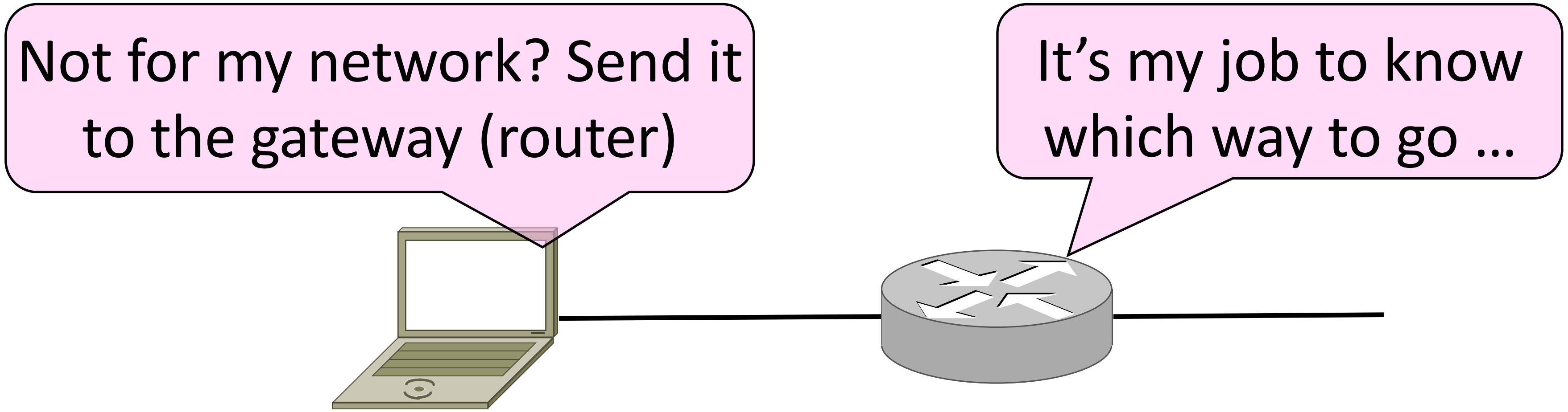
- Nodes use a table that lists the next hop for prefixes
- Lookup the destination address's prefix in the table

| Prefix | Next Hop |
|---|---|
| 102.24.0.0/19 | D |
| 192.24.12.0/22 | B |



# 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

| Prefix | Next Hop |
|---|---|
| My network prefix | Send on local link |
| Default (0.0.0.0/0) | Send to my router |

```
[Ratuls-MacBook-Pro:19wi ratul$ netstat -r -f inet | grep 192
default             192.168.88.1       UGSc          85        30      en0
192.168.88          link#10            UCS            0         0      en0        !
192.168.88.1/32     link#10            UCS            2         0      en0        !
192.168.88.14/32    link#10            UCS            0         0      en0        !
```

# Issues?

- Where does this break down?

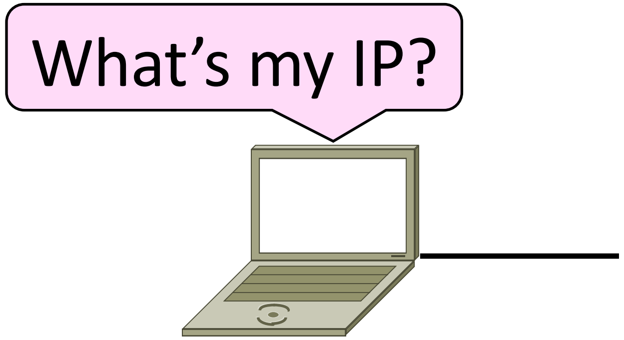
Bootstrapping (DHCP)

Finding Link nodes (ARP)

