Simple Internetworking

Hui Chen, Ph.D.
Department of Engineering & Computer Science
Virginia State University
Petersburg, VA 23806

Acknowledgements

- Some pictures used in this presentation were obtained from the Internet
- The instructor used the following references
  - Larry L. Peterson and Bruce S. Davie, Computer Networks: A Systems Approach, 5th Edition, Elsevier, 2011
  - Andrew S. Tanenbaum, Computer Networks, 5th Edition, Prentice-Hall, 2010
  - James F. Kurose and Keith W. Ross, Computer Networking: A Top-Down Approach, 5th Ed., Addison Wesley, 2009
  - Larry L. Peterson's (http://www.cs.princeton.edu/~llp/) Computer Networks class web site

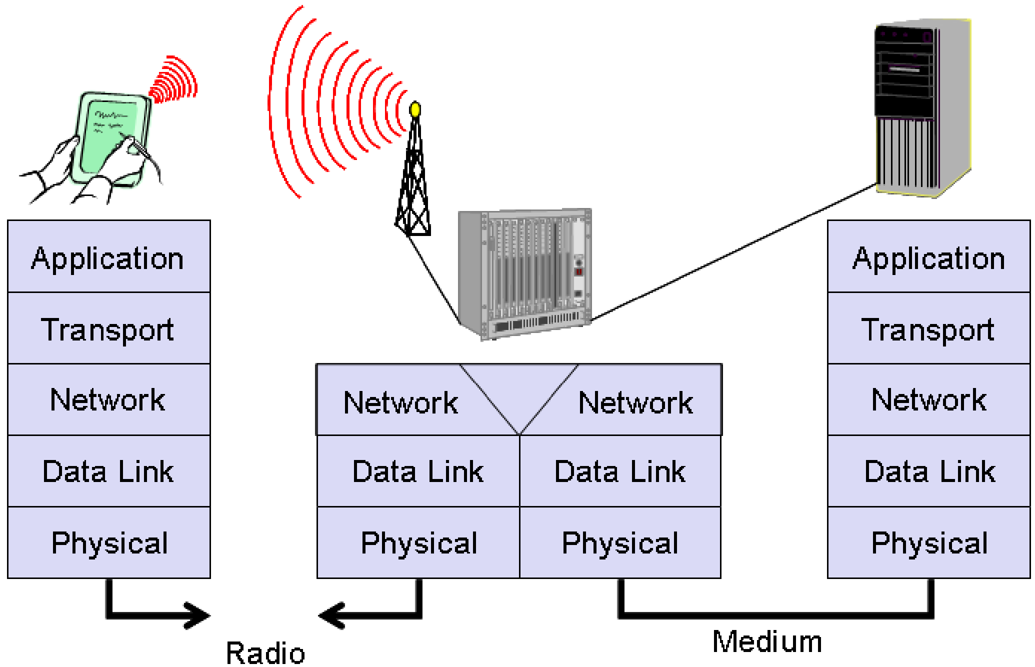
Outline

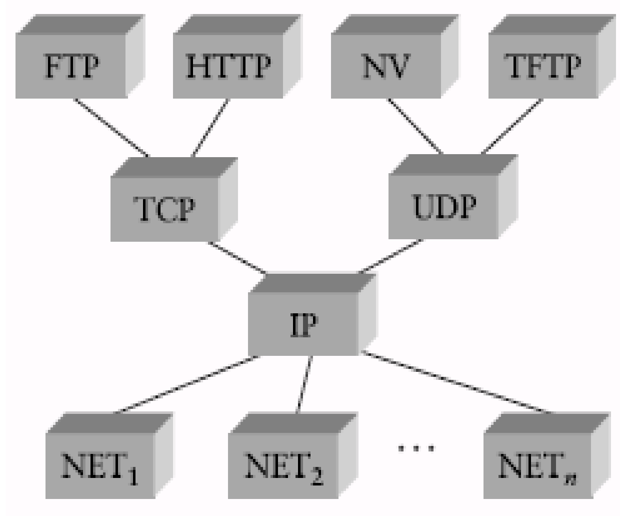
- Topic: internetworking
  - Case study: Internet Protocol (IP) Suite
- Simple interworking
  - internet and the Internet
  - Global addressing scheme
  - Packet fragmentation and assembly
  - Best effort service model and datagram forwarding
  - Address translation
  - Host configuration
  - Error reporting

Heterogeneity and Scalability

- LAN: small in size
- How to extend LAN?
  - Bridges and switches
  - Good for global networks?
    - Spanning tree algorithms → very long path and huge forwarding tables
    - Bridges and switches: link level/layer 2 devices → networks must be using the same type of links
- Problems to deal with
  - Scalability: global networks are huge in size
  - Heterogeneity: networks of different types of links are in use

Solution to Heterogeneity: layered architecture and hourglass design

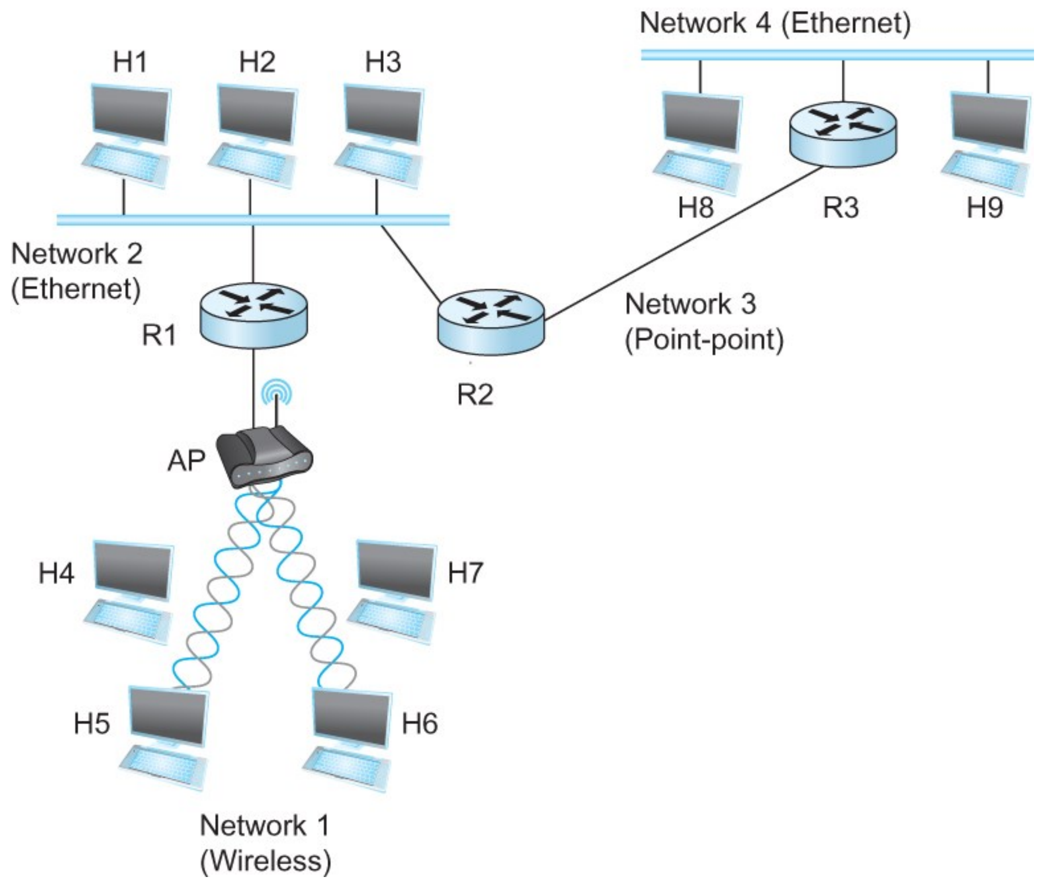




- How do layered architecture and hourglass design work in internetworks?
  - Use the Internet as a case study

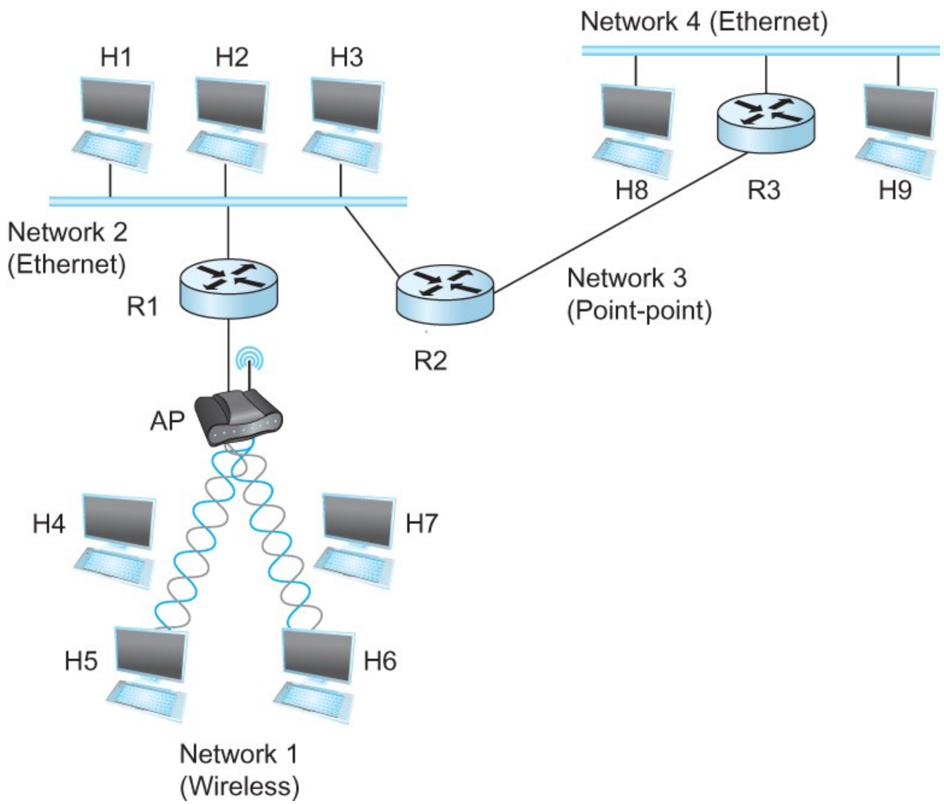
Solution to Heterogeneity and Scalability: Network of Networks

- Forwarding packets to networks from networks



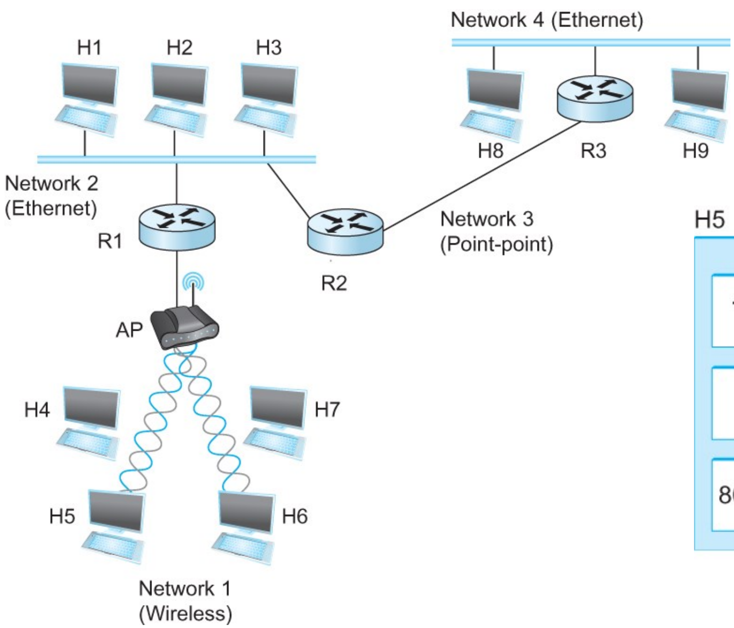
internetworking

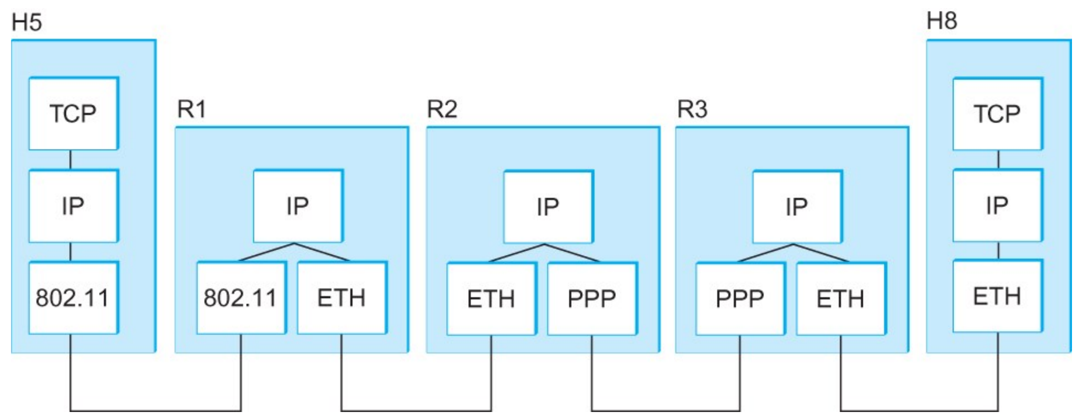
- An arbitrary collection of networks interconnected to provide some sort of host-host to packet delivery service



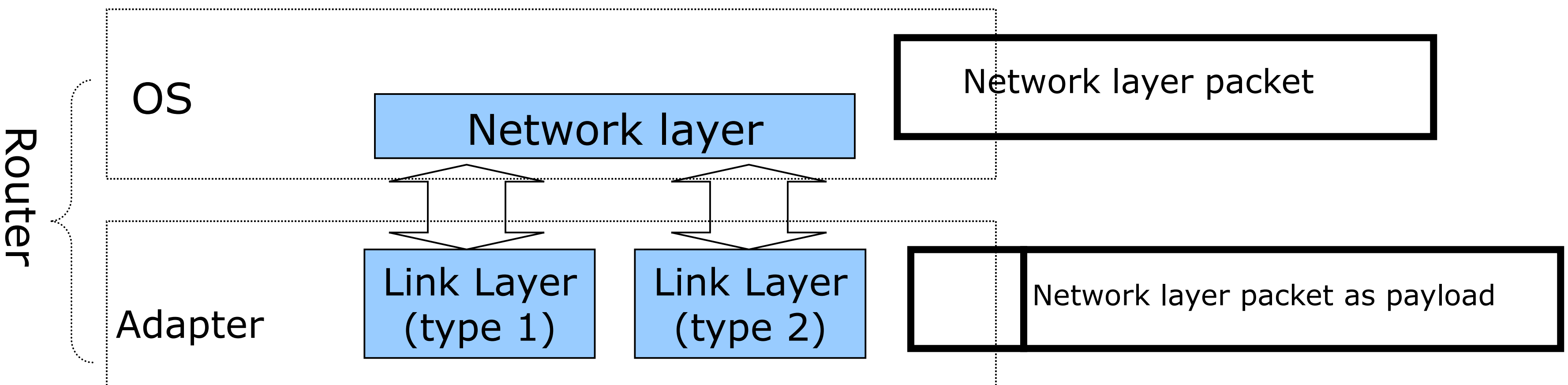
Internet Protocol

- IP = Internet Protocol
- Key tool used today to build scalable, heterogeneous internetworks
  - Routers forward packets to "networks": forwarding tables can be smaller
  - Above link layer: can cope with different link layer technology

