CSCI-1680
Network Layer:
IP & Forwarding

Rodrigo Fonseca

Instructor: Nicholas DeMarinis

Based partly on lecture notes by David Mazières, Phil Levis, John Jannotti

Administrivia

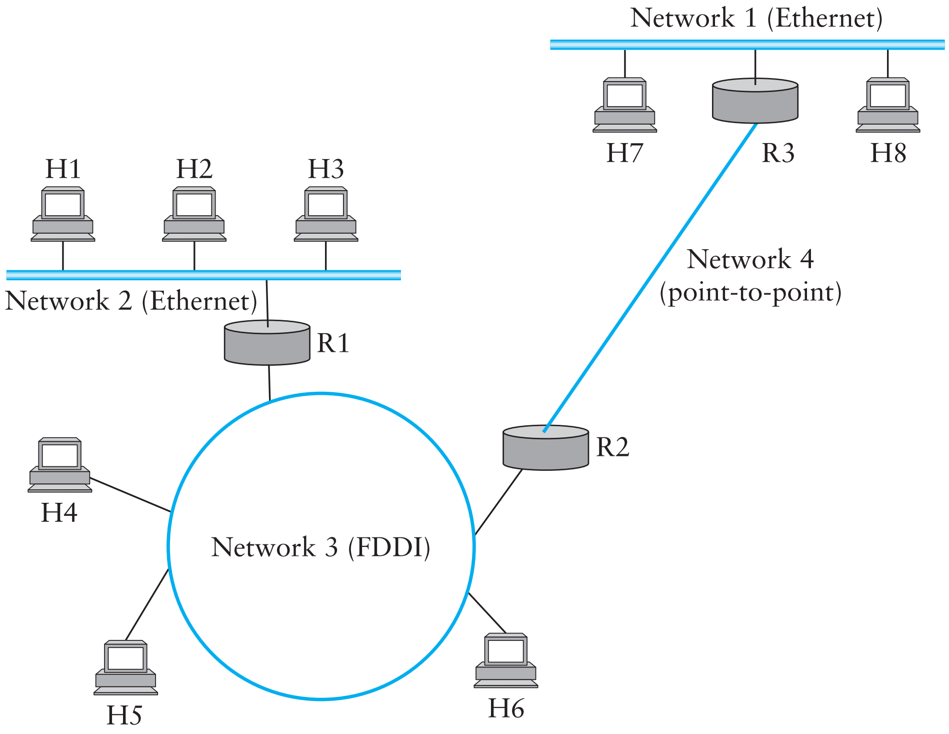
- **IP out today. Your job:**
  - Find partners, get setup with Github
  - Implement IP forwarding and DV routing
  - Get started TODAY ☺
- **HW1 due today**

Today

- **Network layer: Internet Protocol (v4)**
- **Forwarding**
  - Addressing
  - Fragmentation
  - ARP
  - DHCP
  - NATs
- **Next 2 classes: Routing**

Internet Protocol Goal

- **How to connect everybody?**
  - New global network or connect existing networks?
- **Glue lower-level networks together:**
  - allow packets to be sent between any pair or hosts
- **Wasn't this the goal of switching?**



Internetworking Challenges

- **Heterogeneity**
  - Different addresses
  - Different service models
  - Different allowable packet sizes
- **Scaling**
- **Congestion control**

How would you design such a protocol?

- **Circuits or packets?**
  - Predictability
- **Service model**
  - Reliability, timing, bandwidth guarantees
- **Any-to-any**
  - Finding nodes: naming, routing
  - Maintenance (join, leave, add/remove links,…)
  - Forwarding: message formats

IP's Decisions

- **Packet switched**
  - Unpredictability, statistical multiplexing
- **Service model**
  - Lowest common denominator: best effort, connectionless datagram
- **Any-to-any**
  - Common message format
  - Separated routing from forwarding
  - Naming: uniform addresses, hierarchical organization
  - Routing: hierarchical, prefix-based (longest prefix matching)
  - Maintenance: delegated, hierarchical

A Bit of History

- **Packet switched networks: Arpanet’s IMPs**
  - Late 1960’s
  - RFC 1, 1969!
  - Segmentation, framing, routing, reliability, reassembly, primitive flow control
- **Network Control Program (NCP)**
  - Provided connections, flow control
  - Assumed reliable network: IMPs
  - Used by programs like telnet, mail, file transfer
- **Wanted to connect multiple networks**
  - Not all reliable, different formats, etc…

TCP/IP Introduced

- **Vint Cerf, Robert Kahn**
- **Replace NCP**
- **Initial design: single protocol providing a unified reliable pipe**
  - Could support any application
- **Different requirements soon emerged, and the two were separated**
  - IP: basic datagram service among hosts
  - TCP: reliable transport
  - UDP: unreliable *multiplexed* datagram service

