Internet Protocols

IP Header, Fragmentation / Forwarding / Encapsulation / IPv6

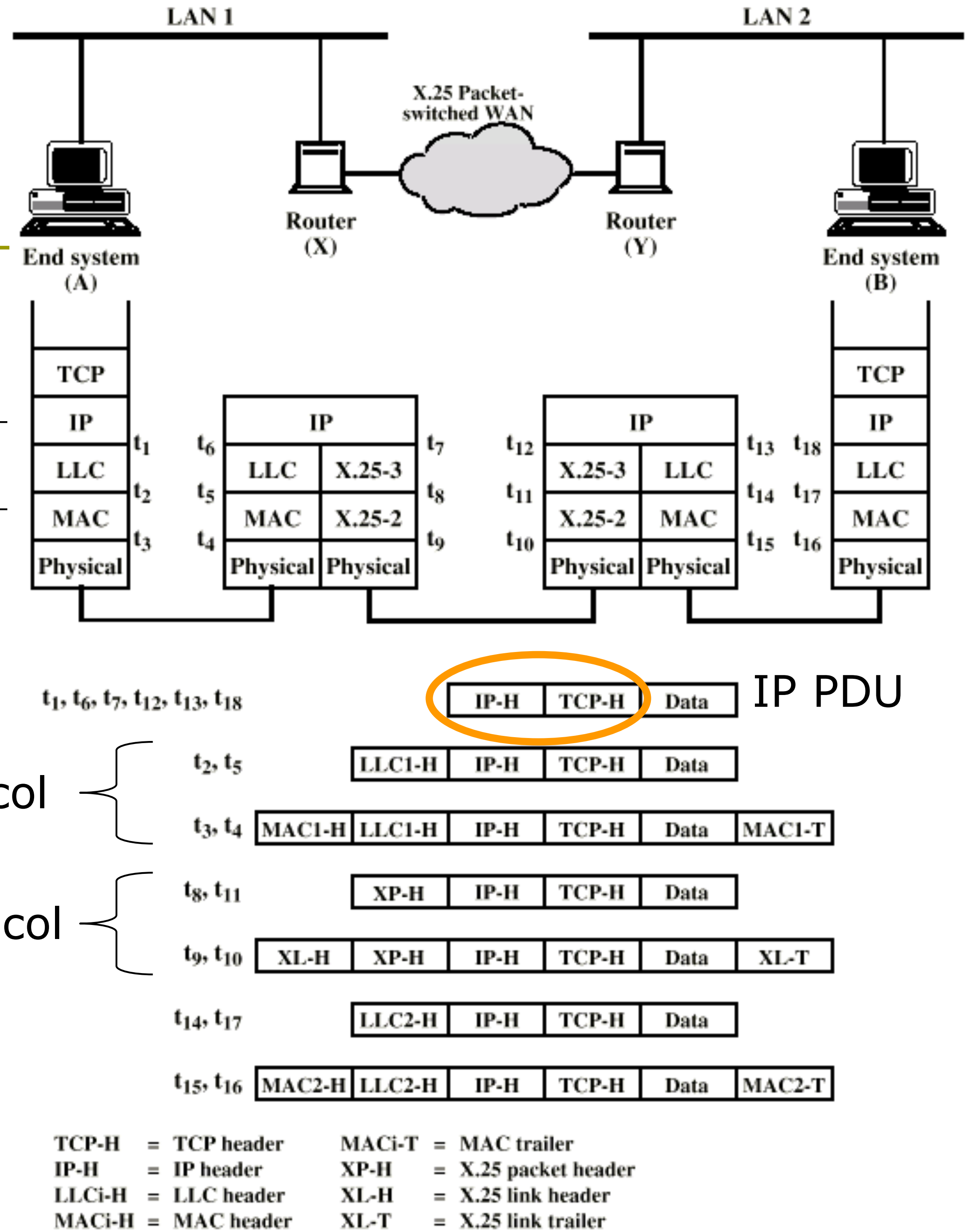
IP Operation

Go to Router X

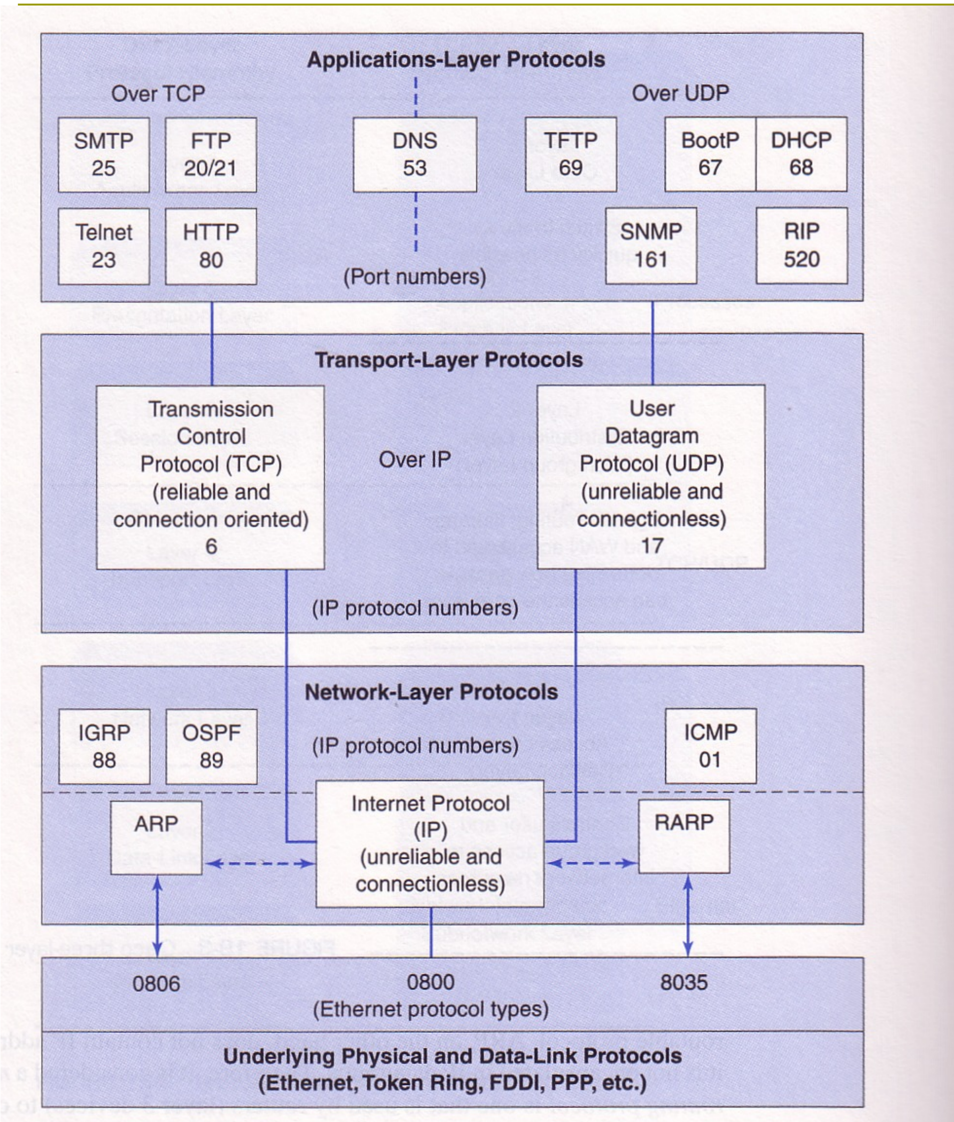
MAC address for Router X

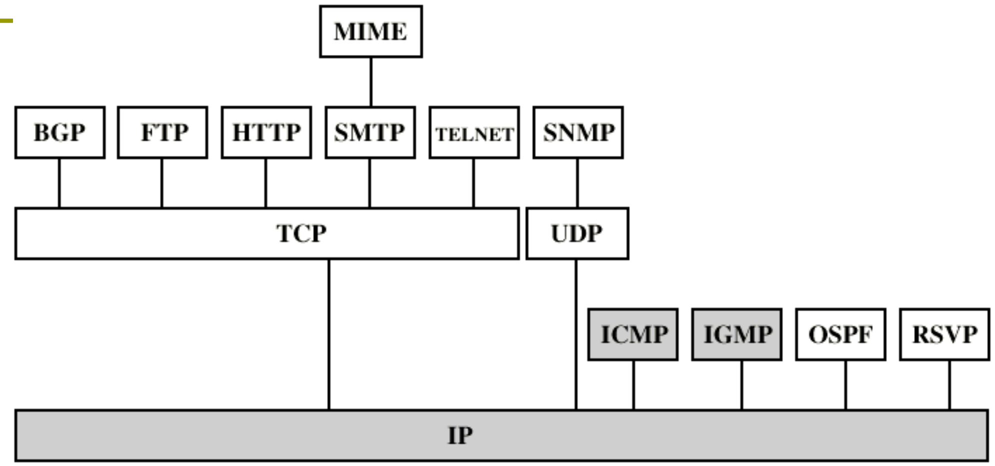
Encapsulated with LAN protocol

Encapsulated with X.25 protocol



TCP/IP Stack Protocol





- Bridge
  - IS used to connect two LANs using similar LAN protocols
  - Address filter passing on packets to the required network only
  - OSI layer 2 (Data Link)
- Router
  - Connects two (possibly dissimilar) networks
  - Uses internet protocol present in each router and end system
  - OSI Layer 3 (Network)

