

CSCI-1680

Network Layer: IP & Forwarding

John Jannotti



Based partly on lecture notes by David Mazières, Phil Levis, Rodrigo Fonseca

Administrivia

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



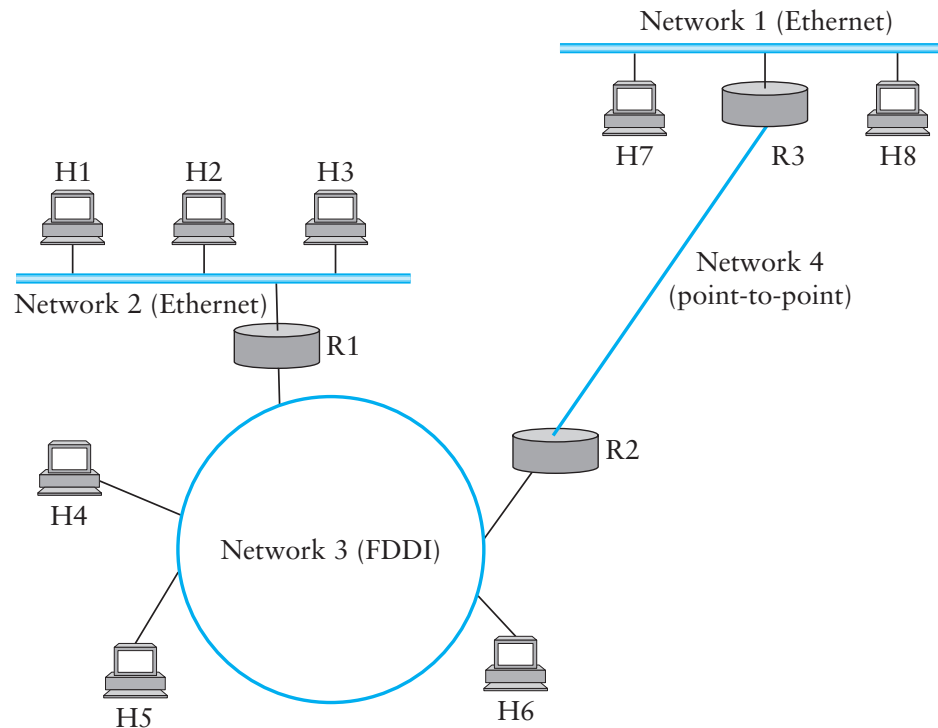
Today

- **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 of 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

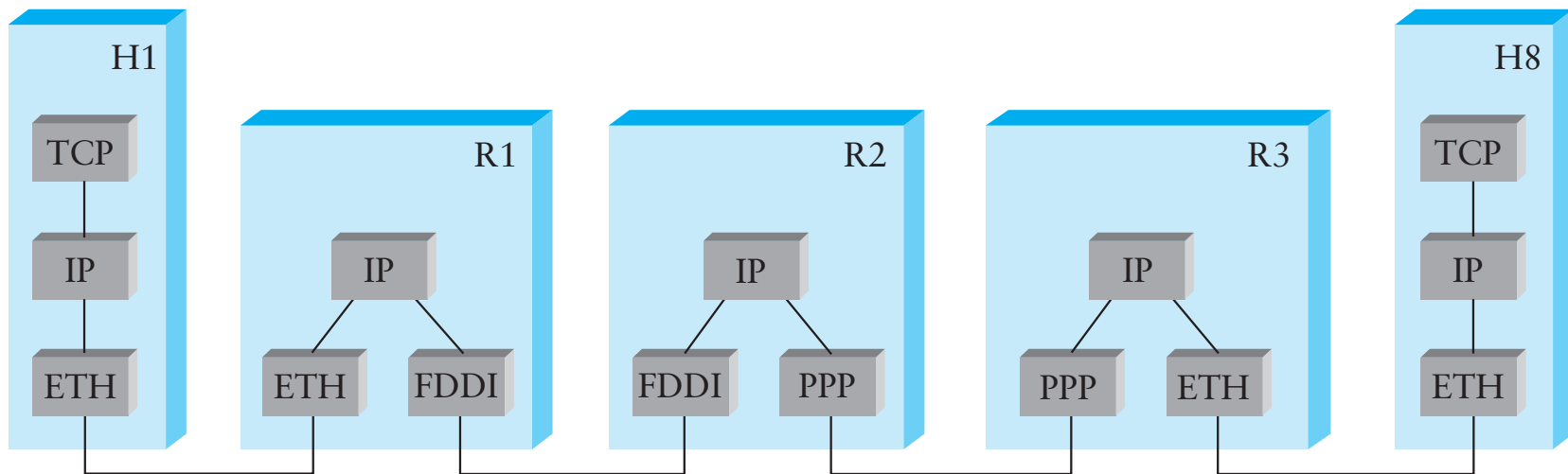
David D. Clark, “The design Philosophy of the DARPA Internet Protocols”, 1988