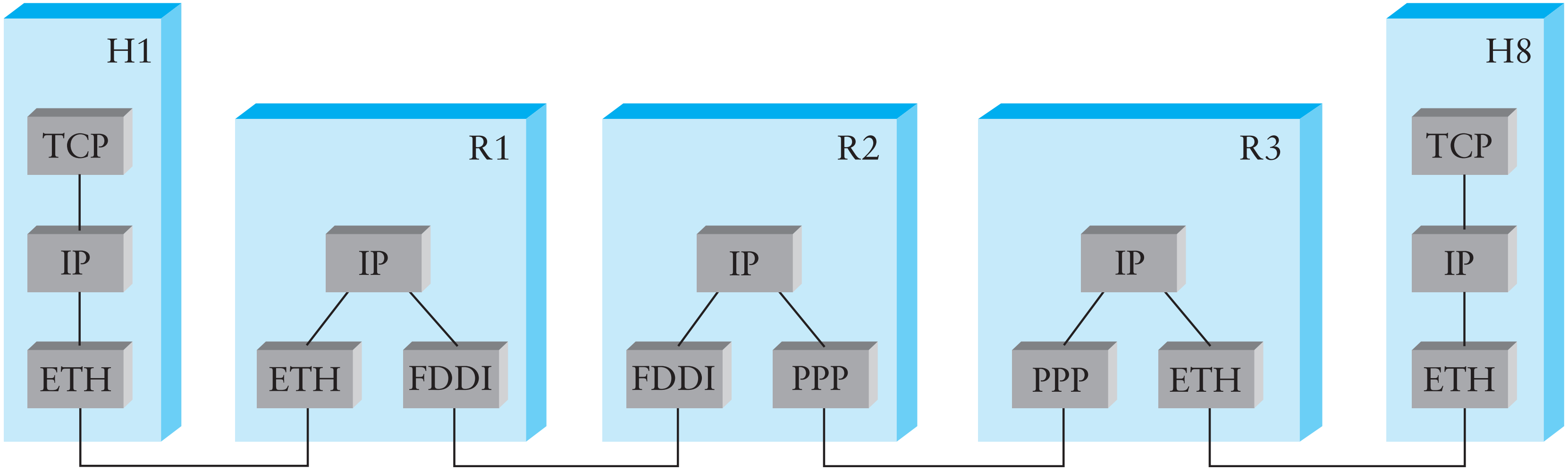
An excellent read

David D. Clark, "The design Philosophy of the DARPA Internet Protocols", 1988

- Primary goal: multiplexed utilization of existing interconnected networks
- Other goals:
  - Communication continues despite loss of networks or gateways
  - Support a variety of communication services
  - Accommodate a variety of networks
  - Permit distributed management of its resources
  - Be cost effective
  - Low effort for host attachment
  - Resources must be accountable

Internet Protocol

- **IP Protocol running on all hosts and *routers***
- **Routers are present in all networks they join**
- **Uniform addressing**
- **Forwarding/Fragmentation**
- **Complementary:**
  - Routing, Error Reporting, Address Translation



IP Protocol

- **Provides addressing and *forwarding***
  - Addressing is a set of conventions for naming nodes in an IP network
  - Forwarding is a local action by a router: passing a packet from input to output port
- **IP forwarding finds output port based on destination address**
  - Also defines certain conventions on how to handle packets (e.g., fragmentation, time to live)
- **Contrast with *routing***
  - Routing is the process of determining how to map packets to output ports (topic of next two lectures)

Service Model

- **Connectionless (datagram-based)**
- **Best-effort delivery (unreliable service)**
  - packets may be lost
  - packets may be delivered out of order
  - duplicate copies of packets may be delivered
  - packets may be delayed for a long time
- **It's the lowest common denominator**
  - A network that delivers no packets fits the bill!
  - All these can be dealt with above IP (if probability of delivery is non-zero…)

IP v4 packet format

<table>
<tr><td>0</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>1</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>2</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>3</td><td></td></tr>
<tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>0</td><td>1</td></tr>
<tr><td colspan="4">vers</td><td colspan="4">hdr len</td><td colspan="8">TOS</td><td colspan="16">Total Length</td></tr>
<tr><td colspan="16">Identification</td><td>0</td><td>DF</td><td>MF</td><td colspan="13">Fragment offset</td></tr>
<tr><td colspan="8">TTL</td><td colspan="8">Protocol</td><td colspan="16">hdr checksum</td></tr>
<tr><td colspan="32">Source IP address</td></tr>
<tr><td colspan="32">Destination IP address</td></tr>
<tr><td colspan="24">Options</td><td colspan="8">Padding</td></tr>
<tr><td colspan="32">Data</td></tr>
</table>