# Dynamic Host Configuration Protocol (DHCP)

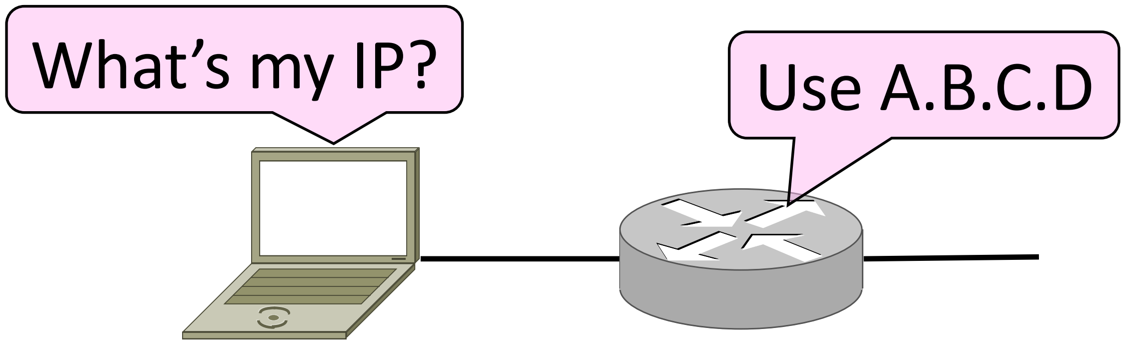
# Bootstrapping

- 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



# DHCP: Dynamic Host Configuration Protocol

- Invented around 1993, widely used now
- It <u>leases</u> IP address to nodes
- Provides other parameters too
  - Network prefix
  - Address of local router
  - DNS server, time server, etc.

# DHCP Protocol Stack

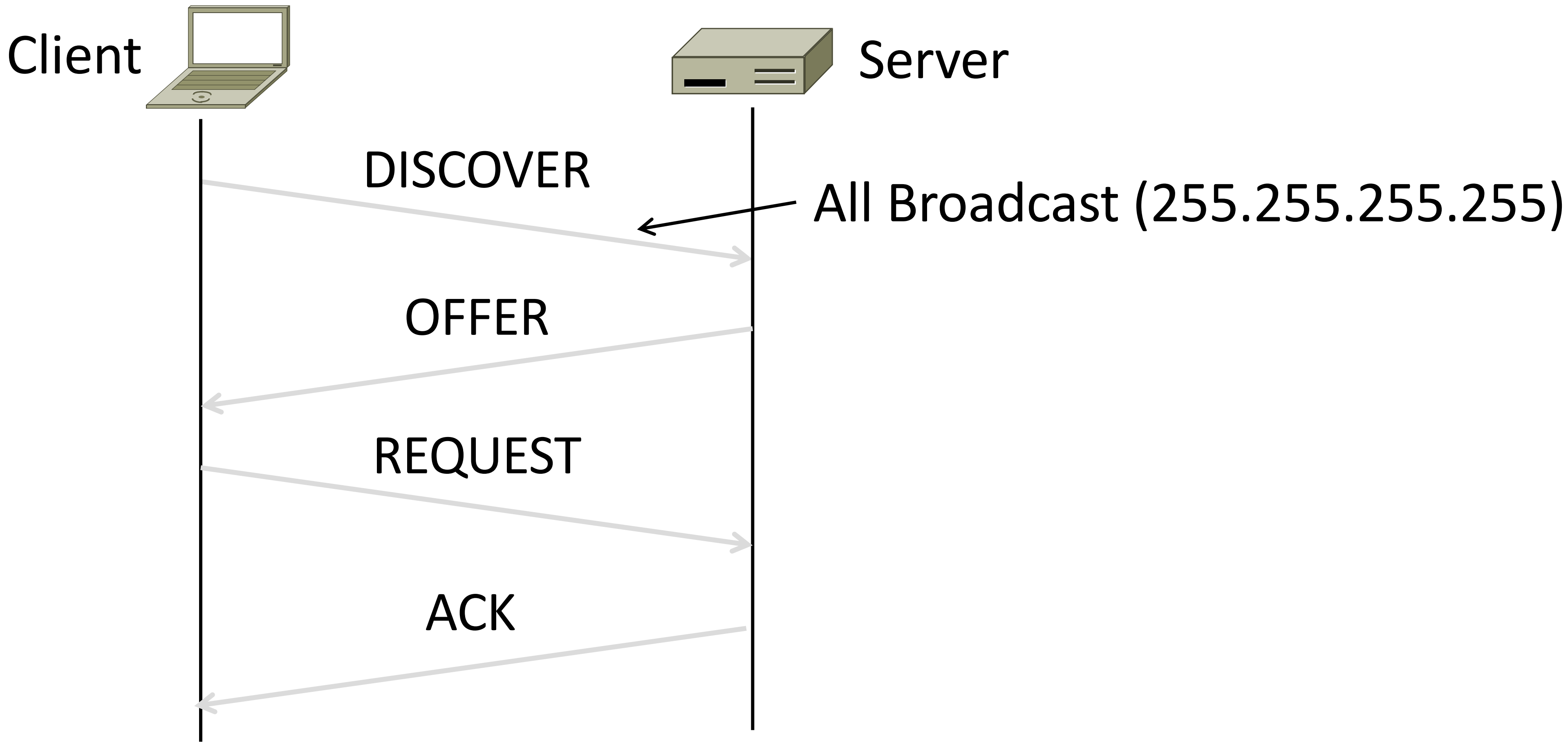
- DHCP is a client-server application
  - Uses UDP ports 67, 68