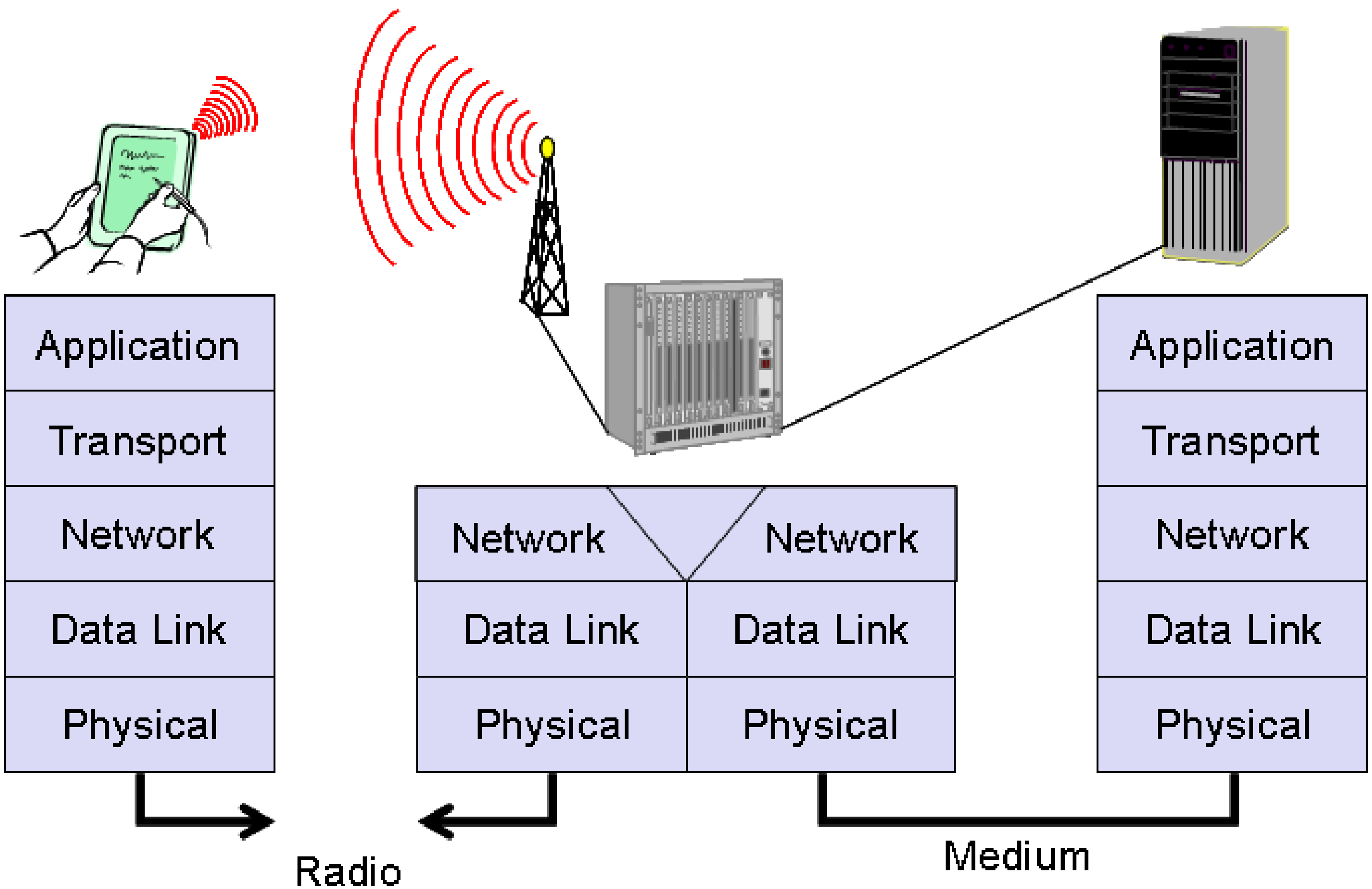




internetworking



internetworking



Case Study: internetworking

- Global internetworks built on IP → The Internet $\neq$ internet
- Using Internet Protocol (IP) as a case study
  - IP packet format and Global IP addressing scheme
  - Deal with different link layer technology
    - Packet fragmentation and assembly
  - Packet forwarding
    - Datagram forwarding and service model
  - Deal with Link layer and network layer interfacing
    - Address translation
  - Other important issues
    - Host configuration
    - Error reporting

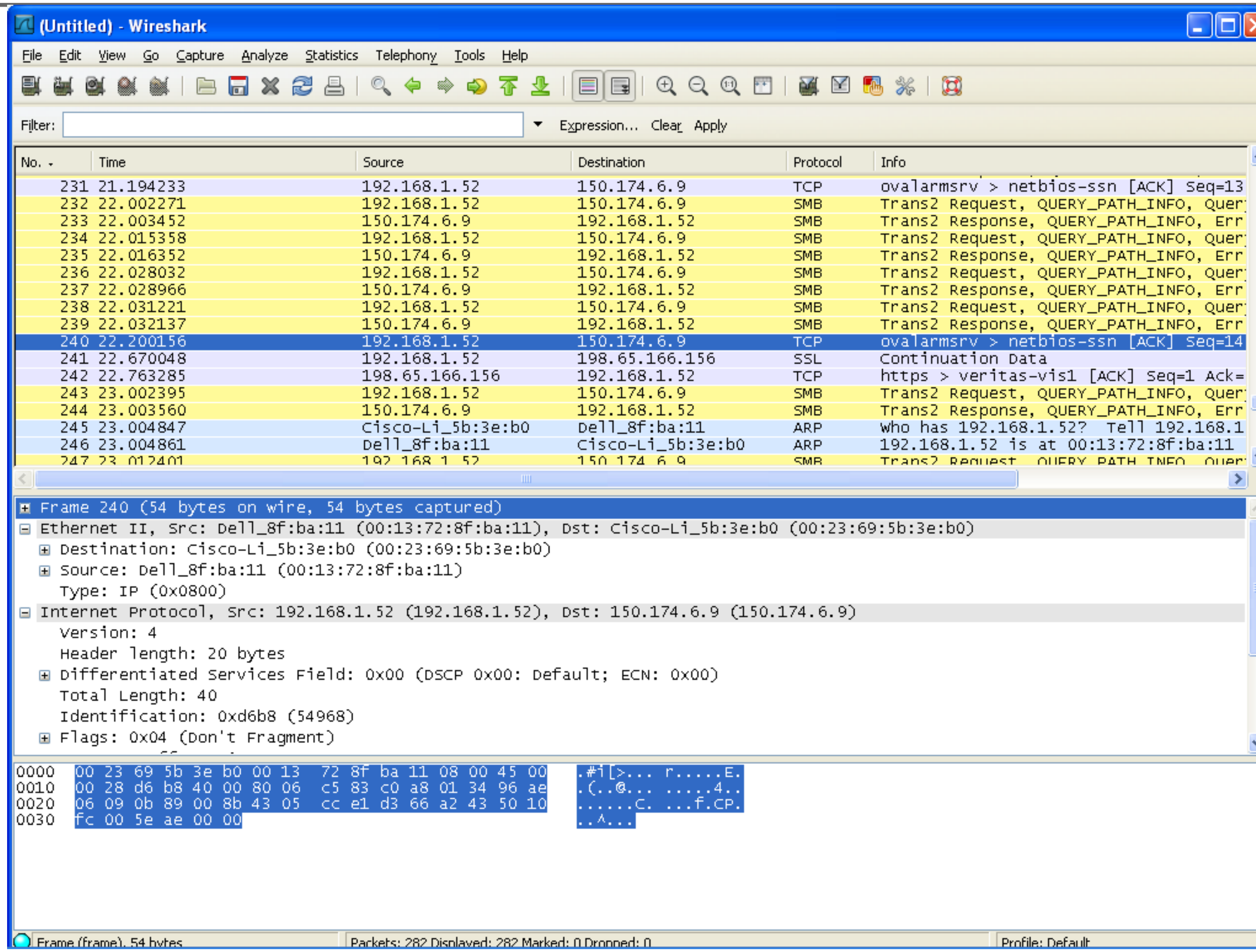
Basic Data Structure: IP Packet

- What is the design?
  - Attributes and purposes
    - Support error detection and handling
    - Support networks as a forwarding source and destinations
    - Support different networking technologies
    - Support multiplexing
    - Support extensibility

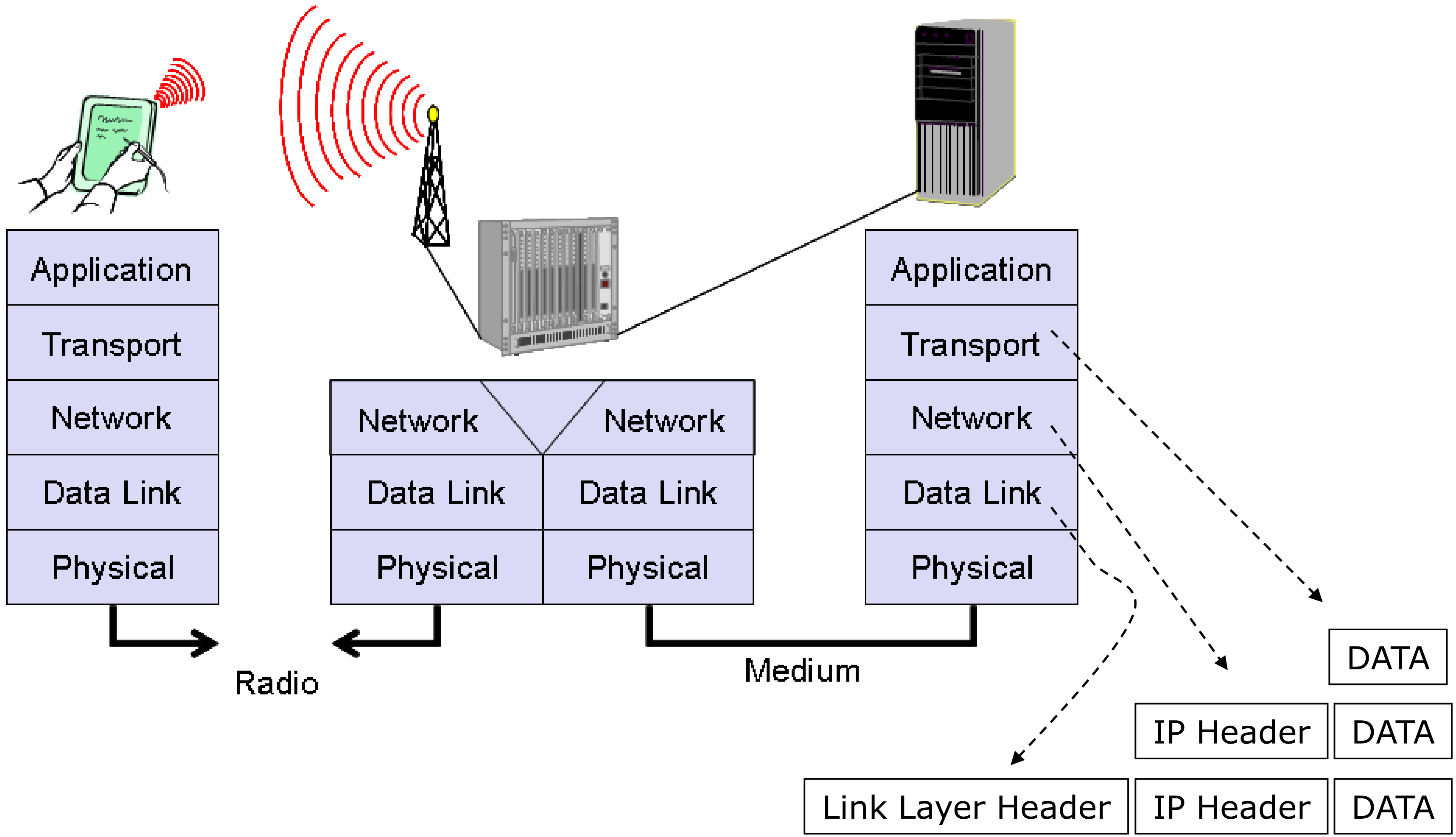
Capturing an IP Packet

- And examining it …
- Use the *ethercap* application (a part of homework 1)
- Use Wireshark
- Use Microsoft Network Monitor (Message Analyzer)
- Use *libpcap* in your own application
- ……

Using Wireshark



IP Packet


Application
Transport
Network
Data Link
Physical
Network
Network
Data Link
Data Link
Physical
Physical
Application
Transport
Network
Data Link
Physical
Radio
Medium
DATA
IP Header
DATA
Link Layer Header
IP Header
DATA

A Captured IP Packet

```
hchen@hecuba:~/Project/csed/csed/networks/ethernet
Interface: eth0
0000   00 23 ae 7b 49 11 00 13 72 8f ba 11 08 00 45 00      .#.{I...r.....E.
0010   00 28 78 41 40 00 80 06 fe d4 c0 a8 01 34 c0 a8      .(xA@........4..
0020   01 35 07 e3 00 16 b6 c0 0a da b6 1e 1a b7 50 10      .5............P.
0030   f1 80 a0 30 00 00 00 00 00 00 00 00                 ...0........
Interface: eth0
0000   00 23 ae 7b 49 11 00 13 72 8f ba 11 08 00 45 00      .#.{I...r.....E.
0010   00 28 78 42 40 00 80 06 fe d3 c0 a8 01 34 c0 a8      .(xB@........4..
0020   01 35 07 e3 00 16 b6 c0 0a da b6 1e 1c 17 50 10      .5............P.
0030   fc 00 94 50 00 00 00 00 00 00 00 00                 ...P........
Interface: eth0
0000   00 23 ae 7b 49 11 00 13 72 8f ba 11 08 00 45 00      .#.{I...r.....E.
0010   00 5c 78 43 40 00 80 06 fe 9e c0 a8 01 34 c0 a8      .\xC@........4..
0020   01 35 07 e3 00 16 b6 c0 0a da b6 1e 1c 17 50 18      .5............P.
0030   fc 00 7f 6a 00 00 3e ad c9 12 24 58 f0 fd e7 96      ...j..>...$X....
0040   79 09 a2 b5 73 4a 88 a1 20 1d c4 87 44 72 e0 8c      y...sJ.. ...Dr..
0050   67 02 37 a3 de f4 c8 cc ec 18 dc ca d1 3a 2a 33      g.7..........:*3
0060   a6 75 c4 14 4d 57 1f 1a 0c f9                       .u..MW....
Interface: eth0
0000   00 13 72 8f ba 11 00 23 ae 7b 49 11 08 00 45 10      ..r....#.{I...E.^C
0000   00 13 72 8f ba 11 00 23 ae 7b 49 11 08 00 45 10      ..r....#.{I...E.

User pressed CTRL-C. Exiting ...
[hchen@hecuba ethernet]$
```

Q: how do we make sense of an IP packet?