IP header details

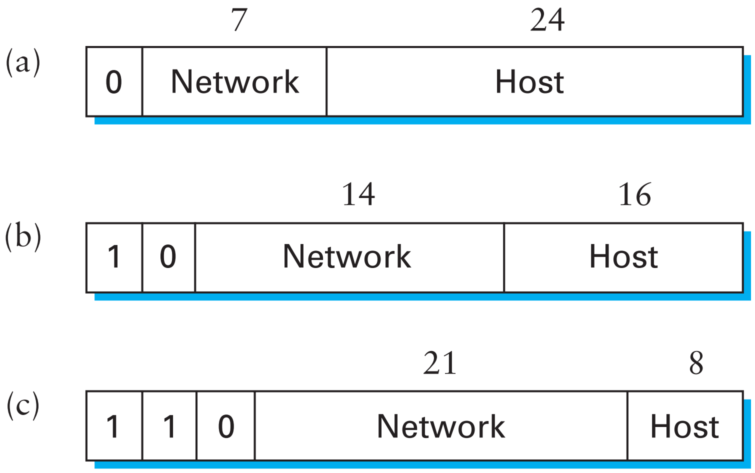
- **Forwarding based on destination address**
- **TTL (time-to-live) decremented at each hop**
  - Originally was in seconds (no longer)
  - Mostly prevents forwarding loops
  - Other cool uses…
- **Fragmentation possible for large packets**
  - Fragmented in network if crossing link w/ small frame
  - MF: more fragments for this IP packet
  - DF: don't fragment (returns error to sender)
- **Following IP header is "payload" data**
  - Typically beginning with TCP or UDP header

Other fields

- Version: 4 (IPv4) for most packets, there's also 6
- Header length: in 32-bit units (>5 implies options)
- Type of service (won't go into this)
- Protocol identifier (TCP: 6, UDP: 17, ICMP: 1, …)
- Checksum over the *header*

Format of IP addresses

- **Globally unique (or made seem that way)**
  - 32-bit integers, read in groups of 8-bits: 128.148.32.110
- **Hierarchical: network + host**
- **Originally, routing prefix embedded in address**



  - Class A (8-bit prefix), B (16-bit), C (24-bit)
  - Routers need only know route for each network

Forwarding Tables

- **Exploit hierarchical structure of addresses: need to know how to reach *networks*, not hosts**

Network	Next Address
212.31.32.*	0.0.0.0
18.*.*.*	212.31.32.5
128.148.*.*	212.31.32.4
Default	212.31.32.1

- **Keyed by network portion, not entire address**
- **Next address should be local: router knows how to reach it directly* (we'll see how soon)**

Classed Addresses

- **Hierarchical: network + host**
  - Saves memory in backbone routers (no default routes)
  - Originally, routing prefix embedded in address
  - Routers in same network must share network part
- **Inefficient use of address space**
  - Class C with 2 hosts (2/255 = 0.78% efficient)
  - Class B with 256 hosts (256/65535 = 0.39% efficient)
  - Shortage of IP addresses
  - Makes address authorities reluctant to give out class B's
- **Still too many networks**
  - Routing tables do not scale
- **Routing protocols do not scale**

CIDR: Classless Inter-Domain Routing

- **Problems: routing table growth, granularity of allocation**
- **Idea: assign blocks of contiguous networks to nearby networks**
- **Represent blocks with a single pair**
  - (first network address, count)
- **Restrict block sizes to powers of 2**
- **Use a bit mask (CIDR mask) to identify block size**
- **Address aggregation: reduce routing tables**

Obtaining IP Addresses

- **Blocks of IP addresses allocated hierarchically**
  - ISP obtains an address block, may subdivide