| |
|---|
| DHCP |
| UDP |
| IP |
| Ethernet |

# DHCP Addressing

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

# DHCP Messages



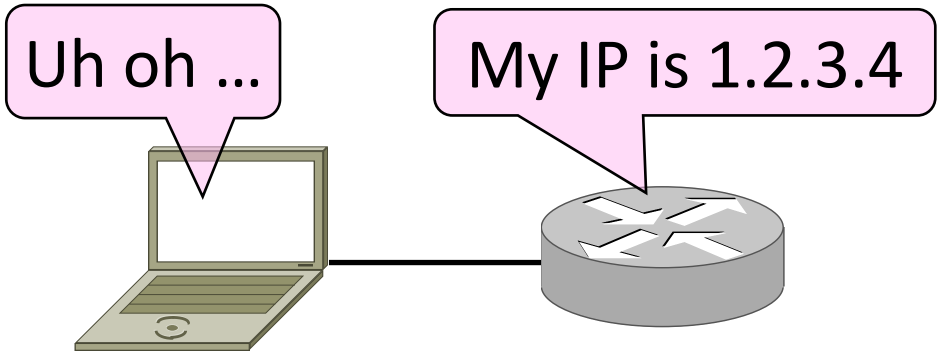
# DHCP Messages (2)

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

# Address Resolution Protocol (ARP)

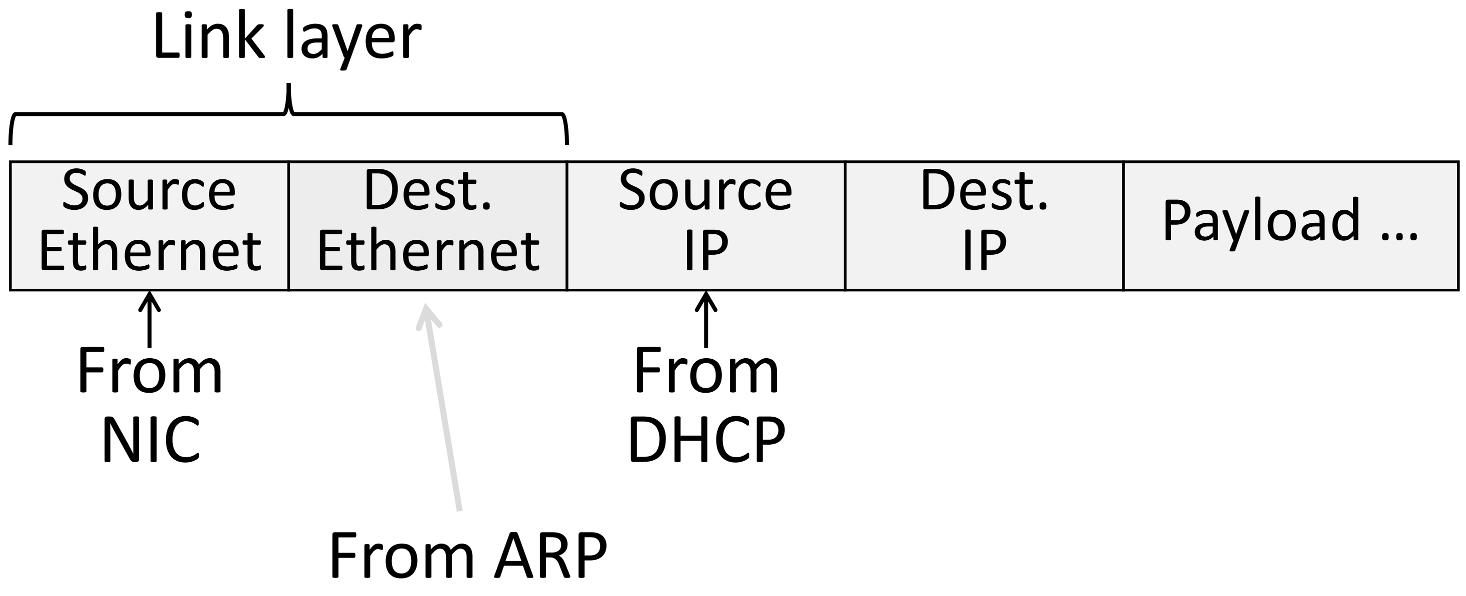
# Sending an IP Packet

- 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?



