# Network Introduction

wangth (2018-2023, CC BY-SA)
? (2009-2017)

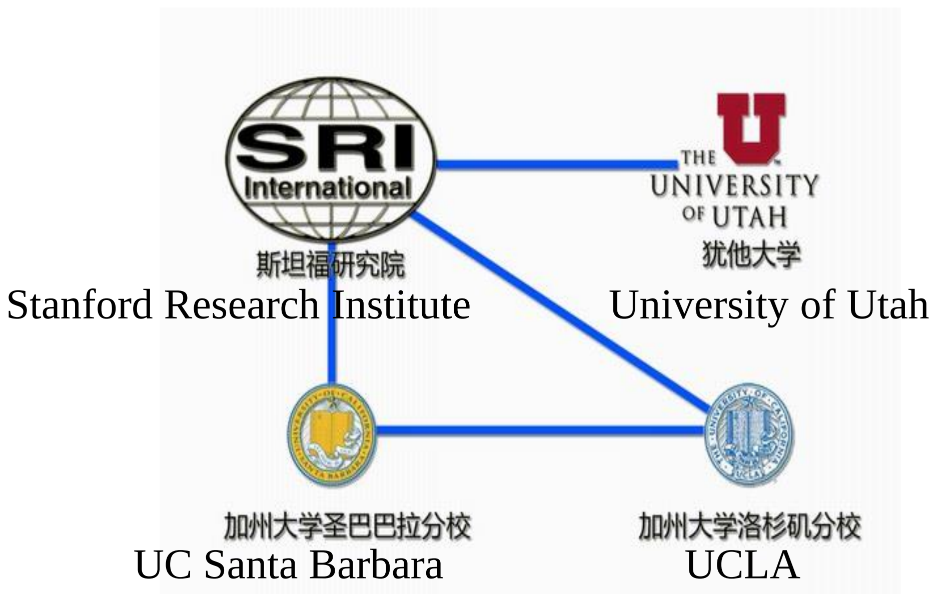
國立陽明交通大學資工系資訊中心
Computer Center of Department of Computer Science, NYCU

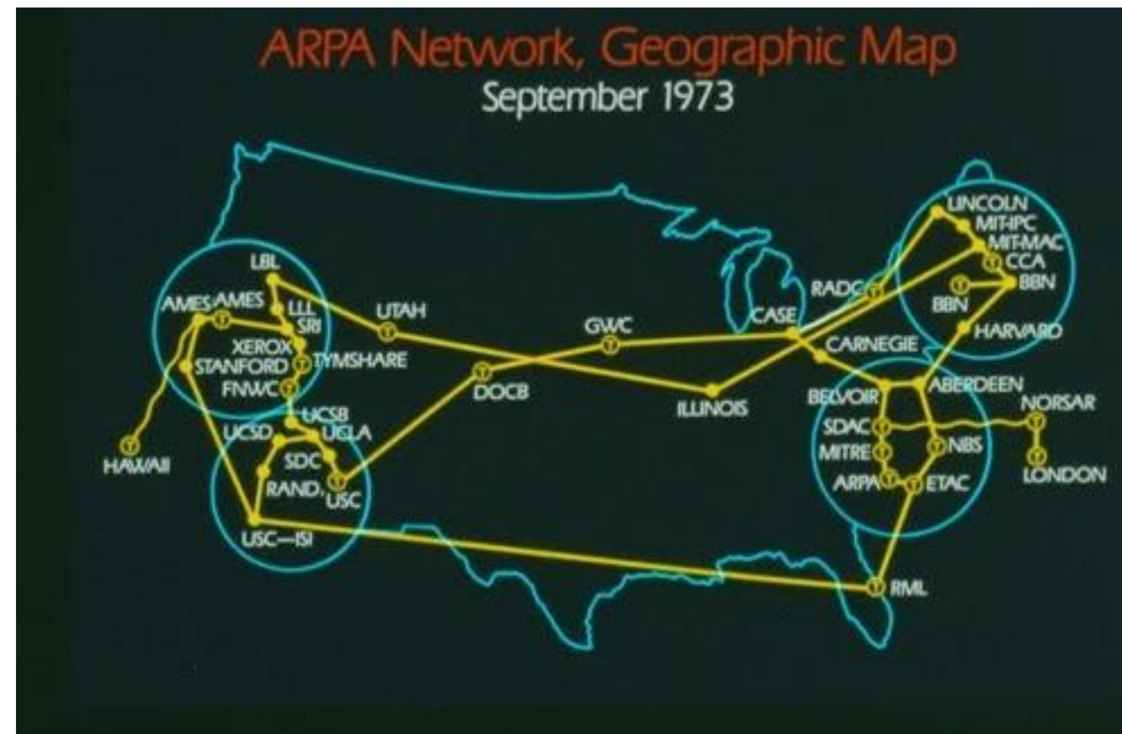
# TCP/IP and the Internet

- In 1969
  - ARPA funded and created the “ARPANet” network
    - 美國高等研究計劃署 (ARPA: Advanced Research Project Agency)
    - NCP – Network Control Protocol
      - Allow an exchange of information between separated computers
- In 1973
  - How to connect ARPANet with SATNet and ALOHANet
  - TCP/IP begun to be developed
- In 1983
  - TCP/IP protocols replaced NCP as the ARPANet’s principal protocol
  - ARPANet → MILNet + ARPANet = Internet
- In 1985
  - The NSF created the NSFNet to connect to Internet
- In 1990
  - ARPANet passed out of existence, and in 1995, the NSFNet became the primary Internet backbone network

NSF: National Science Foundation

# Introduction – ARPANet

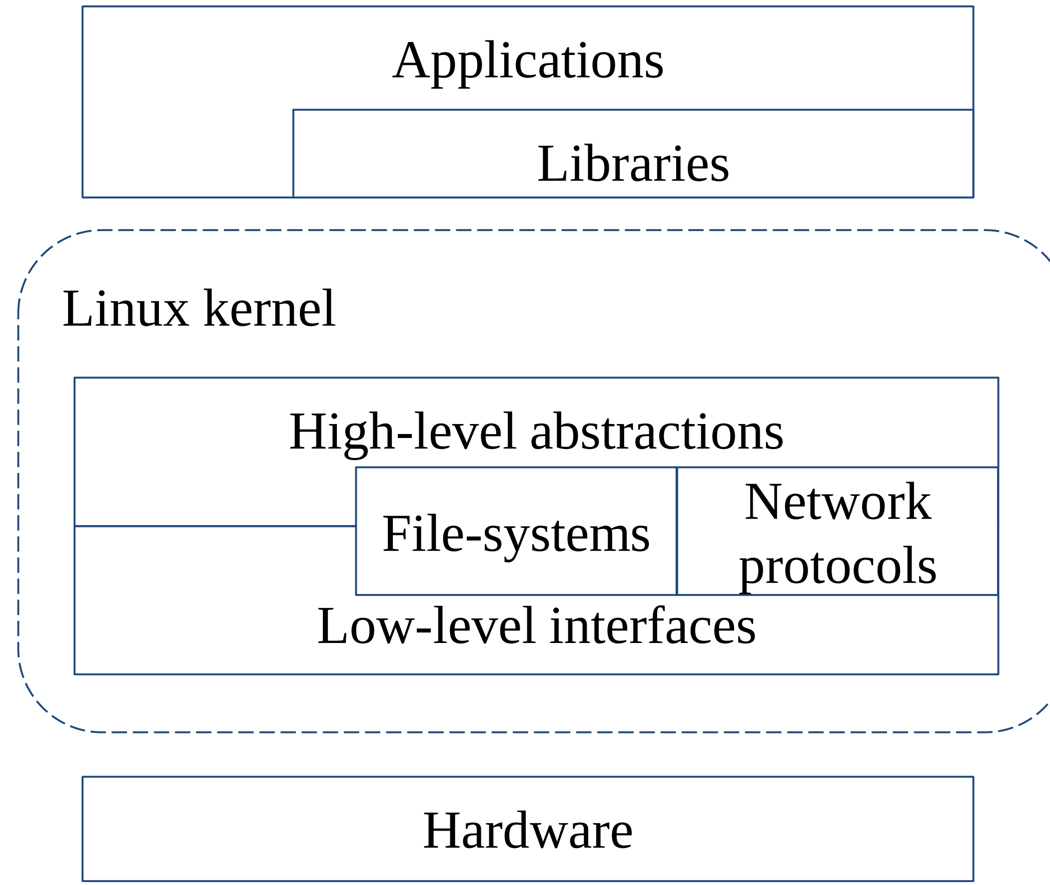



https://inventiontourblog.wordpress.com/2015/03/31/internet-advanced-research-project-agency-arpa-develops-the-first-computer-network/

# Introduction – Why TCP/IP ?

- The gap between applications and Network
  - Network
    - 802.3 Ethernet
    - 802.4 Token bus
    - 802.5 Token Ring
    - 802.11 Wireless
    - 802.16 WiMAX
  - Application
    - Reliable
    - Performance



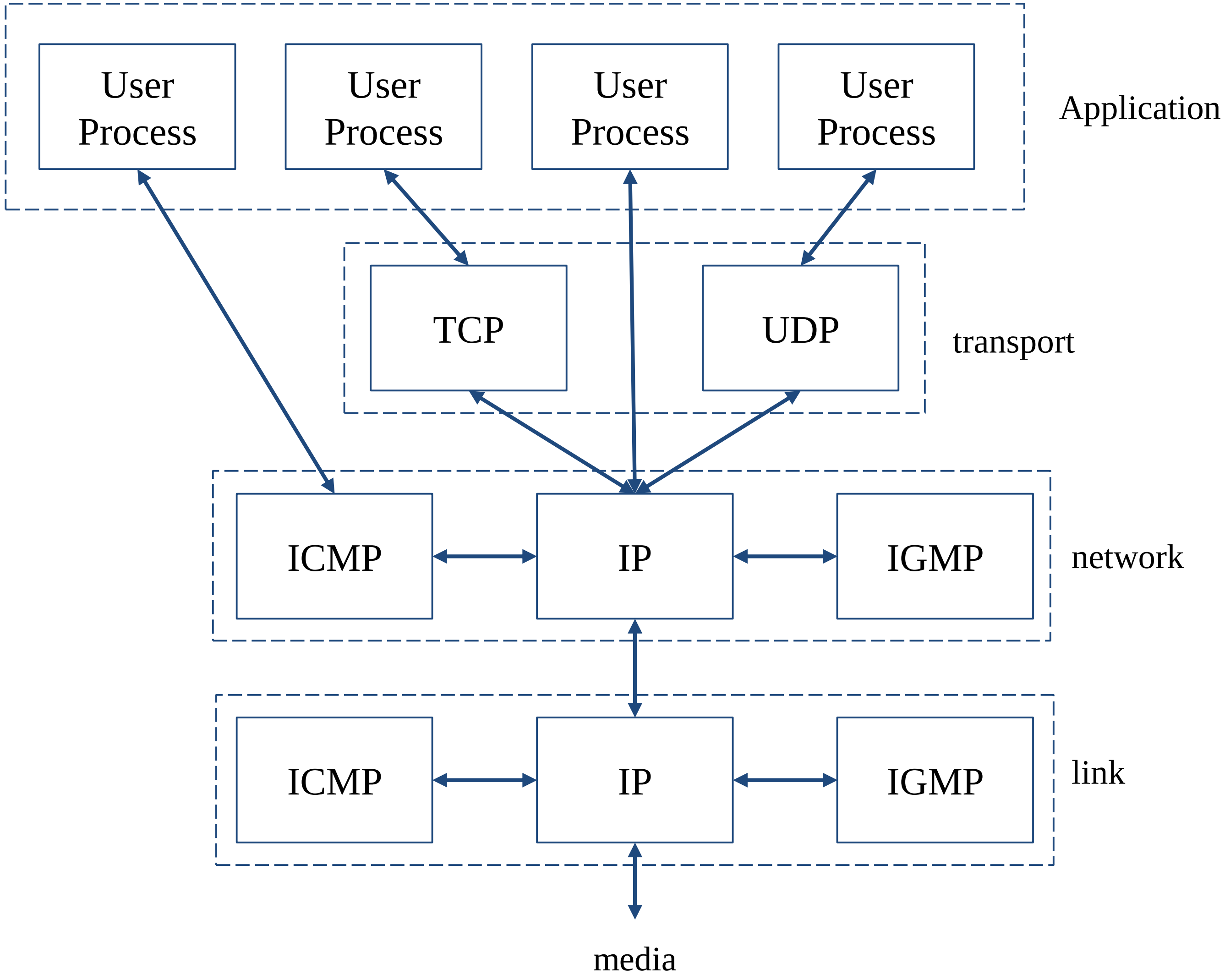

**We need something to do the translating work!**
**TCP/IP it is!!**

# Introduction – Layers of TCP/IP (1)

- TCP/IP is a suite of networking protocols
  - 4-layer architecture
    - Link layer (data-link layer)
      - Include device drivers to handle hardware details
    - Network layer (IP)
      - Handle the movement of packets around the network
    - Transport layer (Port)
      - Handle flow of data between hosts
    - Application

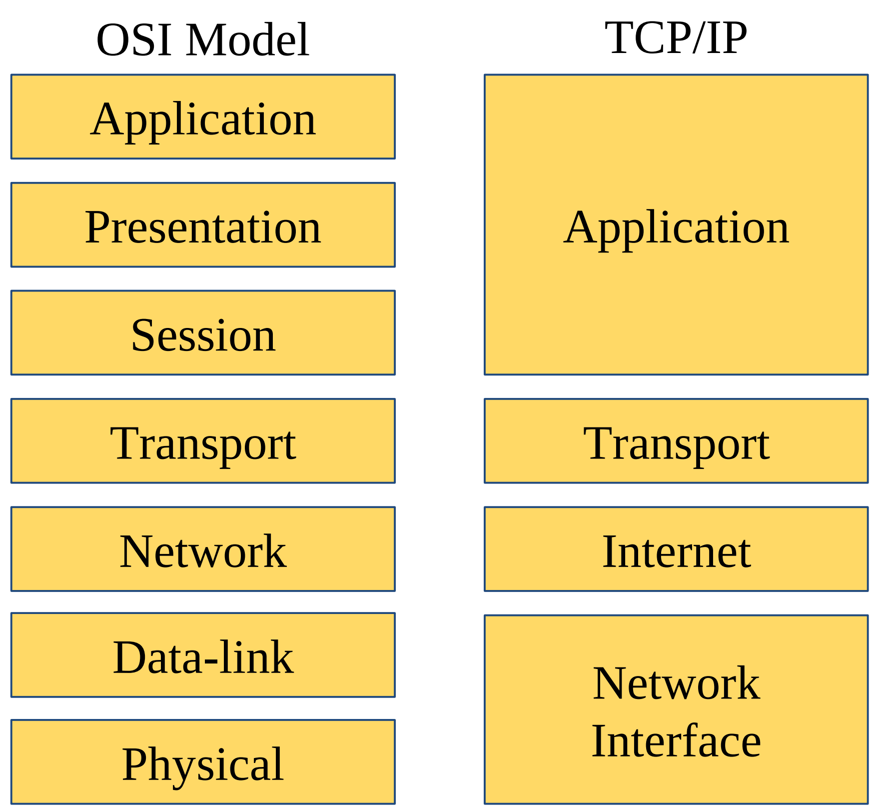
# Introduction – Layers of TCP/IP (2)

- Each layer has several protocols
  - A layer define a data communication function that may be performed by certain protocols
  - A protocol provides a service suitable to the function of that layer



# Introduction – Layers of TCP/IP (2)

- ISO/OSI Model (International Organization for Standardization / Open System Interconnection Reference Model)
- TCP/IP Model




TCP/IP and the OSI model

# Introduction

- TCP/IP
    - Used to provide data communication between hosts
        - How to delivery data reliably
        - How to address remote host on the network
        - How to handle different type of hardware device

# Introduction – Addressing

- Addressing
  - MAC Address
    - Media Access Control Address
    - 48-bit Network Interface Card Hardware Address
      - 24-bit manufacture ID
      - 24-bit serial number
    - Ex:
      - 00:07:e9:10:e6:6b
  - IP Address
    - 32-bit Internet Address (IPv4)
    - Ex:
      - 140.113.209.64
  - Port
    - 16-bit uniquely identify application (1 ~ 65536)
    - Ex:
      - FTP port 21, SSH port 22, Telnet port 23

# Link Layer

# Link Layer – Introduction of Link Layer

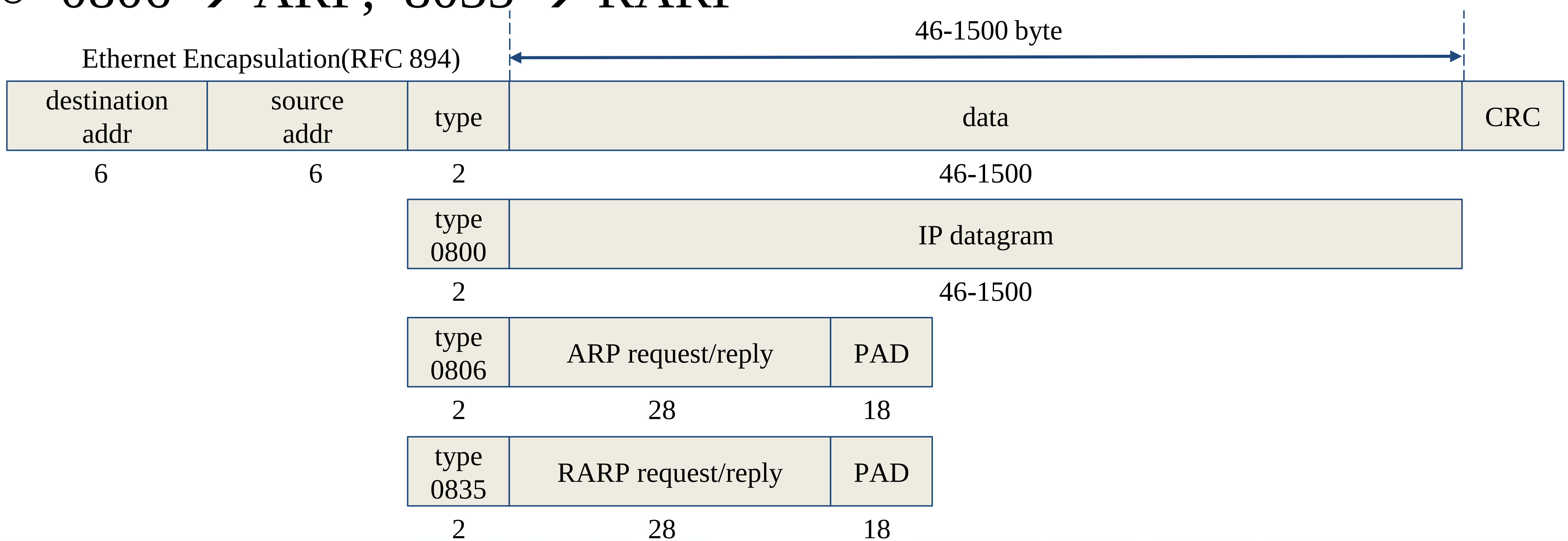
- Purpose of the link layer
  - Send and receive IP datagram for IP module
  - ARP request and reply
  - RARP request and reply
- TCP/IP support various link layers, depending on the type of hardware used:
  - Ethernet
    - Teach in this class
  - Token Ring
  - FDDI (Fiber Distributed Data Interface)
  - Serial Line

# Link Layer – Ethernet

- Features
  - Predominant form of local LAN technology used today
  - Use CSMA/CD
    - Carrier Sense, Multiple Access with Collision Detection
  - Use 48-bit MAC address
  - Operate at 10 Mbps
    - Fast Ethernet at 100 Mbps
    - Gigabit Ethernet at 1000 Mbps
    - 10 Gigabit Ethernet at 10,000 Mbps (10Gbps)
  - Ethernet frame format is defined in RFC 894
    - This is the actually used format in reality