- 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)



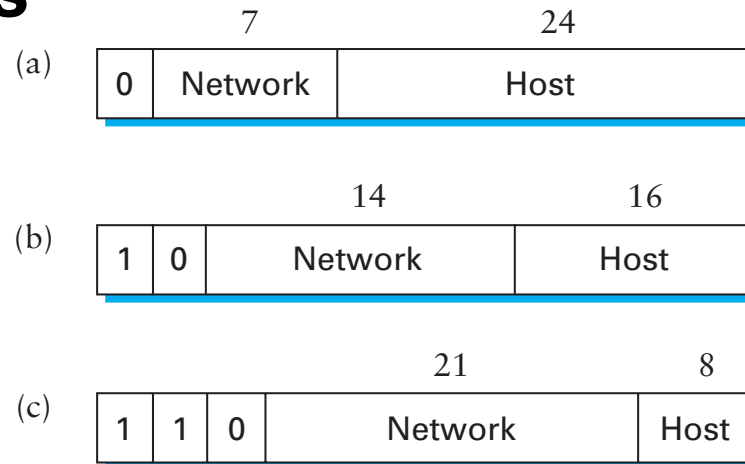
Service Model

- **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**
 - All these can be dealt with above IP (if probability of delivery is non-zero...)



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**



Subnetting

Network number	Host number
----------------	-------------

Class B address

111111111111111111111111	00000000
--------------------------	----------

Subnet mask (255.255.255.0)

Network number	Subnet ID	Host ID
----------------	-----------	---------

Subnetted address

- **Add another level to address/routing hierarchy**
- **Subnet mask** defines variable portion of host part
- **Subnets visible only within site**
- **Better use of address space**



Scaling: Supernetting

- **Problem: routing table growth**
- **Idea: assign blocks of contiguous networks to nearby networks**
- **Called CIDR: Classless Inter-Domain Routing**
- **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**