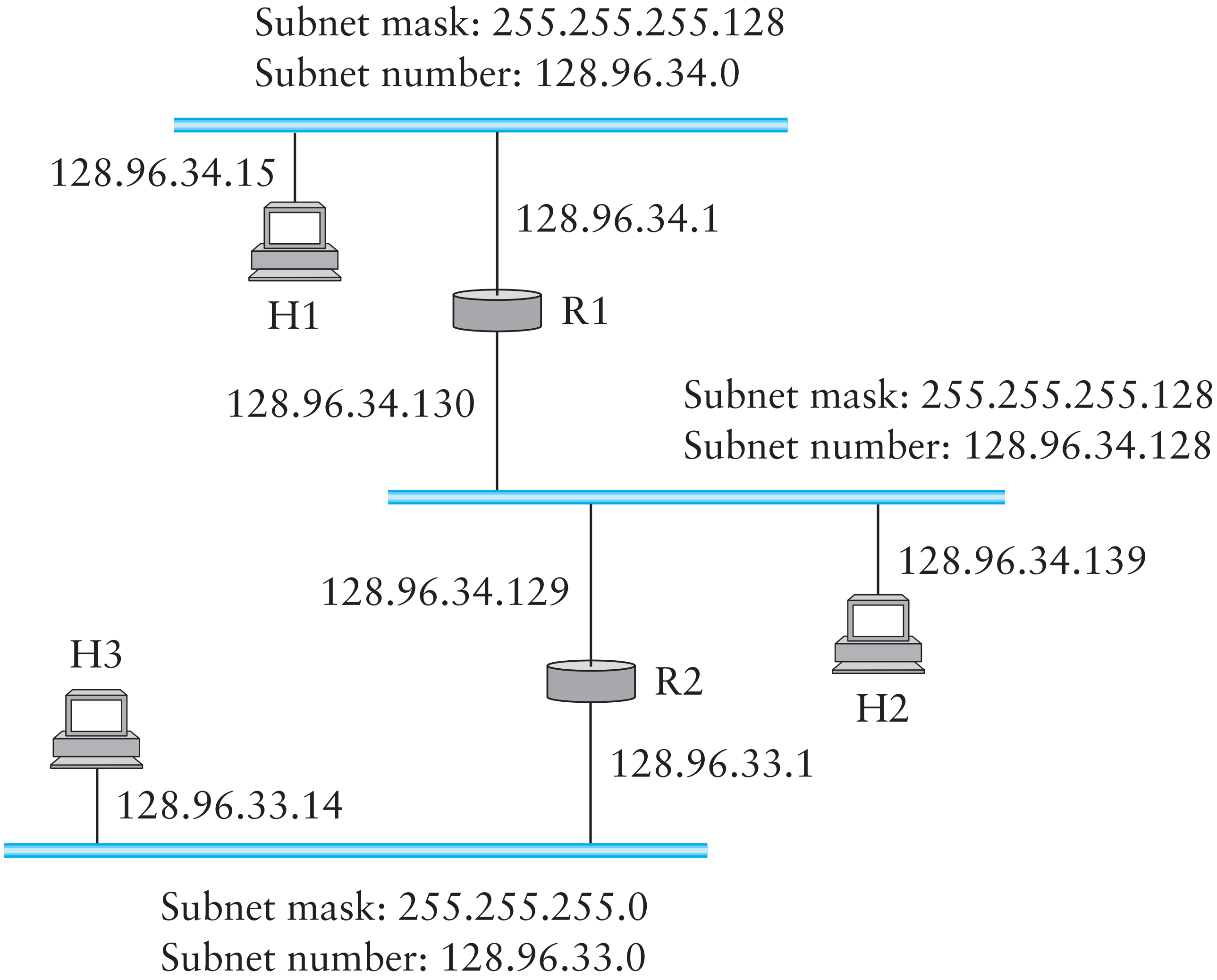
ISP: 128.35.16/20 10000000 00100011 00010000 00000000
Client 1: 128.35.16/22 10000000 00100011 00010000 00000000
Client 2: 128.35.20/22 10000000 00100011 00010100 00000000
Client 3: 128.35.24/21 10000000 00100011 00011000 00000000

- **Global allocation: ICANN, /8's (ran out!)**
- **Regional registries: ARIN, RIPE, APNIC, LACNIC, AFRINIC**

CIDR Forwarding Table

Network	Next Address
212.31.32/24	0.0.0.0
18/8	212.31.32.5
128.148/16	212.31.32.4
128.148.128/17	212.31.32.8
0/0	212.31.32.1