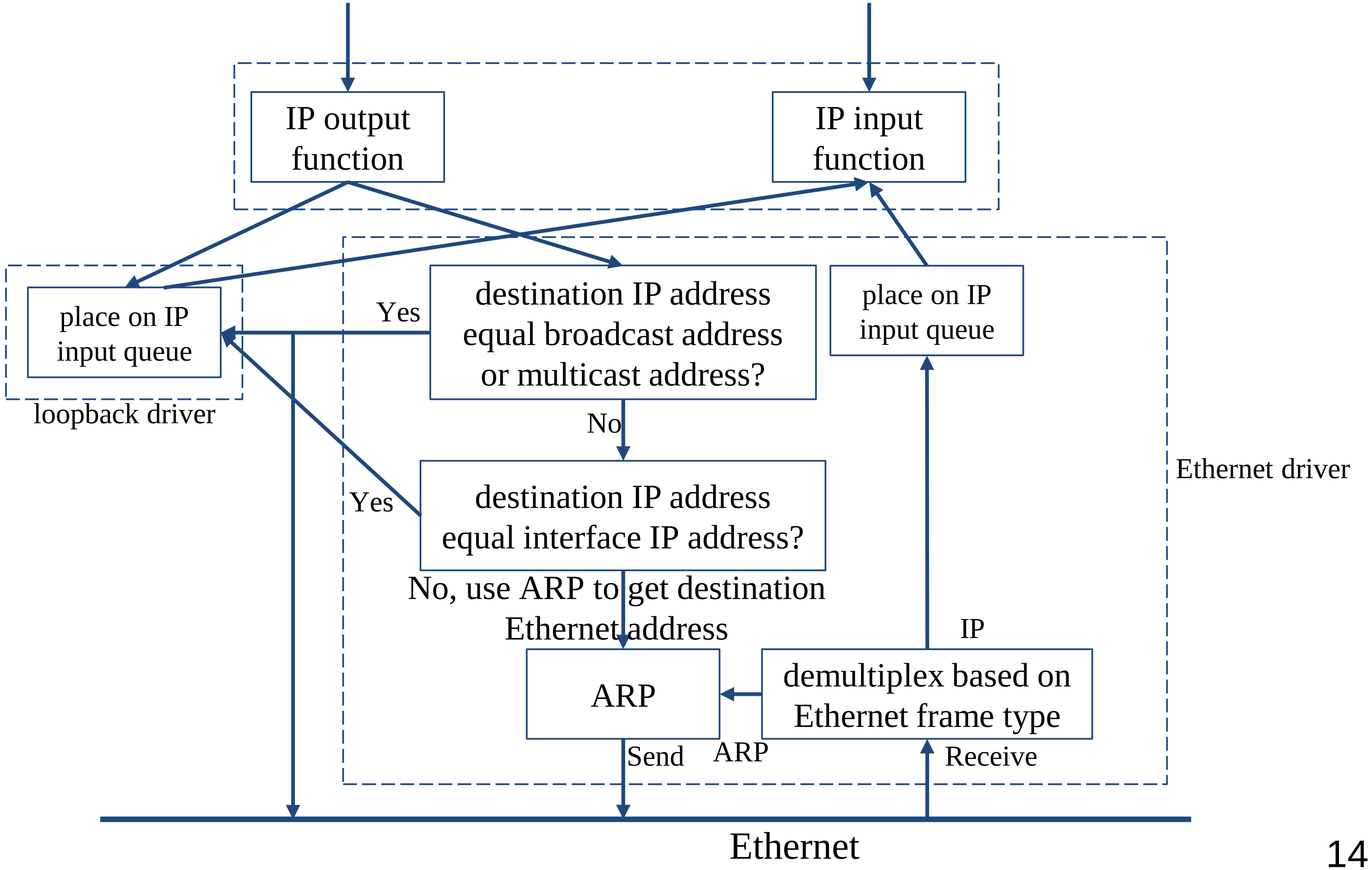
# Link Layer – Ethernet Frame Format

- 48-bit hardware address
  - For both destination and source address
- 16-bit type is used to specify the type of following data
  - 0800 → IP datagram
  - 0806 → ARP, 8035 → RARP



# Link Layer – Loopback Interface

- Pseudo NIC
  - Allow client and server on the same host to communicate with each other using TCP/IP
  - IP
    - 127.0.0.1
  - Hostname
    - localhost



# Link Layer – MTU

- Maximum Transmission Unit
  - Limit size of payload part of Ethernet frame
    - 1500 bytes
  - If the IP datagram is larger than MTU,
    - IP performs “fragmentation”
- MTU of various physical device
- Path MTU
  - Smallest MTU of any data link MTU between the two hosts
  - Depend on route

| Network | MTU (bytes) |
| --- | --- |
| Hyperchannel | 65536 |
| 16 Mbits/sec token ring (IMB) | 17914 |
| 4 Mbits/sec token ring (IEEE 802.5) | 4464 |
| FDDI | 4352 |
| Ethernet | 1500 |
| IEEE 802.3/802.2 | 1492 |
| X.25 | 576 |
| Point-to-point (low delay) | 296 |

# Link Layer – MTU

- To get MTU info

```
$ ifconfig
em0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> mtu 9000
        options=b<RXCSUM,TXCSUM,VLAN_MTU>
        inet 192.168.7.1 netmask 0xffffff00 broadcast 192.168.7.255
        ether 00:0e:0c:01:d7:c8
        media: Ethernet autoselect (1000baseTX <full-duplex>)
        status: active
fxp0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> mtu 1500
        options=b<RXCSUM,TXCSUM,VLAN_MTU>
        inet 140.113.17.24 netmask 0xffffff00 broadcast 140.113.17.255
        ether 00:02:b3:99:3e:71
        media: Ethernet autoselect (100baseTX <full-duplex>)
        status: active
```

# Network Layer

# Network Layer – Introduction to Network Layer

- Unreliable and connectionless datagram delivery service
  - IP Routing
  - IP provides best effort service (unreliable)
  - IP datagram can be delivered out of order (connectionless)
- Protocols using IP
  - TCP, UDP, ICMP, IGMP

# Network Layer – IP Header

- 20 bytes in total length, except options

<table>
<tr><td>0</td><td></td><td>15</td><td>16</td><td>31</td><td></td></tr>
<tr><td>4-bit version</td><td>4-bit header length</td><td>8-bit type of service (TOS)</td><td colspan="2">16-bit total length (in bytes)</td><td rowspan="5">20 bytes</td></tr>
<tr><td colspan="3">16-bit identification</td><td>3-bit flags</td><td>13-bit fragment offset</td></tr>
<tr><td colspan="2">8-bit time to live (TTL)</td><td>8-bit protocol</td><td colspan="2">16-bit header checksum</td></tr>
<tr><td colspan="5">32-bit source IP address</td></tr>
<tr><td colspan="5">32-bit destination IP address</td></tr>
<tr><td colspan="5">options (if any)</td><td></td></tr>
<tr><td colspan="5">data</td><td></td></tr>
</table>

# The Network Layer – IP Address

- 32-bit long
  - Network part
    - Identify a logical network
  - Host part
    - Identify a machine on certain network
- E.g.,
  - NCTU
    - Class B address: 140.113.0.0
    - Network ID: 140.113
    - Number of hosts: 256*256 = 65536
- IP address category

| Class | 1st byte | Format | Comments |
|---|---|---|---|
| A | 1-126 | N.H.H.H | Very early networks, or reserved for DOD |
| B | 128-191 | N.N.H.H | Large sites, usually subnetted, were to get |
| C | 192-223 | N.N.N.H | Easy to get, often obtained in sets |
| D | 224-239 | - | Multicast addresses, not permanently assigned |
| E | 240-254 | - | Experimental addresses |

# Network Layer – Subnetting, CIDR, and Netmask (1)

- Problems of Class A or B network
  - Number of hosts is enormous
  - Hard to maintain and management
  - Solution => Subnetting
- Problems of Class C network
  - 255*255*255 number of Class C network make the size of Internet routes huge
  - Solution => Classless Inter-Domain Routing

# Network Layer – Subnetting, CIDR, and Netmask (2)

- Subnetting
  - Borrow some bits from network ID to extends hosts ID
  - E.g.,
    - Class B address : 140.113.0.0
      = 256 Class C-like IP addresses
      in N.N.N.H subnetting method
    - 140.113.209.0 subnet
- Benefits of subnetting
  - Reduce the routing table size of Internet routers
  - Ex:
    - All external routers have only one entry for 140.113 Class B network

# Network Layer – Subnetting, CIDR, and Netmask (3)

- Netmask
  - Specify how many bits of network-ID are used for network-ID
  - Continuous 1 bits form the network part
  - E.g.,
    - 255.255.255.0 in NCTU-CS example
      - 256 hosts available
    - 255.255.255.248 in ADSL example
      - Only 8 hosts available
  - Shorthand notation
    - Address/prefix-length
      - Ex: 140.113.209.8/24

# Network Layer – Subnetting, CIDR, and Netmask (4)

- How to determine your network ID?
  - Bitwise-AND IP and netmask
  - E.g.,
  - 140.113.214.37 & 255.255.255.0 => 140.113.214.0
  - 140.113.209.37 & 255.255.255.0 => 140.113.209.0

  - 140.113.214.37 & 255.255.0.0 => 140.113.0.0
  - 140.113.209.37 & 255.255.0.0 => 140.113.0.0

  - 211.23.188.78 & 255.255.255.248 => 211.23.188.72
    - 78 = 01001110
    - 78 & 248 = 01001110 & 11111000 = 72

# Network Layer – Subnetting, CIDR, and Netmask (5)

- In a subnet, not all IP are available
- The first one IP → network ID
- The last one IP → broadcast address
- E.g.,

```
Netmask 255.255.255.0
140.113.209.32/24

140.113.209.0   => network ID
140.113.209.255 => broadcast address
1 ~ 254, total 254 IPs are usable
```

```
Netmask 255.255.255.252
211.23.188.78/29

211.23.188.72 => network ID
211.23.188.79 => broadcast address
73 ~ 78, total 6 IPs are usable
```

# Network Layer – Subnetting, CIDR, and Netmask (6)

- The smallest subnetting
  - Network portion : 30 bits
  - Host portion : 2 bits

    => 4 hosts, but only 2 IPs are available
- ipcalc
  - $ pkg install ipcalc
  - /usr/ports/net-mgmt/ipcalc

```
$ ipcalc 140.113.235.100/28

Address:   140.113.235.100      10001100.01110001.11101011.0110 0100
Netmask:   255.255.255.240 = 28 11111111.11111111.11111111.1111 0000
Wildcard:  0.0.0.15             00000000.00000000.00000000.0000 1111
=>
Network:   140.113.235.96/28    10001100.01110001.11101011.0110 0000
HostMin:   140.113.235.97       10001100.01110001.11101011.0110 0001
HostMax:   140.113.235.110      10001100.01110001.11101011.0110 1110
Broadcast: 140.113.235.111      10001100.01110001.11101011.0110 1111
Hosts/Net: 14                    Class B
```

# Network Layer – Subnetting, CIDR, and Netmask (7)

- Network configuration for various lengths of netmask

| Length | Host bits | Hosts/net | Dec. netmask | Hex netmask |
|---|---|---|---|---|
| /20 | 12 | 4094 | 255.255.240.0 | 0xFFFFF000 |
| /21 | 11 | 2046 | 255.255.248.0 | 0xFFFFF800 |
| /22 | 10 | 1022 | 255.255.252.0 | 0xFFFFFC00 |
| /23 | 9 | 510 | 255.255.254.0 | 0xFFFFFE00 |
| /24 | 8 | 254 | 255.255.255.0 | 0xFFFFFF00 |
| /25 | 7 | 126 | 255.255.255.128 | 0xFFFFFF80 |
| /26 | 6 | 62 | 255.255.255.192 | 0xFFFFFFC0 |
| /27 | 5 | 30 | 255.255.255.224 | 0xFFFFFFE0 |
| /28 | 4 | 14 | 255.255.255.240 | 0xFFFFFFF0 |
| /29 | 3 | 6 | 255.255.255.248 | 0xFFFFFFF8 |
| /30 | 2 | 2 | 255.255.255.252 | 0xFFFFFFFC |

# Network Layer – Subnetting, CIDR, and Netmask (8)

- CIDR (Classless Inter-Domain Routing)
  - Use address mask instead of old address classes to determine the destination network
  - CIDR requires modifications to routers and routing protocols
    - Need to transmit both destination address and mask
  - Ex:
    - We can merge two Class C network: 203.19.68.0/24, 203.19.69.0/24 => 203.19.68.0/23
  - Benefit of CIDR
    - We can allocate continuous Class C network to organization
      - Reflect physical network topology
      - Reduce the size of routing table

# Network Layer – IP Routing (1)

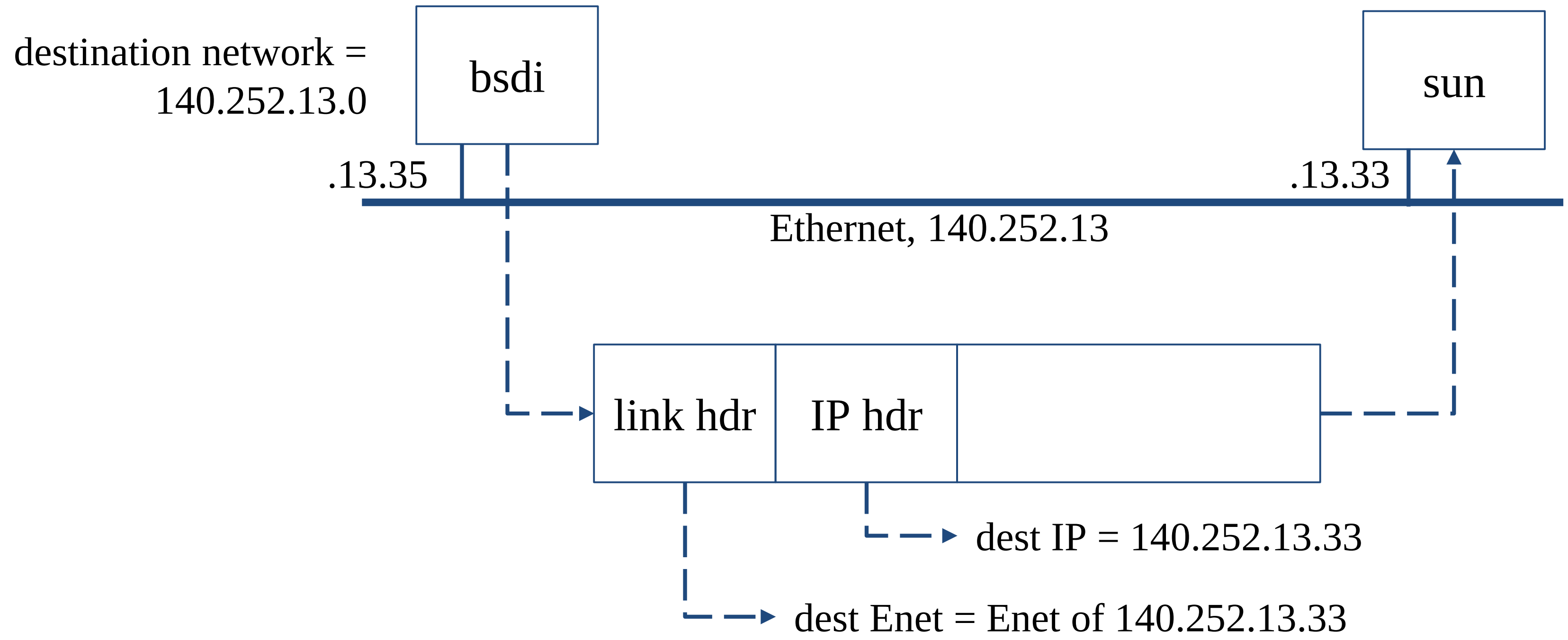
- Difference between Host and Router
  - Router forwards datagram from one of its interface to another, while host does not
  - Almost every Unix system can be configured to act as a router or both
    - net.inet.ip.forwarding=1
- Router
  - IP layer has a routing table, which is used to store the information for forwarding datagram
  - When router receiving a datagram
    - If Dst. IP = my IP, demultiplex to other protocol
    - Other, forward the IP based on routing table

# Network Layer – IP Routing (2)

- Routing table information
  - Destination IP
  - IP address of next-hop router or IP address of a directly connected network
  - Flags
  - Next interface
- IP routing
  - Done on a hop-by-hop basis
  - It assumes that the next-hop router is closer to the destination
  - Steps:
    - Search routing table for complete matched IP address
      - Send to next-hop router or to the directly connected NIC
    - Search routing table for matched network ID
      - Send to next-hop router or to the directly connected NIC
    - Search routing table for default route
      - Send to this default next-hop router
    - host or network unreachable

# Network Layer – IP Routing (3)

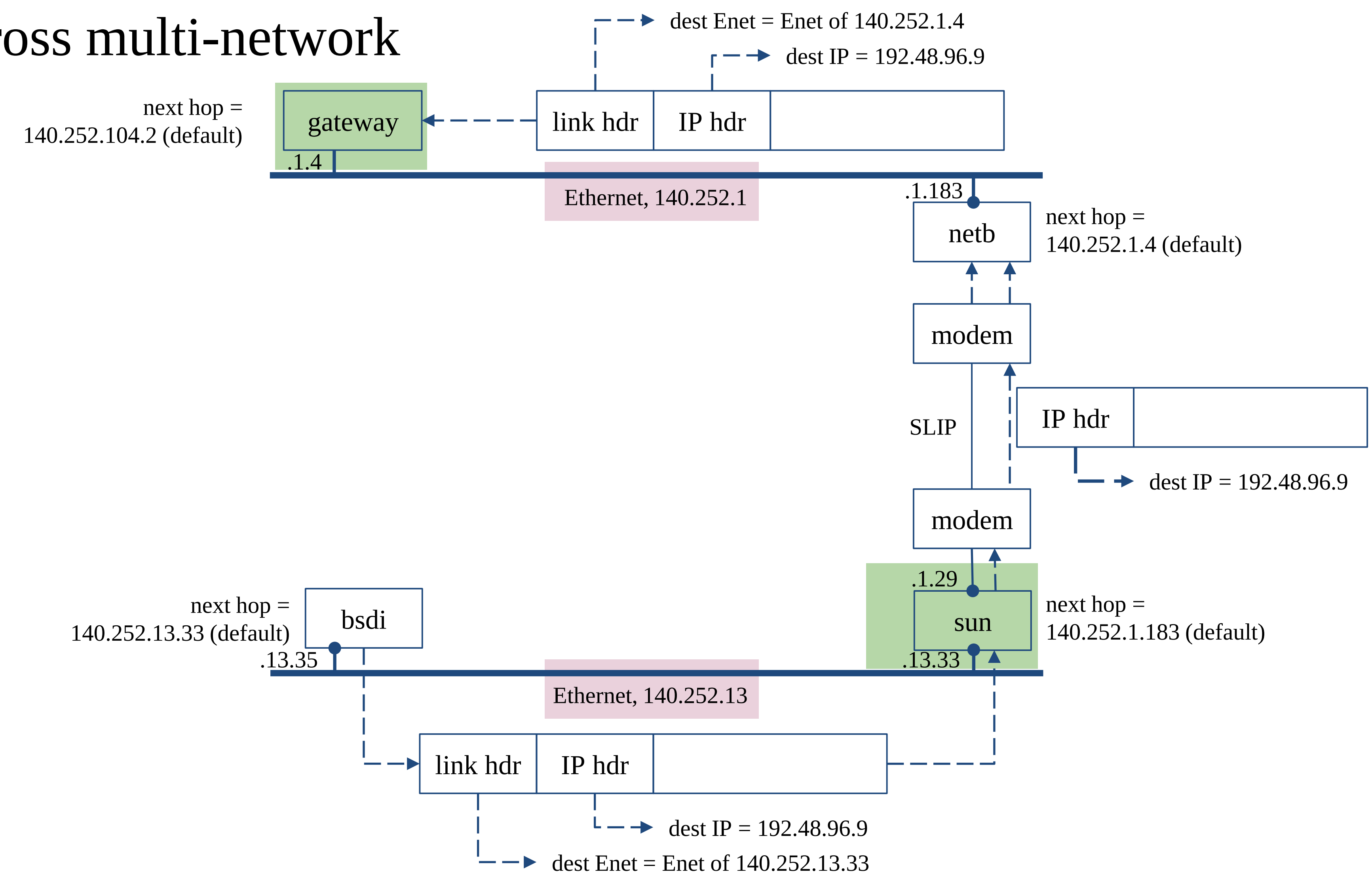
- Ex1: routing in the same network
  - bsdi: 140.252.13.35
  - sun: 140.252.13.33




| | | | |
|---|---|---|---|
| Ex Routing table: | | | |
| 140.252.13.33 | 00:d0:59:83:d9:16 | UHLW | fxp1 |