IP Header Format

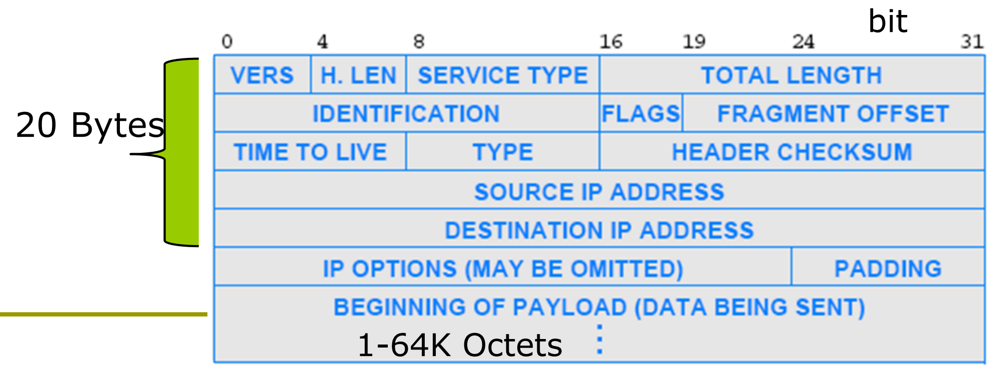
- Source address
- Destination address
- Protocol
  - Recipient e.g. TCP
- Type of Service
  - Specify treatment of data unit during transmission through networks
- Identification
  - Source, destination address and user protocol
  - Uniquely identifies PDU
  - Needed for re-assembly and error reporting
  - Send only

+	0 - 3	4 - 7	8 - 15	16 - 18	19 - 31
0	Version	Header length	Type of Service	Total Length	
32	Identification			Flags	Fragment Offset
64	Time to Live		Protocol	Header Checksum	
96	Source Address				
128	Destination Address				
160	Options + padding				
192	1-64K Octets Data				