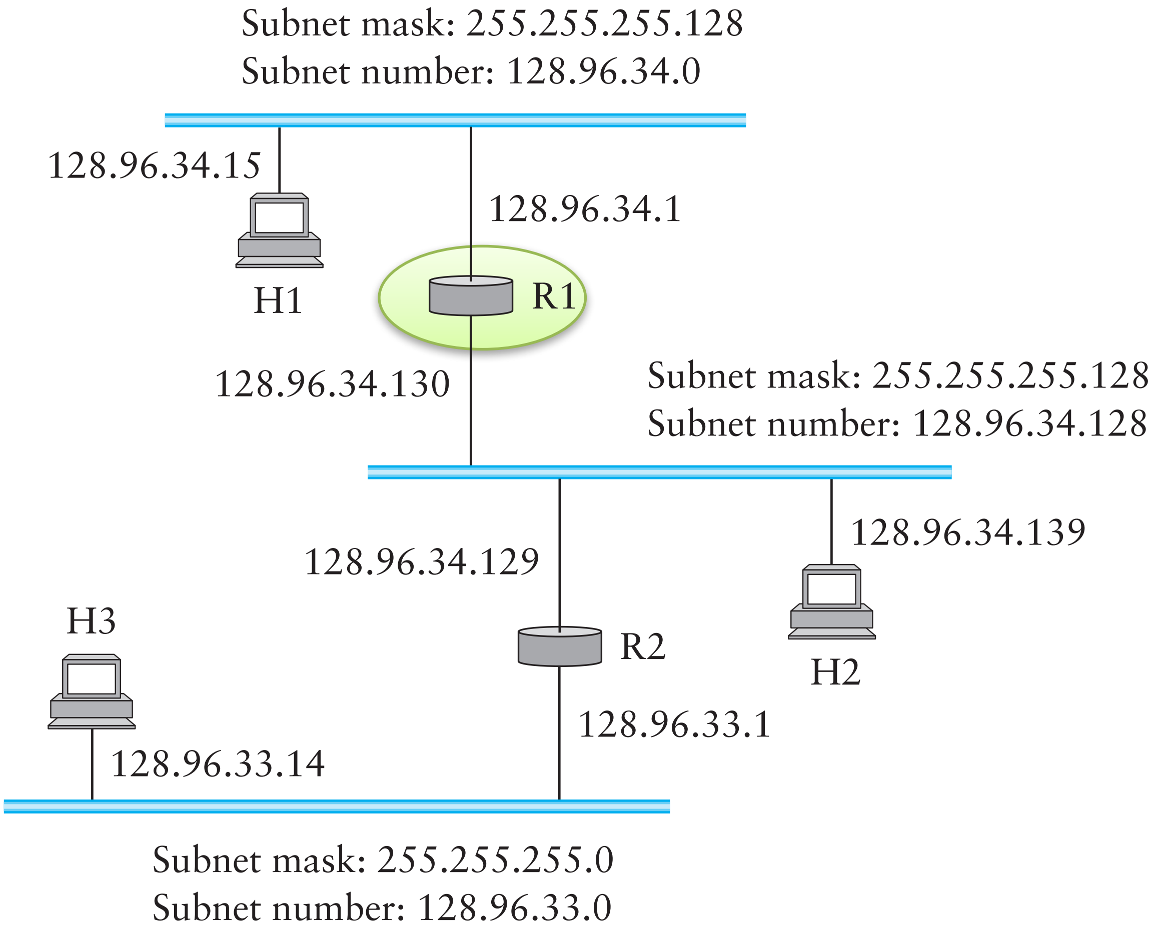
Example



H1-> H2: H2.ip & H1.mask != H1.subnet => no direct path

R1’s Forwarding Table

Network	Subnet Mask	Next Address
128.96.34.0	255.255.255.128	128.96.34.1
128.96.34.128	255.255.255.128	128.96.34.130
128.96.33.0	255.255.255.0	128.96.34.129



Translating IP to lower level addresses or… How to reach these *local* addresses?

- **Map IP addresses into physical addresses**
  - E.g., Ethernet address of destination host
  - or Ethernet address of next hop router
- **Techniques**
  - Encode physical address in host part of IP address (IPv6)
  - Each network node maintains lookup table (IP->phys)

ARP – *address resolution protocol*

- Dynamically builds table of IP to physical address bindings for a *local network*
- Broadcast request if IP address not in table
- All learn IP address of requesting node (broadcast)
- Target machine responds with its physical address
- Table entries are discarded if not refreshed

ARP Ethernet frame format

<table>
<tr><th>0</th><th>8</th><th>16</th><th>31</th></tr>
<tr><td colspan="2">Hardware type = 1</td><td colspan="2">ProtocolType = 0x0800</td></tr>
<tr><td>HLen = 48</td><td>PLen = 32</td><td colspan="2">Operation</td></tr>
<tr><td colspan="4">SourceHardwareAddr (bytes 0–3)</td></tr>
<tr><td colspan="2">SourceHardwareAddr (bytes 4–5)</td><td colspan="2">SourceProtocolAddr (bytes 0–1)</td></tr>
<tr><td colspan="2">SourceProtocolAddr (bytes 2–3)</td><td colspan="2">TargetHardwareAddr (bytes 0–1)</td></tr>
<tr><td colspan="4">TargetHardwareAddr (bytes 2–5)</td></tr>
<tr><td colspan="4">TargetProtocolAddr (bytes 0–3)</td></tr>
</table>

- **Why include source hardware address?**

Obtaining Host IP Addresses - DHCP

- **Networks are free to assign addresses within block to hosts**
- **Tedious and error-prone: e.g., laptop going from CIT to library to coffee shop**
- **Solution: Dynamic Host Configuration Protocol**
  - Client: DHCP Discover to 255.255.255.255 (broadcast)
  - Server(s): DHCP Offer to 255.255.255.255 (why broadcast?)
  - Client: choose offer, DHCP Request (broadcast, why?)
  - Server: DHCP ACK (again broadcast)
- **Result: address, gateway, netmask, DNS server**