# Network Layer – IP Routing (4)

- Ex2:
    - routing across multi-network

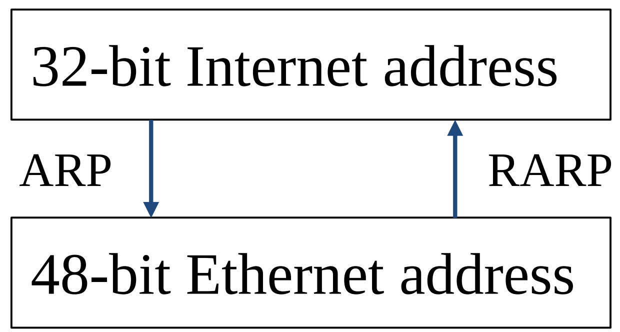


# ARP and RARP

Something between MAC (link layer) And IP (network layer)

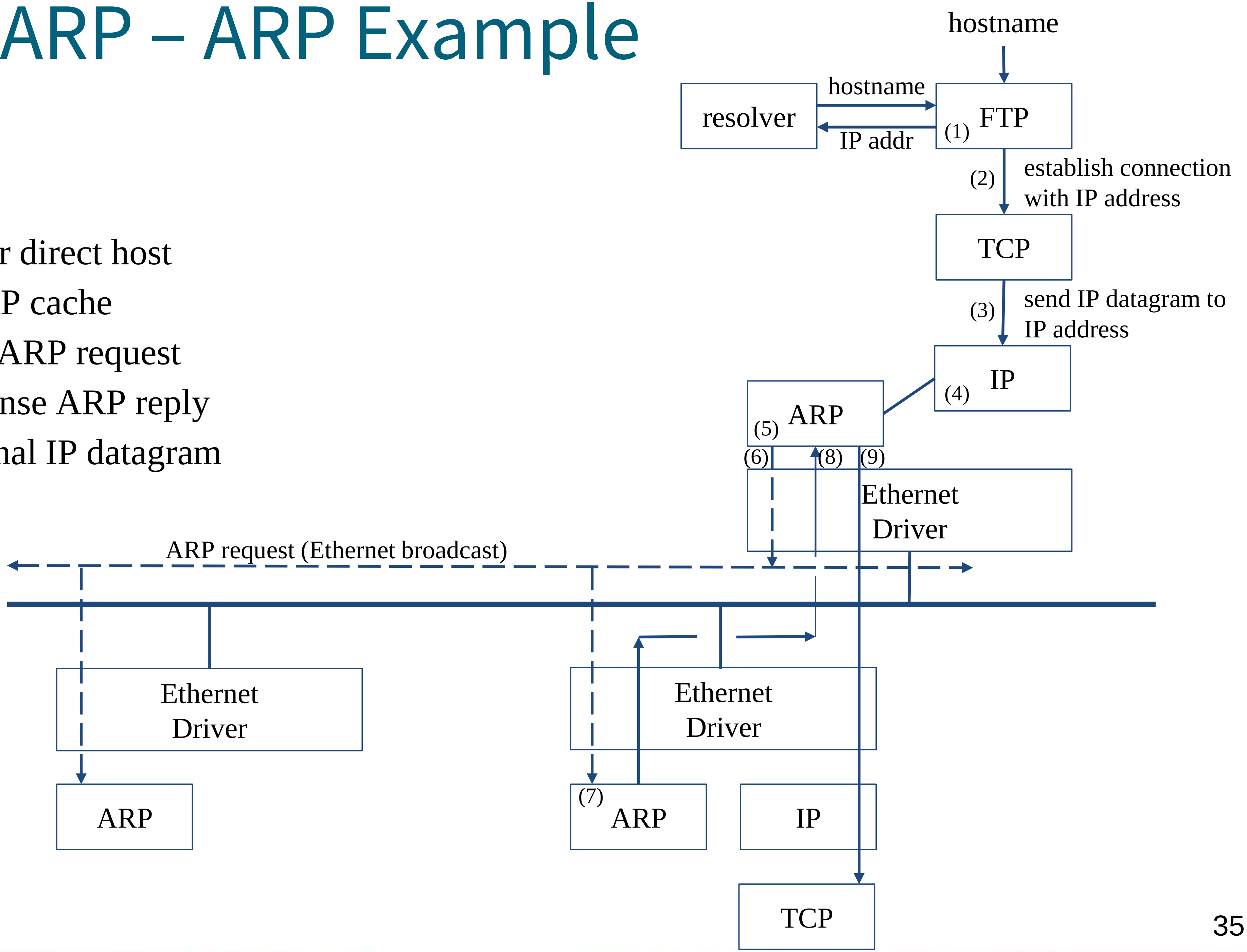
# ARP and RARP

- ARP – Address Resolution Protocol and
  RARP – Reverse ARP
  - Mapping between IP and Ethernet address
- When an Ethernet frame is sent on LAN from one host to another,
  - It is the 48-bit Ethernet address that determines for which interface the frame is destined



# ARP and RARP – ARP Example

- Example
  - % ftp bsd1
  - (4) next-hop or direct host
  - (5) Search ARP cache
  - (6) Broadcast ARP request
  - (7) bsd1 response ARP reply
  - (9) Send original IP datagram



# ARP and RARP – ARP Cache

- Maintain recent ARP results
  - Come from both ARP request and reply
  - Expiration time
    - Complete entry = 20 minutes
    - Incomplete entry = 3 minutes
  - Use arp command to see the cache
  - E.g.:
    - $ arp -a
    - $ arp -da
    - $ arp -S 140.113.235.132 00:0e:a6:94:24:6e

```
$ arp -a
crypto23.csie.nctu.edu.tw  (140.113.208.143) at 00:16:e6:5b:fa:e9 on fxp1 [ethernet]
e3rtn-208.csie.nctu.edu.tw (140.113.208.254) at 00:0e:38:a4:c2:00 on fxp1 [ethernet]
e3rtn-210.csie.nctu.edu.tw (140.113.210.254) at 00:0e:38:a4:c2:00 on fxp2 [ethernet]
```

# ARP and RARP – ARP/RARP Packet Format

- Ethernet destination addr: all 1's (broadcast)
- Known value for IP <-> Ethernet
    - Frame type: 0x0806 for ARP, 0x8035 for RARP
    - Hardware type: type of hardware address (1 for Ethernet)
    - Protocol type: type of upper layer address (0x0800 for IP)
    - Hard size: size in bytes of hardware address (6 for Ethernet)
    - Protocol size: size in bytes of upper layer address (4 for IP)
    - Op: 1, 2, 3, 4 for ARP request, reply, RARP request, reply

hard size
prot size

| Ethernet destination addr | Ethernet source addr | frame type | hard type | prot type | | | op | sender Ethernet addr | sender IP addr | target Ethernet addr | target IP addr |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 6 | 6 | 2 | 2 | 2 | 1 | 1 | 2 | 6 | 4 | 6 | 4 |

Ethernet header — 28 byte ARP request/reply

# ARP and RARP – Use tcpdump to see ARP

- Host 140.113.17.212 => 140.113.17.215
  - Clear ARP cache of 140.113.17.212
    - $ sudo arp -d 140.113.17.215
  - Run tcpdump on 140.113.17.215 (00:11:d8:06:1e:81)
    - $ sudo tcpdump -i sk0 -e arp
    - $ sudo tcpdump -i sk0 -n -e arp
    - $ sudo tcpdump -i sk0 -n -t -e arp
  - On 140.113.17.212, ssh to 140.113.17.215

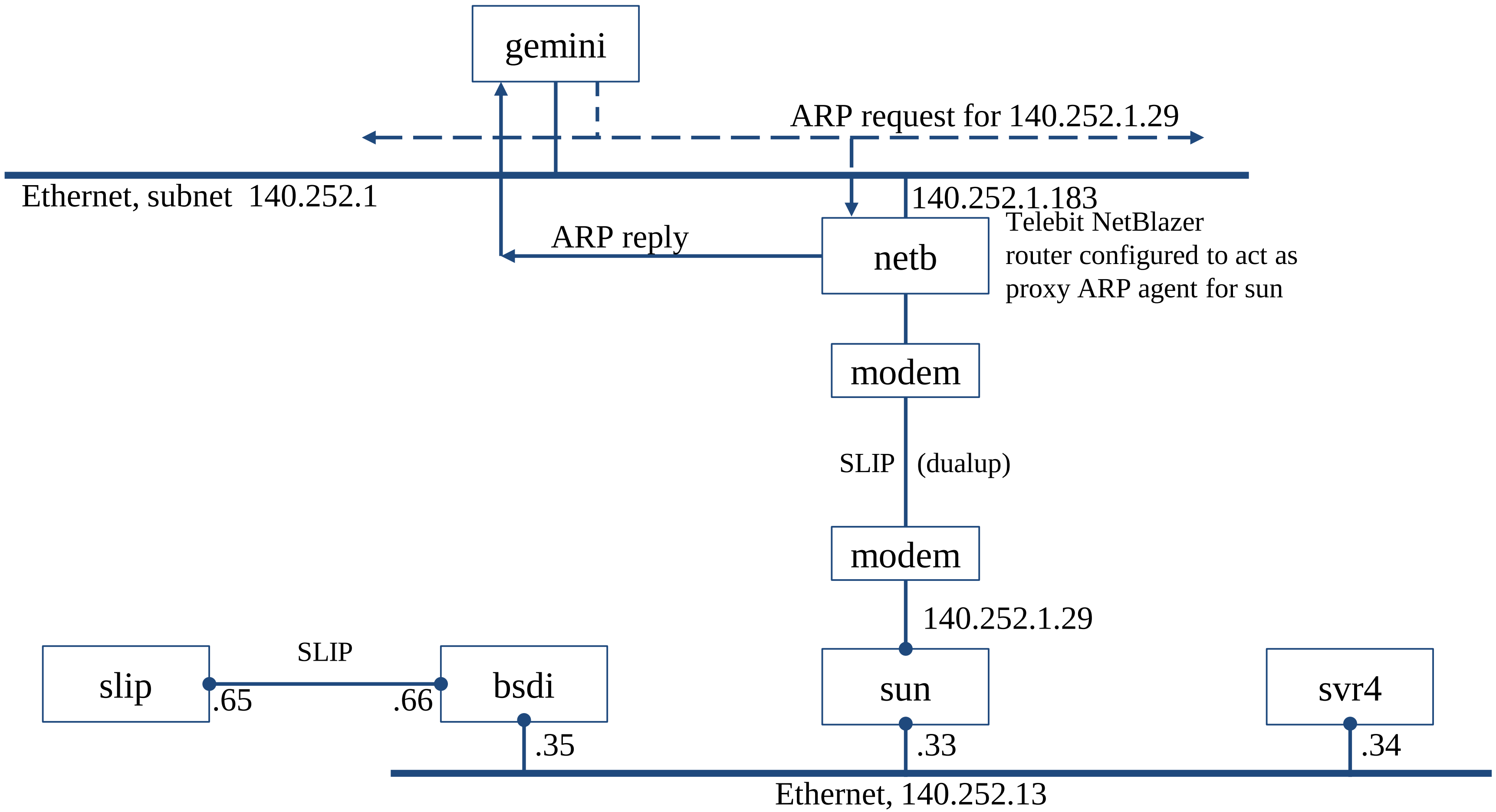
```
15:18:54.899779 00:90:96:23:8f:7d > Broadcast, ethertype ARP (0x0806), length 60:
    arp who-has nabsd tell chbsd.csie.nctu.edu.tw
15:18:54.899792 00:11:d8:06:1e:81 > 00:90:96:23:8f:7d, ethertype ARP (0x0806), length 42:
    arp reply nabsd is-at 00:11:d8:06:1e:81
```

```
15:26:13.847417 00:90:96:23:8f:7d > ff:ff:ff:ff:ff:ff, ethertype ARP (0x0806), length 60:
    arp who-has 140.113.17.215 tell 140.113.17.212
15:26:13.847434 00:11:d8:06:1e:81 > 00:90:96:23:8f:7d, ethertype ARP (0x0806), length 42:
    arp reply 140.113.17.215 is-at 00:11:d8:06:1e:81
```

```
00:90:96:23:8f:7d > ff:ff:ff:ff:ff:ff, ethertype ARP (0x0806), length 60:
    arp who-has 140.113.17.215 tell 140.113.17.212
00:11:d8:06:1e:81 > 00:90:96:23:8f:7d, ethertype ARP (0x0806), length 42:
    arp reply 140.113.17.215 is-at 00:11:d8:06:1e:81
```

# ARP and RARP – Proxy ARP

- Let router answer ARP request on one of its networks for a host on another of its network



# ARP and RARP – Gratuitous ARP

- Gratuitous ARP
  - The host sends an ARP request looking for its own IP
  - Provide two features
    - Used to determine whether there is another host configured with the same IP
    - Used to cause any other host to update ARP cache when changing hardware address

# ARP and RARP – RARP

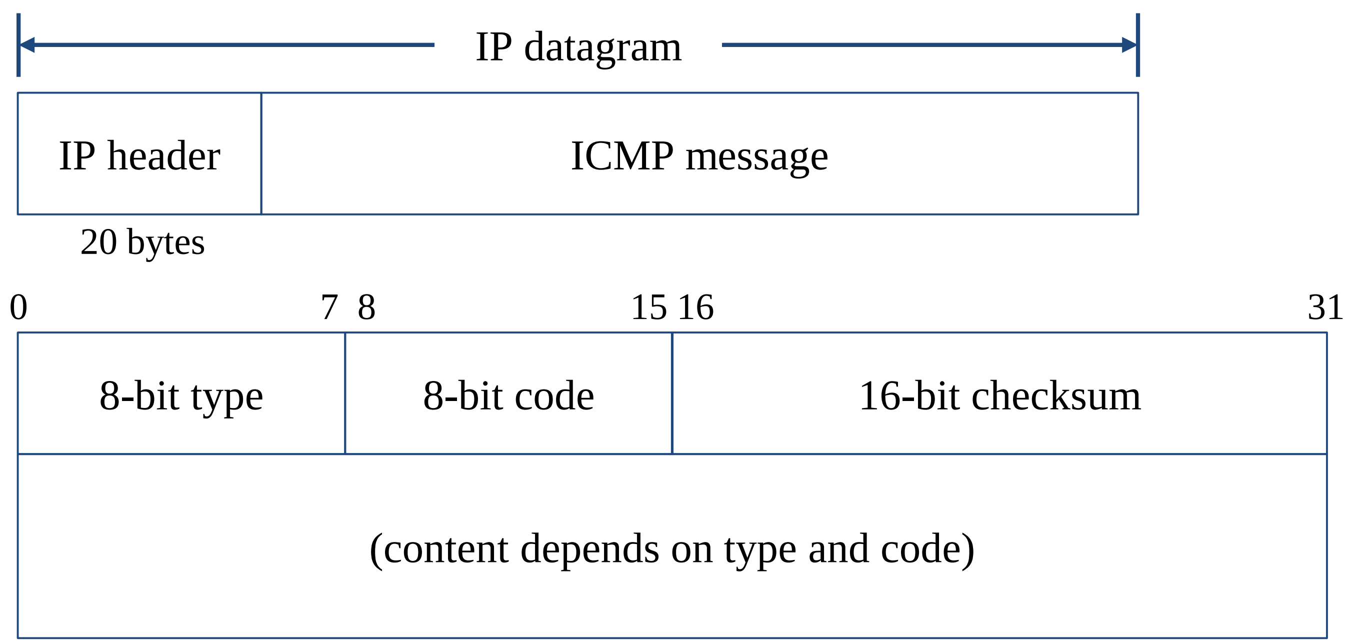
- Principle
    - Used for the diskless system to read its hardware address from the NIC and send an RARP request to gain its IP
- RARP Server Design
    - RARP server must maintain the map from hardware address to an IP address for many host
    - Link-layer broadcast
        - This prevent most routers from forwarding an RARP request

# ICMP

Internet Control Message Protocol

# ICMP – Introduction

- Part of the IP layer
  - ICMP messages are transmitted within IP datagram
  - ICMP communicates error messages and other conditions that require attention for other protocols
- ICMP message format



# ICMP – Message Type (1)

| type | code | Description | Query | Error |
|---|---|---|---|---|
| 0 | 0 | echo reply (Ping reply) | • | |
| 3 | | destination unreachable: | | |
| | 0 | > network unreachable | | • |
| | 1 | > host unreachable | | • |
| | 2 | > protocol unreachable | | • |
| | 3 | > port unreachable | | • |
| | 4 | > fragmentation needed but don't fragment bit set | | • |
| | 5 | > source route failed | | • |
| | 6 | > destination network unknown | | • |
| | 7 | > destination host unknown | | • |
| | 8 | > source host isolated (obsolete) | | • |
| | 9 | > destination network administratively prohibited | | • |
| | 10 | > destination host administratively prohibited | | • |

| type | code | Description | Query | Error |
|---|---|---|---|---|
| | 11 | > network unreachable for TOS | | • |
| | 12 | > host unreachable for TOS | | • |
| | 13 | > communication administratively prohibited by filtering | | • |
| | 14 | > host precedence violation | | • |
| | 15 | > precedence cutoff effect | | • |

# ICMP – Message Type (2)

| type | code | Description | Query | Error |
|---|---|---|---|---|
| 4 | 0 | source quench (elementary flow control) | | • |
| 5 | | redirect: | | |
| | 0 | > redirect for network | | • |
| | 1 | > redirect for host | | • |
| | 2 | > redirect for type-of-service and network | | • |
| | 3 | > redirect for type-of-service and host | | • |
| 8 | 0 | echo request (Ping request) | • | |
| 9 | 0 | router advertisement | • | |
| 10 | 0 | router solicitation | • | |
| 11 | | time exceeded: | | |
| | 0 | > time-to-live equals 0 during transit (Traceroute) | | • |
| | 1 | > time-to-live equals 0 during reassembly | | • |

| type | code | Description | Query | Error |
|---|---|---|---|---|
| 12 | | parameter problem: | | |
| | 0 | > IP header bad (catchall error) | | • |
| | 1 | > required option missing | | • |
| 13 | 0 | timestamp request | • | |
| 14 | 0 | timestamp reply | • | |
| 15 | 0 | information request (obsolete) | • | |
| 16 | 0 | information reply (obsolete) | • | |
| 17 | 0 | address mask request | • | |
| 18 | 0 | address mask reply | | |

# ICMP – Query Message – Address Mask Request/Reply (1)

- Address Mask Request and Reply
  - Used for diskless system to obtain its subnet mask
  - Identifier and sequence number
    - Can be set to anything for sender to match reply with request
  - The receiver will response an ICMP reply with the subnet mask of the receiving NIC

| 0 ... 7 | 8 ... 15 | 16 ... 31 |
| --- | --- | --- |
| type (17 or 18) | code (0) | checksum |
| identifier | | sequence number |
| address mask | | |