20 Bytes (rows 0–128)

HL=5 rows→ 20 octet (variable) / 8*20/32

IP Header Format

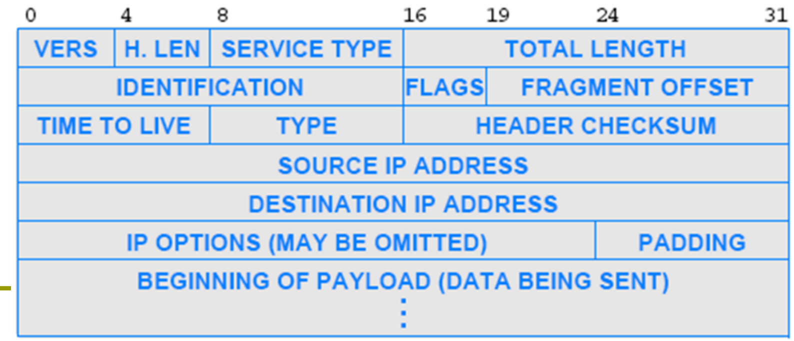


- VERS
  - Each datagram begins with a 4-bit protocol version number (the figure shows a version 4 header)
- H.LEN (Header Length)
  - Number of 32-bit rows in the Header → Header Length
  - 4-bit header specifies the number of 32-bit quantities in the header (in the figure we have 5 32-bit rows)
  - If no options are present, the value is 5

HL=5 → 20 octet (variable), thus 8*20/32

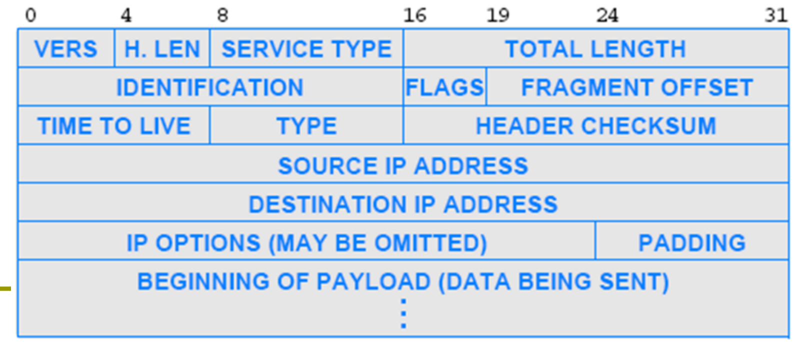
- SERVICE TYPE
  - 8-bit field that carries a class of service for the datagram
  - Seldom used in practice
  - Chapter 28 explains the DiffServ interpretation of the service type field
- TOTAL LENGTH
  - 16-bit integer that specifies the total number of bytes in the datagram
  - Includes both the header and the data

IP Header Format



- IDENTIFICATION
  - 16-bit number (usually sequential) assigned to the datagram
    - used to gather all fragments for reassembly of the datagram
- FLAGS
  - 3-bit field with individual bits specifying whether the datagram is a fragment
    - If so, then whether the fragment corresponds to the rightmost piece of the original datagram
- FRAGMENT OFFSET
  - 13-bit field that specifies where in the original datagram the data in this fragment belongs
  - the value of the field is multiplied by 8 to obtain an offset