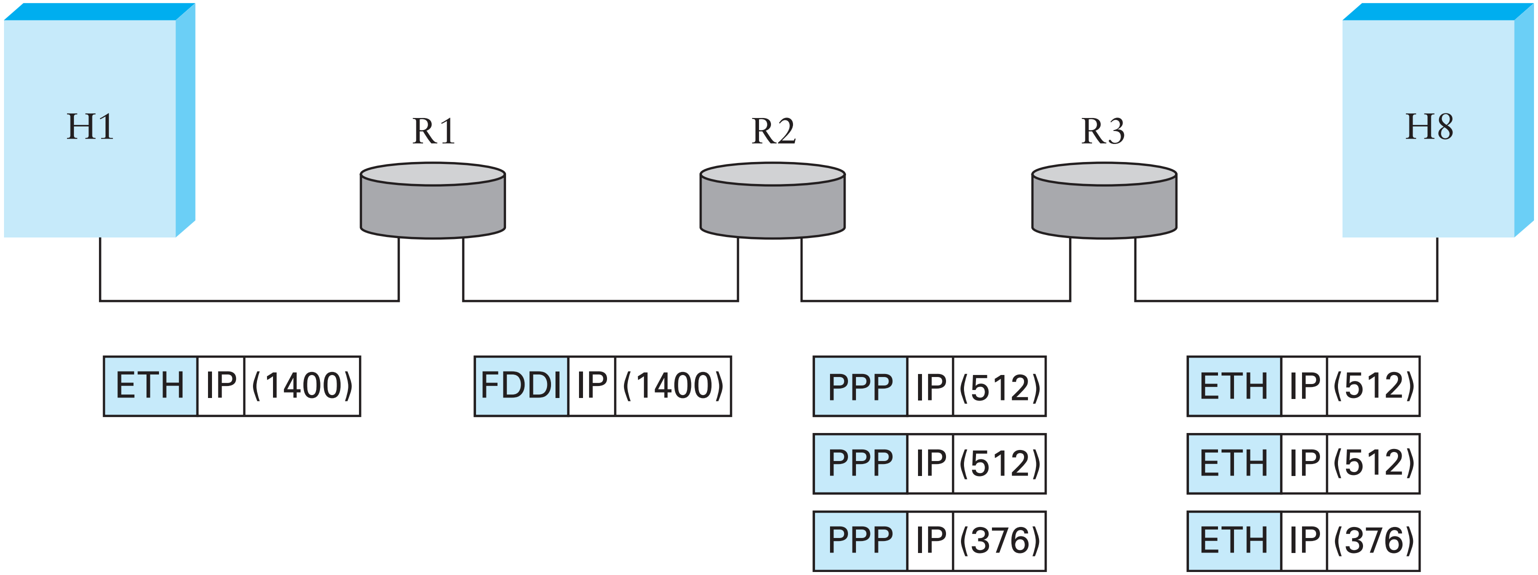
Network Address Translation (NAT)

- Despite CIDR, it's still difficult to allocate addresses (2^{32} is only 4 billion)
- We'll talk about IPv6 later
- NAT "hides" entire network behind one address
- Hosts are given *private* addresses
- Routers map outgoing packets to a free address/port
- Router reverse maps incoming packets
- Problems?

Fragmentation & Reassembly

- **Each network has maximum transmission unit (MTU)**
- **Strategy**
  - Fragment when necessary (MTU < size of datagram)
  - Source tries to avoid fragmentation (why?)
  - Re-fragmentation is possible
  - Fragments are self-contained datagrams
  - Delay reassembly until destination host
  - No recovery of lost fragments