# ICMP – Query Message
# – Address Mask Request/Reply (2)

- Example:

```
$ ping -M m sun1.cs.nctu.edu.tw
ICMP_MASKREQ
PING sun1.cs.nctu.edu.tw (140.113.235.171): 56 data bytes
68 bytes from 140.113.235.171: icmp_seq=0 ttl=251 time=0.663 ms mask=255.255.255.0
68 bytes from 140.113.235.171: icmp_seq=1 ttl=251 time=1.018 ms mask=255.255.255.0
68 bytes from 140.113.235.171: icmp_seq=2 ttl=251 time=1.028 ms mask=255.255.255.0
68 bytes from 140.113.235.171: icmp_seq=3 ttl=251 time=1.026 ms mask=255.255.255.0
^C
--- sun1.cs.nctu.edu.tw ping statistics ---
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max/stddev = 0.663/0.934/1.028/0.156 ms

$ icmpquery -m sun1
sun1                                    :   0xFFFFFF00
```

※ icmpquery can be found in /usr/ports/net-mgmt/icmpquery

# ICMP – Query Message – Timestamp Request/Reply (1)

- Timestamp request and reply
  - Allow a system to query another for the current time
  - Milliseconds resolution, since midnight UTC
  - Requestor
    - Fill in the originate timestamp and send
  - Reply system
    - Fill in the receive timestamp when it receives the request and the transmit time when it sends the reply

| 0 – 7 | 8 – 15 | 16 – 31 |
|---|---|---|
| type (13 or 14) | code (0) | checksum |
| identifier | | sequence number |
| originate timestamp | | |
| receive timestamp | | |
| transmit timestamp | | |

# ICMP – Query Message – Timestamp Request/Reply (2)

- Example

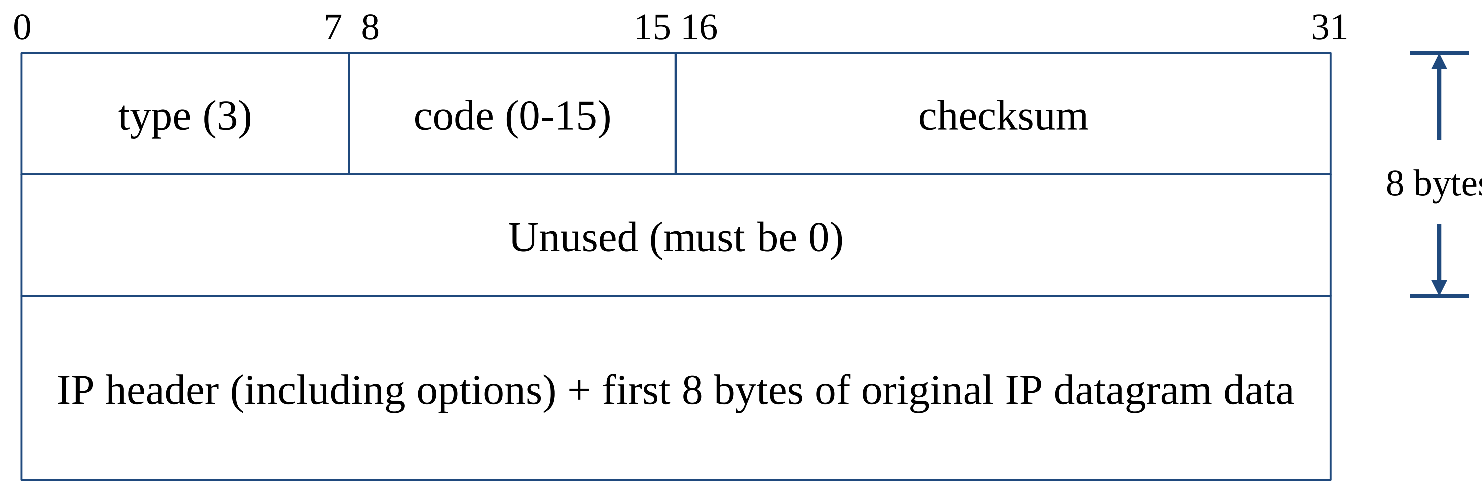
```
$ ping -M time nabsd
ICMP_TSTAMP
PING nabsd.cs.nctu.edu.tw (140.113.17.215): 56 data bytes
76 bytes from 140.113.17.215: icmp_seq=0 ttl=64 time=0.663 ms
    tso=06:47:46 tsr=06:48:24 tst=06:48:24
76 bytes from 140.113.17.215: icmp_seq=1 ttl=64 time=1.016 ms
    tso=06:47:47 tsr=06:48:25 tst=06:48:25

$ icmpquery -t nabsd
nabsd                                    :  14:54:47
```

```
$ sudo tcpdump -i sk0 -e icmp
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on sk0, link-type EN10MB (Ethernet), capture size 96 bytes
14:48:24.999106 00:90:96:23:8f:7d > 00:11:d8:06:1e:81, ethertype IPv4 (0x0800), length 110:
    chbsd.csie.nctu.edu.tw > nabsd: ICMP time stamp query id 18514 seq 0, length 76
14:48:24.999148 00:11:d8:06:1e:81 > 00:90:96:23:8f:7d, ethertype IPv4 (0x0800), length 110:
    nabsd > chbsd.csie.nctu.edu.tw: ICMP time stamp reply id 18514 seq 0: org 06:47:46.326,
    recv 06:48:24.998, xmit 06:48:24.998, length 76
14:48:26.000598 00:90:96:23:8f:7d > 00:11:d8:06:1e:81, ethertype IPv4 (0x0800), length 110:
    chbsd.csie.nctu.edu.tw > nabsd: ICMP time stamp query id 18514 seq 1, length 76
14:48:26.000618 00:11:d8:06:1e:81 > 00:90:96:23:8f:7d, ethertype IPv4 (0x0800), length 110:
    nabsd > chbsd.csie.nctu.edu.tw: ICMP time stamp reply id 18514 seq 1: org 06:47:47.327,
    recv 06:48:25.999, xmit 06:48:25.999, length 76
```

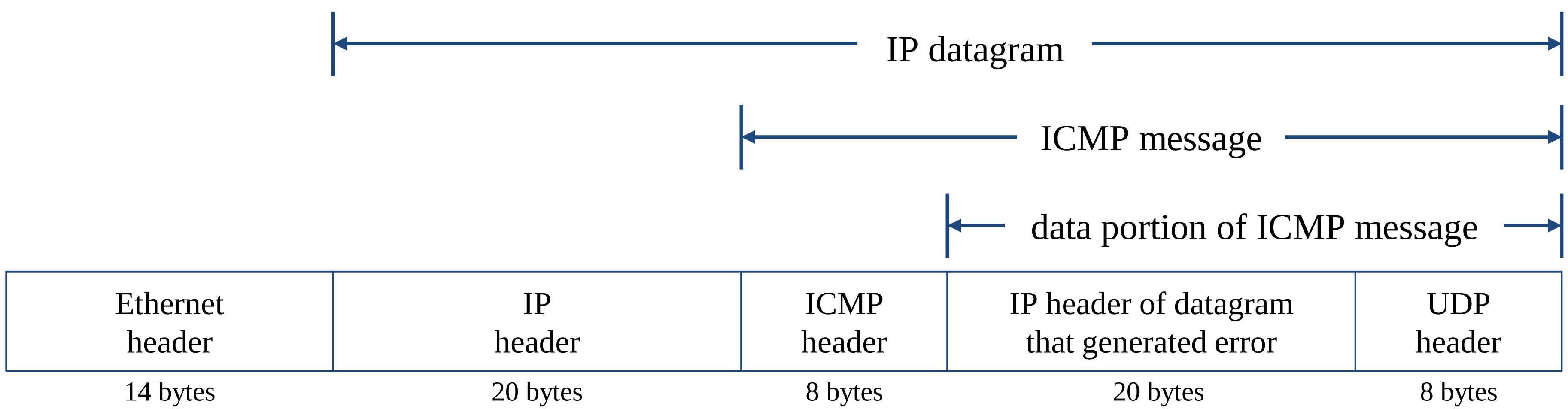
# ICMP – Error Message
# – Destination Unreachable Error Message

- Format
  - 8 bytes ICMP Header
  - Application-depend data portion
    - IP header
      - Let ICMP know how to interpret the 8 bytes that follow
    - first 8 bytes that followed this IP header
      - Information about who generates the error



# ICMP – Error Message – Port Unreachable (1)

- ICMP port unreachable
  - Type = 3 , code = 3
  - Host receives a UDP datagram but the destination port does not correspond to a port that some process has in use



# ICMP – Error Message – Port Unreachable (2)

- Example:
  - Using TFTP (Trivial File Transfer Protocol)
    - Original port: 69

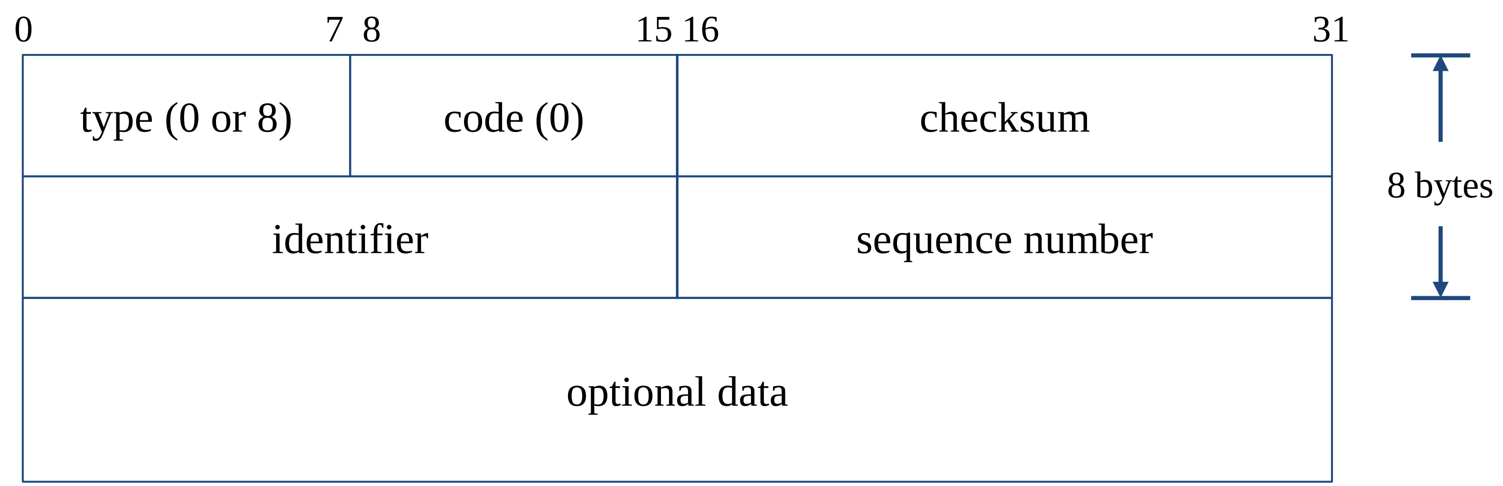
```
$ tftp
tftp> connect localhost 8888
tftp> get temp.foo
Transfer timed out.

tftp>
```

```
$ sudo tcpdump -i lo0
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on lo0, link-type NULL (BSD loopback), capture size 96 bytes
15:01:24.788511 IP localhost.62089 > localhost.8888: UDP, length 16
15:01:24.788554 IP localhost > localhost:
    ICMP localhost udp port 8888 unreachable, length 36
15:01:29.788626 IP localhost.62089 > localhost.8888: UDP, length 16
15:01:29.788691 IP localhost > localhost:
    ICMP localhost udp port 8888 unreachable, length 36
```

# ICMP – Ping Program (1)

- Use ICMP to test whether another host is reachable
  - Type 8, ICMP echo request
  - Type 0, ICMP echo reply
- ICMP echo request/reply format
  - Identifier: process ID of the sending process
  - Sequence number: start with 0
  - Optional data: any optional data sent must be echoed



# ICMP – Ping Program (2)

- Ex:
  - ServerA ping ServerB
  - execute “tcpdump -i sk0 -X -e icmp” on ServerB

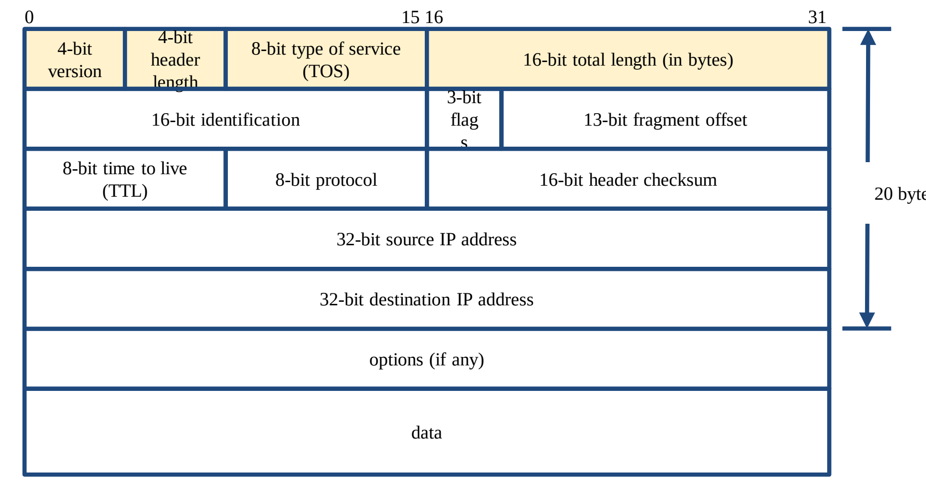
```
ServerA $ ping ServerB
PING ServerB.cs.nctu.edu.tw (140.113.17.215): 56 data bytes
64 bytes from 140.113.17.215: icmp_seq=0 ttl=64 time=0.520 ms
```

```
15:08:12.631925 00:90:96:23:8f:7d > 00:11:d8:06:1e:81, ethertype IPv4 (0x0800), length 98:
    ServerA.cs.nctu.edu.tw > ServerB: ICMP echo request, id 56914, seq 0, length 64
        0x0000:  4500 0054 f688 0000 4001 4793 8c71 11d4  E..T....@.G..q..
        0x0010:  8c71 11d7 0800 a715 de52 0000 45f7 9f35  .q.......R..E..5
        0x0020:  000d a25a 0809 0a0b 0c0d 0e0f 1011 1213  ...Z............
        0x0030:  1415 1617 1819 1a1b 1c1d 1e1f 2021 2223  ..............!"#
        0x0040:  2425 2627 2829 2a2b 2c2d 2e2f 3031 3233  $%&'()*+,-./0123
        0x0050:  3435                                     45
15:08:12.631968 00:11:d8:06:1e:81 > 00:90:96:23:8f:7d, ethertype IPv4 (0x0800), length 98:
     ServerB > ServerA.cs.nctu.edu.tw: ICMP echo reply, id 56914, seq 0, length 64
        0x0000:  4500 0054 d97d 0000 4001 649e 8c71 11d7  E..T.}..@.d..q..
        0x0010:  8c71 11d4 0000 af15 de52 0000 45f7 9f35  .q.......R..E..5
        0x0020:  000d a25a 0809 0a0b 0c0d 0e0f 1011 1213  ...Z............
        0x0030:  1415 1617 1819 1a1b 1c1d 1e1f 2021 2223  ..............!"#
        0x0040:  2425 2627 2829 2a2b 2c2d 2e2f 3031 3233  $%&'()*+,-./0123
        0x0050:  3435                                     45
```

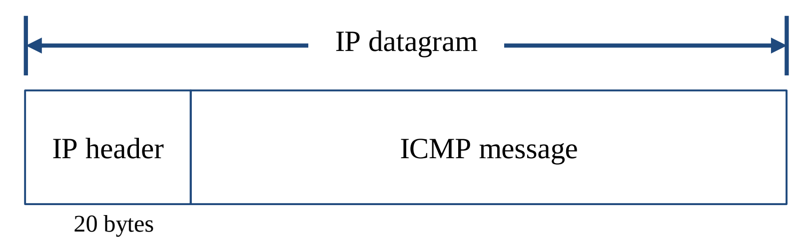
Type/Code

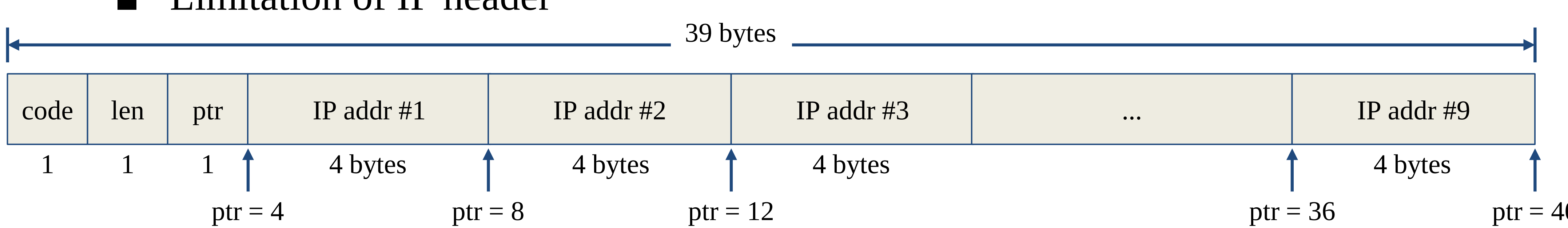
ID

# ICMP – Ping Program (3)



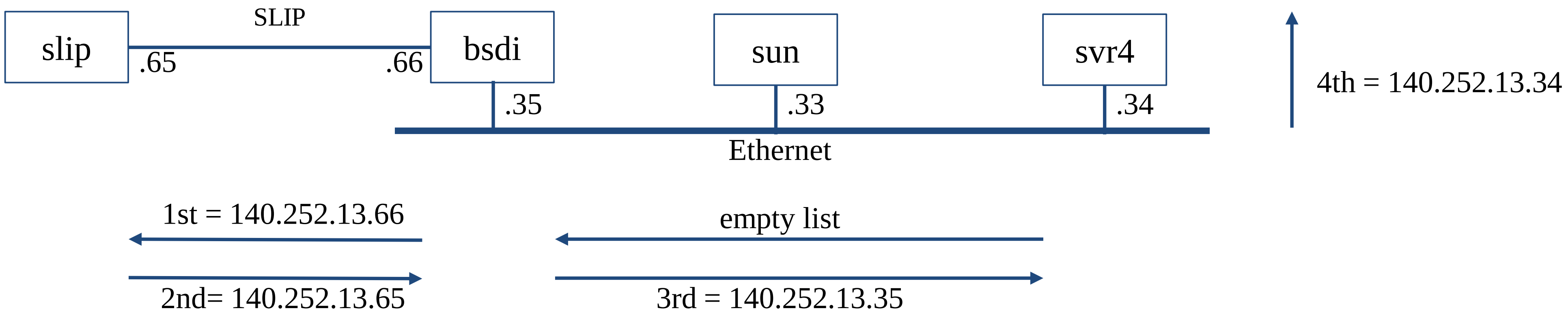

- To get the route that packets take to host
  - Taking use of “IP Record Route Option”
  - Command: ping -R
  - Cause every router that handles the datagram to add its (outgoing) IP address to a list in the options field.
  - Format of Option field for IP RR Option
    - code: type of IP Option (7 for RR)
    - len: total number of bytes of the RR option
    - ptr:4 ~ 40 used to point to the next IP address
  - Only 9 IP addresses can be stored
    - Limitation of IP header