CIDR Forwarding Table

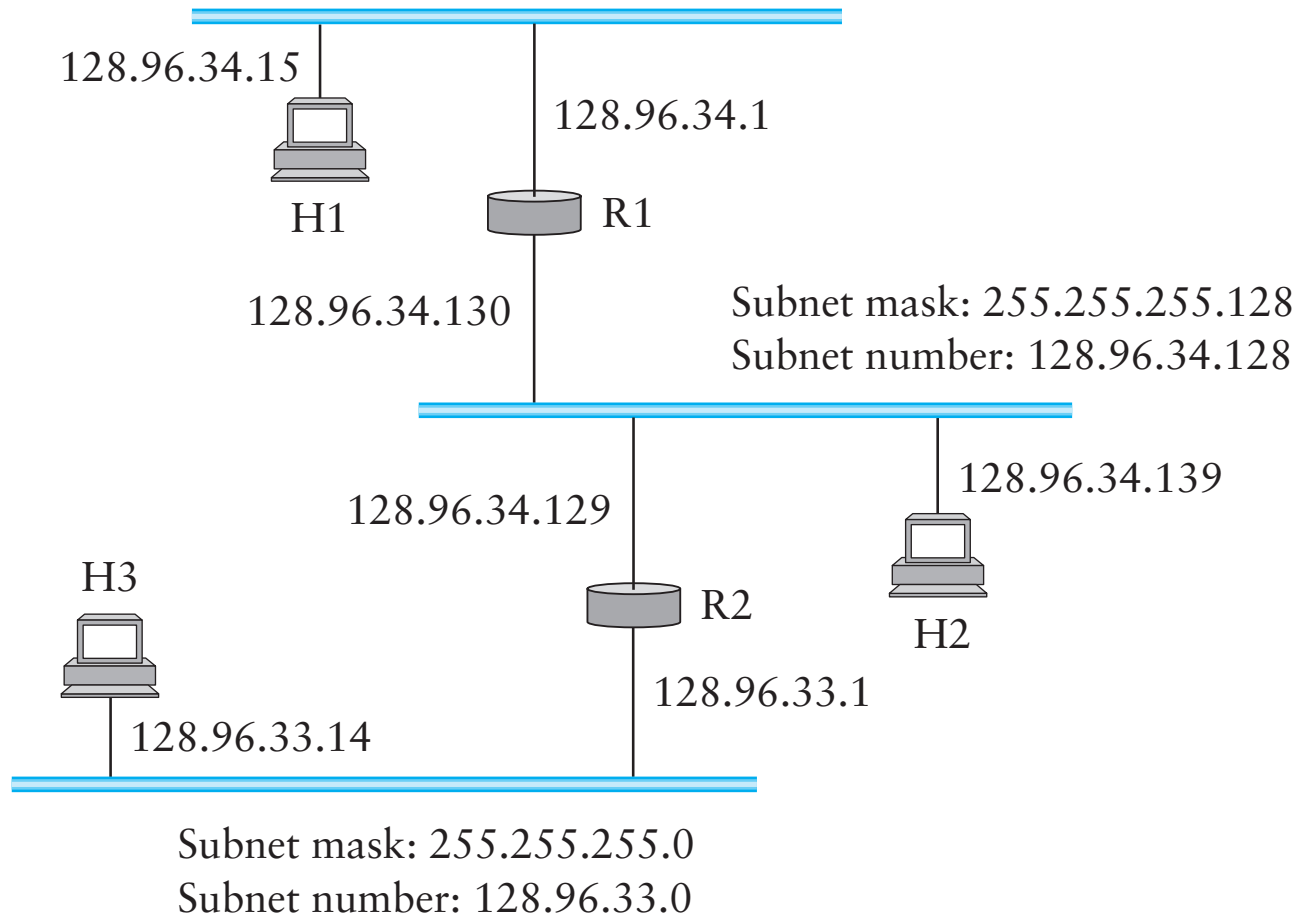
Network	Next Address
212.31.32/24	0.0.0.0
18/8	212.31.32.5
128.148/16	212.31.32.4
128.148.128/17	212.31.32.8
0/0	212.31.32.1