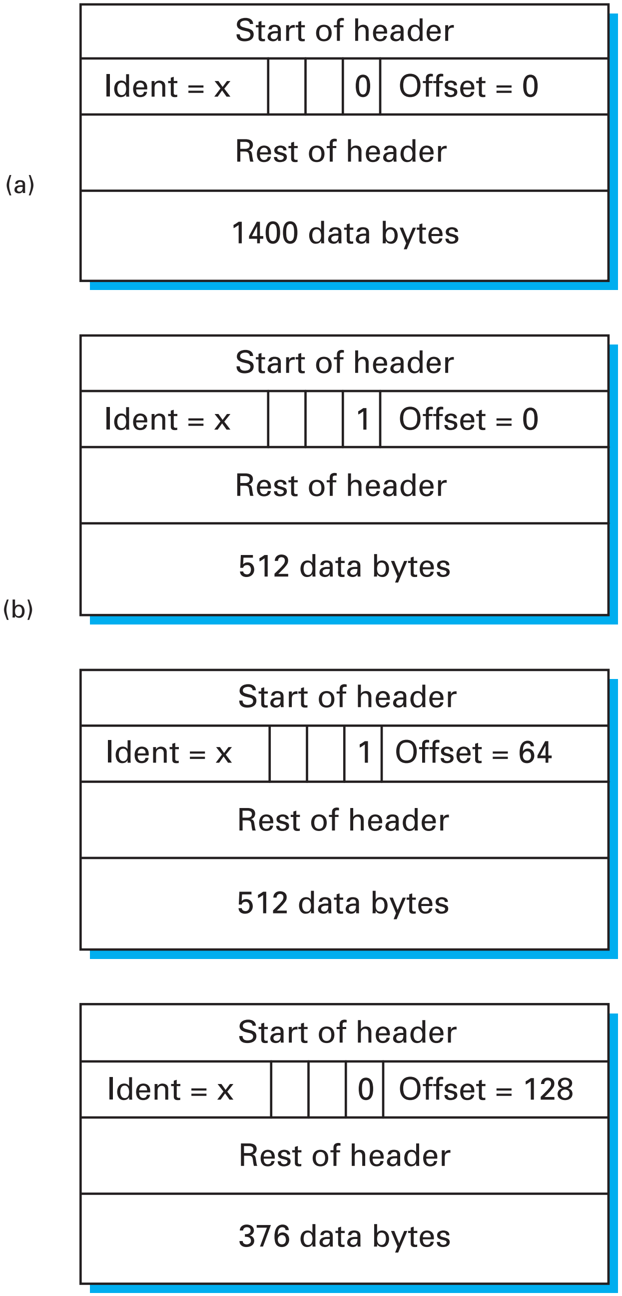
Fragmentation Example



- **Ethernet MTU is 1,500 bytes**
- **PPP MTU is 576 bytes**
  - R2 must fragment IP packets to forward them

Fragmentation Example (cont)

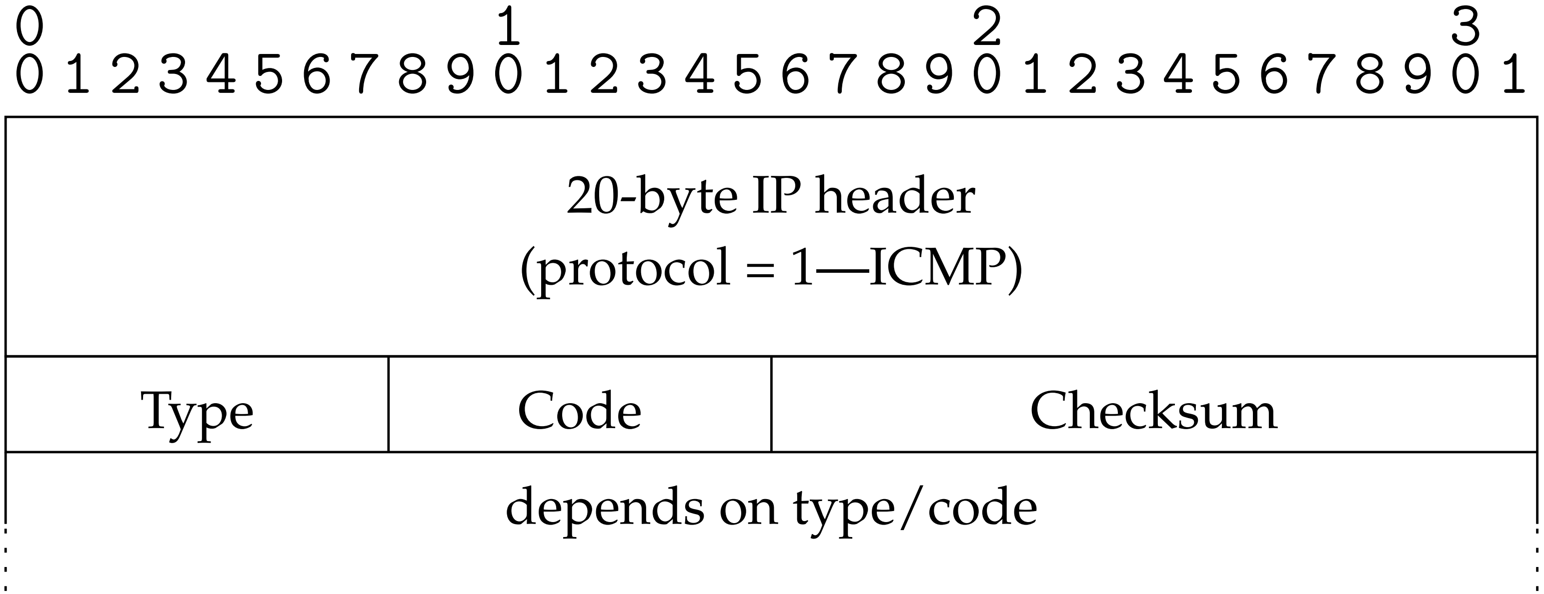
- IP addresses plus ident field identify fragments of same packet
- MF (more fragments bit) is 1 in all but last fragment
- Fragment offset multiple of 8 bytes
  - – Multiply offset by 8 for fragment position original packet



Internet Control Message Protocol (ICMP)

- Echo (ping)
- Redirect
- Destination unreachable (protocol, port, or host)
- TTL exceeded
- Checksum failed
- Reassembly failed
- Can’t fragment
- Many ICMP messages include part of packet that triggered them
- See http://www.iana.org/assignments/icmp-parameters