Ethernet Protocol ID's

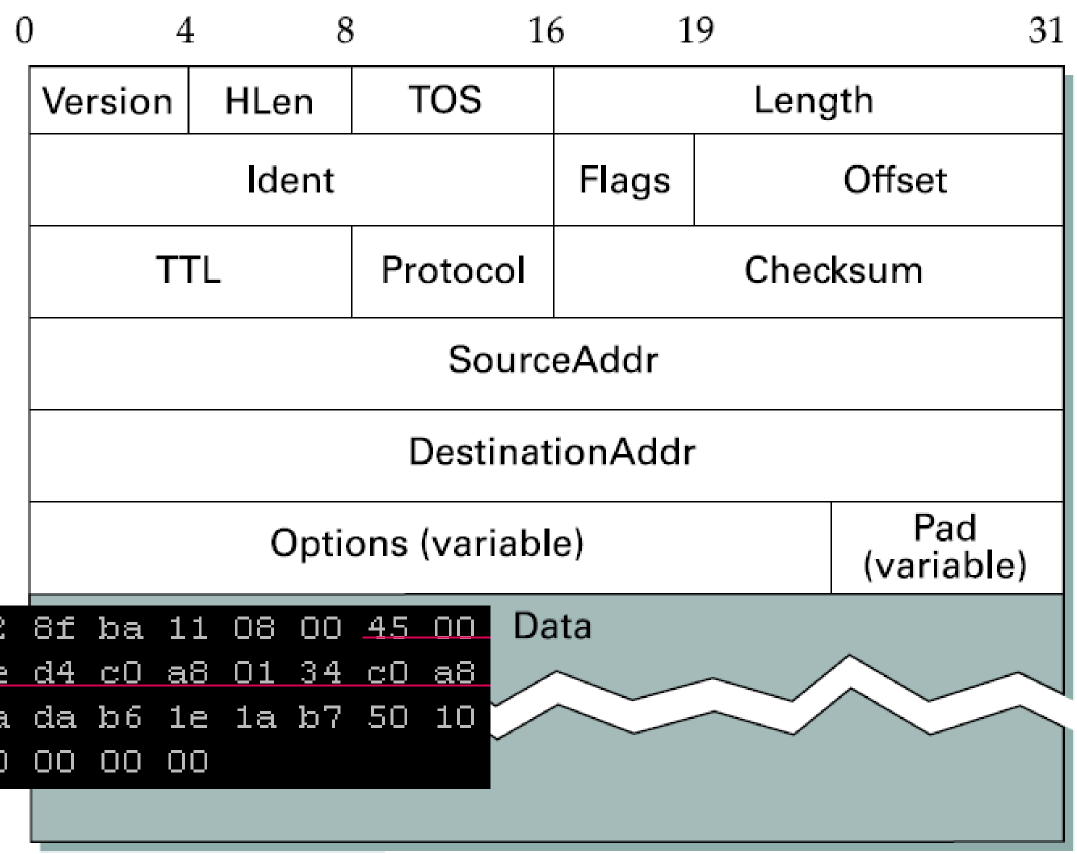
VIM - /usr/include/net/ethernet.h

```
/* 10Mb/s ethernet header */
struct ether_header
{
  u_int8_t  ether_dhost[ETH_ALEN];  /* destination eth addr */
  u_int8_t  ether_shost[ETH_ALEN];  /* source ether addr    */
  u_int16_t ether_type;             /* packet type ID field */
} __attribute__ ((__packed__));

/* Ethernet protocol ID's */
#define ETHERTYPE_PUP       0x0200          /* Xerox PUP */
#define ETHERTYPE_SPRITE    0x0500      /* Sprite */
#define ETHERTYPE_IP        0x0800      /* IP */
#define ETHERTYPE_ARP       0x0806      /* Address resolution */
#define ETHERTYPE_REVARP    0x8035      /* Reverse ARP */
#define ETHERTYPE_AT        0x809B      /* AppleTalk protocol */
#define ETHERTYPE_AARP      0x80F3      /* AppleTalk ARP */
#define ETHERTYPE_VLAN      0x8100      /* IEEE 802.1Q VLAN tagging */
#define ETHERTYPE_IPX       0x8137      /* IPX */
#define ETHERTYPE_IPV6      0x86dd      /* IP protocol version 6 */
#define ETHERTYPE_LOOPBACK  0x9000      /* used to test interfaces */


#define ETHER_ADDR_LEN  ETH_ALEN                 /* size of ethernet addr */
"/usr/include/net/ethernet.h" [readonly] 84L, 3221C           49,1          60%
```

IP Packet Format (1)

- Current version: IP version 4 (IPv4)

Convention used to illustrate IP packet
- •32 bit words
- •Top word transmit first
- •Left-most byte transmit first

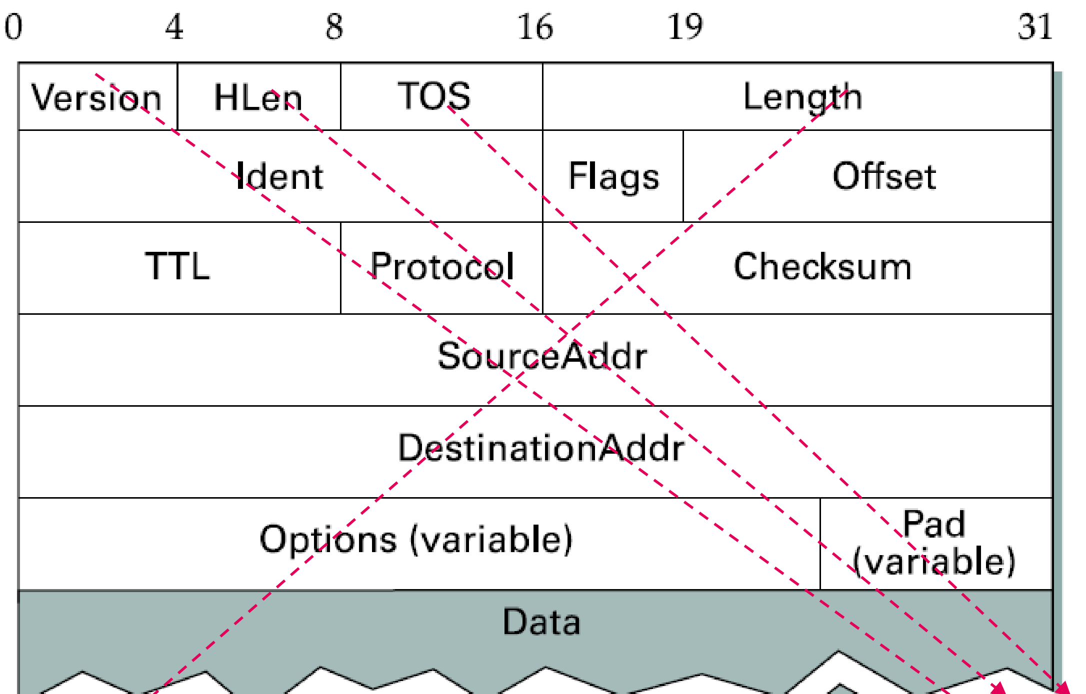



```
0000   00 23 ae 7b 49 11 00 13 72 8f ba 11 08 00 45 00
0010   00 28 78 41 40 00 80 06 fe d4 c0 a8 01 34 c0 a8
0020   01 35 07 e3 00 16 b6 c0 0a da b6 1e 1a b7 50 10
0030   f1 80 a0 30 00 00 00 00 00 00 00 00
```

IPv4 Packet format

IP Packet Format (2)

- Version
- HLen: Header Length in 32-bit words
- TOS: Type Of Service, used to treat packet different based on application needs
  - Usages of TOS discussed in later chapters. See DiffServ
- Length: length of the packet/datagram (including header) in bytes

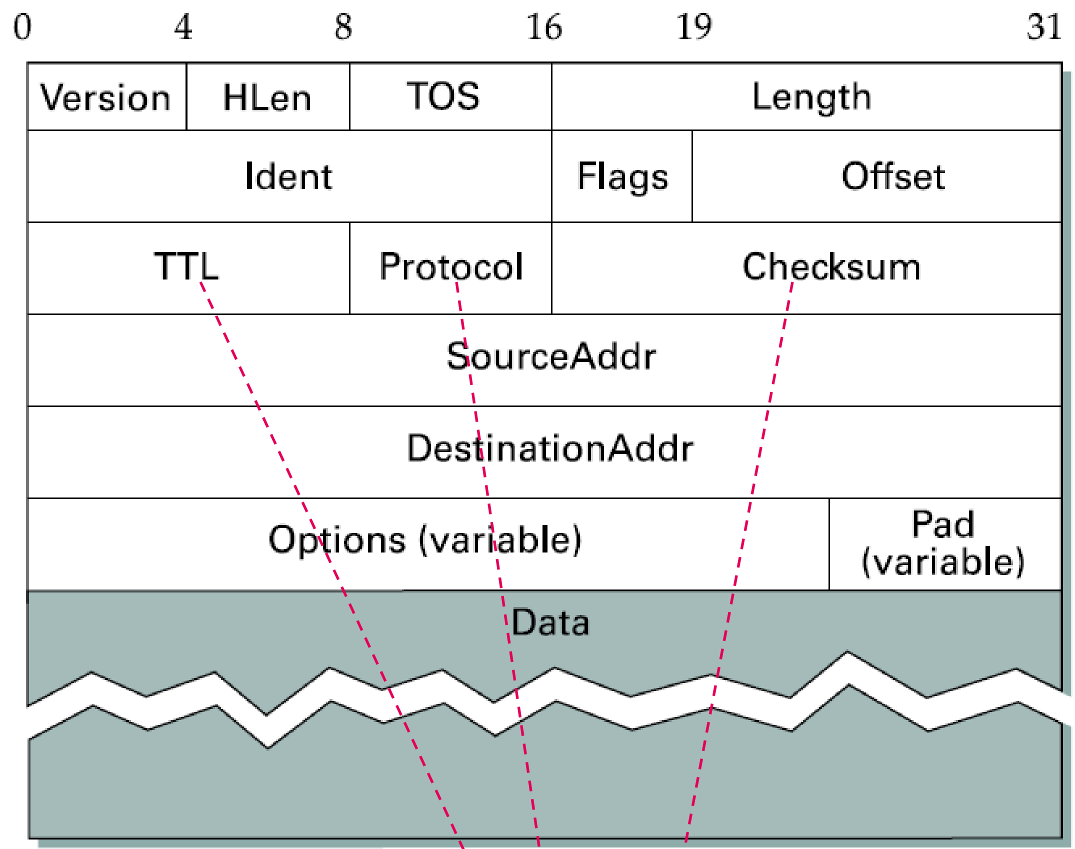



```
0000   00 23 ae 7b 49 11 00 13 72 8f ba 11 08 00 45 00
0010   00 28 78 41 40 00 80 06 fe d4 c0 a8 01 34 c0 a8
0020   01 35 07 e3 00 16 b6 c0 0a da b6 1e 1a b7 50 10
0030   f1 80 a0 30 00 00 00 00 00 00 00 00
```

- Q1: what is the length in bytes of the largest IP packet? What is the corresponding Length?
- Q2: what is the length in bytes of the smallest IP packet? What is the corresponding Length?
- Q3: **which byte is the last byte of THIS IP packet?**

IP Packet Format (3)

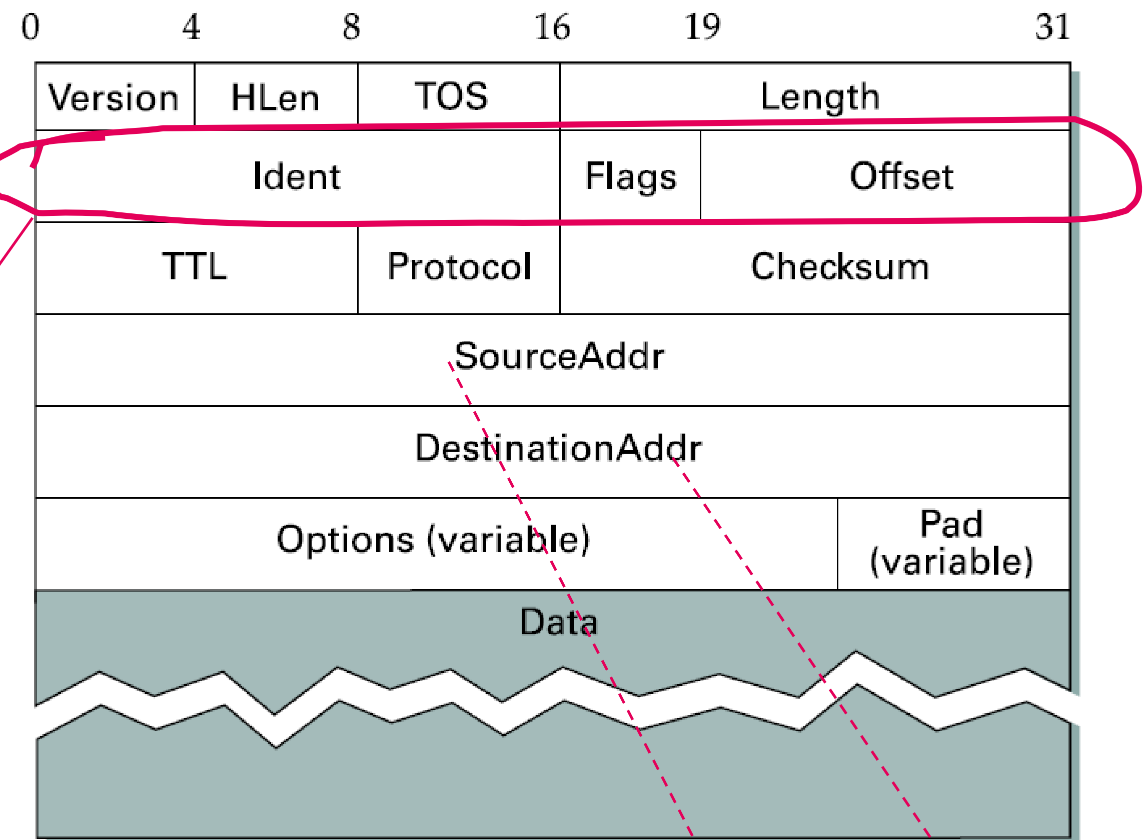
- TTL: Time-To-Live, use as hope count today
  - Set by hosts
  - Default: 64
- Protocol: to which upper layer protocol this packet should be delivered, e.g., 6=TCP, 17=UDP. See [IANA]
- Checksum: Internet checksum of IP header with *checksum field as 0s*




```
0000   00 23 ae 7b 49 11 00 13 72 8f ba 11 08 00 45 00
0010   00 28 78 41 40 00 80 06 fe d4 c0 a8 01 34 c0 a8
0020   01 35 07 e3 00 16 b6 c0 0a da b6 1e 1a b7 50 10
0030   f1 80 a0 30 00 00 00 00 00 00 00 00
```

IP Packet Format (4)

- SourceAddr: IP address of the originating host
- DestinationAddr: indented destination



What are these?