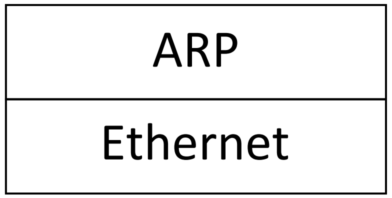
# ARP (Address Resolution Protocol)

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

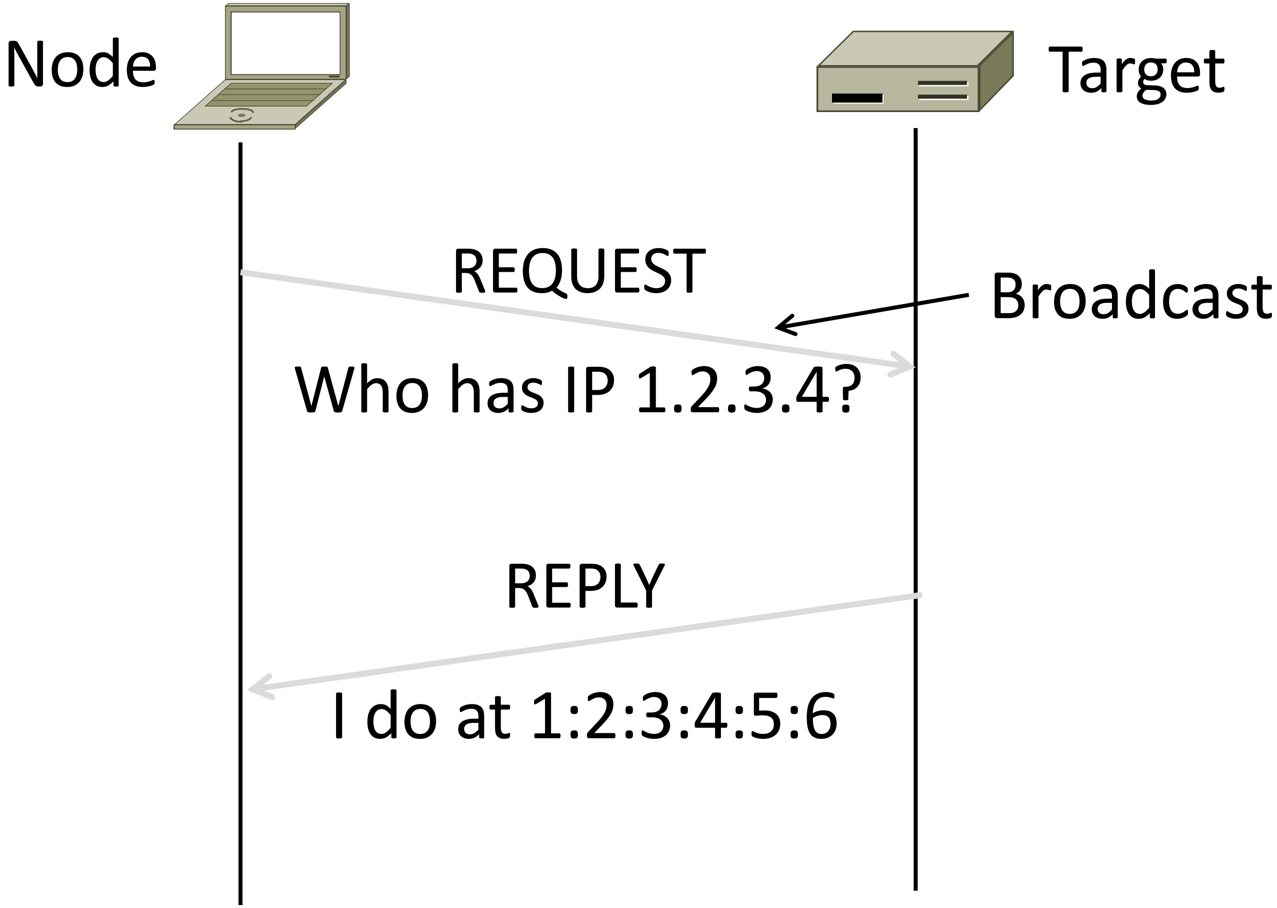


# ARP Protocol Stack

- 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




[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

```
[Ratuls-MacBook-Pro:19wi ratul$ arp -a | grep 192
? (192.168.88.1) at e4:8d:8c:54:0:52 on en0 ifscope [ethernet]
```

# Discovery Protocols

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