# ICMP – Ping Program (4)

- Example:




```
srv4 $ ping -R slip
PING slip (140.252.13.65): 56 data bytrs
64 bytes from 140.252.13.65: icmp_seq=0 ttl=254 time=280 ms
RR      bsdi (140.252.13.66)
        bsdi (140.252.13.65)
        bsdi (140.252.13.35)
        bsdi (140.252.13.34)
64 bytes from 140.252.13.65: icmp_seq=1 ttl=254 time=280 ms (same route)
64 bytes from 140.252.13.65: icmp_seq=2 ttl=254 time=270 ms (same route)
^?
--- slip ping statistics
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max = 270/276/280 ms
```

# ICMP – Ping Program (5)

- Example:

```
$ ping -R www.nctu.edu.tw
PING www.nctu.edu.tw (140.113.250.5): 56 data bytes
64 bytes from 140.113.250.5: icmp_seq=0 ttl=61 time=2.361 ms
RR:     ProjE27-253.NCTU.edu.tw (140.113.27.253)
        140.113.0.57
        CC250-gw.NCTU.edu.tw (140.113.250.253)
        www.NCTU.edu.tw (140.113.250.5)
        www.NCTU.edu.tw (140.113.250.5)
        140.113.0.58
        ProjE27-254.NCTU.edu.tw (140.113.27.254)
        e3rtn.csie.nctu.edu.tw (140.113.17.254)
        chbsd.csie.nctu.edu.tw (140.113.17.212)
64 bytes from 140.113.250.5: icmp_seq=1 ttl=61 time=3.018 ms    (same route)
```

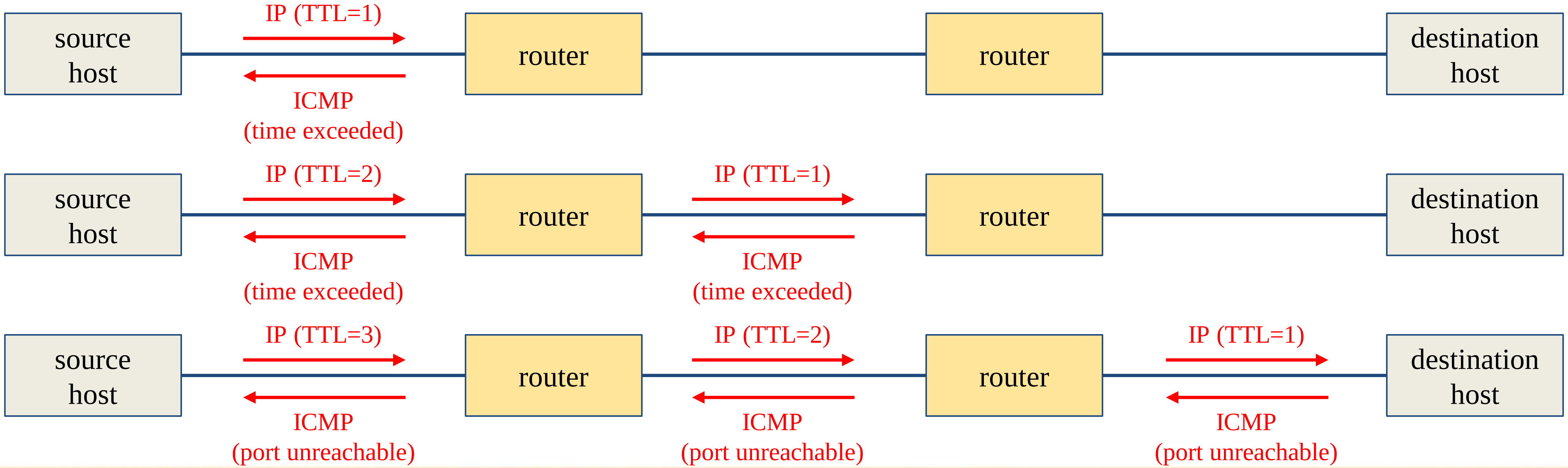
```
$ sudo tcpdump -v -n -i dc0 -e icmp
tcpdump: listening on dc0, link-type EN10MB (Ethernet), capture size 96 bytes
22:57:04.507271 00:90:96:23:8f:7d > 00:90:69:64:ec:00, ethertype IPv4 (0x0800), length 138:
   (tos 0x0, ttl  64, id 17878, offset 0, flags [none], proto: ICMP (1), length: 124,
   options ( RR (7) len 390.0.0.00.0.0.00.0.0.00.0.0.00.0.0.00.0.0.00.0.0.00.0.0.00.0.0.0EOL
   (0) len 1 )) 140.113.17.212 > 140.113.250.5: ICMP echo request, id 45561, seq 0, length 64
22:57:04.509521 00:90:69:64:ec:00 > 00:90:96:23:8f:7d, ethertype IPv4 (0x0800), length 138:
   (tos 0x0, ttl  61, id 33700, offset 0, flags [none], proto: ICMP (1), length: 124,
   options ( RR (7) len 39140.113.27.253, 140.113.0.57, 140.113.250.253, 140.113.250.5,
   140.113.250.5, 140.113.0.58, 140.113.27.254, 140.113.17.254, 0.0.0.0EOL (0) len 1 ))
   140.113.250.5 > 140.113.17.212: ICMP echo reply, id 45561, seq 0, length 64
```

# Traceroute Program (1)

- To print the route packets take to network host
- Drawbacks of IP RR options (ping -R)
  - Not all routers have supported the IP RR option
  - Limitation of IP header length
- Background knowledge of traceroute
  - When a router receive a datagram, it will decrement the TTL by one
  - When a router receive a datagram with TTL = 0  or 1,
    - it will through away the datagram and
    - sends back a “Time exceeded” ICMP message
  - Unused UDP port will generate a “port unreachable” ICMP message

# Traceroute Program (2)

- Operation of traceroute
  - Send UDP with port > 30000, encapsulated with IP header with TTL = 1, 2, 3, … continuously
  - When router receives the datagram and TTL = 1, it returns a “Time exceeded” ICMP message
  - When destination host receives the datagram and TTL = 1, it returns a “Port unreachable” ICMP message



# Traceroute Program (3)

- Time exceed ICMP message
    - Type = 11, code = 0 or 1
        - Code = 0 means TTL=0 during transit
        - Code = 1 means TTL=0 during reassembly
    - First 8 bytes of datagram
        - UDP header

| 0 | 8 | 16 | 31 |
|---|---|---|---|
| **TYPE (11)** | **CODE (0 or 1)** | **CHECKSUM** | |
| **UNUSED (MUST BE ZERO)** | | | |
| **INTERNET HEADER + FIRST 64 BITS OF DATAGRAM** | | | |
| **...** | | | |

# Traceroute Program (4)

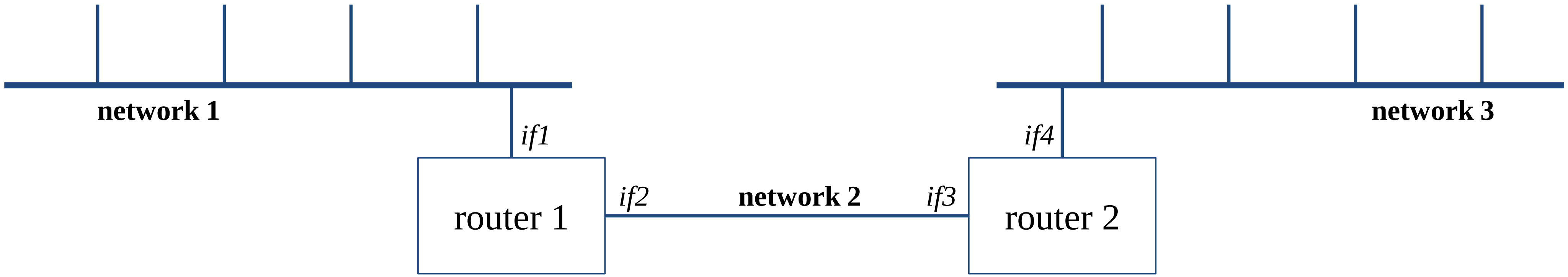
- Example

```
$ traceroute bsd1.cs.nctu.edu.tw
traceroute to bsd1.cs.nctu.edu.tw (140.113.235.131), 64 hops max, 40 byte packets
 1  e3rtn.csie.nctu.edu.tw (140.113.17.254)  0.377 ms  0.365 ms  0.293 ms
 2  ProjE27-254.NCTU.edu.tw (140.113.27.254)  0.390 ms  0.284 ms  0.391 ms
 3  140.113.0.58 (140.113.0.58)  0.292 ms  0.282 ms  0.293 ms
 4  140.113.0.165 (140.113.0.165)  0.492 ms  0.385 ms  0.294 ms
 5  bsd1.cs.nctu.edu.tw (140.113.235.131)  0.393 ms  0.281 ms  0.393 ms
```

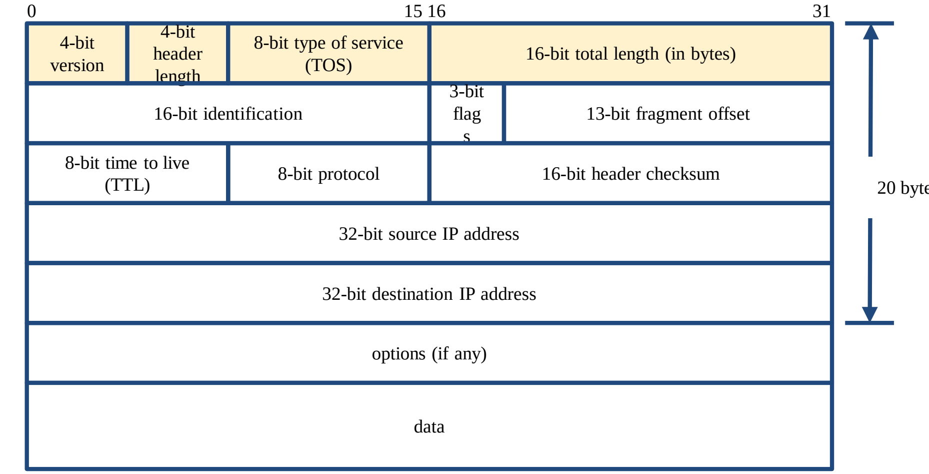
```
$ sudo tcpdump -i sk0 -t icmp
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on sk0, link-type EN10MB (Ethernet), capture size 96 bytes
IP e3rtn.csie.nctu.edu.tw > nabsd: ICMP time exceeded in-transit, length 36
IP e3rtn.csie.nctu.edu.tw > nabsd: ICMP time exceeded in-transit, length 36
IP e3rtn.csie.nctu.edu.tw > nabsd: ICMP time exceeded in-transit, length 36
IP ProjE27-254.NCTU.edu.tw > nabsd: ICMP time exceeded in-transit, length 36
IP ProjE27-254.NCTU.edu.tw > nabsd: ICMP time exceeded in-transit, length 36
IP ProjE27-254.NCTU.edu.tw > nabsd: ICMP time exceeded in-transit, length 36
IP 140.113.0.58 > nabsd: ICMP time exceeded in-transit, length 36
IP 140.113.0.58 > nabsd: ICMP time exceeded in-transit, length 36
IP 140.113.0.58 > nabsd: ICMP time exceeded in-transit, length 36
IP 140.113.0.165 > nabsd: ICMP time exceeded in-transit, length 36
IP 140.113.0.165 > nabsd: ICMP time exceeded in-transit, length 36
IP 140.113.0.165 > nabsd: ICMP time exceeded in-transit, length 36
IP bsd1.cs.nctu.edu.tw > nabsd: ICMP bsd1.cs.nctu.edu.tw udp port 33447 unreachable, length 36
IP bsd1.cs.nctu.edu.tw > nabsd: ICMP bsd1.cs.nctu.edu.tw udp port 33448 unreachable, length 36
IP bsd1.cs.nctu.edu.tw > nabsd: ICMP bsd1.cs.nctu.edu.tw udp port 33449 unreachable, length 36
```

# Traceroute Program (5)

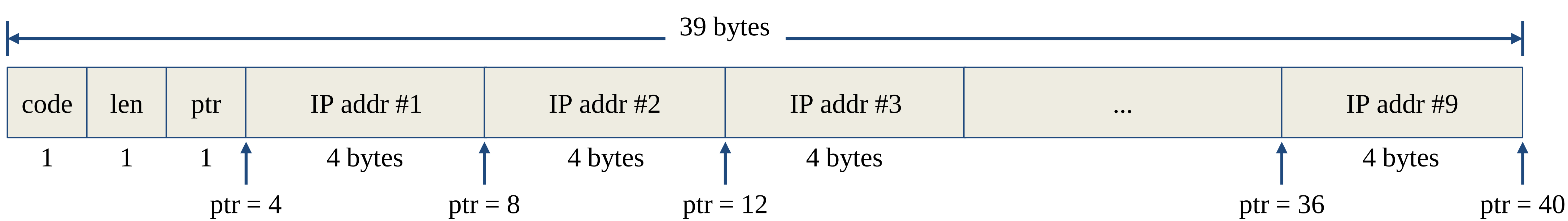
- The router IP in traceroute is the interface that receives the datagram. (incoming IP)
  - Traceroute from left host to right host
    - if1, if3
  - Traceroute from right host to left host
    - if4, if2



# Traceroute Program – IP Source Routing Option (1)

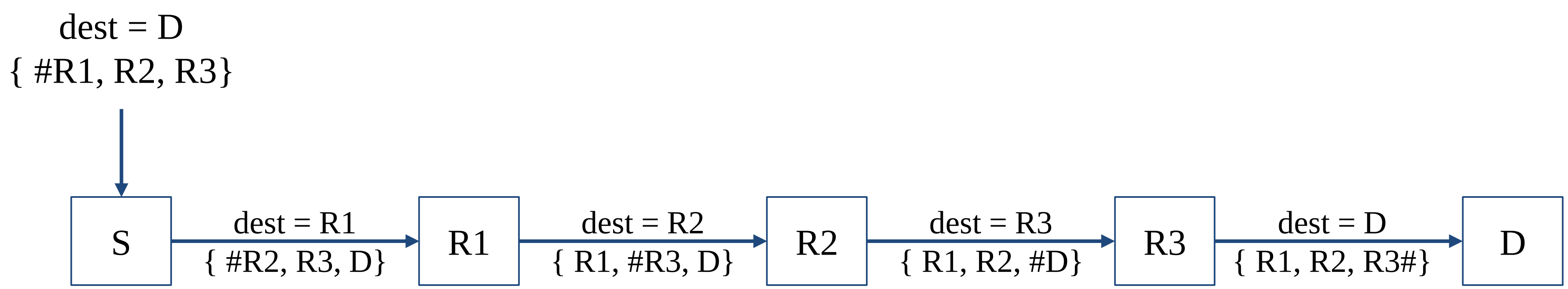



- Source Routing
  - Sender specifies the route
- Two forms of source routing
  - Strict source routing
    - Sender specifies the exact path that the IP datagram must follow
  - Loose source routing
    - As strict source routing, but the datagram can pass through other routers between any two addresses in the list
- Format of IP header option field
  - Code = 0x89 for strict and code = 0x83 for loose SR option



# Traceroute Program – IP Source Routing Option (2)

- Scenario of source routing
  - Sending host
    - Remove first entry and append destination address in the final entry of the list
  - Receiving router != destination
    - Loose source route, forward it as normal
  - Receiving router = destination
    - Next address in the list becomes the destination
    - Change source address
    - Increment the pointer

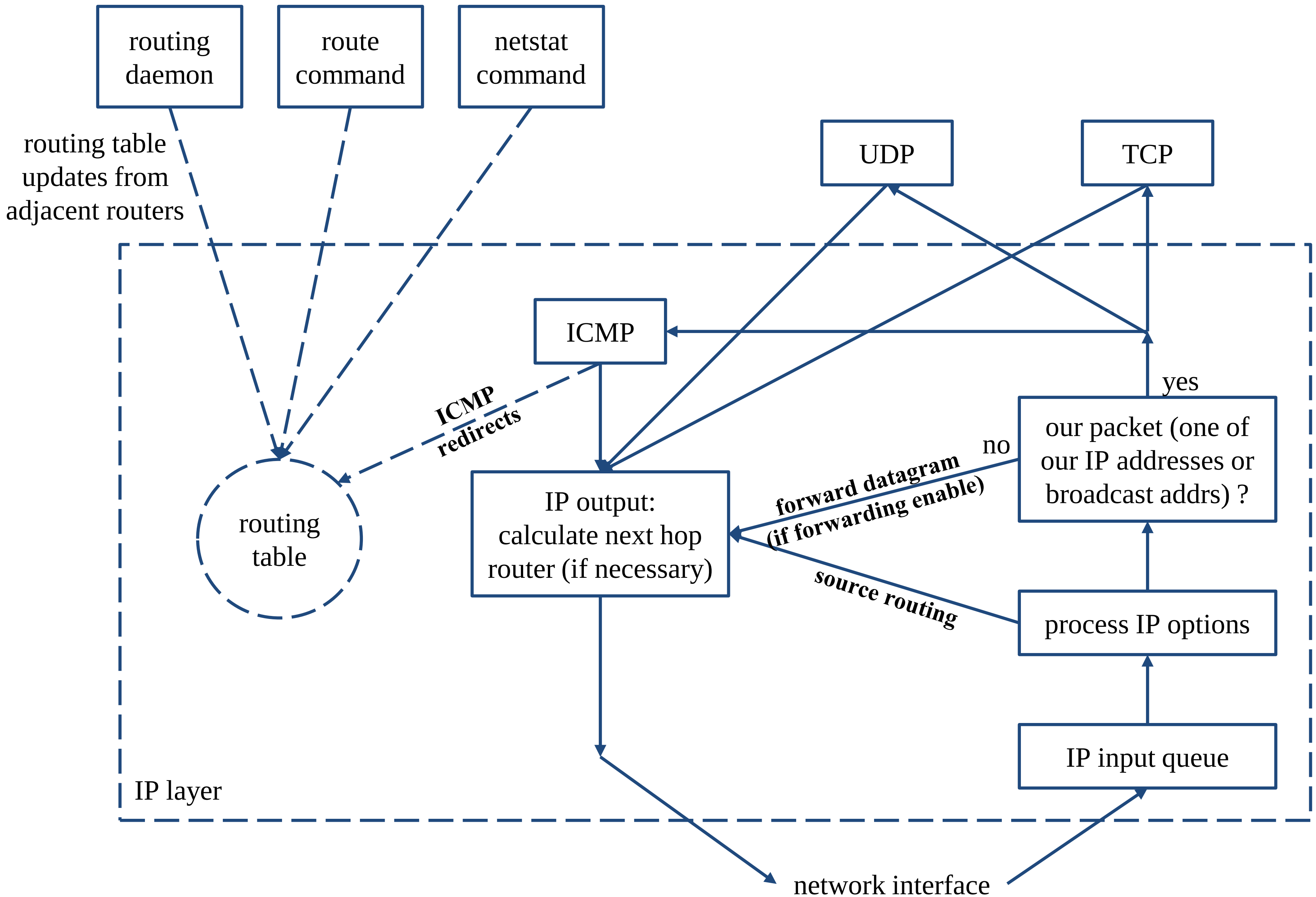


# Traceroute Program – IP Source Routing Option (3)

- Traceroute using IP loose SR option
- Example:

```
$ traceroute u2.nctu.edu.tw
traceroute to u2.nctu.edu.tw (211.76.240.193), 64 hops max, 40 byte packets
 1  e3rtn-235 (140.113.235.254)  0.549 ms  0.434 ms  0.337 ms
 2  140.113.0.166 (140.113.0.166)  108.726 ms  4.469 ms  0.362 ms
 3  v255-194.NTCU.net (211.76.255.194)  0.529 ms  3.446 ms  5.464 ms
 4  v255-229.NTCU.net (211.76.255.229)  1.406 ms  2.017 ms  0.560 ms
 5  h240-193.NTCU.net (211.76.240.193)  0.520 ms  0.456 ms  0.315 ms
$ traceroute -g 140.113.0.149 u2.nctu.edu.tw
traceroute to u2.nctu.edu.tw (211.76.240.193), 64 hops max, 48 byte packets
 1  e3rtn-235 (140.113.235.254)  0.543 ms  0.392 ms  0.365 ms
 2  140.113.0.166 (140.113.0.166)  0.562 ms  9.506 ms  0.624 ms
 3  140.113.0.149 (140.113.0.149)  7.002 ms  1.047 ms  1.107 ms
 4  140.113.0.150 (140.113.0.150)  1.497 ms  6.653 ms  1.595 ms
 5  v255-194.NTCU.net (211.76.255.194)  1.639 ms  7.214 ms  1.586 ms
 6  v255-229.NTCU.net (211.76.255.229)  1.831 ms  9.244 ms  1.877 ms
 7  h240-193.NTCU.net (211.76.240.193)  1.440 ms !S  2.249 ms !S  1.737 ms !S
```