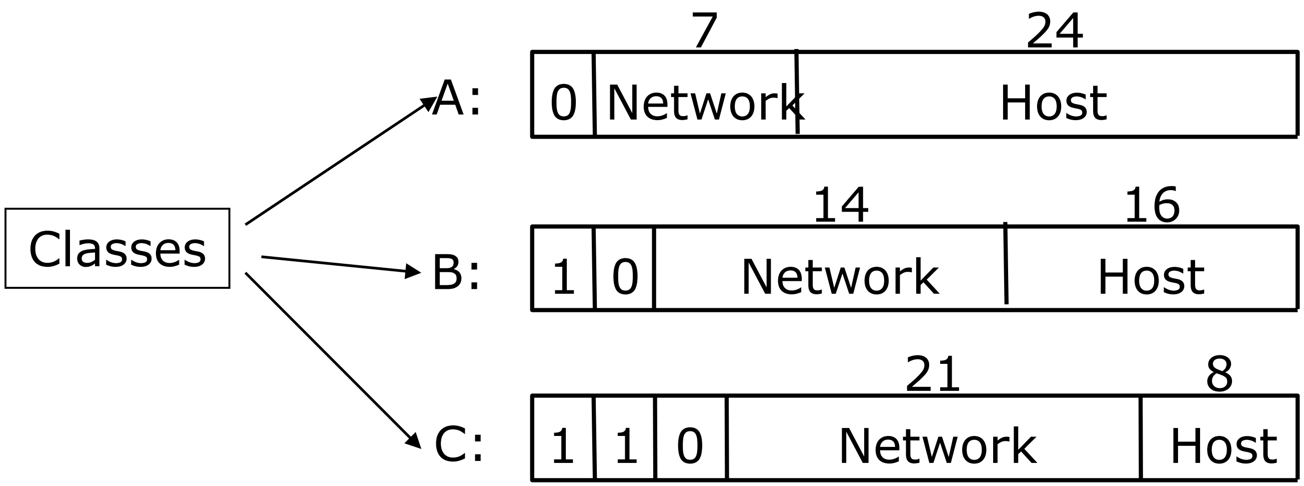
```
0000   00 23 ae 7b 49 11 00 13 72 8f ba 11 08 00 45 00
0010   00 28 78 41 40 00 80 06 fe d4 c0 a8 01 34 c0 a8
0020   01 35 07 e3 00 16 b6 c0 0a da b6 1e 1a b7 50 10
0030   f1 80 a0 30 00 00 00 00 00 00 00 00
```

Global Addresses

- Internet Protocol (IP) Address
  - Public addresses are unique
  - Hierarchical: **network** + host
- IPv4
  - 32 bit integer
  - Human-readable form
    - 150.174.44.57
  - Facing exhaustion of address space, moving to IPv6

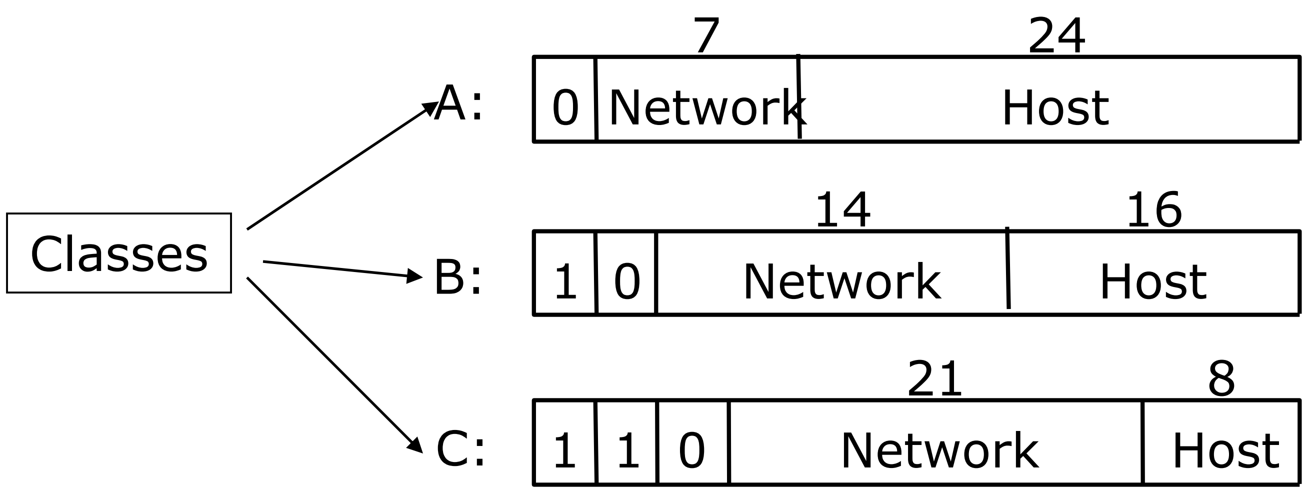
IPv4 Address Classes (Legacy)

- Classes (legacy)
  - To express networks



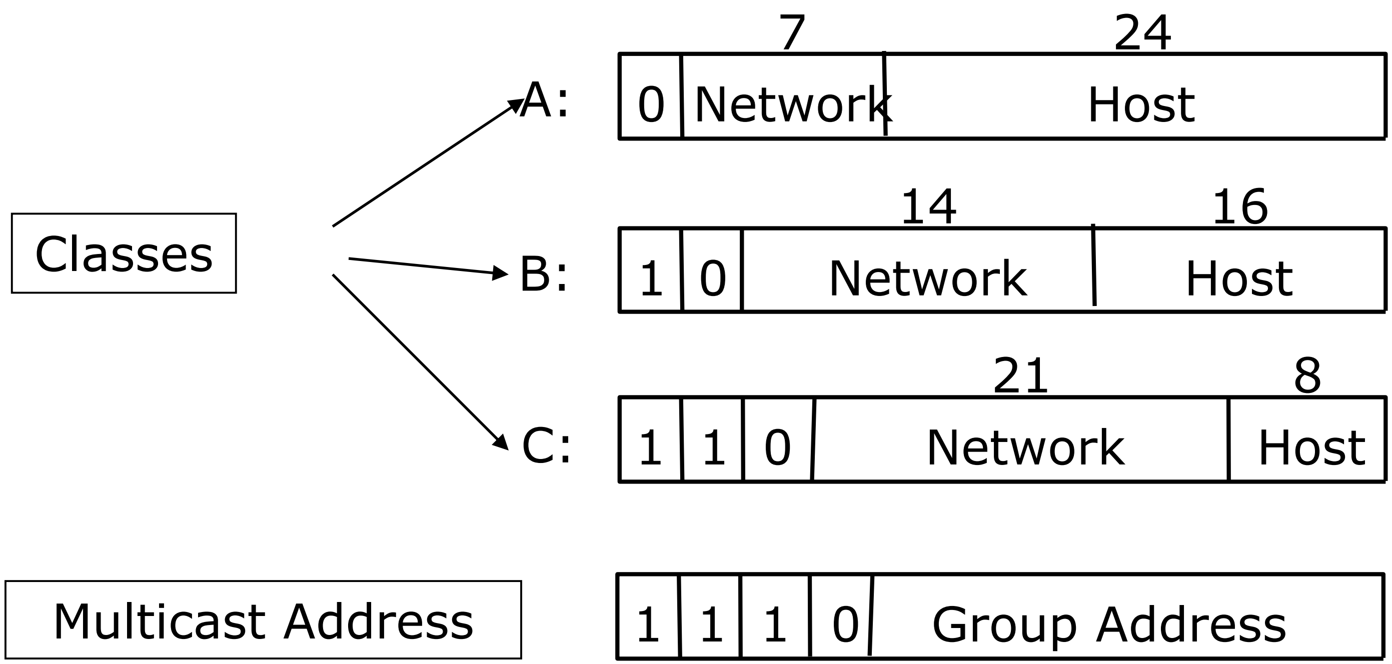
Broadcast and Multicast Addresses

- Do the classes of IP addresses discussed including any IP addresses starting with bits 111?



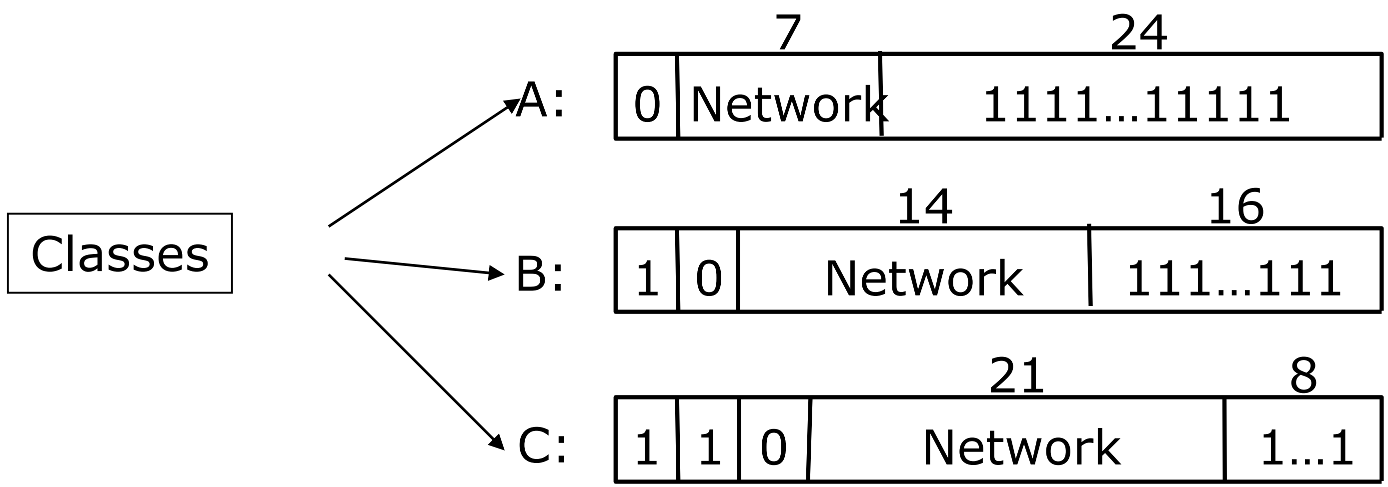
IPv4 Multicast Address

- Addresses starting with 1110



IPv4 Broadcast Address

- setting all the host bits to 1



Private IPv4 Address Spaces

- See [RFC 1918]
- Private networks
  - 24-bit block 10.0.0.0–10.255.255.255
  - 20-bit block 172.16.0.0–172.31.255.255
  - 16-bit block 192.168.0.0–192.168.255.255
- Routers do not forward these IP packets to other networks

Link Local IPv4 Address

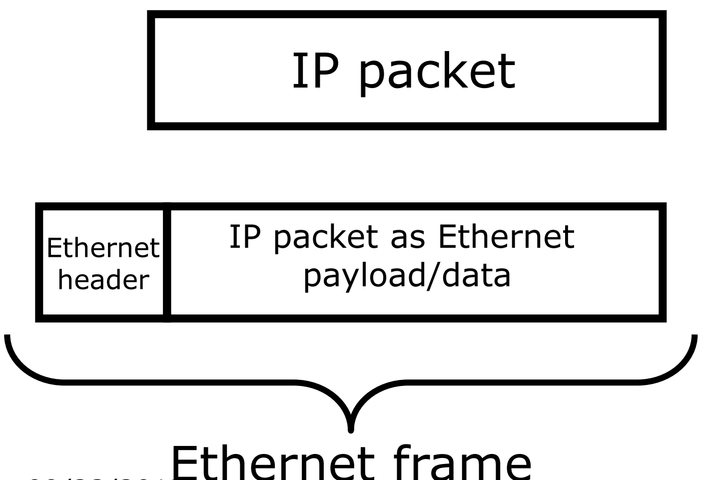
- See RFC 3927
- Link-Local IPv4 Address
  - 16-bit block 169.254.0.0–169.254.255.255

Exercise L10-1

- Find out IPv4 addresses of following hosts and indicate the class to which the IP addresses belong
  - www.vsu.edu
  - www.drsr.sk
  - www.google.com
- Remark
  - There are many ways to find out the IP address of a host given a domain name
    - Example: nslookup www.vsu.edu (which works on most platforms including Windows, Unix/Linux, and Mac OS X)
  - Convert the first number (from left) to a binary number, then take a look at the 1^{st}, and/or 2^{nd}, and/or 3^{rd} bit

Fragmentation and Reassembly (1)

- Different network has different MTU
  - Maximum Transmission Unit
  - Examples
    - MTU of typical Ethernet = 1500 bytes
    - MTU of typical FDDI = 4500 bytes



Q: What if an IP packet is greater than the MTU of the underlying network?

MTU Example: MS Windows

Command Prompt

```
H:\>ping

Usage: ping [-t] [-a] [-n count] [-l size] [-f] [-i TTL] [-v TOS]
            [-r count] [-s count] [[-j host-list] | [-k host-list]]
            [-w timeout] target_name

Options:
    -t             Ping the specified host until stopped.
                   To see statistics and continue - type Control-Break;
                   To stop -
    -a             Resolve ad
    -n count       Number of
    -l size        Send buffe
    -f             Set Don't
    -i TTL         Time To Li
    -v TOS         Type Of Se
    -r count       Record rou
    -s count       Timestamp
    -j host-list   Loose sour
    -k host-list   Strict sou
    -w timeout     Timeout in

H:\>_
```

Command Prompt

```
H:\>ipconfig

Windows IP Configuration


Ethernet adapter Local Area Connection:

        Connection-specific DNS Suffix  . :
        IP Address. . . . . . . . . . . . : 192.168.1.52
        Subnet Mask . . . . . . . . . . . : 255.255.255.0
        Default Gateway . . . . . . . . . : 192.168.1.1

H:\>ping -f -l 1492 -n 1 192.168.1.52

Pinging 192.168.1.52 with 1492 bytes of data:

Reply from 192.168.1.52: bytes=1492 time<1ms TTL=128

Ping statistics for 192.168.1.52:
    Packets: Sent = 1, Received = 1, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 0ms, Average = 0ms

H:\>ping -f -l 1493 -n 1 192.168.1.52

Pinging 192.168.1.52 with 1493 bytes of data:

Packet needs to be fragmented but DF set.
```

What is the MTU for the network interface on this Windows box?