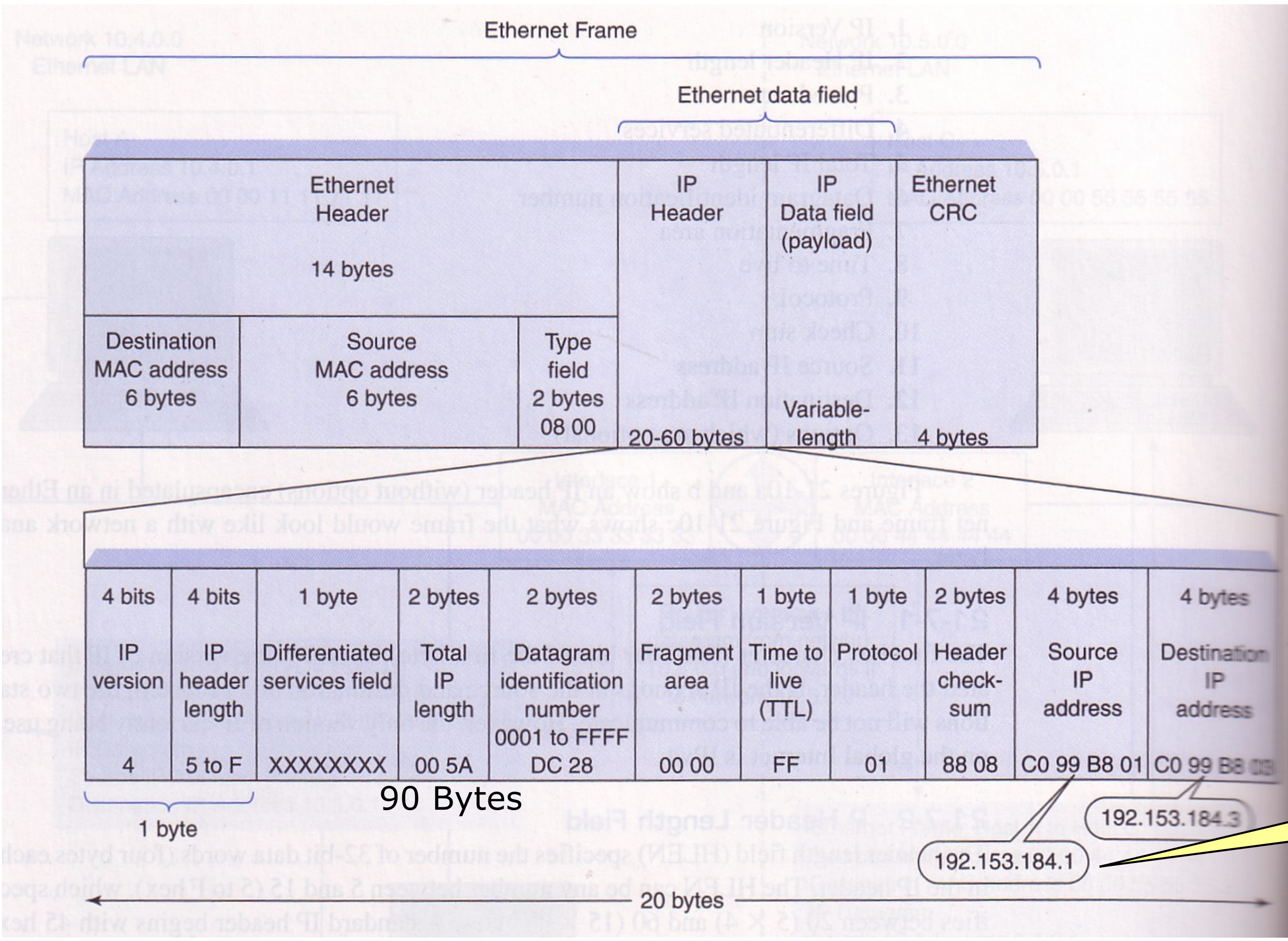
IP Header Format



- TIME TO LIVE
  - 8-bit integer initialized by the original sender
  - Represents the max. number of hops the packets can visit
  - it is decremented by each router that processes the datagram
  - if the value reaches zero (0)
    - the datagram is discarded and an error message is sent back to the source
- TYPE
  - 8-bit field that specifies the type of the payload
- HEADER CHECKSUM
  - 16-bit ones-complement checksum of header fields
- SOURCE IP ADDRESS
  - 32-bit Internet address of the original sender
  - The addresses of intermediate routers do not appear in the header

Example:
Encapsulated IP Packet in Ethernet Frame

Ethernet Frame Carrying IP Packet

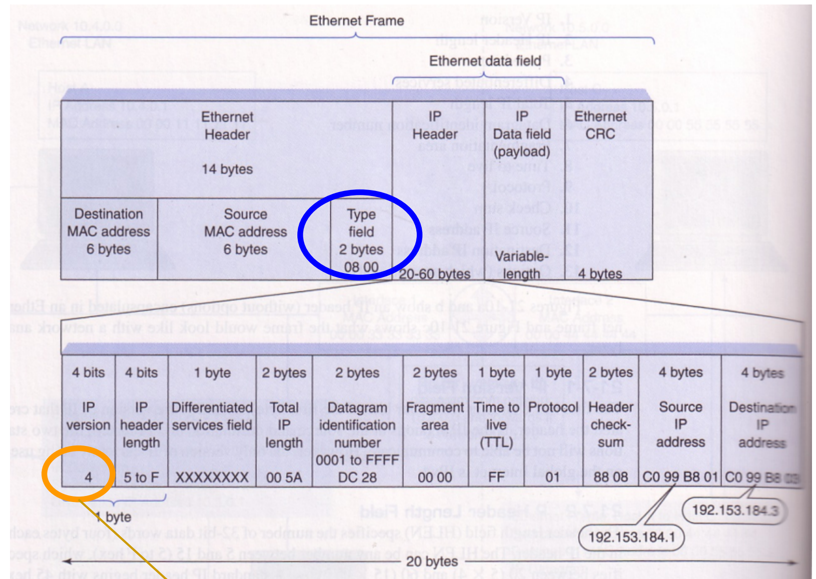


Important!

MAC and Associated IP address

Example:
Encapsulated IP Packet in Ethernet Frame

Ethernet Frame Carrying IP Packet



An Ethernet frame containing IP information has **08 00** in its type field

IP starting with **45** Hex indicates IPv4 with standard HED length of 20 bytes = 5 rows x 32/8

IP starting with 4F Hex indicates IPv4 with HED length of 60 bytes = 15 rows x 32/8

Remember: 2^4=16; **45= 0100 0101**= One Byte

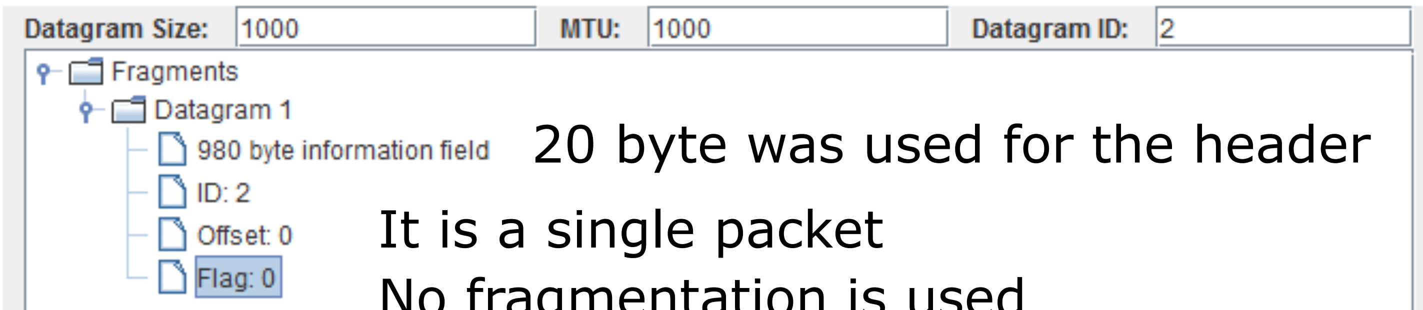
Protocol Analyzer Display:

0000 00 00 C0 A0 51 24 00 C0 93 21 88 A7 08 00 45 08
0010 00 5A DC 28 00 00 FF 01 88 08 C0 99 B8 01 C0 99
0020 B8 03 2a B4 DD …..