# IP Routing – Processing in IP Layer



# IP Routing – Routing Table (1)

- Routing Table
  - Command to list: netstat -rn
  - Flag
    - U: the route is up
    - G: the route is to a router (indirect route)
      - Indirect route: IP is the dest. IP, MAC is the router's MAC
    - H: the route is to a host (Not to a network)
      - The dest. filed is either an IP address or network address
    - S: the route is static
  - Expire: expiration time for each route

```
$ netstat -rn
Routing tables

Internet:
Destination    Gateway        Flags  Netif  Expire
Default        140.113.17.254UGS    em0
127.0.0.1      link#2         UH     lo0
140.113.17.0/24      link#1 U      em0
140.113.17.225link#1         UHS    lo0
```

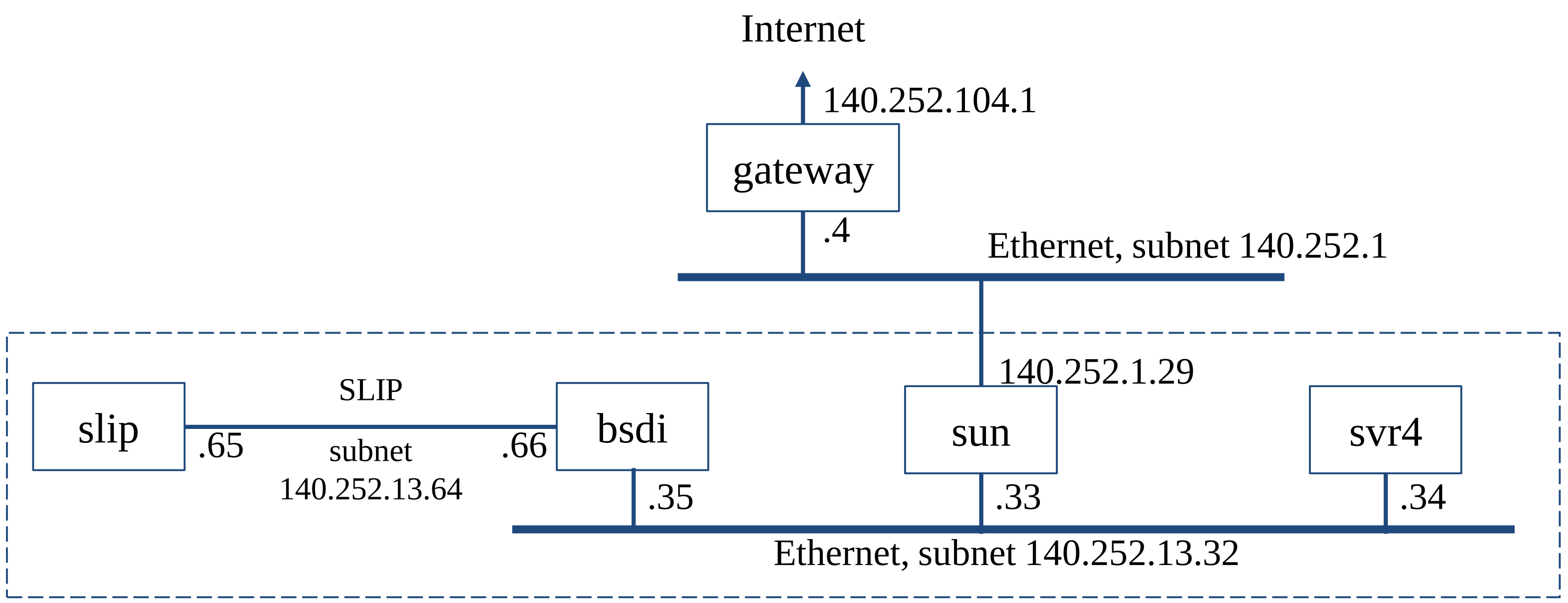
# IP Routing – Routing Table (2)

1. dst. = sun
2. dst. = slip
3. dst. = 192.207.117.2
4. dst. = svr4 or 140.252.13.34
5. dst. = 127.0.0.1

- Example:

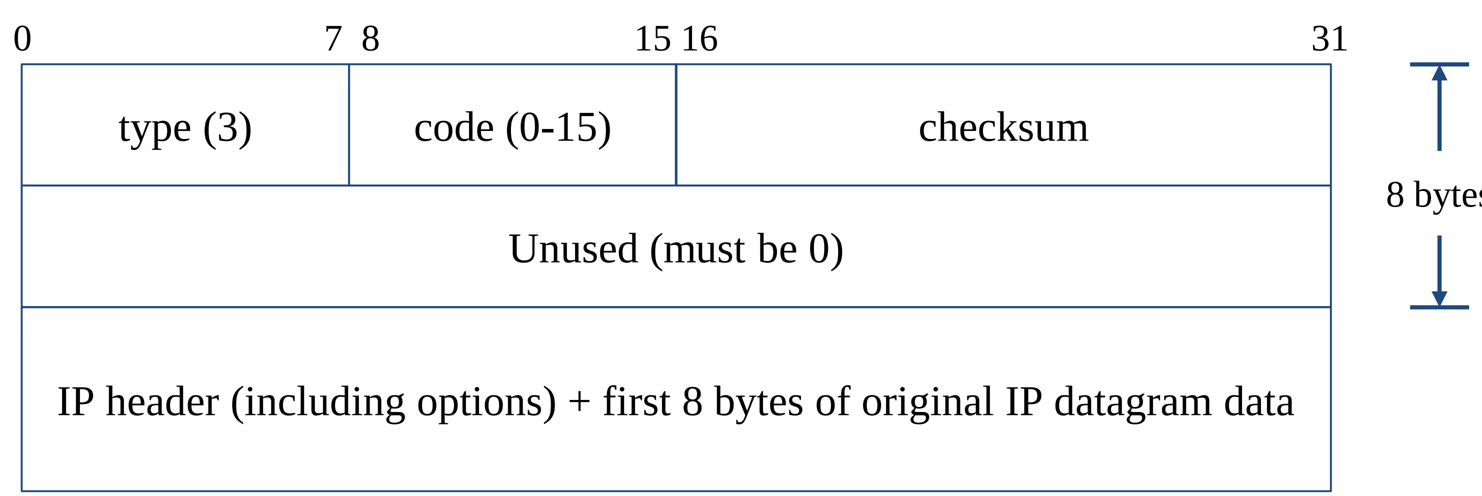
```
srv4 $ netstat -rn
Routing tables
Destination      Gateway          Flags  Refcnt  Use    Interface
140.252.13.65    140.252.13.35    UGH    0       0      emd0
127.0.0.1        127.0.0.1        UH     1       0      lo0
default          140.252.13.33    UG     0       0      emd0
140.252.13.32    140.252.13.34    U      4       25043  emd0
```

loopback



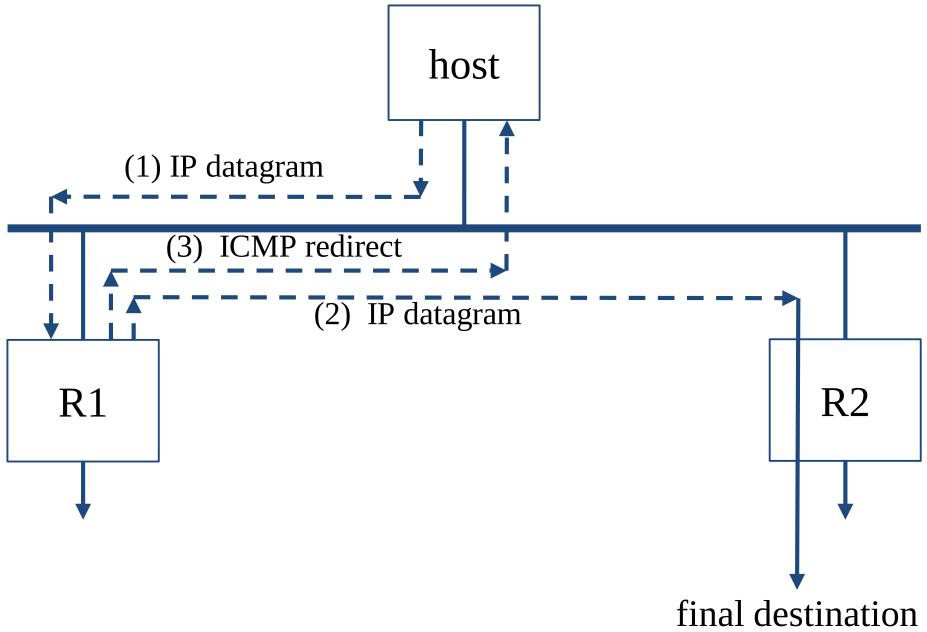
# ICMP – No Route to Destination

- If there is no match in routing table
  - If the IP datagram is generated on the host
    - “host unreachable” or “network unreachable”
  - If the IP datagram is being forwarded
    - ICMP “host unreachable” error message is generated and sends back to sending host
    - ICMP message
      - Type = 3, code = 0 for host unreachable
      - Type = 3, code = 1 for network unreachable



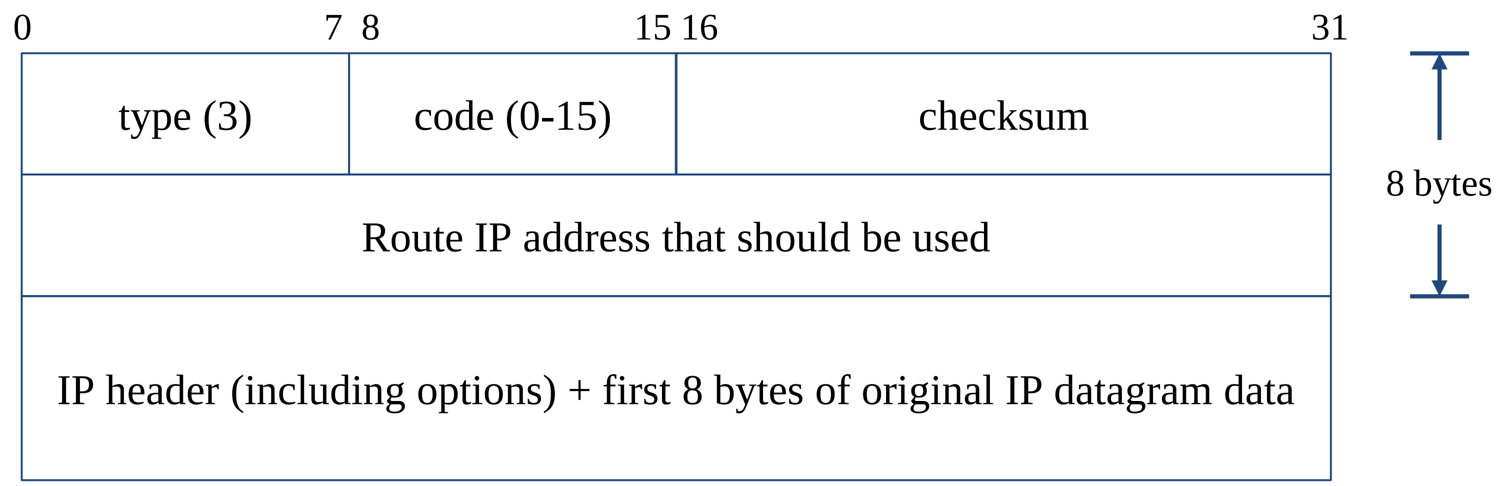
# ICMP – Redirect Error Message (1)

- Concept
  - Used by router to inform the sender that the datagram should be sent to a different router
  - This will happen if the host has a choice of routers to send the packet to
    - Ex:
      - R1 found sending and receiving interface are the same



# ICMP – Redirect Error Message (2)

- ICMP redirect message format
  - Code 0: redirect for network
  - Code 1: redirect for host
  - Code 2: redirect for TOS and network (RFC 1349)
  - Code 3: redirect for TOS and hosts (RFC 1349)



# ICMP – Router Discovery Messages (1)

- Dynamic update host's routing table
  - ICMP router solicitation message (懇求)
    - Host broadcast or multicast after bootstrapping
  - ICMP router advertisement message
    - Router response
    - Router periodically broadcast or multicast
- Format of ICMP router solicitation message

| 0     7 | 8     15 | 16     31 | |
|---|---|---|---|
| type (0) | code (0) | checksum | 8 bytes |
| Unused (sent as 0) | | | |

# ICMP – Router Discovery Messages (2)

- Format of ICMP router advertisement message
  - Router address
    - Must be one of the router’s IP address
  - Preference level
    - Preference as a default router address

| 0 – 7 | 8 – 15 | 16 – 31 |
|---|---|---|
| type (0) | code (0) | checksum |
| number of addresses | address entry size (2) | lifetime |
| router address [1] | | |
| preference level [1] | | |
| router address [2] | | |
| preference level [2] | | |
| … | | |

8 bytes

# UDP – User Datagram Protocol

# UDP

- No reliability
  - Datagram-oriented, not stream-oriented protocol
- UDP header
  - 8 bytes
  - Source port and destination port
    - Identify sending and receiving process
  - UDP length: $\geqq$ 8

| 0 7 8 15 | 16 31 | |
|---|---|---|
| 16-bit source port number | 16-bit destination port number | 8 bytes |
| 16-bit UDP length | 16-bit UDP checksum | |
| data (if any) | | |

# UDP

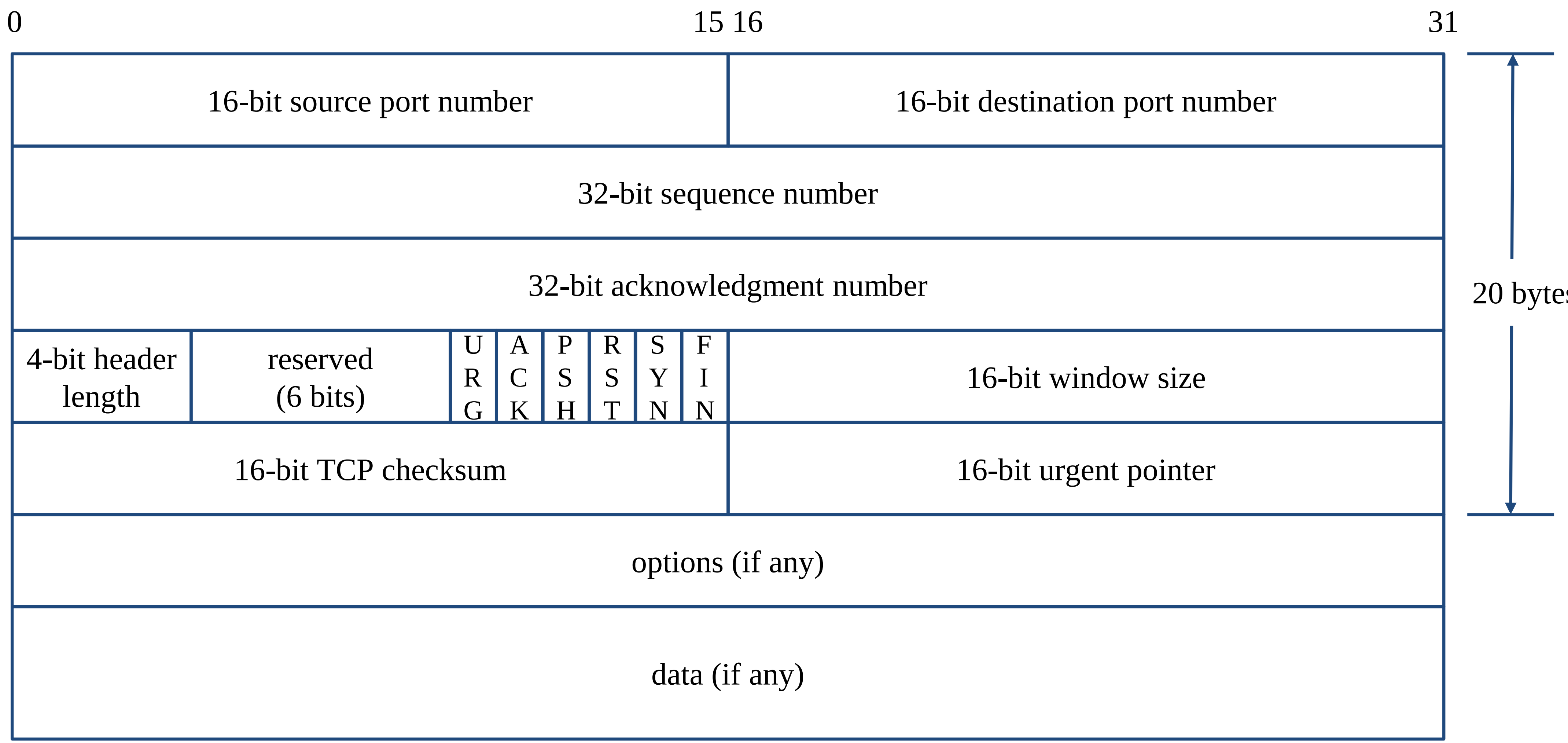
- Application
  - VoIP
  - VPN (OpenVPN over UDP)
  - DNS
  - SNMP
  - Quick UDP Internet Connections (QUIC)
    - Designed by Google, based on UDP
    - Renamed to “HTTP/3”
    - Keep reliability as TCP, but less latency
      - As most HTTP connections will demand TLS, QUIC makes the exchange of setup keys and supported protocols part of the initial handshake process.
      - During network-switch events, reuse old connection instead of creating a new one as TCP does.