MTU Example: Linux

debian@dVM1: ~

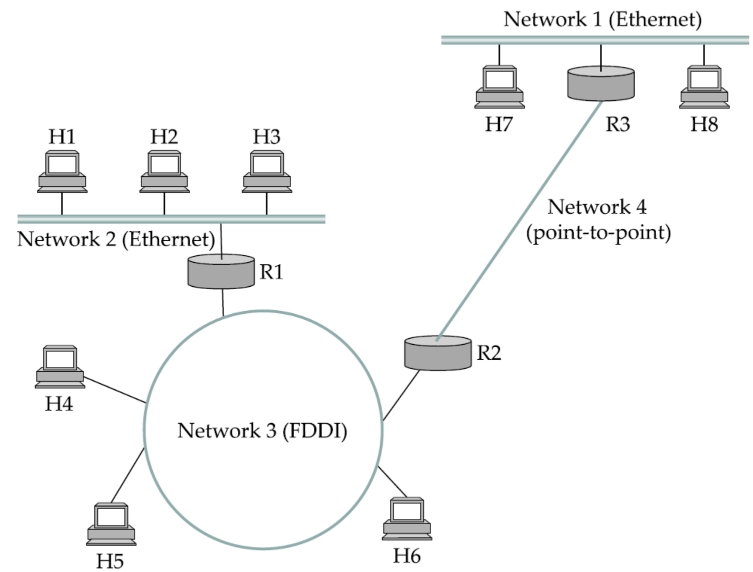
```
debian@dVM1:~$ ip link show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode DEFAULT
 group default
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP mo
de DEFAULT group default qlen 1000
    link/ether 08:00:27:18:91:8d brd ff:ff:ff:ff:ff:ff
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP mo
de DEFAULT group default qlen 1000
    link/ether 08:00:27:cb:a5:01 brd ff:ff:ff:ff:ff:ff
4: eth2: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN mode DEFAULT group
 default qlen 1000
    link/ether 08:00:27:8d:80:e4 brd ff:ff:ff:ff:ff:ff
5: eth3: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN mode DEFAULT group
 default qlen 1000
    link/ether 08:00:27:6e:88:91 brd ff:ff:ff:ff:ff:ff
debian@dVM1:~$
```

Exercise L10-2

- Use the approaches introduced to find the MTU for the network interface of
  - the Windows box in front of you
  - The Linux virtual machine

Example: internet requires fragmentation

- Assume
  - IP packet
    - Data: 1400 bytes
    - IP header: 20 bytes
  - MTU
    - Ethernet=1500
    - FDDI=4500
    - PPP=532

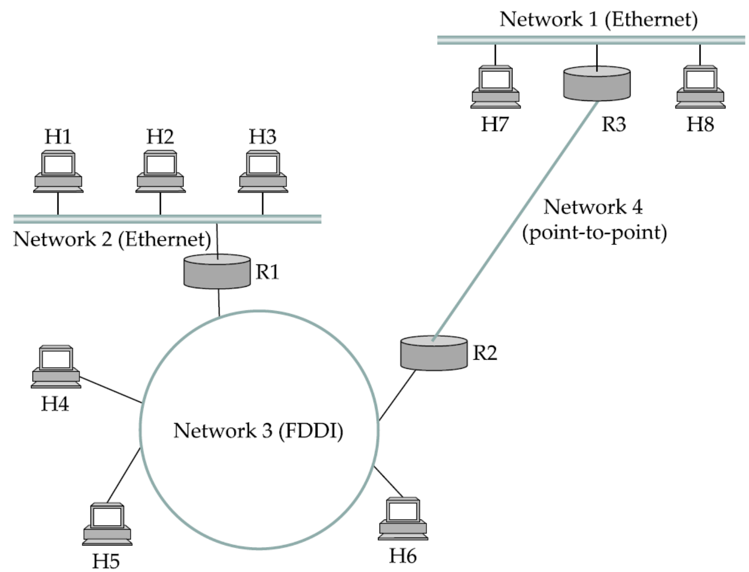


Fragmentation and Assembly (2)

- Fragmentation
  - **Router** divides the received IP packet into many small ones if necessary
    - Fragments
    - Each fragment is an IP packet
  - Send them using underlying network
- Assembly
  - **Receiving host** assembles the received fragments and put them together

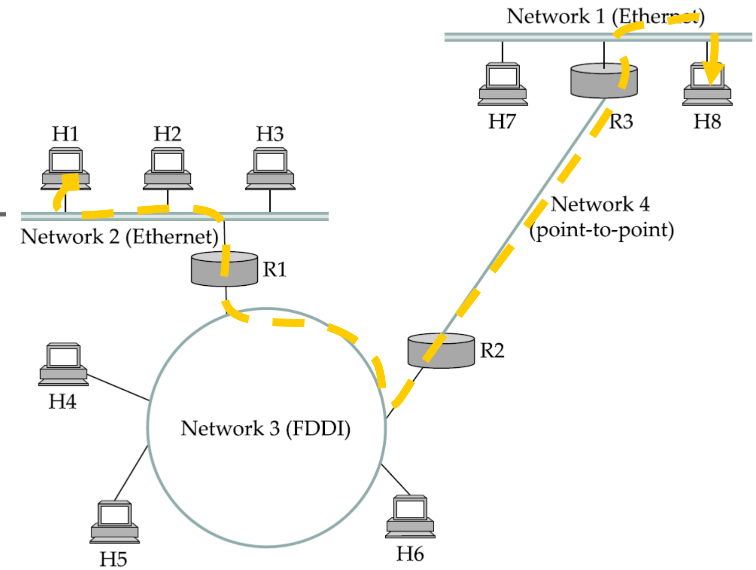
Example: internet requires fragmentation

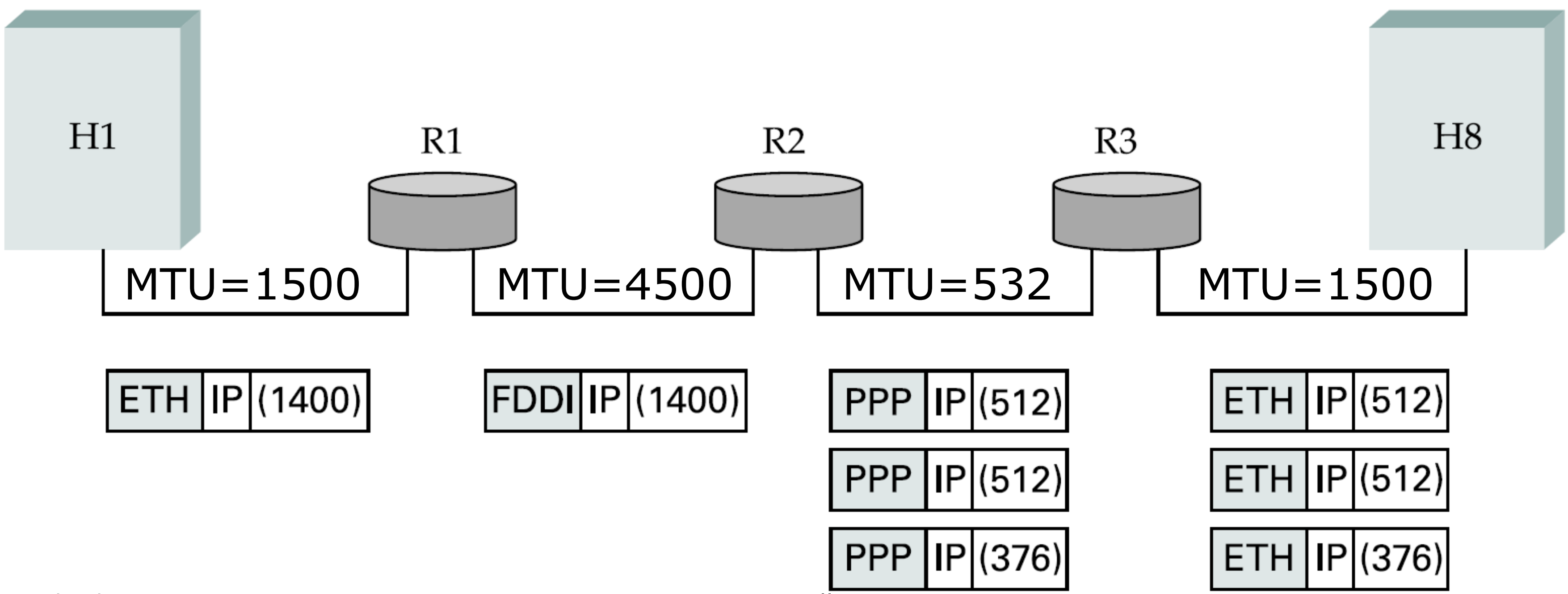
- Assume
  - IP packet
    - Data: 1400 bytes
    - IP header: 20 bytes
  - MTU
    - Ethernet=1500
    - FDDI=4500
    - PPP=532

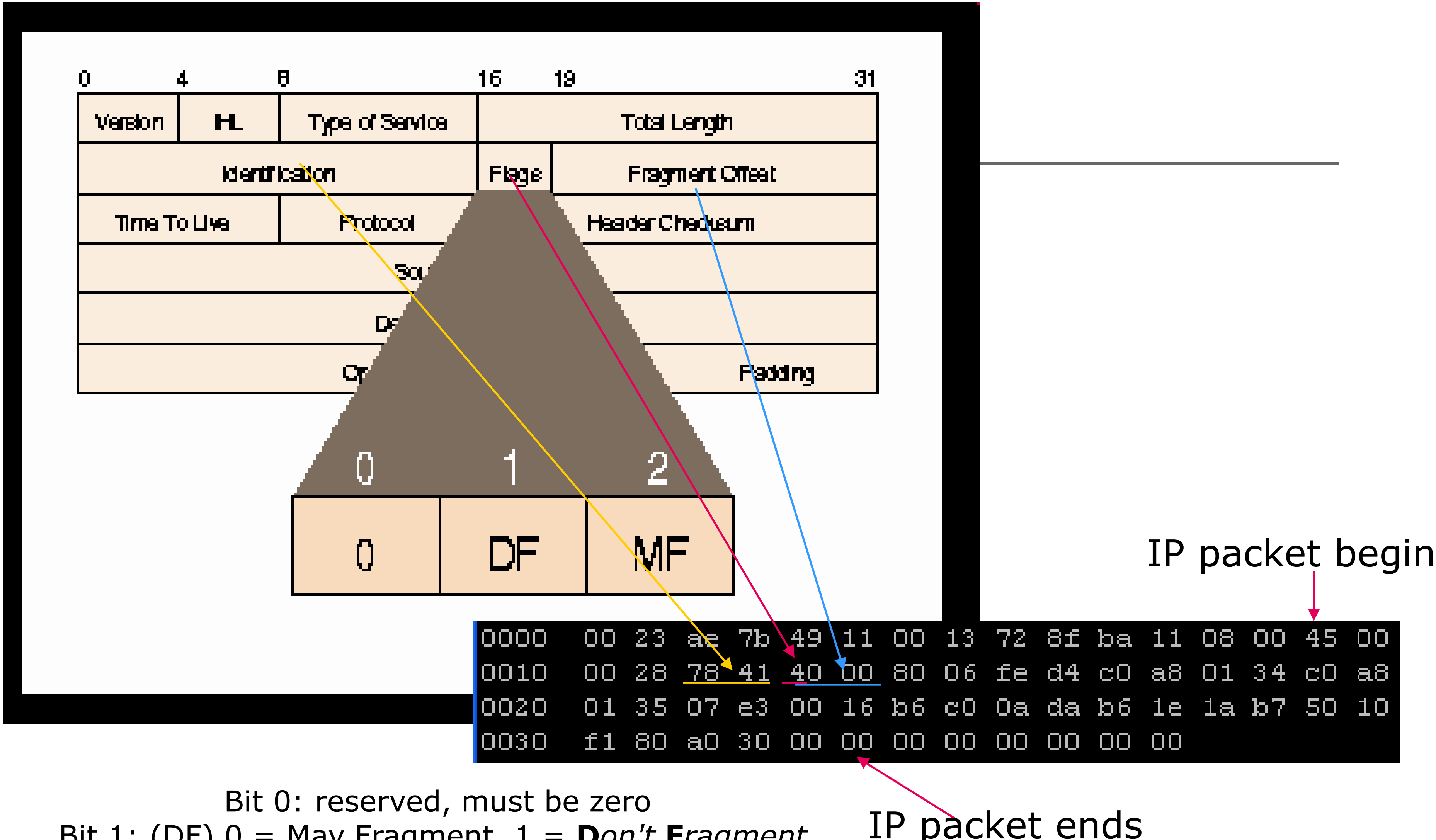


Example

IP packet
Data: 1400 bytes
IP header: 20 bytes



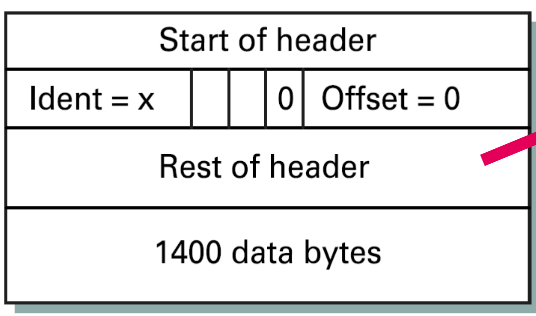




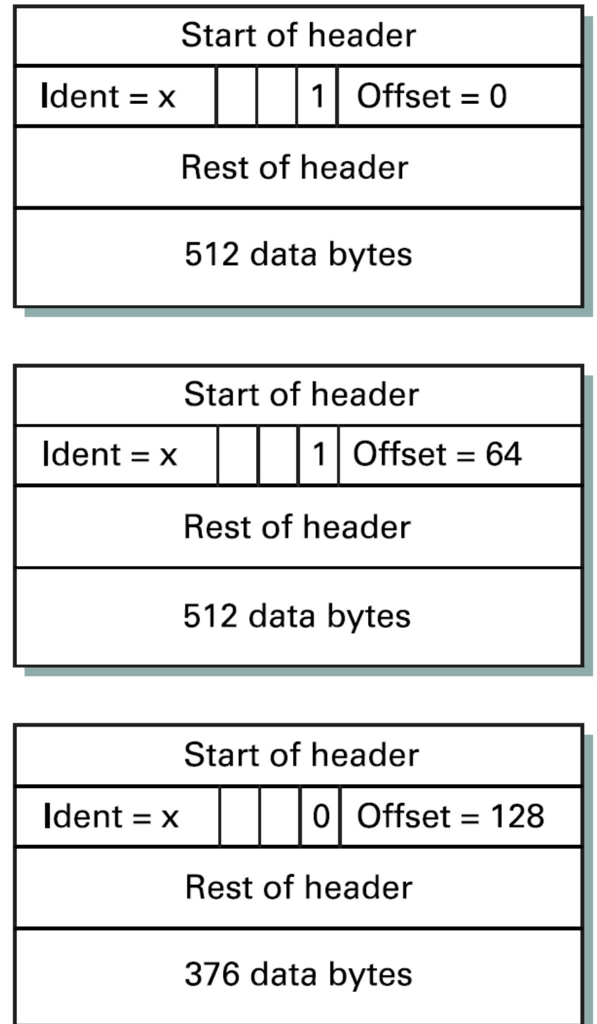
Bit 0: reserved, must be zero
Bit 1: (DF) 0 = May Fragment, 1 = **D**_on't_ **F**_ragment_.
Bit 2: (MF) 0 = Last Fragment, 1 = **M**_ore_ **F**_ragments_.
Source: http://www.freesoft.org/CIE/Course/Section3/7.htm

Example

Ident:
Same across all fragments
Unique for each packet
MF (M_{ore} $F_{ragments}$) bit in Flags
set → more fragments to follow
0 → last fragment
Offset: in terms of 8 byte chunks