Example

Subnet mask: 255.255.255.128

Subnet number: 128.96.34.0

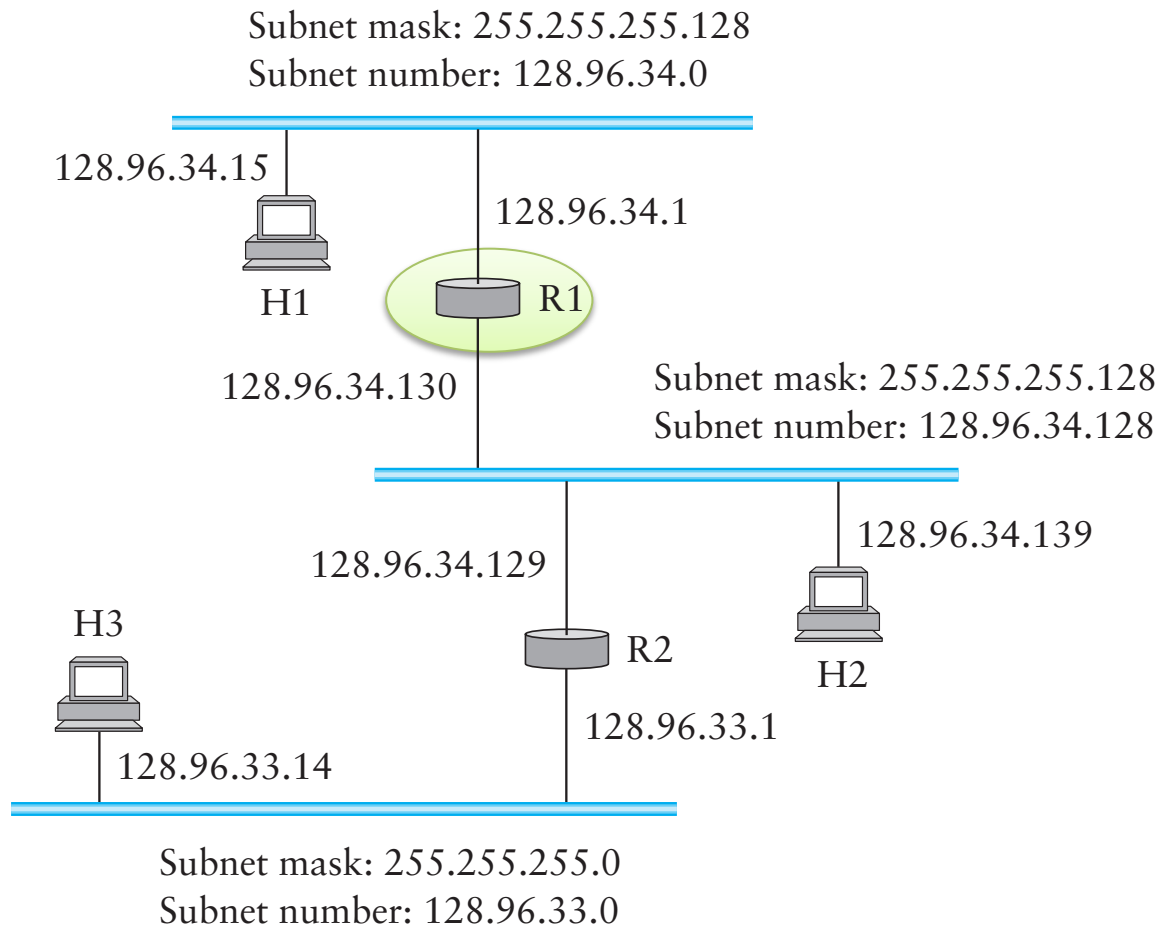


H1-> H2: H2.ip & H1.mask != H1.subnet => no direct path

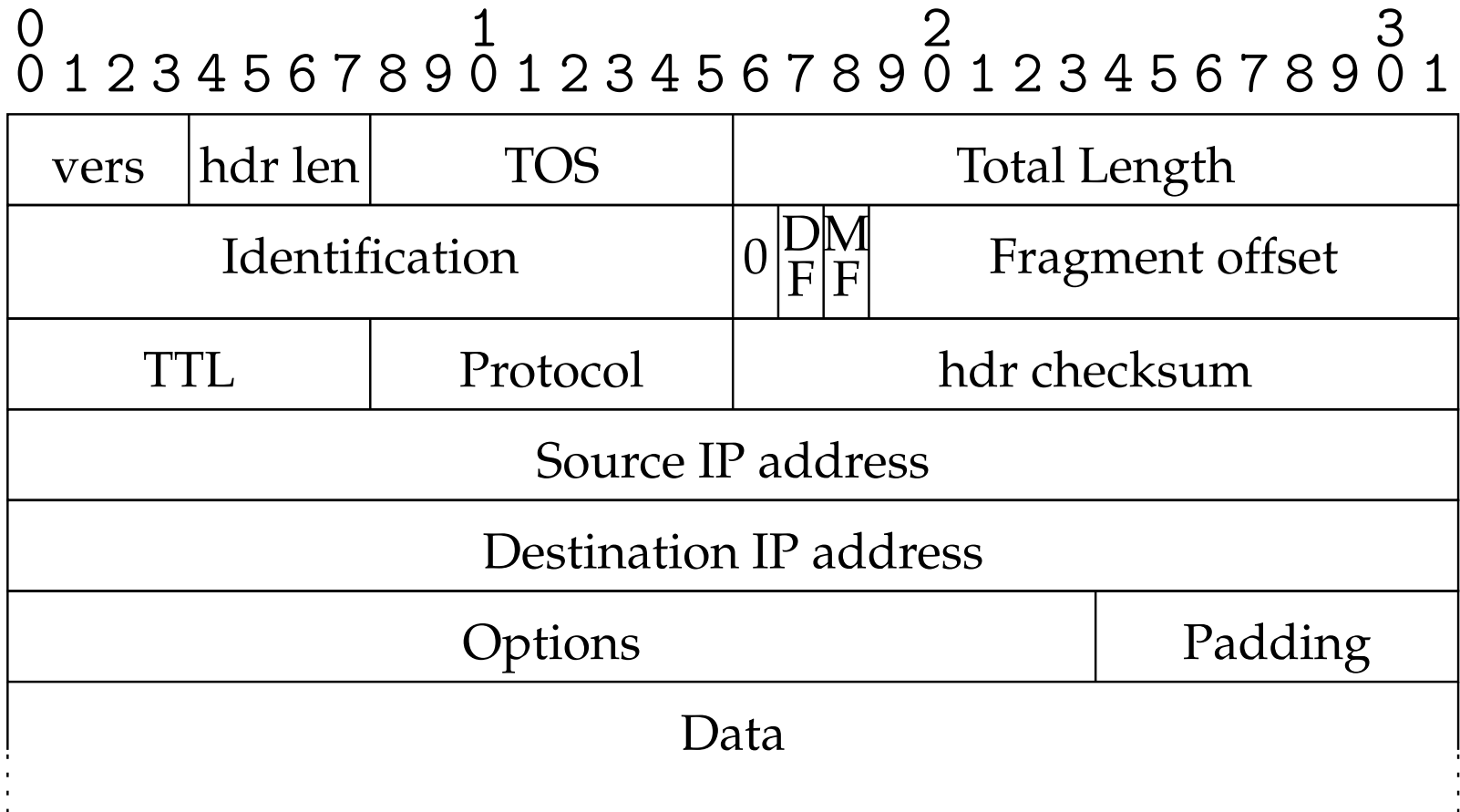


R1's Forwarding Table

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



IP v4 packet format



IP header details

- **Forwarding based on destination address**
- **TTL (time-to-live) decremented at each hop**
 - Originally was intended to be seconds (no longer)
 - Mostly prevents forwarding loops
 - Other cool uses...
- **Fragmentation possible for large packets**
 - Fragmented if crossing link w/ small maximum frame
 - MF: more fragments for this IP packet
 - DF: don't fragment (returns error if would fragment)
- **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***