# TCP – Transmission Control Protocol

# TCP

- Services
  - Connection-oriented
    - Establish TCP connection before exchanging data
  - Reliability
    - Acknowledgement when receiving data
    - Retransmission when timeout
    - Ordering
    - Discard duplicated data
    - Flow control

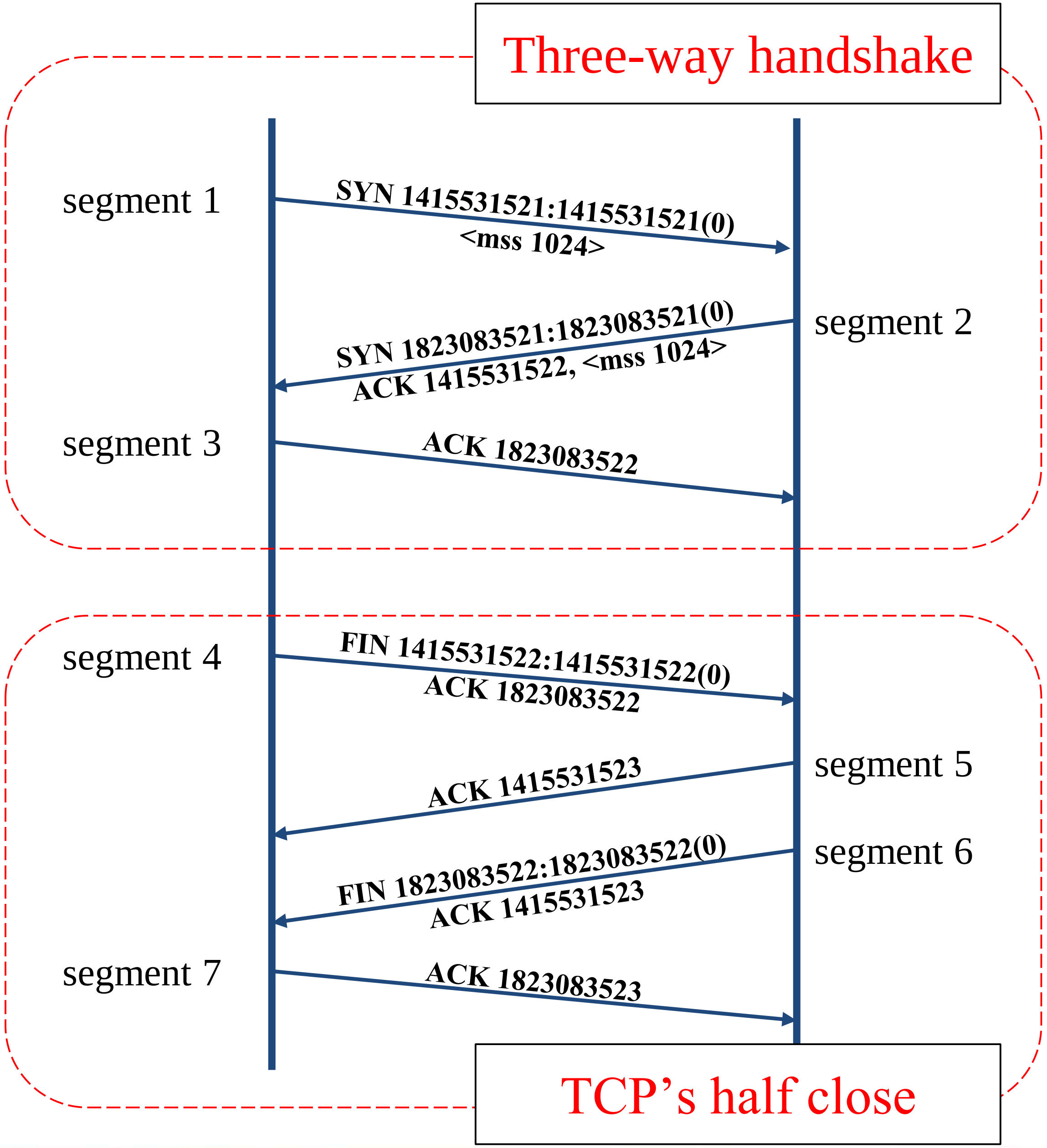
# TCP – Header (1)


0
15 16
31
16-bit source port number
16-bit destination port number
32-bit sequence number
32-bit acknowledgment number
4-bit header length
reserved (6 bits)
URG
ACK
PSH
RST
SYN
FIN
16-bit window size
16-bit TCP checksum
16-bit urgent pointer
options (if any)
data (if any)
20 bytes

# TCP – Header (2)

- Flags
  - SYN
    - Establish new connection
  - ACK
    - Acknowledgement number is valid
    - Used to ack previous data that host has received
  - RST
    - Reset connection
  - FIN
    - The sender is finished sending data

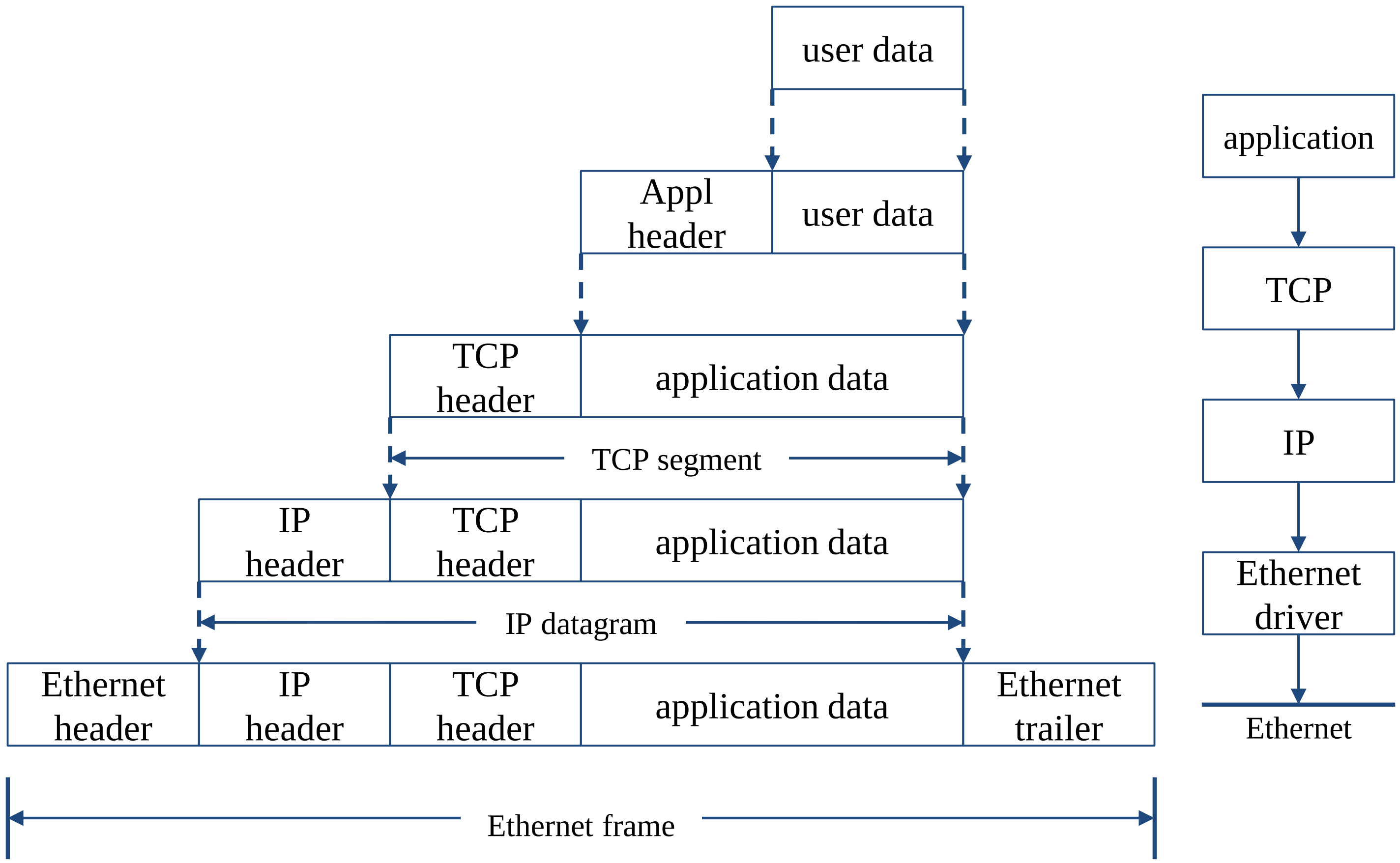
# TCP connection – establishment and termination



# Appendix

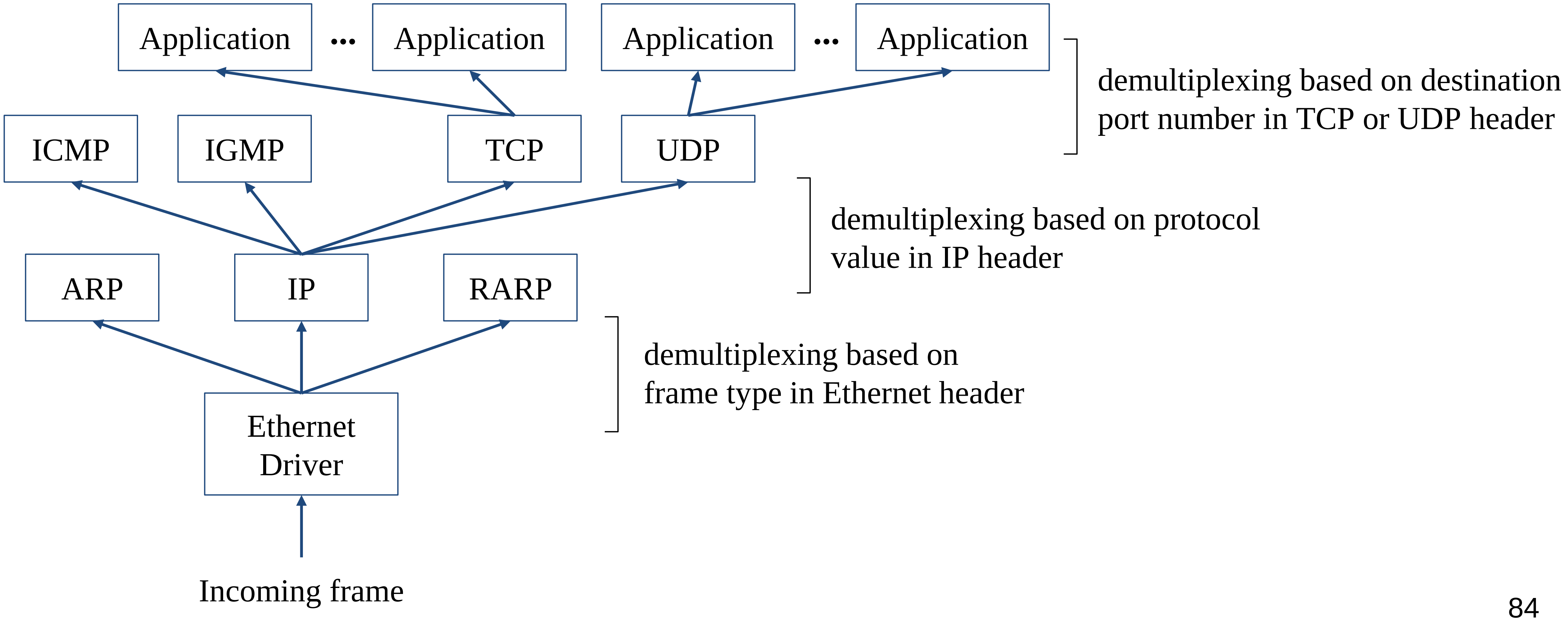
# Introduction – Encapsulation

- Multiplexing
  - Gathering data from multiple sockets, enveloping data with header



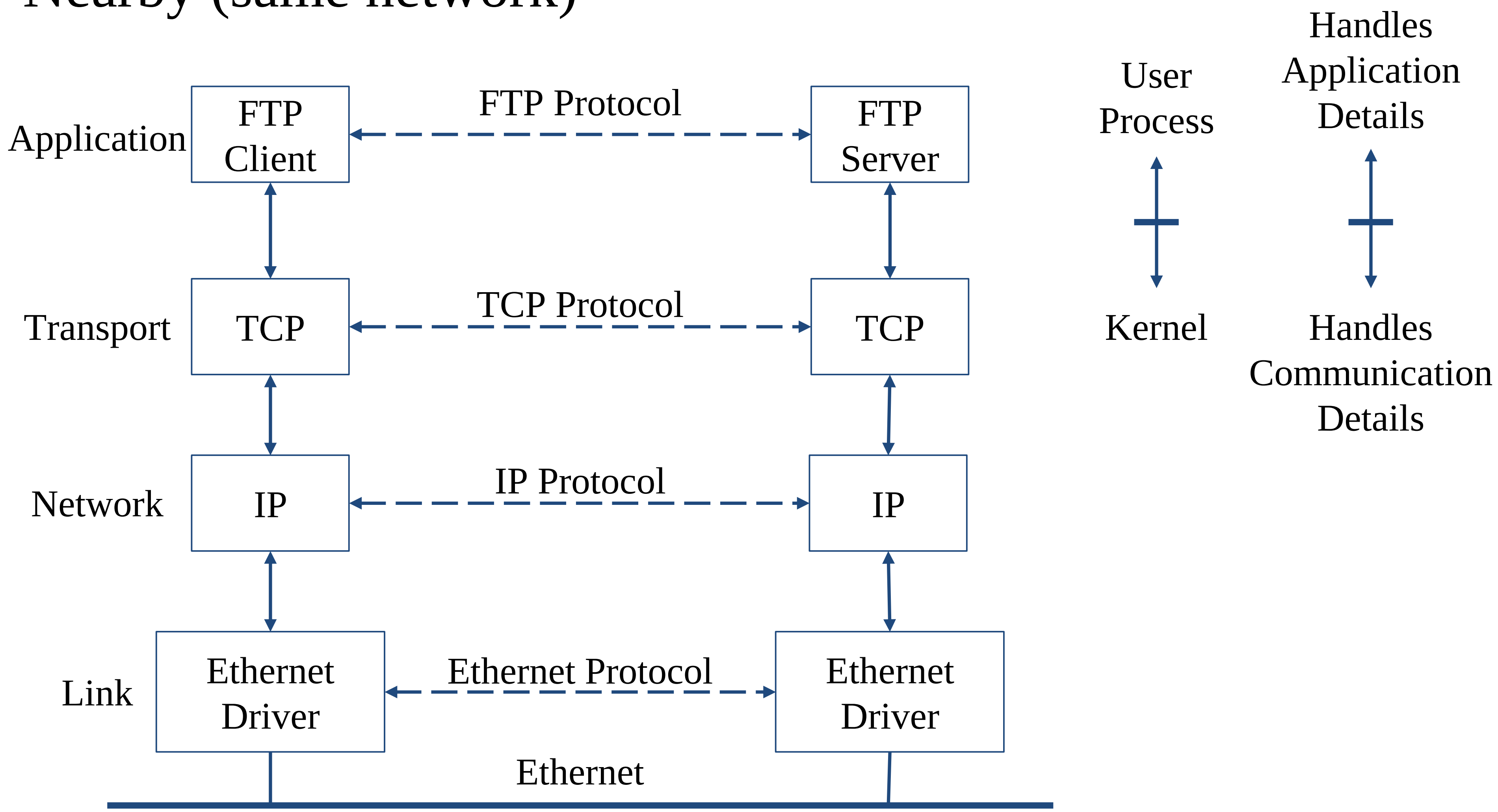
# Introduction – Decapsulation

- Demultiplexing
  - Delivering received segments to correct socket



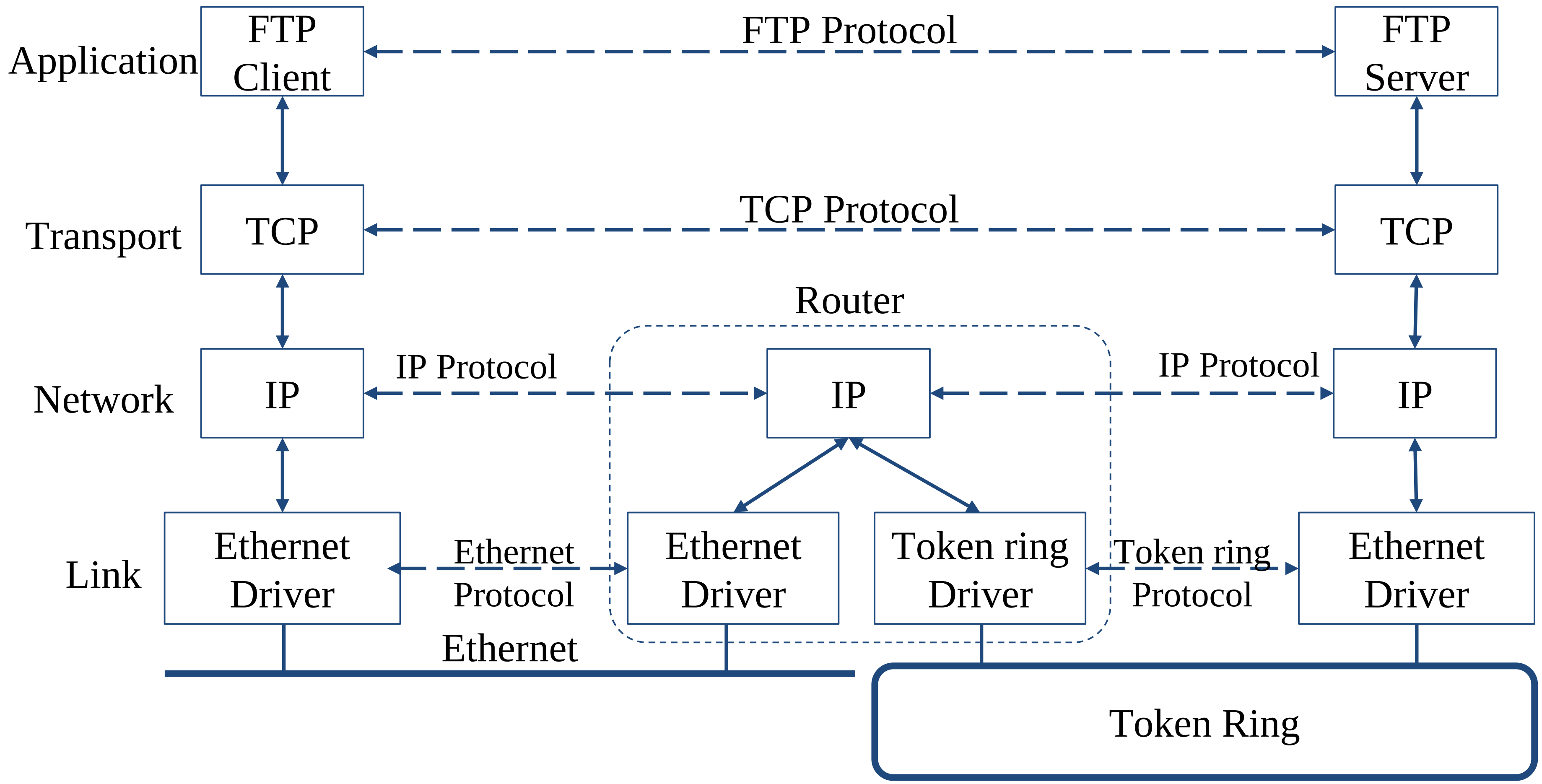
# Introduction – Addressing

- Addressing
  - Nearby (same network)

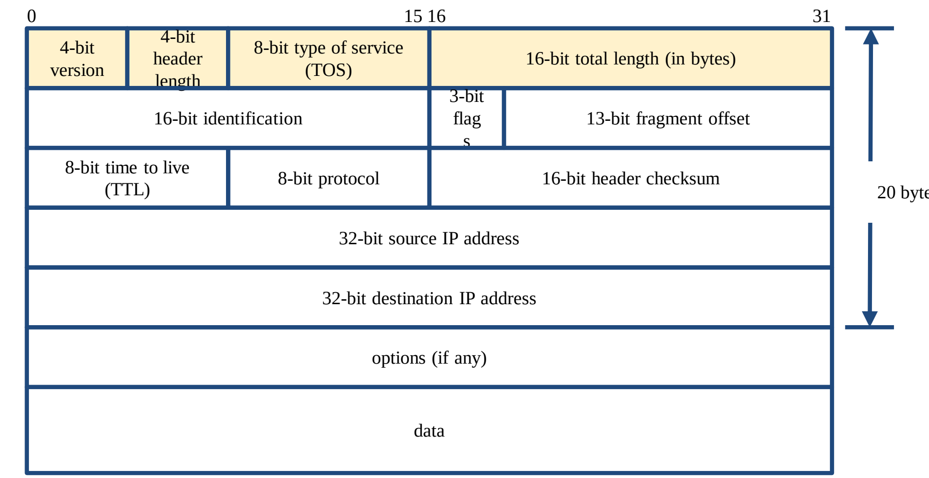


# Introduction – Addressing

- Addressing
    - Faraway (across network)