Example:

99 is one byte
1001 1001

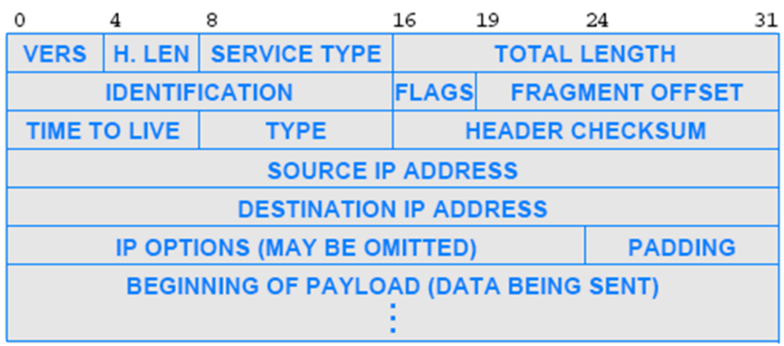
Example of a Single IP Packet



20 byte was used for the header

It is a single packet

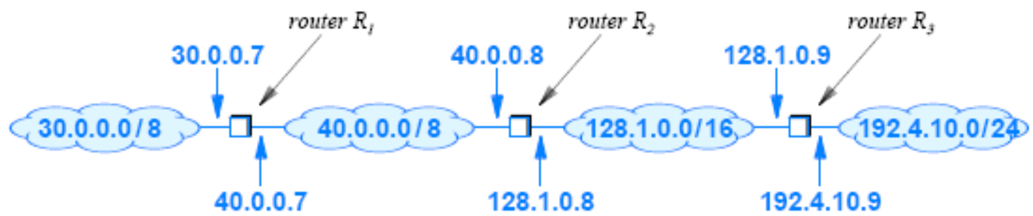
No fragmentation is used



http://media.pearsoncmg.com/aw/aw_kurose_network_2/applets/ip/ipfragmentation.html

Forwarding

- The Internet uses next-hop forwarding
- To make the selection of a next hop efficient, an IP router uses a forwarding table
- Mask field is used to direct the incoming packet
- Number of entries in the table can be very large
- A forwarding table is initialized when the router boots
  - Forwarding table must be updated if the topology changes or hardware fails



(a)

Destination	Mask	Next Hop
30.0.0.0	255.0.0.0	40.0.0.7
40.0.0.0	255.0.0.0	deliver direct
128.1.0.0	255.255.0.0	deliver direct
192.4.10.0	255.255.255.0	128.1.0.9

(b)

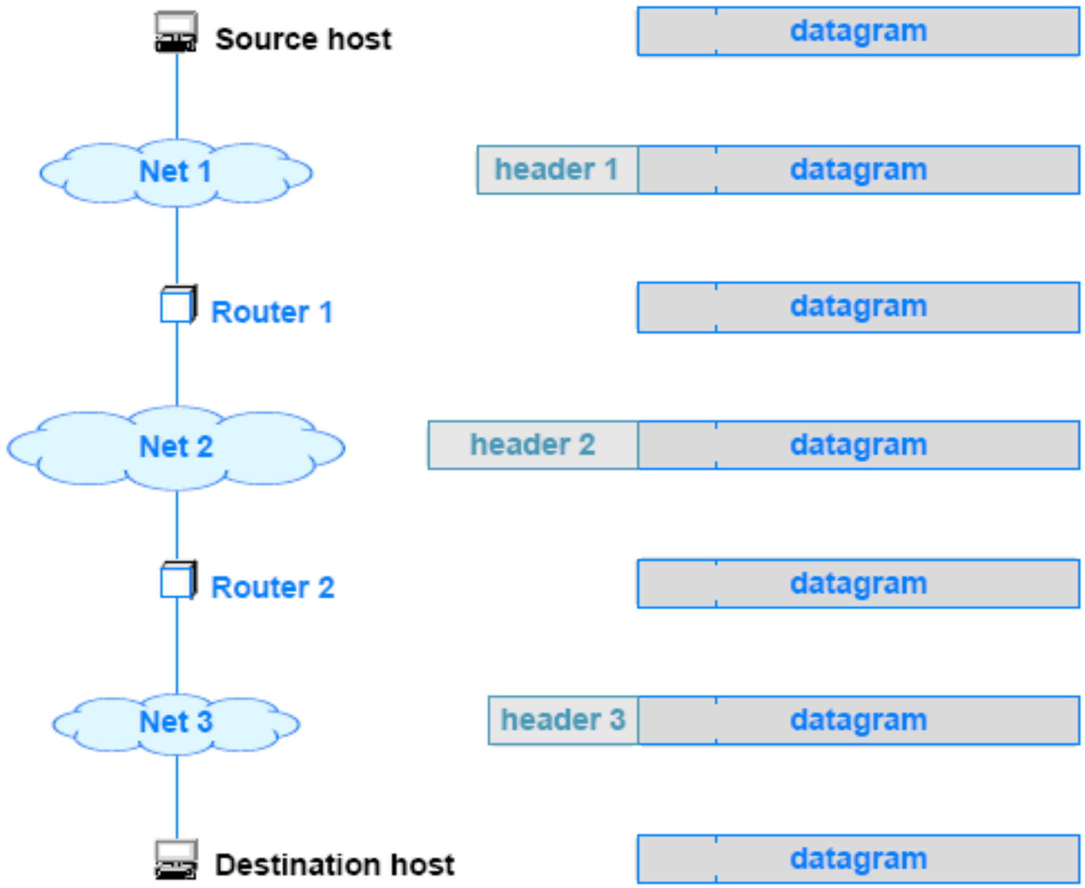
Routing Table: e.g., If a packet with destination 30.0.0.0 arrives at R2 → The next hop will be 40.0.0.7

Longest Prefix

- Suppose a router's forwarding table contains entries for the following two network prefixes:

  128.10.0.0/16 and 128.10.2.0/24
- What happens if a datagram arrives destined to 128.10.2.3?
- Matching procedure succeeds for both of the entries
  - a Boolean and of a 16-bit mask will produce 128.10.0.0
  - a Boolean and with a 24-bit mask will produce 128.10.2.0
- Which entry should be used?
  - To handle ambiguity that arises from overlapping address masks, Internet forwarding uses a longest prefix match
    - Instead of examining the entries in arbitrary order
    - forwarding software arranges to examine entries with the longest prefix first
- In the example above, Internet forwarding will choose the entry that corresponds to 128.10.2.0/24