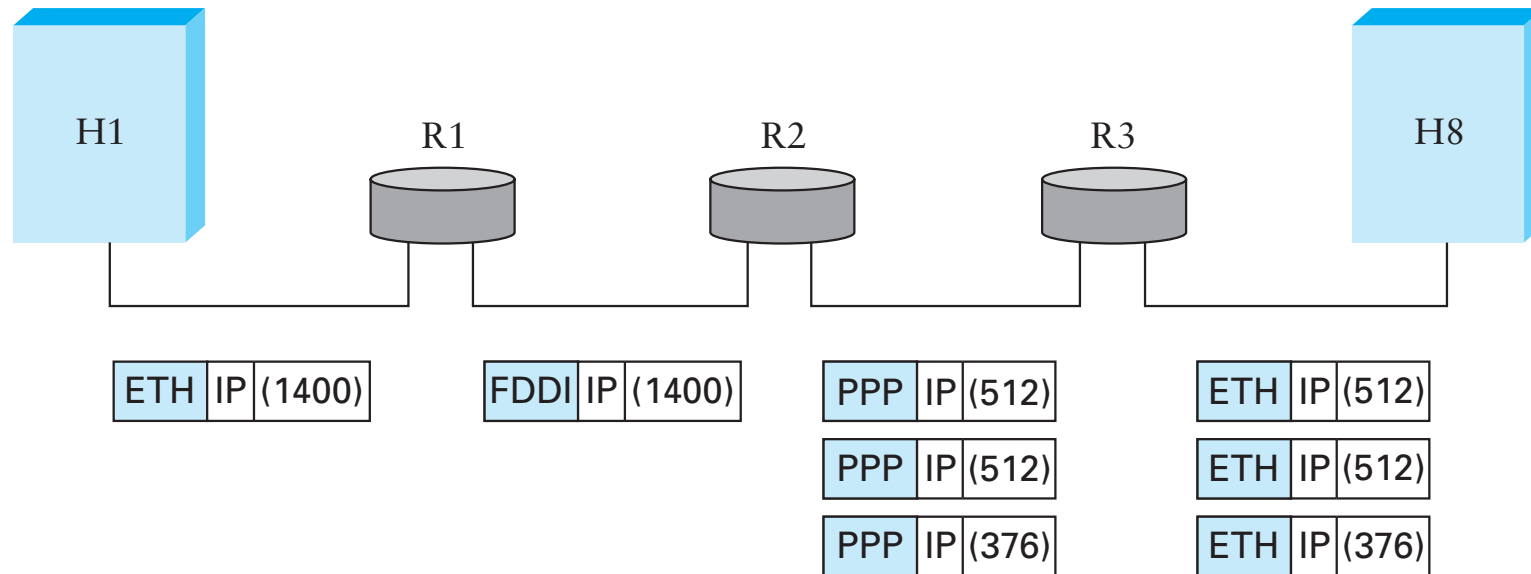


Fragmentation & Reassembly

- **Each network has maximum transmission unit (MTU)**
- **Strategy**
 - Fragment when necessary ($MTU < \text{size of datagram}$)
 - 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

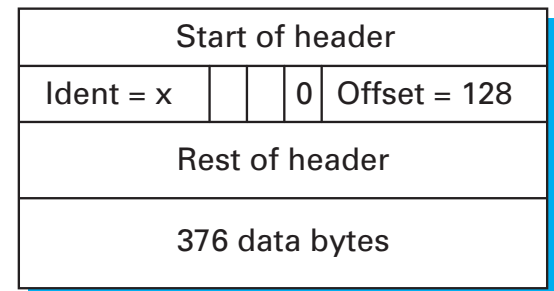
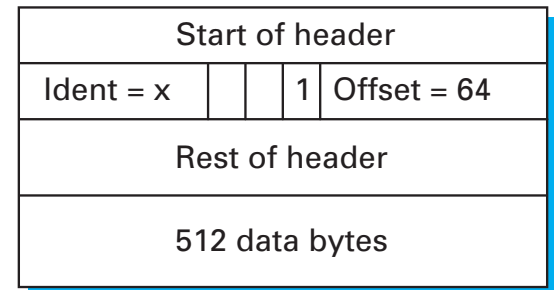
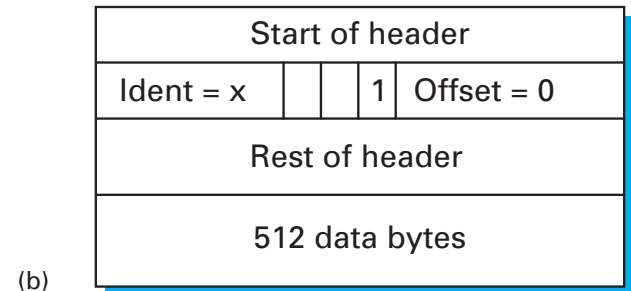
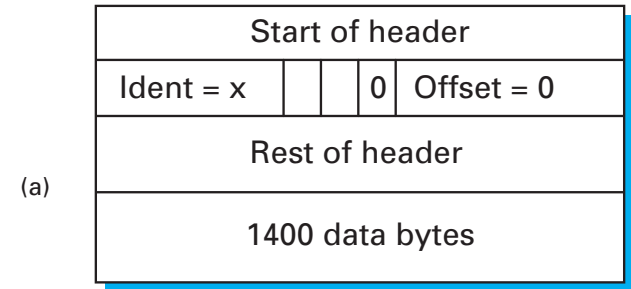