# Network Layer – IP Header (1)



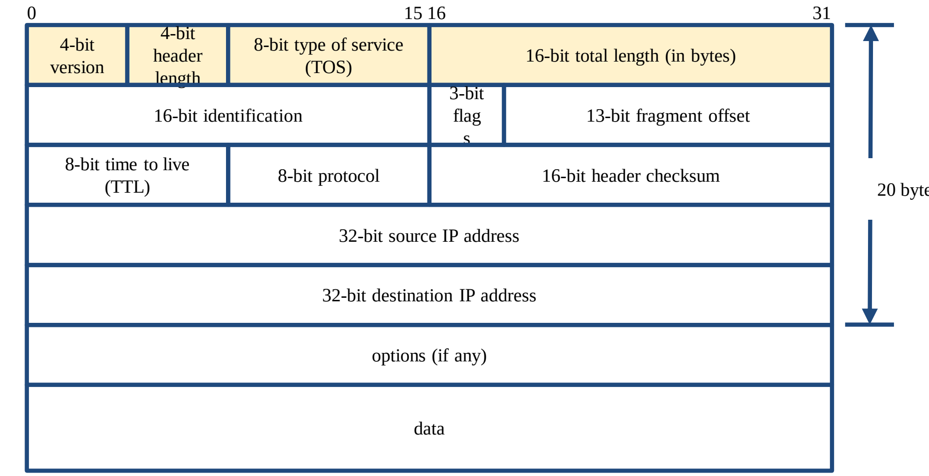

- Version (4-bit)
  - 4 for IPv4 and 6 for IPv6
- Header length (4-bit)
  - The number of 32-bit words in the header (15*4=60 bytes)
  - Normally, the value is 5 (no option)
- TOS - Type of Service (8-bit)
  - IP Precedence: 3-bit precedence + 4-bit TOS + 1-bit unused
  - DSCP: 3-bit major class + 3-bit drop preference + 2-bit ECN
- Total length (16-bit)
  - Total length of the IP datagram in bytes

DSCP: Differentiated Services Code Point
ECN: Explicit Congestion Notification

| Application | Minimize delay | Maximize throughput | Maximize reliability | Minimize monetary cost | Hex value |
|---|---|---|---|---|---|
| Telnet/Rlogin | 1 | 0 | 0 | 0 | 0x10 |
| FTP control | 1 | 0 | 0 | 0 | 0x10 |
| FTP data | 0 | 1 | 0 | 0 | 0x08 |
| any bulk data | 0 | 1 | 0 | 0 | 0x08 |
| TFTP | 1 | 0 | 0 | 0 | 0x10 |
| SMTP command phase | 1 | 0 | 0 | 0 | 0x10 |

| Name | Binary Value |
|---|---|
| Routine | 000 |
| Priority | 001 |
| Unneduate | 010 |
| Flash | 011 |
| Flash Override | 100 |
| Critic/critical | 010 |
| Internetwork Control | 110 |
| Network Control | 111 |

# Network Layer – IP Header (2)



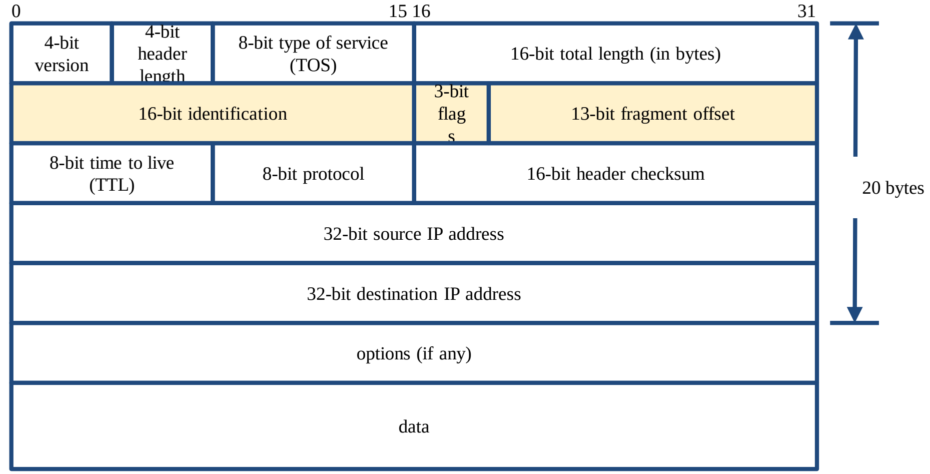

- DSCP - Differentiated Services Code Point (6-bit)
  - Supersede the ToS field in IPv4 to make
  - per-hop behavior (PHB) decisions
    - Default
      - Best-effort traffic
    - Expedited Forwarding (EF)
      - Dedicated to low-loss, low-latency traffic
    - Class Selector
      - Backward compatibility with the IP Precedence field
    - Assured Forwarding (AF)
      - Give assurance of delivery under prescribed conditions
- ECN: Explicit Congestion Notification (2-bit)
  - FreeBSD 8.0 implement ECN support for TCP
    - Enable ECN via sysctl(8)
      - net.inet.tcp.ecn.enable=1
    - Linux Kernel supports ECN for TCP since version 2.4.20

| **DSCP Class Selector Names** | **Binary DSCP Values** | **IPP Binary Values** | **IPP Names** |
|---|---|---|---|
| Default/CS0* | 000000 | 000 | Routine |
| CS1 | 001000 | 001 | Priority |
| CS2 | 010000 | 010 | Immediate |
| CS3 | 011000 | 011 | Flash |
| CS4 | 100000 | 100 | Flash Override |
| CS5 | 101000 | 101 | Critic/Critical |
| CS6 | 110000 | 110 | Internetwork Control |
| CS7 | 111000 | 111 | Network Control |

| Queue Class | Low Drop Probability | Medium Drop Probability | High Drop Probability |
|---|---|---|---|
| | Name/Dec/Bin | Name/Dec/Bin | Name/Dec/Bin |
| 1 | AF11 / 10 / 001010 | AF12 / 12 / 001100 | AF13 / 14 / 001110 |
| 2 | AF21 / 18 / 010010 | AF22 / 20 / 010100 | AF23 / 22 / 010110 |
| 4 | AF31 / 26 / 011010 | AF32 / 28 / 011100 | AF33 / 30 / 011110 |
| 5 | AF41 / 34 /100010 | AF42 / 36 / 100100 | AF43 /38 / 100110 |

| Binary Value | Description |
|---|---|
| 00 | Non ECN-Capable Transport, Non-ECT |
| 10 | ECN Capable Transport, ECT(0) |
| 01 | ECN Capable Transport, ECT(1) |

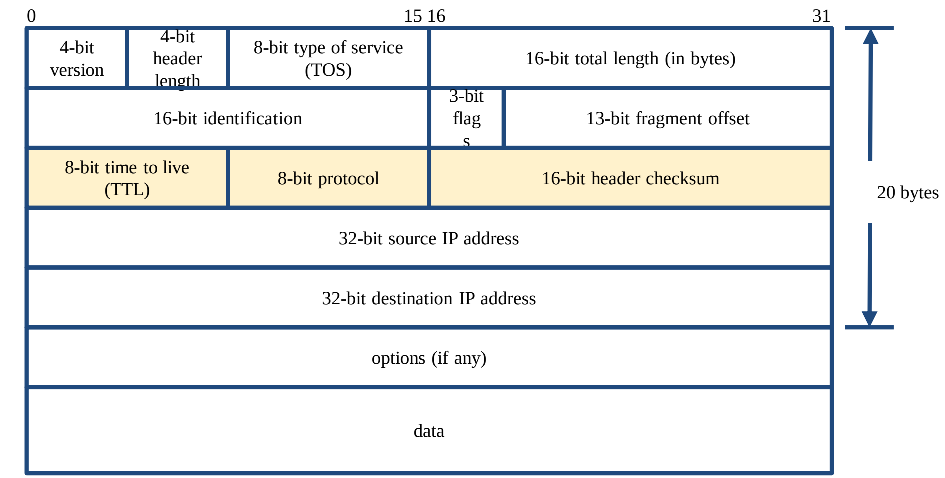
# Network Layer – IP Header (3)




- Identification (16-bit)
  - Identify the group of fragments of a single IP datagram
- Fragmentation offset (13-bit)
  - Specify the offset of a particular fragment relative to the beginning of the original unfragmented IP datagram
- Flags (3-bit)
  - All these three fields are used for fragmentation

| | | |
|---|---|---|
| Reserved | Don’t Fragment (DF) | More Fragments (MF) |

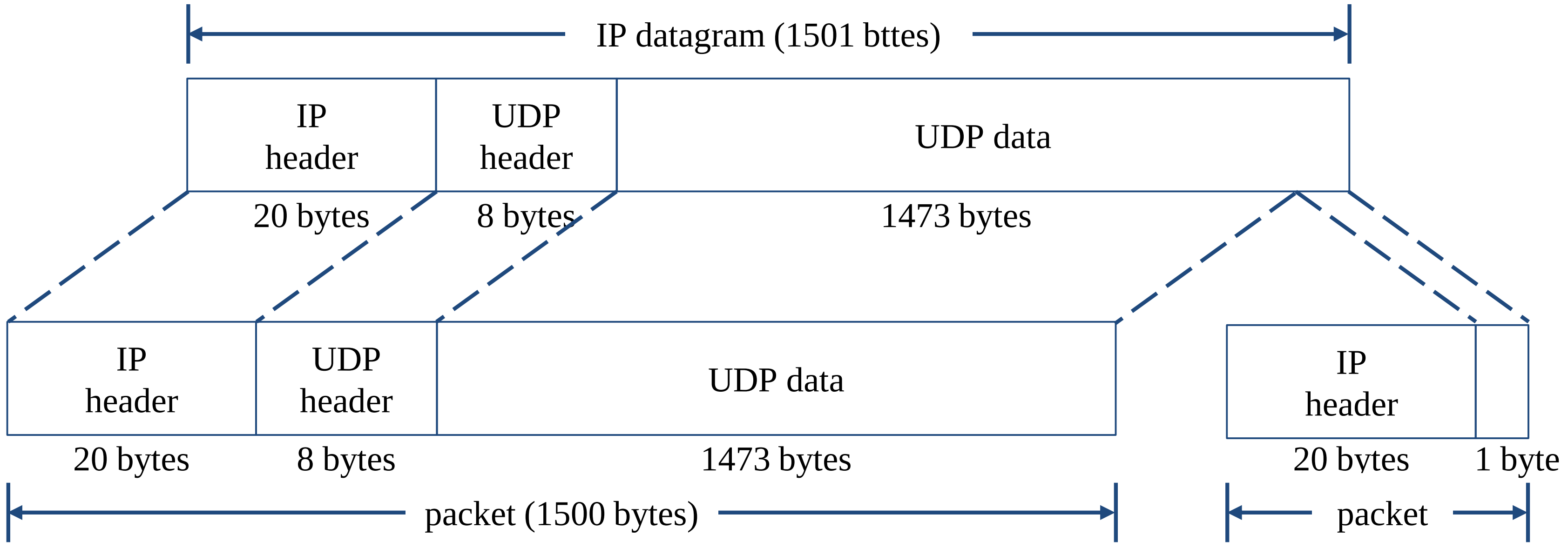
# Network Layer – IP Header (4)




- TTL (8-bit)
  - Limit of next hop count of routers
- Protocol (8-bit)
  - Used to demultiplex to other protocols
  - TCP, UDP, ICMP, IGMP
- Header checksum (16-bit)
  - Calculated over the IP header only
  - If checksum error, IP discards the datagram and no error message is generated

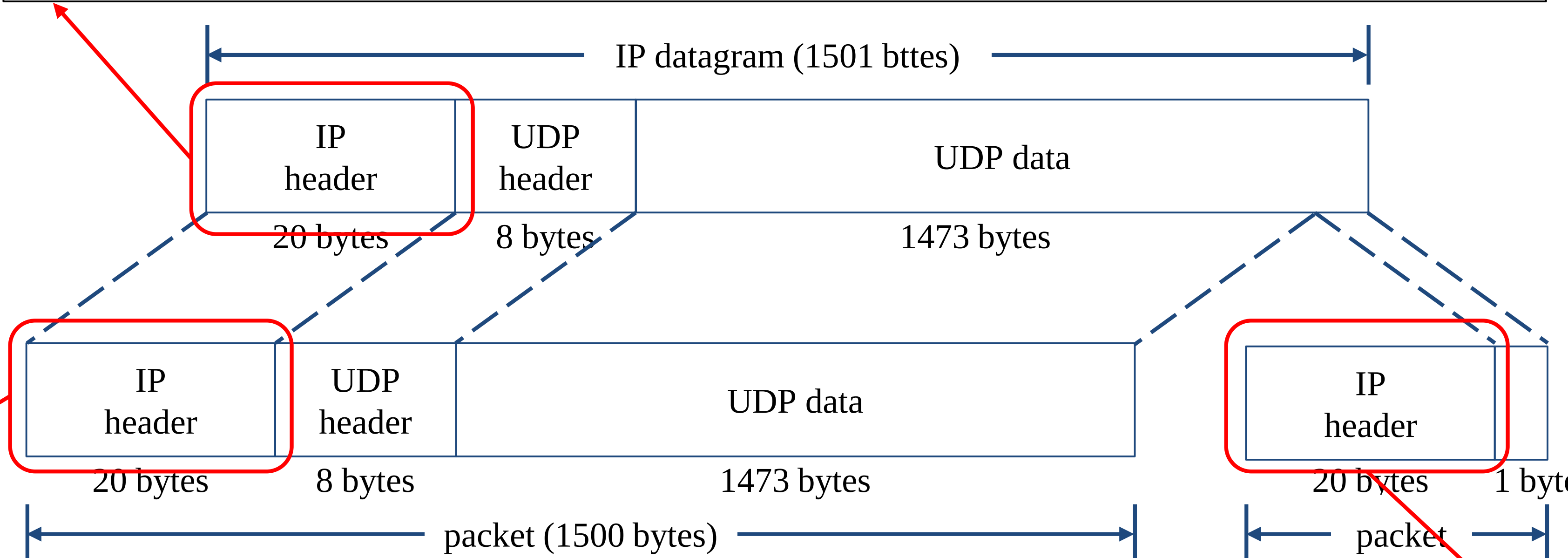
# IP Fragmentation (1)

- MTU limitation
  - Before network-layer to link-layer
    - IP will check the size and link-layer MTU
    - Do fragmentation if necessary
  - Fragmentation may be done at sending host or routers
  - Reassembly is done only in receiving host



# IP Fragmentation (1)

| | |
|---|---|
| identification: | which unique IP datagram |
| flags: | more fragments? |
| fragment offset | offset of this datagram from the beginning of original datagram |

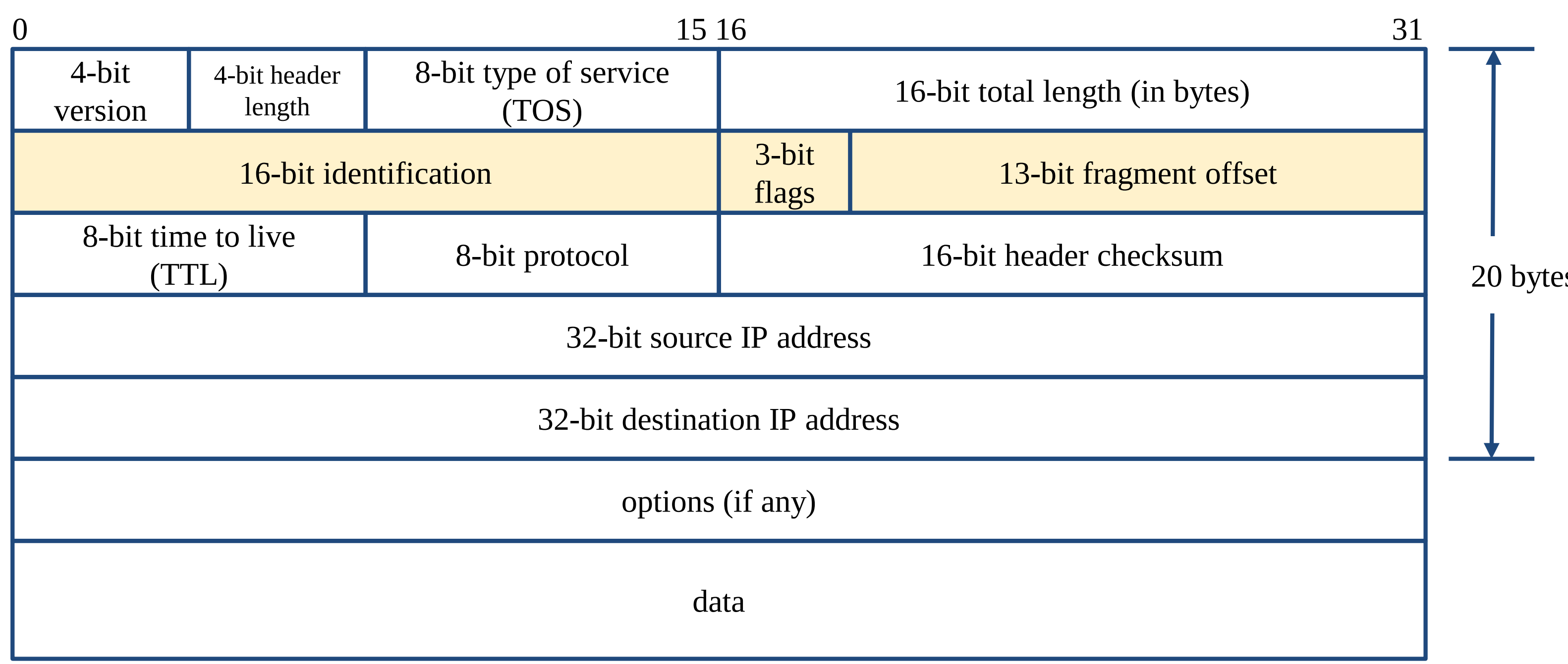



| | |
|---|---|
| identification: | the same |
| flags: | more fragments |
| fragment offset | 0 |

| | |
|---|---|
| identification: | the same |
| flags: | end of fragments |
| fragment offset | 1480 |

# IP Fragmentation (3)

- Issues of fragmentation
  - One fragment lost, entire datagram must be retransmitted
  - If the fragmentation is performed by intermediate router, there is no way for sending host how fragmentation did
  - Fragmentation is often avoided
    - There is a “don’t fragment” bit in flags of IP header



# ICMP Unreachable Error – Fragmentation Required

- Type=3, code=4
  - Router will generate this error message if the datagram needs to be fragmented, but the “don’t fragment” bit is turn on in IP header
- Message format

<table>
<tr><td>0 … 7</td><td>8 … 15</td><td>16 … 31</td><td></td></tr>
<tr><td>type (3)</td><td>code (4)</td><td>checksum</td><td rowspan="2">8 bytes</td></tr>
<tr><td colspan="2">Unused (must be 0)</td><td>MTU of next-hop network</td></tr>
<tr><td colspan="3">IP header (including options) + first 8 bytes of original IP datagram data</td><td></td></tr>
</table>

# ICMP – Source Quench Error

- Type=4, code=0
  - May be generated by system when it receives datagram at a rate that is too fast to be processed
  - Host receiving more than it can handle datagram
    - Send ICMP source quench or
    - Throw it away
  - Host receiving UDP source quench message
    - Ignore it or
    - Notify application

# Appendix of IP Options: IP Timestamp Option

- IP Timestamp Option
  - Similar to RR option
  - Record Timestamp in option field
    - code, len, ptr are the same as IP RR option
    - OF
      - Overflow field
      - Router will increment OF if it can’t add a timestamp because of no room left
    - FL
      - Flags
      - 0: only timestamp
      - 1: both timestamp and IP address
      - 3: the sender initiates the options with up to 4 pairs of IP address and timestamp

40 bytes

| code | len | ptr | OF | FL | timestamp #1 | timestamp #2 | timestamp #3 | ... | timestamp #9 |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | | | 4 bytes | 4 bytes | 4 bytes | | 4 bytes |