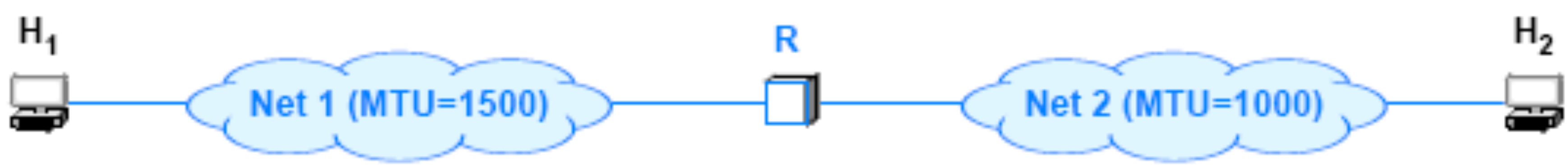
Transmission Across the Internet



Header can change – Going through WiFi or Ethernet

Transmission Across the Internet

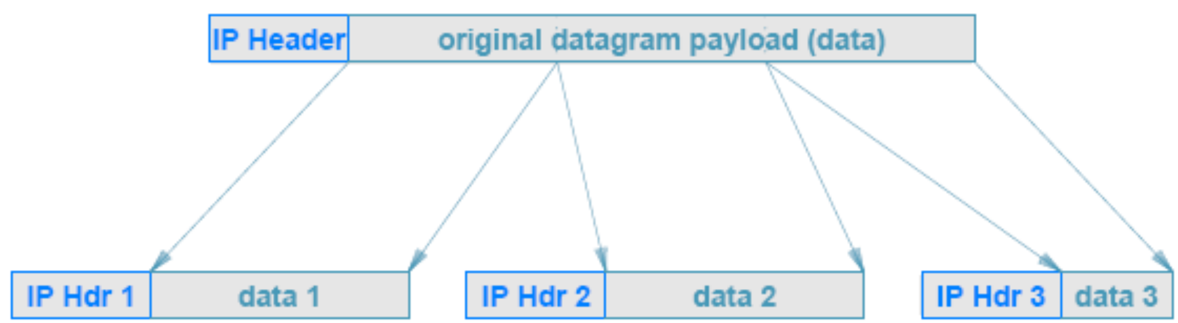
- Each hardware technology specifies the maximum amount of data that a frame can carry
  - The limit is known as a Maximum Transmission Unit (MTU)
- There is no exception to the MTU limit
  - Network hardware is not designed to accept or transfer frames that carry more data than the MTU allows
  - A datagram must be smaller or equal to the network MTU
- In an internet that contains heterogeneous networks, MTU restrictions create a problem
- A router can connect networks with different MTU values
  - a datagram that a router receives over one network can be too large to send over another network



Two networks with different MTU
(a heterogeneous network)

IP Fragmentation (1)

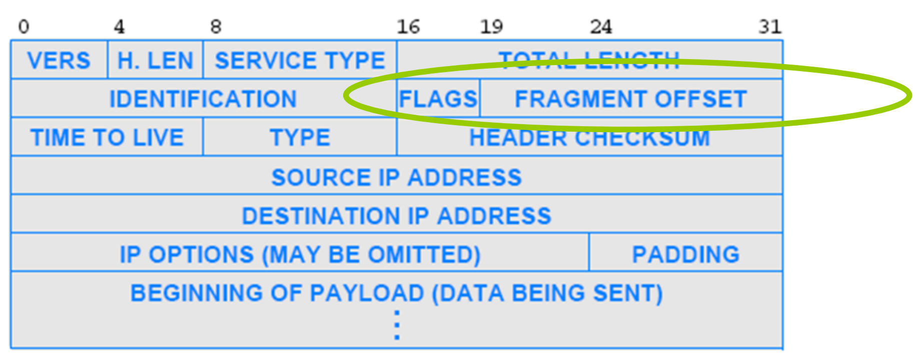
- IP re-assembles at destination only
- Uses fields in header
  - Data Unit Identifier (ID)
    - Identifies end system originated datagram
      - Source and destination address
      - Protocol layer generating data (e.g. TCP)
      - Identification supplied by that layer



Fragmentation

IP Fragmentation (2)