- **Ethernet MTU is 1,500 bytes**
- **PPP MTU is 576 bytes**
 - R2 must fragment IP packets to forward them



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



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

0	8	16	31
Hardware type = 1		ProtocolType = 0x0800	
HLen = 48	PLen = 32		Operation
SourceHardwareAddr (bytes 0–3)			
SourceHardwareAddr (bytes 4–5)		SourceProtocolAddr (bytes 0–1)	
SourceProtocolAddr (bytes 2–3)		TargetHardwareAddr (bytes 0–1)	
TargetHardwareAddr (bytes 2–5)			
TargetProtocolAddr (bytes 0–3)			

- **Why include source hardware address?**



Obtaining Host IP Addresses - DHCP

- **Administrators are free to assign addresses within their allocated 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**



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**



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” an 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?

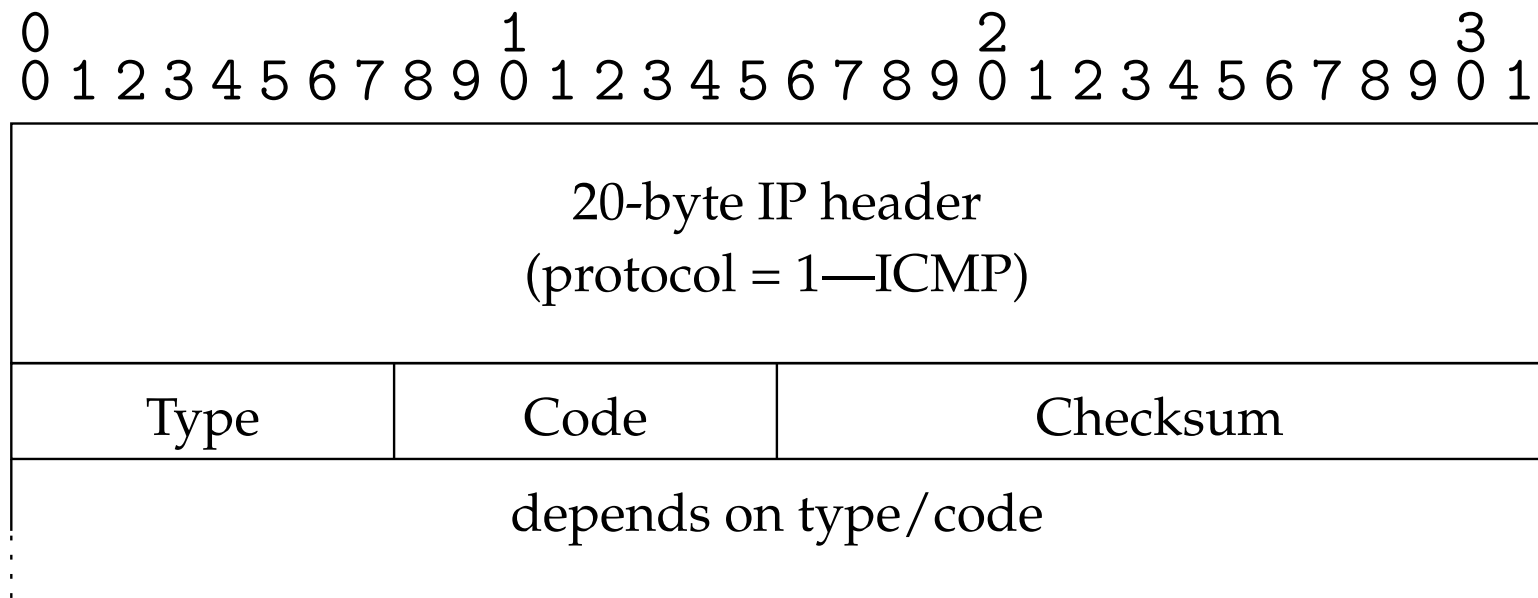


Internet Control Message Protocol (ICMP)