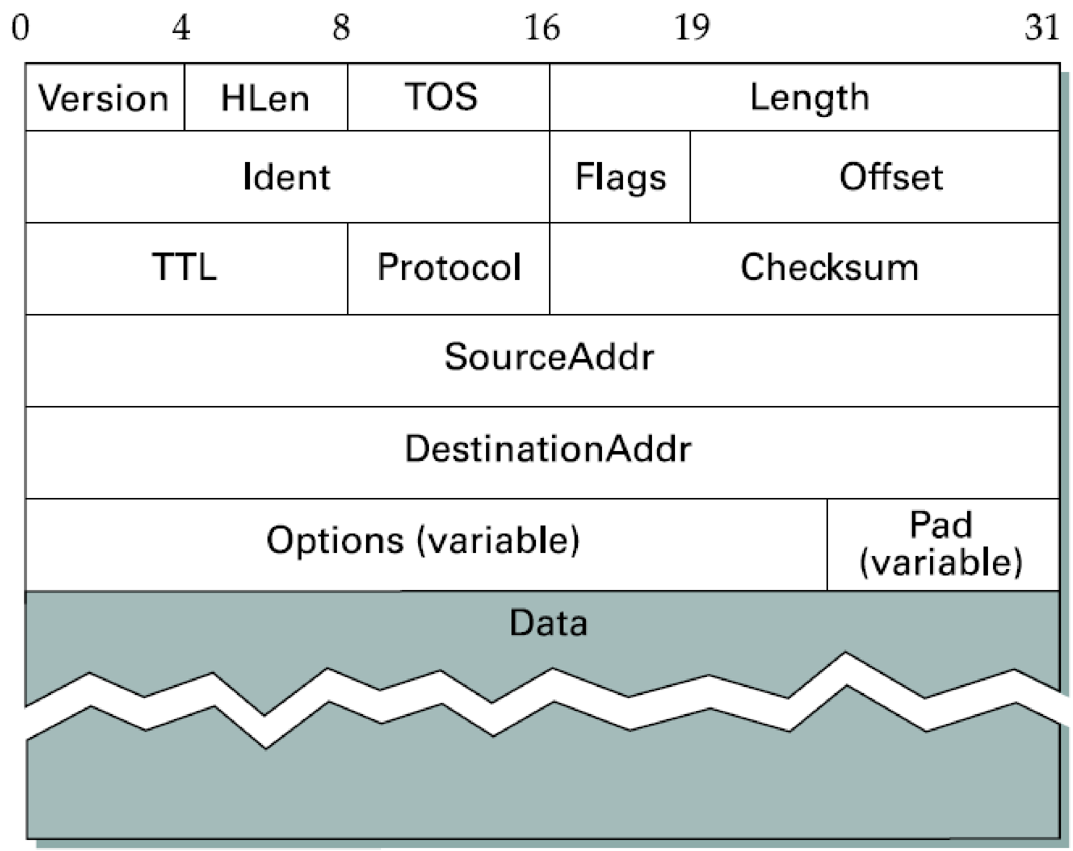


Fragmented into three fragments

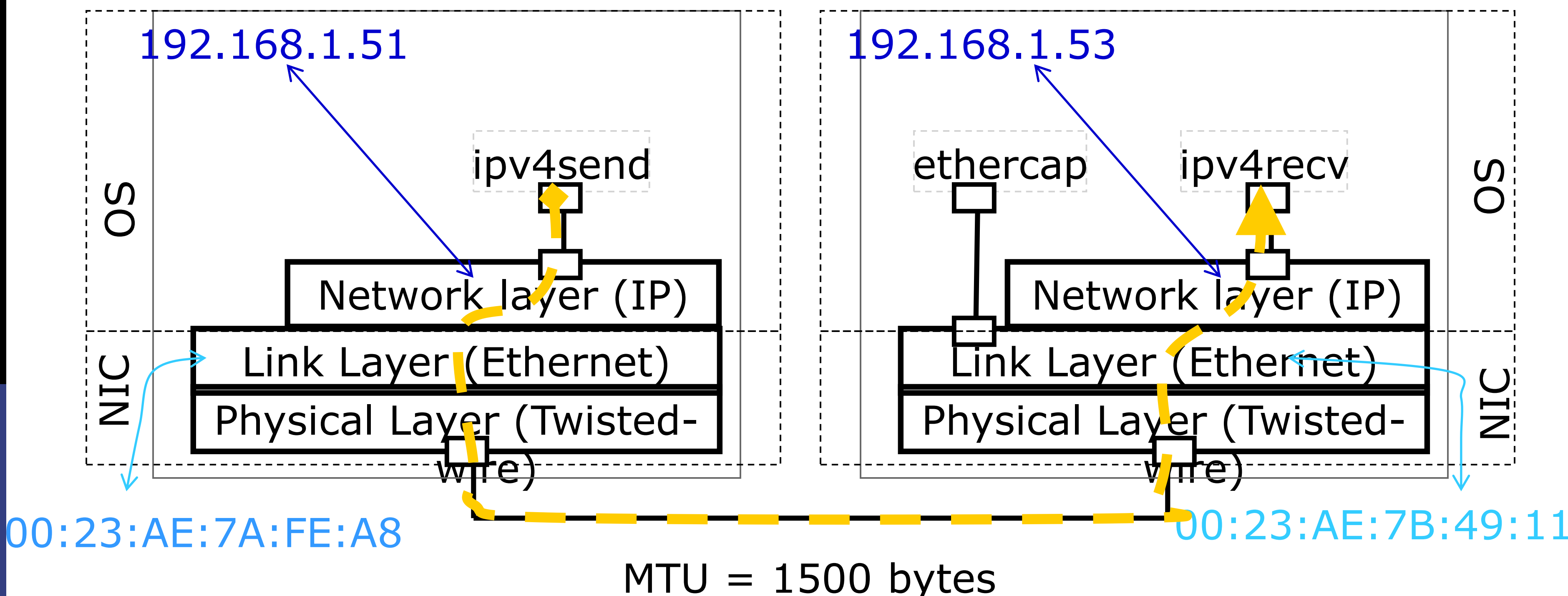


Q: why 8-byte chunks?

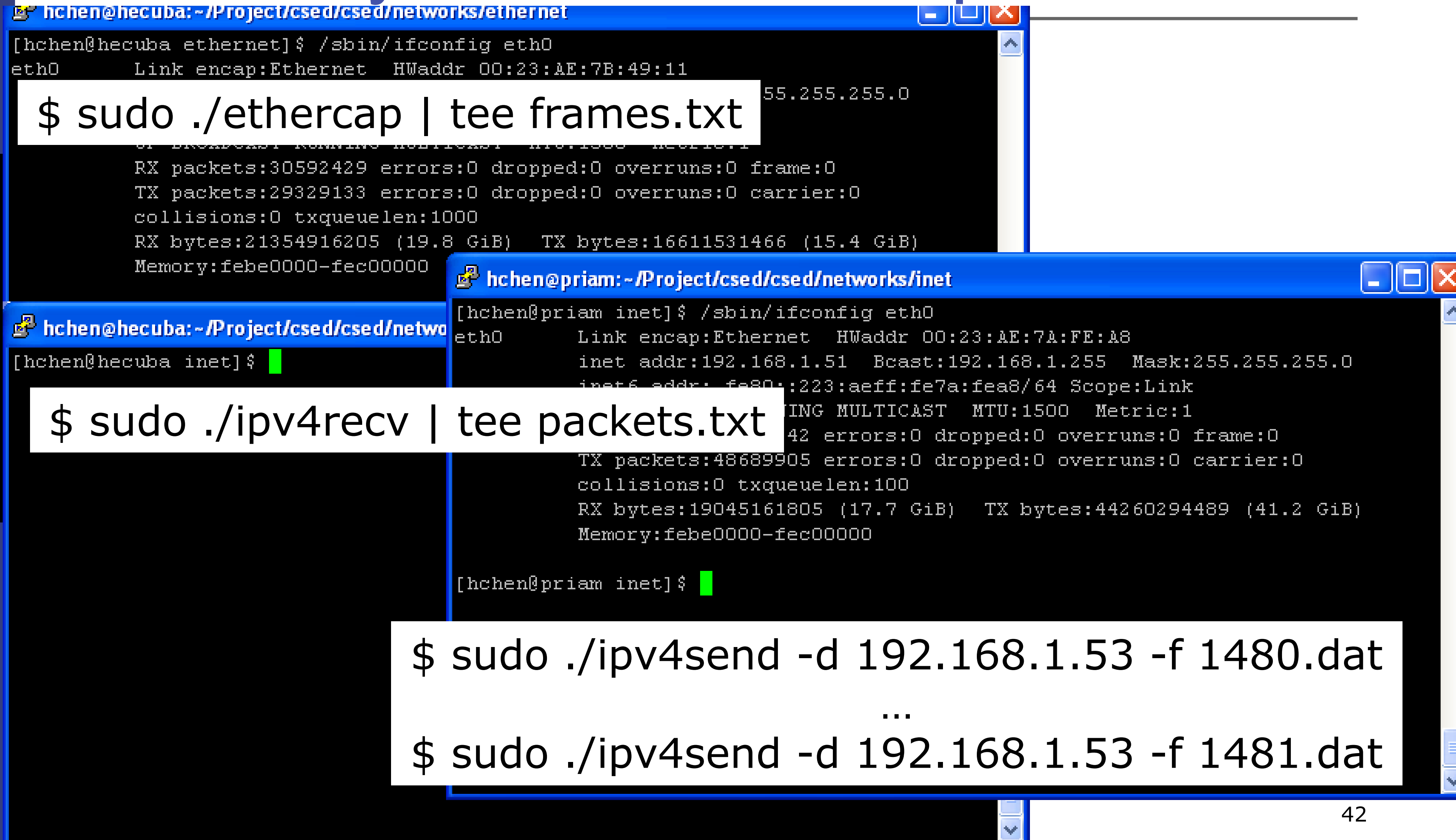
Hint for “*Why 8-byte Chunk?*”



Experiment: IP Fragmentation and Assembly in Practice - Setup



Experiment: IP Fragmentation and Assembly in Practice - Experiment




```
$ sudo ./ethercap | tee frames.txt
```

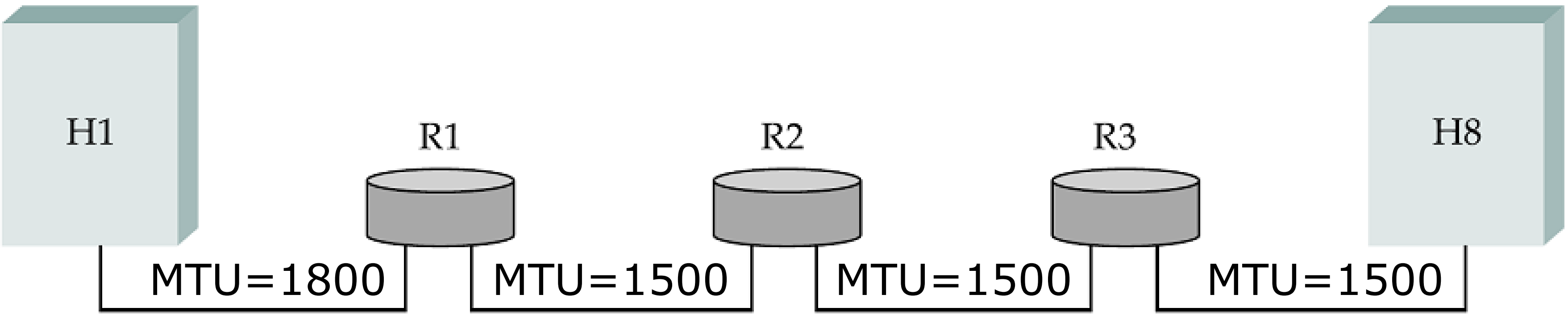
```
$ sudo ./ipv4recv | tee packets.txt
```

```
$ sudo ./ipv4send -d 192.168.1.53 -f 1480.dat
…
$ sudo ./ipv4send -d 192.168.1.53 -f 1481.dat
```

Experiment: IP Fragmentation and Assembly in Practice - Experiment

- Demonstration
  - Experiment 1: Transmit a file (or message) of MTU bytes
  - Experiment 2: Transmit a file (or message) of MTU + 1 bytes
  - Observe Ethernet headers and IP headers of relevant frames/packets

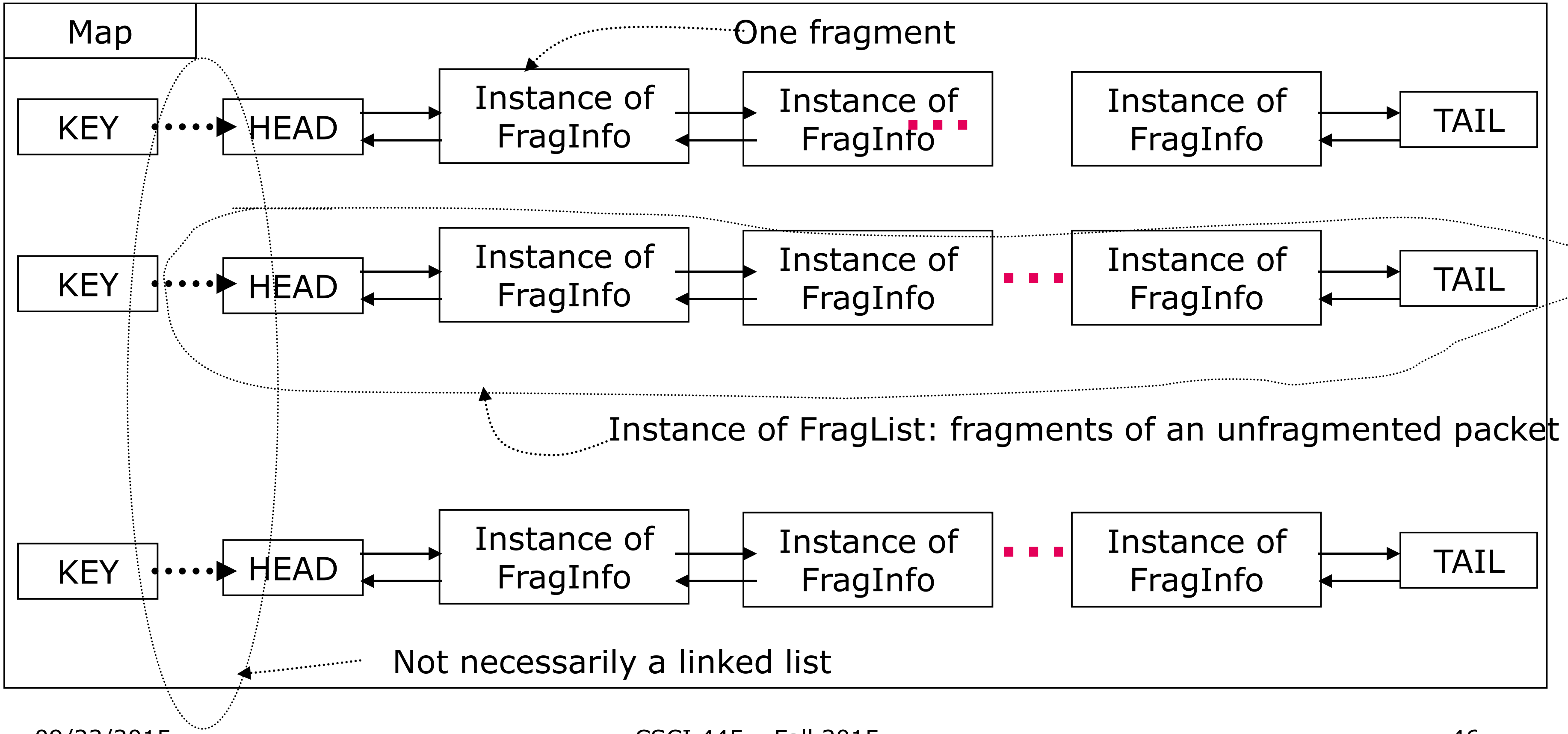
Example



Q: H1 sends an IP packet of 1800 bytes including IP header to H8. Please show
1) IP datagrams traversing the sequence of physical network graphed above
2) Header fields used at each router and hosts