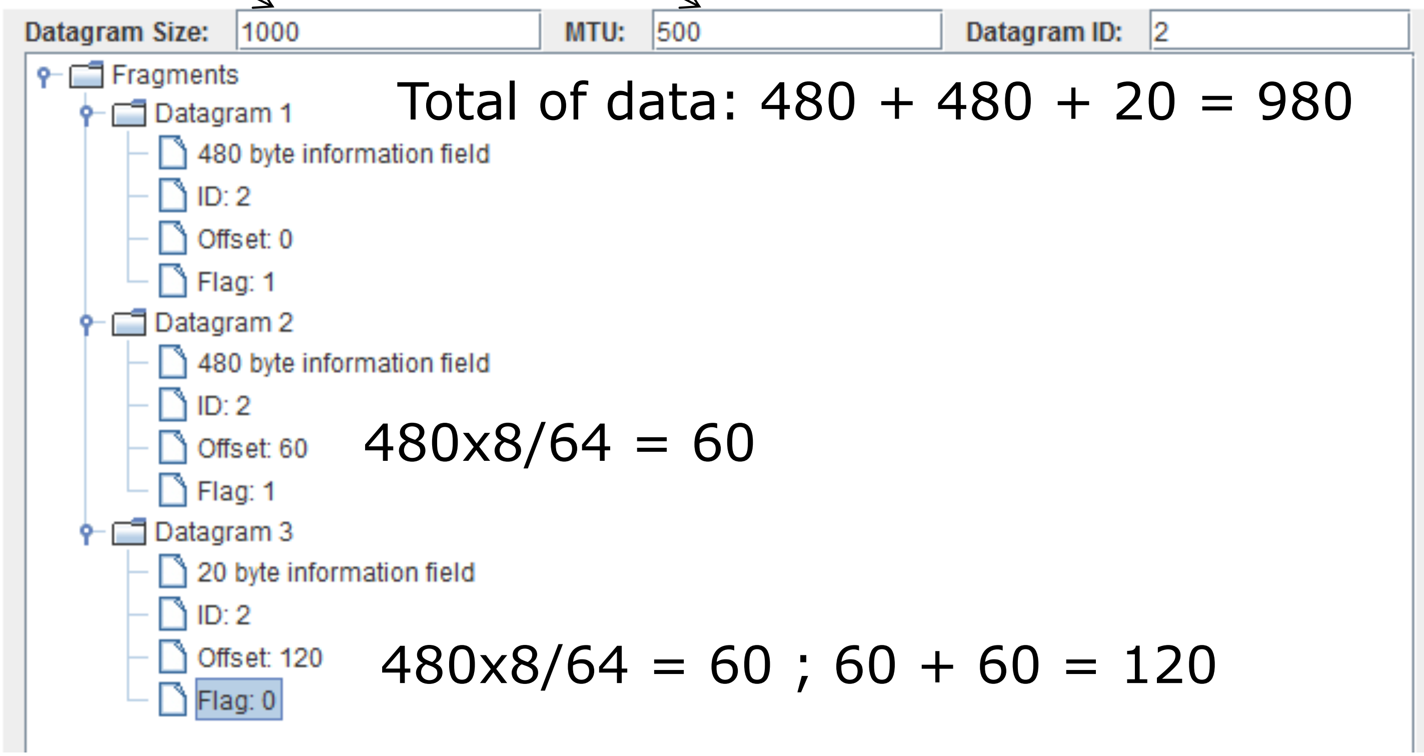
- Offset
  - Position of fragment of user data in original datagram
  - In multiples of 64 bits (8 octets)
- *More* flag (more is coming!)
  - Indicates that this is not the last fragment



IP Fragmentation (3)

Data Size =
Data + Header

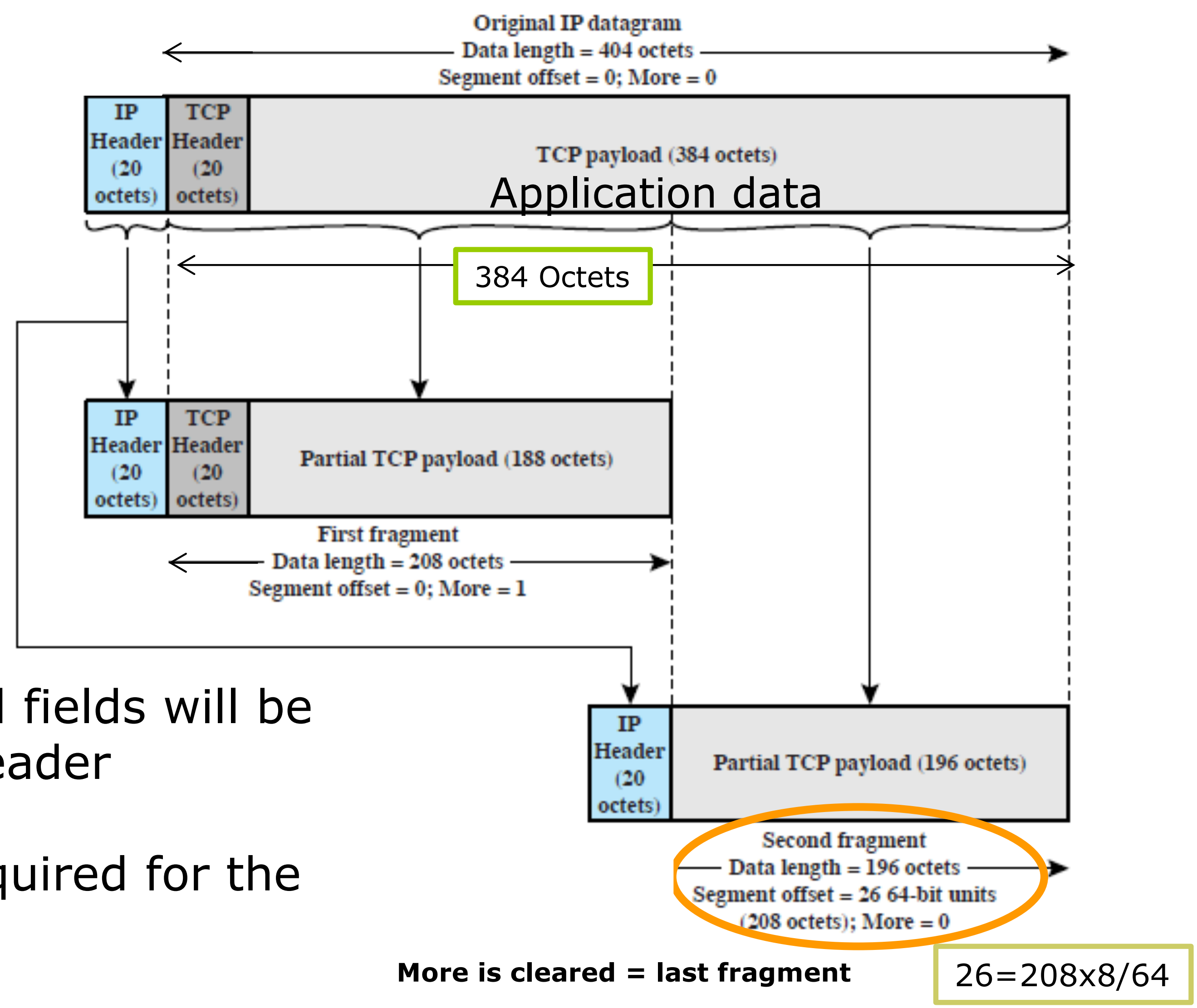
MTU = Max Data Size =
Data + Header



- Offset
  - Position of fragment of user data in original datagram
  - In multiples of 64 bits (8 octets)
- *More* flag
  - Indicates that this is not the last fragment

http://media.pearsoncmg.com/aw/aw_kurose_network_2/applets/ip/ipfragmentation.html

Fragmentation Example



Fragmentation-related fields will be modified in each IP header

TCP header is only required for the first fragment

Fragmentation Example

IP + TCP + Data = Data Size

Max. Datagram Size= IP + TCP + Data

Compare with the previous figure!