ICMP message format



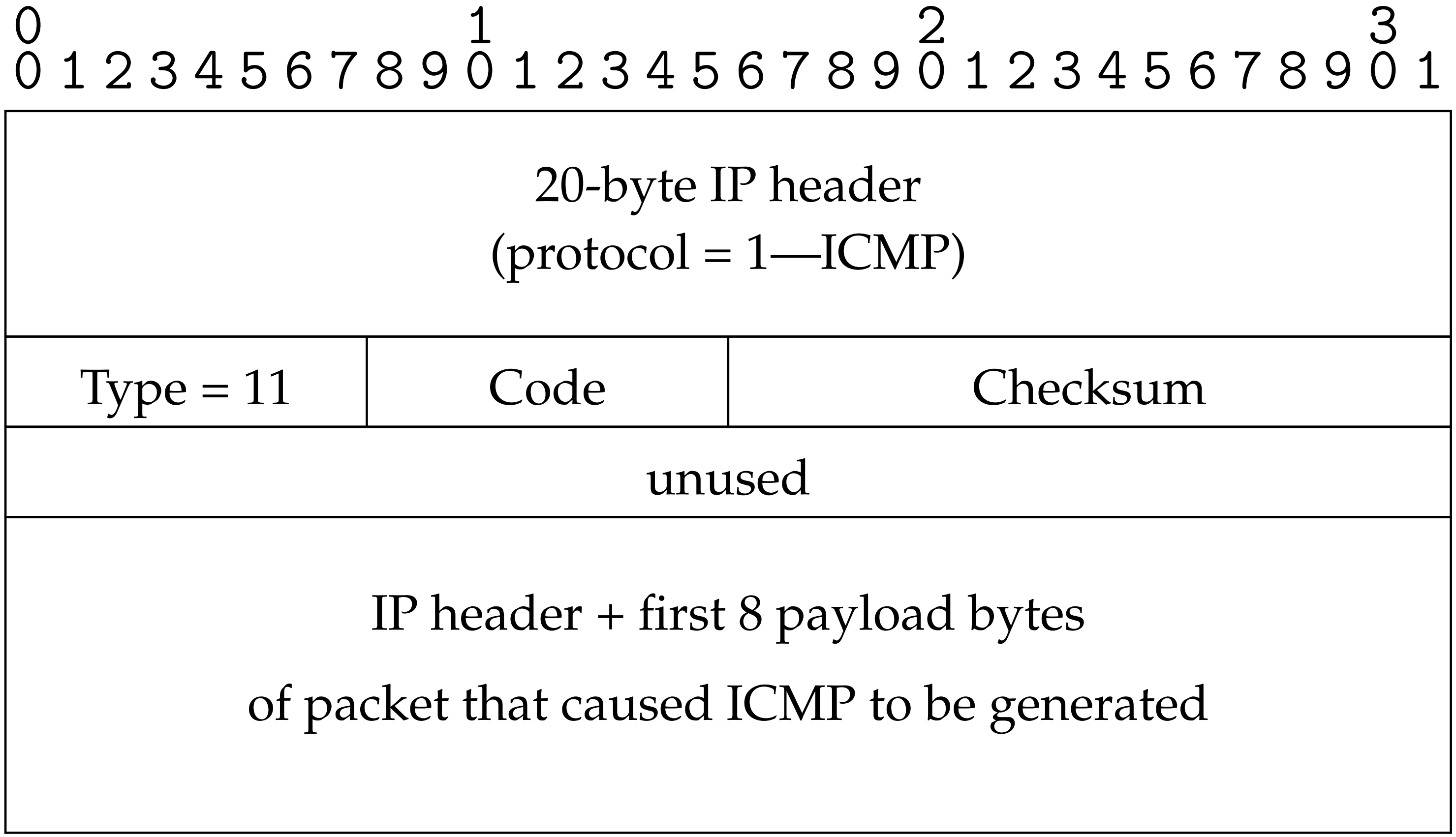
```
0                   1                   2                   3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
```

20-byte IP header
(protocol = 1—ICMP)

Type	Code	Checksum

depends on type/code

Example: Time Exceeded



- **Code usually 0 (TTL exceeded in transit)**
- **Discussion: traceroute**

Example: Can’t Fragment

- **Sent if DF=1 and packet length > MTU**
- **What can you use this for?**
- **Path MTU Discovery**
  - Can do binary search on packet sizes
  - But better: base algorithm on most common MTUs

Coming Up

- **Routing: how do we fill the routing tables?**
  - Intra-domain routing: Tuesday, 10/4
  - Inter-domain routing: Thursday, 10/6

Example

```
# arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
172.17.44.1              ether   00:12:80:01:34:55   C                     eth0
172.17.44.25             ether   10:dd:b1:89:d5:f3   C                     eth0
172.17.44.6              ether   b8:27:eb:55:c3:45   C                     eth0
172.17.44.5              ether   00:1b:21:22:e0:22   C                     eth0
```

```
# ip route
127.0.0.0/8 via 127.0.0.1 dev lo
172.17.44.0/24 dev enp7s0  proto kernel  scope link  src 172.17.44.22  metric 204
default via 172.17.44.1 dev eth0 src 172.17.44.22  metric 204
```