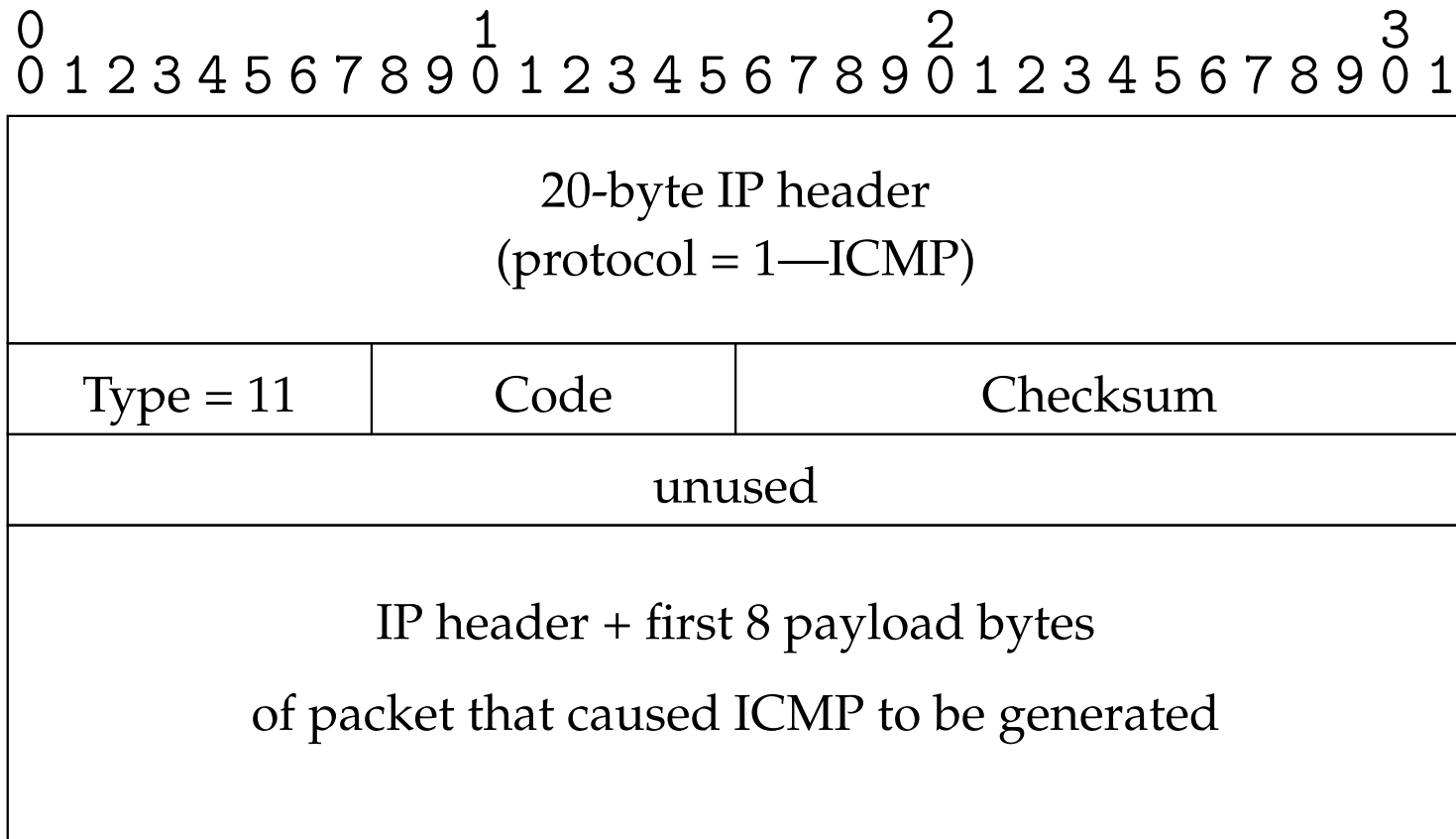
- Echo (ping)
- Redirect
- Destination unreachable (protocol, port, or host)
- TTL exceeded
- Checksum failed
- Reassembly failed
- Can't fragment
- Many ICMP messages include part of packet that triggered them
- See <http://www.iana.org/assignments/icmp-parameters>



ICMP message format



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 10/6
 - Inter-domain routing: Thursday, 10/8