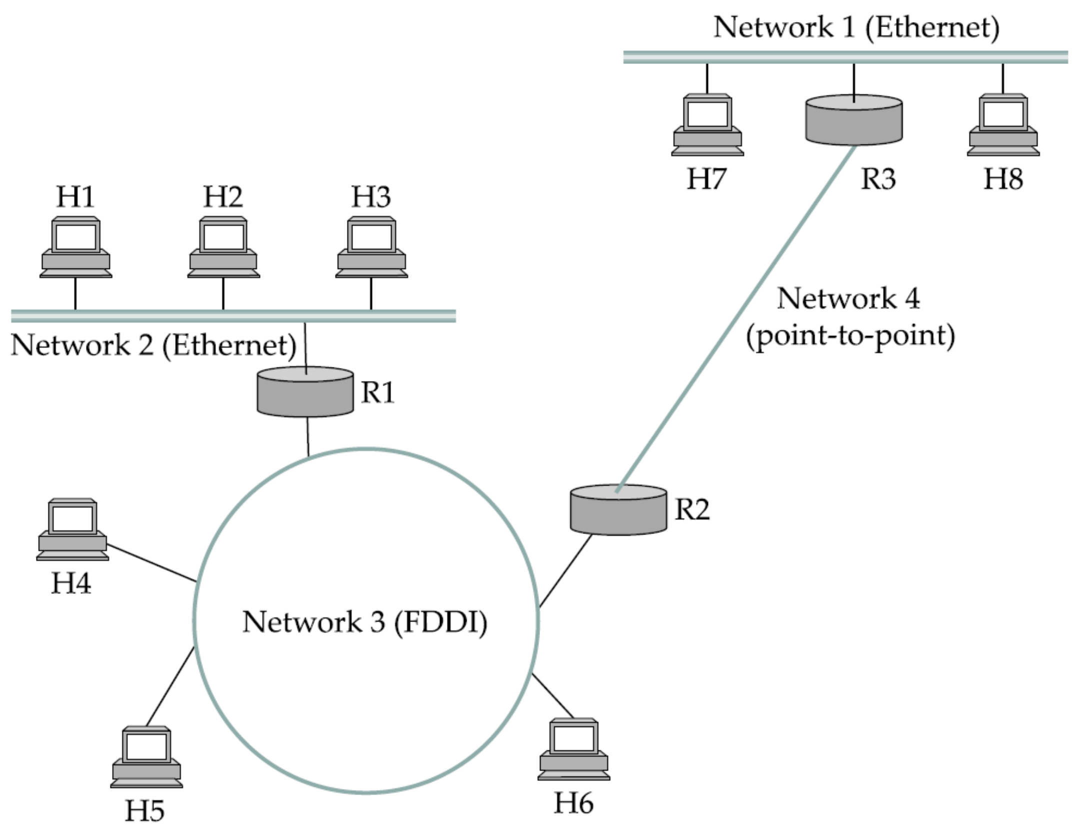
Implementation of Reassembly

- Hints to understand the program (pp.243-247)



H1 Sends to H8?

- Networks connected by routers
  - Networks may be different
  - Routers: nodes that internet networks
    - gateways
    - R1, R2, R3



Packet Forwarding: Datagram Forwarding - Internet Protocol Service Model

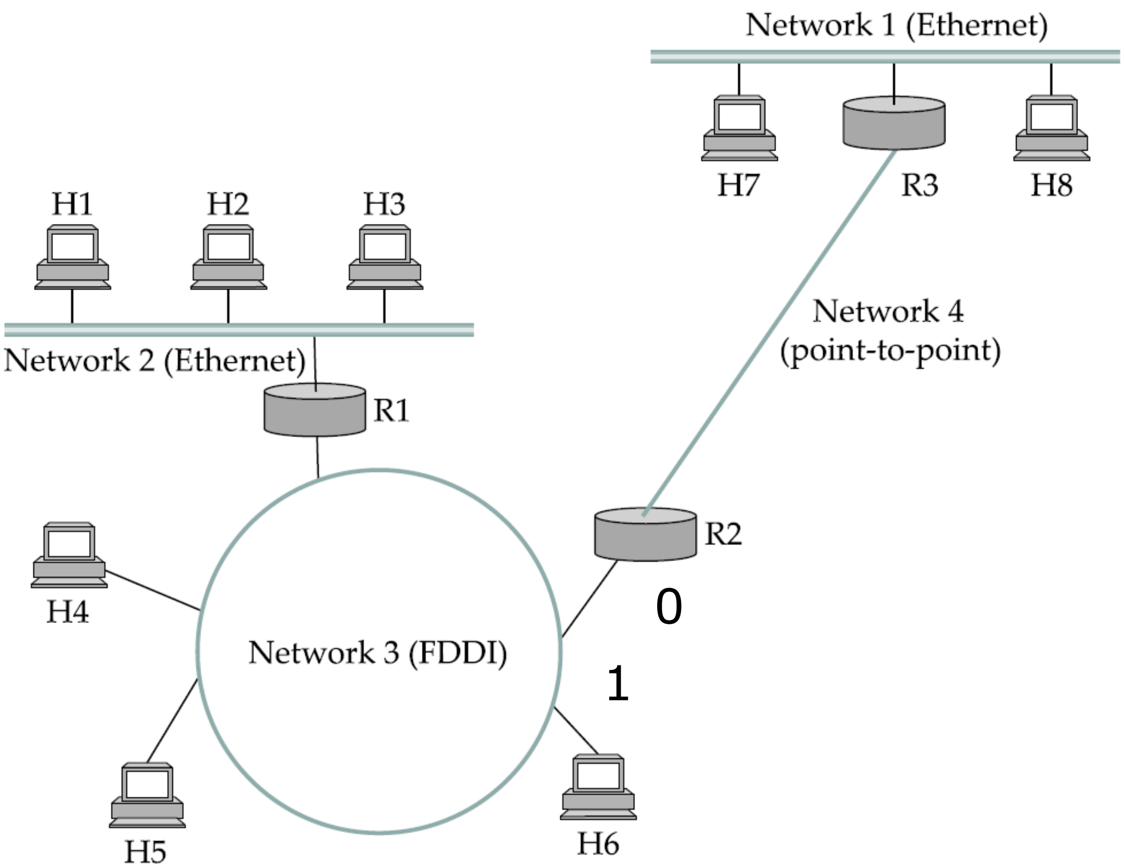
- Datagram delivery
  - Connectionless
  - Best-effort delivery
    - packets may be lost
    - packets may be delivered out of order
    - duplicate copies of a packet may be delivered
    - packets can be delayed for a long time
- Easy to run on top of any types of networks

Packet Forwarding: Datagram Forwarding – Forwarding Strategy

- Strategy
  - Every datagram contains destination's address
  - If directly connected to destination network, then forward to host
  - If not directly connected to destination network, then forward to some router based on a forwarding table
    - Each router maintains a forwarding table
    - Forwarding table maps network number into next hop
    - Each host has a default router

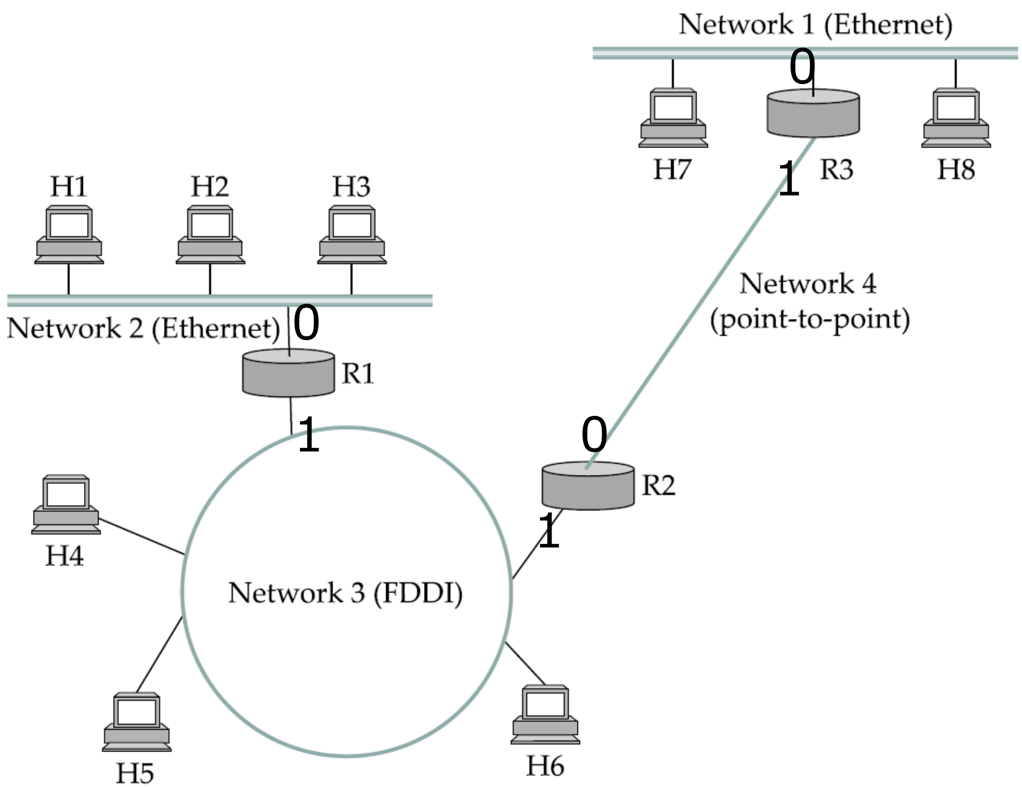
Datagram Forwarding

- Example: Forwarding Table of Router R2



Network Number	Next Hop
1	R3
2	R1
3	interface 1
4	interface 0

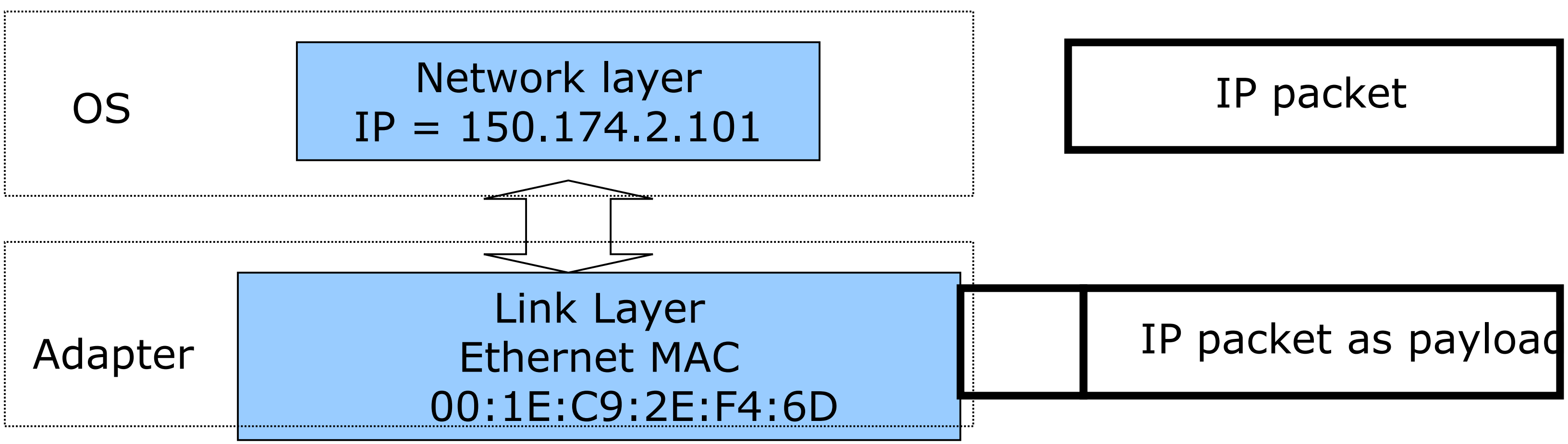
Exercise L10-4



Q: construct forwarding tables for routers R1 and R3

IP address → Physical Address?

- Questions:
  - In an IP network, an IP packet is the payload of one or more Ethernet frames. Who prepares the Ethernet frame headers which contain destination Ethernet/physical address of the destination node?
  - How does the source node know the Ethernet/physical address of the destination node?

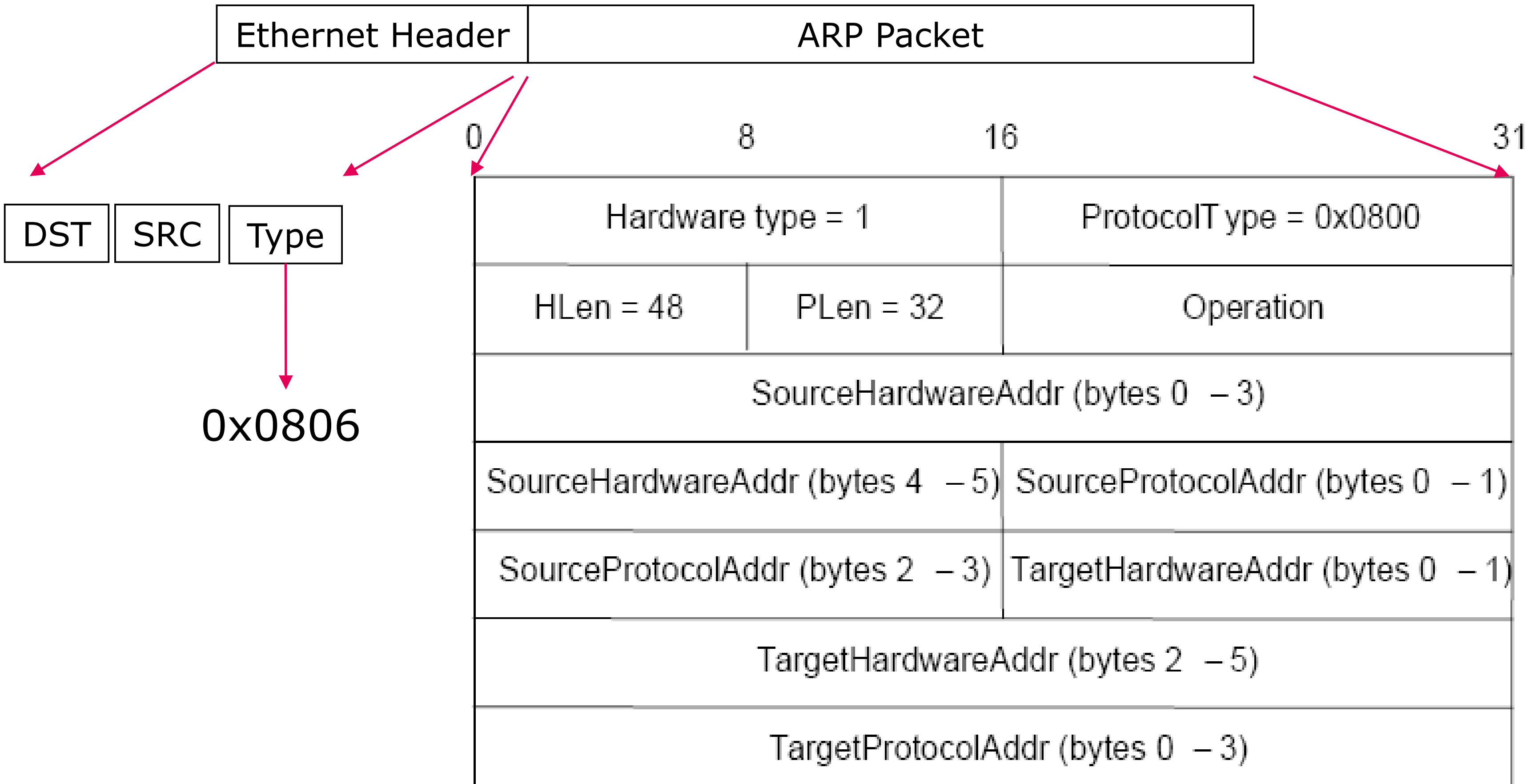


IP Address → Physical Address: Solution

- Map IP addresses into physical addresses
  - destination host
  - next hop router
- Address Resolution Protocol (ARP)
  - Data structure
    - Table of IP to physical address bindings (ARP table/ARP cache)
  - Mechanism
    - Broadcast request if IP address not in table
    - Target machine responds with its physical address
    - Table entries are discarded if not refreshed

ARP Packet Format

- An ARP packet is the payload of a frame



ARP Packet: Examples

Ethernet Header	ARP Packet

DST	SRC	Type

0x0806

<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">ProtocolT ype = 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>

Operation (opcode)
- 1 request
- 2 reply
- 3 request reverse
- 4 reply reverse

```
Interface: eth0
0000  ff ff ff ff ff ff 00 0c 29 a0 51 6d 08 06 00 01    ........).Qm....
0010  08 00 06 04 00 01 00 0c 29 a0 51 6d c0 a8 01 48    ........).Qm...H
0020  00 00 00 00 00 00 c0 a8 01 33                      ..........3
Interface: eth0
0000  ff ff ff ff ff ff 00 0c 29 a0 51 6d 08 06 00 01    ........).Qm....
0010  08 00 06 04 00 01 00 0c 29 a0 51 6d c0 a8 01 48    ........).Qm...H
0020  00 00 00 00 00 00 c0 a8 01 33                      ..........3
```

ARP

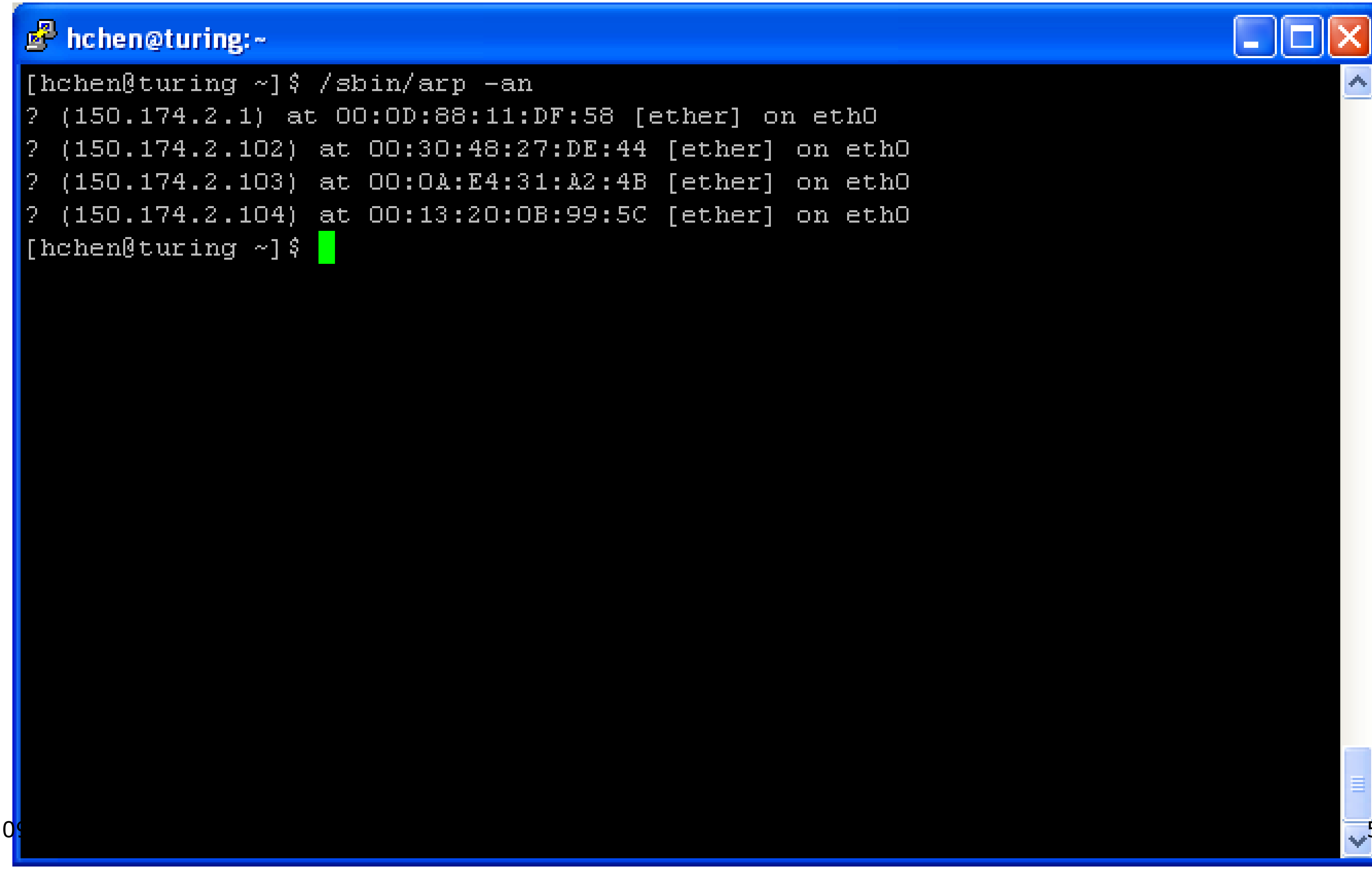
- Request Format
  - HardwareType: type of physical network (e.g., Ethernet)
  - ProtocolType: type of higher layer protocol (e.g., IP)
  - HLEN & PLEN: length of physical and protocol addresses
  - Operation: request or response
  - Source/Target-Physical/Protocol addresses
- Notes
  - Prevent stalled entries
    - Table entries will timeout (~15 minutes)
    - Do not refresh table entries upon reference
  - Fresh entries (reset timer)
    - Update table if already have an entry
  - Reduce ARP messages
    - Update table with source when you are the target in ARP request messages

ARP in Practice (1)

Command Prompt

```
H:\>arp -a

Interface: 192.168.1.52 --- 0x2
  Internet Address      Physical Address      Type
  192.168.1.1           00-23-69-5b-3e-b0     dynamic
  192.168.1.51          00-23-ae-7a-fe-a8     dynamic

H:\>
```

ARP in Practice (2)



Host Configuration

- Network configuration
  - IP addresses
    - Unique on a network
    - Reflect structure of the network
  - Default router/gateway
- Mechanism
  - Manual configuration
    - Does not scale up
    - Error-prone
  - Automatic configuration
    - Dynamic Host Configuration Protocol (DHCP)

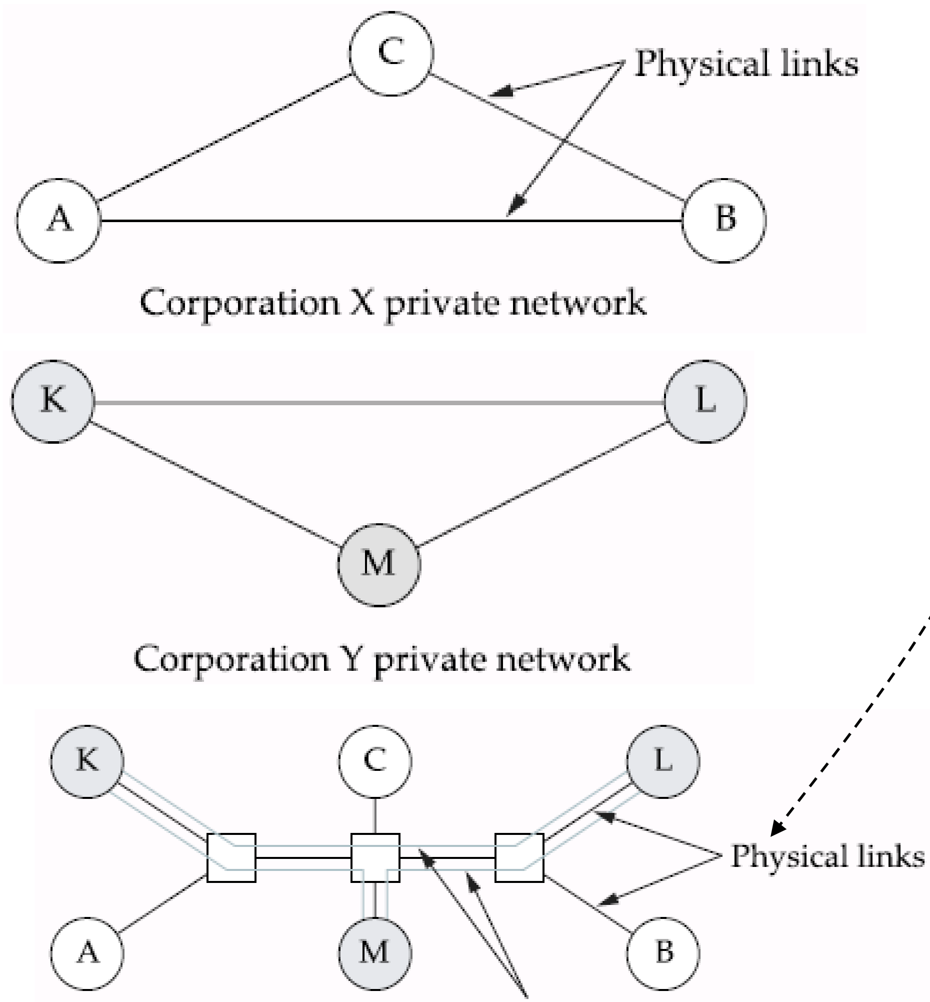
Error Reporting: Internet Control Message Protocol (ICMP)

- Echo (ping)
- Redirect (from router to source host)
- Destination unreachable (protocol, port, or host)
- TTL exceeded (so datagrams don’t cycle forever)
- Checksum failed
- Reassembly failed
- Cannot fragment

Virtual Networks and Tunnels

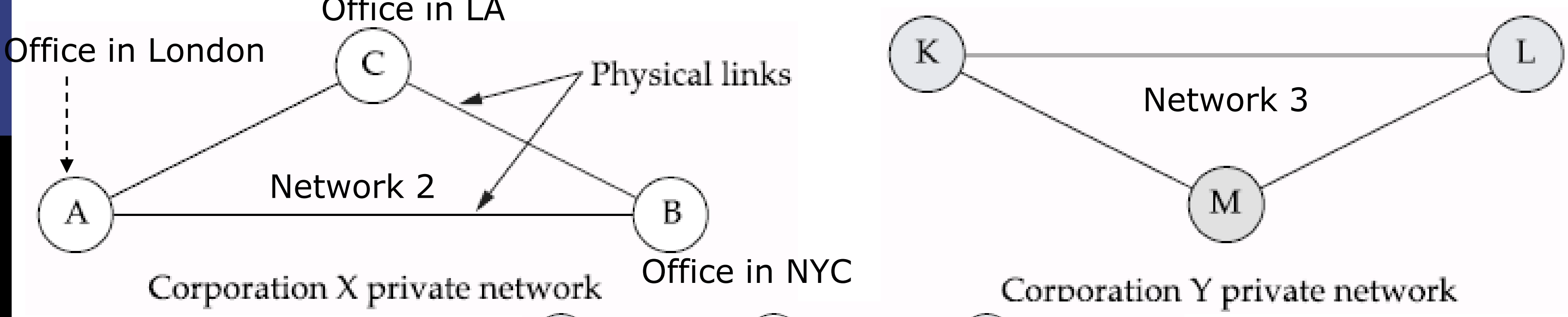
- Internetworks often have shared infrastructure networks
- Data packets may not be forwarded without restriction
- Virtual Price Networks (VPN)
  - VPN is a heavily overused and definitions vary
  - An “private” network utilizing an shared network infrastructure

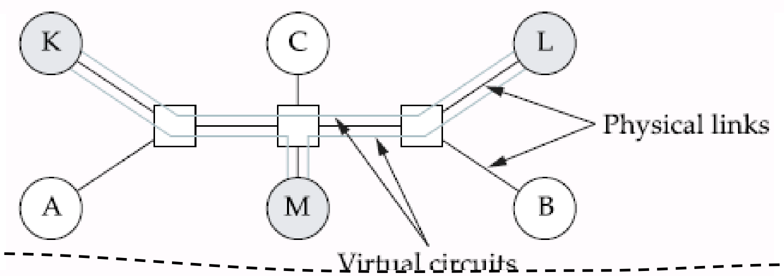
Virtual Private Networks: Example



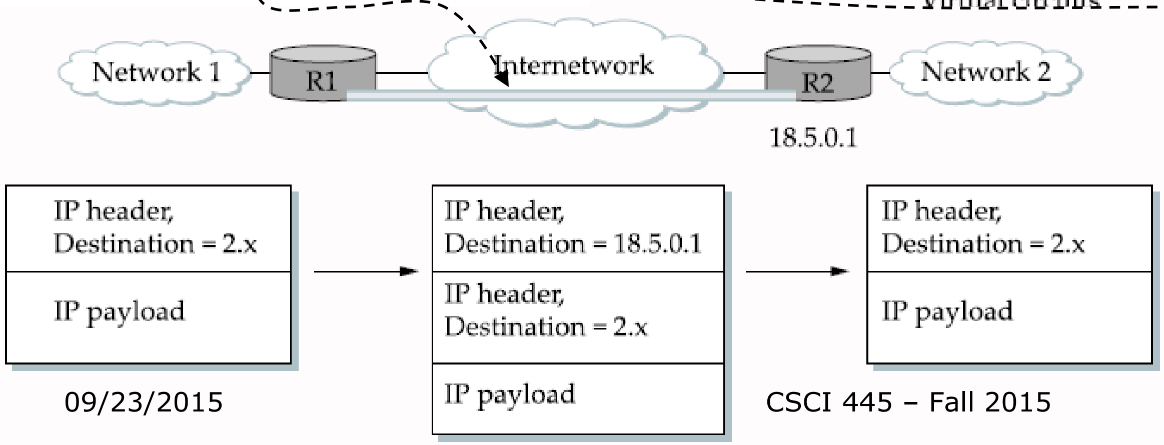
- Corporations X and Y want their own networks via "leased lines" belonging to other networks
- X wants to keep their data private
- So does Y
- X and Y have "virtual" private networks
- "virtualization" can be done on different layers
  - Layer 2 VPN
  - Layer 3 VPN

Virtual Private Networks via IP Tunneling





Pre-configured tunnel



Forwarding Table at R1

NetworkNum	NextHop
1	Interface 0
2	Virtual interface 0
Default	Interface 1

Summary

- internet and the Internet
- Global addressing scheme
- Packet fragmentation and assembly
- Best effort service model and datagram forwarding
- Address translation
- Host configuration
- Error reporting