Length	Info
1514	Echo (ping) request id=0x0001, seq=57/14592, ttl=128
1514	Fragmented IP protocol (proto=ICMP 1, off=1480, ID=7474)
582	Fragmented IP protocol (proto=ICMP 1, off=2960, ID=7474)
1514	Fragmented IP protocol (proto=ICMP 1, off=0, ID=07f8) [Reassembled
1514	Fragmented IP protocol (proto=ICMP 1, off=1480, ID=07f8) [Reassemb
582	Echo (ping) reply id=0x0001, seq=57/14592, ttl=127

Fragmentation

Source	Destination	Protoco
10.10.0.3	192.168.0.128	ICMP
10.10.0.3	192.168.0.128	IPv4
10.10.0.3	192.168.0.128	IPv4
192.168.0.128	10.10.0.3	IPv4
192.168.0.128	10.10.0.3	IPv4
192.168.0.128	10.10.0.3	ICMP

```
▷ Frame 1: 1514 bytes on wire (12112 bits), 1514
▷ Ethernet II, Src: 00:25:b3:bf:91:ee (00:25:b3:b
▽ Internet Protocol Version 4, Src: 10.10.0.3 (10
    Version: 4
    Header length: 20 bytes
  ▷ Differentiated Services Field: 0x00 (DSCP 0x0
    Total Length: 1500
    Identification: 0x7474 (29812)
  ▽ Flags: 0x01 (More Fragments)
      0... .... = Reserved bit: Not set
      .0.. .... = Don't fragment: Not set
      ..1. .... = More fragments: Set
    Fragment offset: 0
    Time to live: 128
    Protocol: ICMP (1)
  ▷ Header checksum: 0x0000 [incorrect, should be
    Source: 10.10.0.3 (10.10.0.3)
    Destination: 192.168.0.128 (192.168.0.128)
    [Source GeoIP: Unknown]
    [Destination GeoIP: Unknown]
▽ Internet Control Message Protocol
    Type: 8 (Echo (ping) request)
    Code: 0
    Checksum: 0xdeaa
    Identifier (BE): 1 (0x0001)
    Identifier (LE): 256 (0x0100)
    Sequence number (BE): 57 (0x0039)
    Sequence number (LE): 14592 (0x3900)
    [Response In: 6]
  ▽ Data (1472 bytes)
```

Flag = 1

Creating a large ICMP frame:

"ping –s 3000 -M do 192.168.1.1"

Fragmentation

```
▷ Frame 2: 1514 bytes on wire (12112 bits), 1514 l
▷ Ethernet II, Src: 00:25:b3:bf:91:ee (00:25:b3:b
▽ Internet Protocol Version 4, Src: 10.10.0.3 (10
    Version: 4
    Header length: 20 bytes
  ▷ Differentiated Services Field: 0x00 (DSCP 0x0
    Total Length: 1500
    Identification: 0x7474 (29812)
  ▽ Flags: 0x01 (More Fragments)
      0... .... = Reserved bit: Not set
      .0.. .... = Don't fragment: Not set
      ..1. .... = More fragments: Set
    Fragment offset: 1480
    Time to live: 128
    Protocol: ICMP (1)
  ▷ Header checksum: 0x0000 [incorrect, should be
    Source: 10.10.0.3 (10.10.0.3)
    Destination: 192.168.0.128 (192.168.0.128)
    [Source GeoIP: Unknown]
    [Destination GeoIP: Unknown]
▽ Data (1480 bytes)
    Data: 6162636465666768696a6b6c6d6e6f707172737
    [Length: 1480]
```

```
▷ Frame 3: 582 bytes on wire (4656 bits), 582 byt
▷ Ethernet II, Src: 00:25:b3:bf:91:ee (00:25:b3:b
▽ Internet Protocol Version 4, Src: 10.10.0.3 (10
    Version: 4
    Header length: 20 bytes
  ▷ Differentiated Services Field: 0x00 (DSCP 0x
    Total Length: 568
    Identification: 0x7474 (29812)
  ▽ Flags: 0x00
      0... .... = Reserved bit: Not set
      .0.. .... = Don't fragment: Not set
      ..0. .... = More fragments: Not set
    Fragment offset: 2960
    Time to live: 128
    Protocol: ICMP (1)
  ▷ Header checksum: 0x0000 [incorrect, should be
    Source: 10.10.0.3 (10.10.0.3)
    Destination: 192.168.0.128 (192.168.0.128)
    [Source GeoIP: Unknown]
    [Destination GeoIP: Unknown]
▽ Data (548 bytes)
    Data: 696a6b6c6d6e6f70717273747576777616263646
    [Length: 548]
```

Dealing with Failure

- Re-assembly may fail if some fragments get lost
  - Requires buffer
  - Failures must be detected
- Ways to deal with failures (two approaches)
  - Use re-assembly time-out
    - Assigned to first fragment to arrive
    - If timeout expires before all fragments arrive, discard partial data
  - Use packet lifetime (TTL in IP) -
    - If time-to-live runs out, kill partial data
    - Note: TTL can be in hops or sec.

Sub-fragments

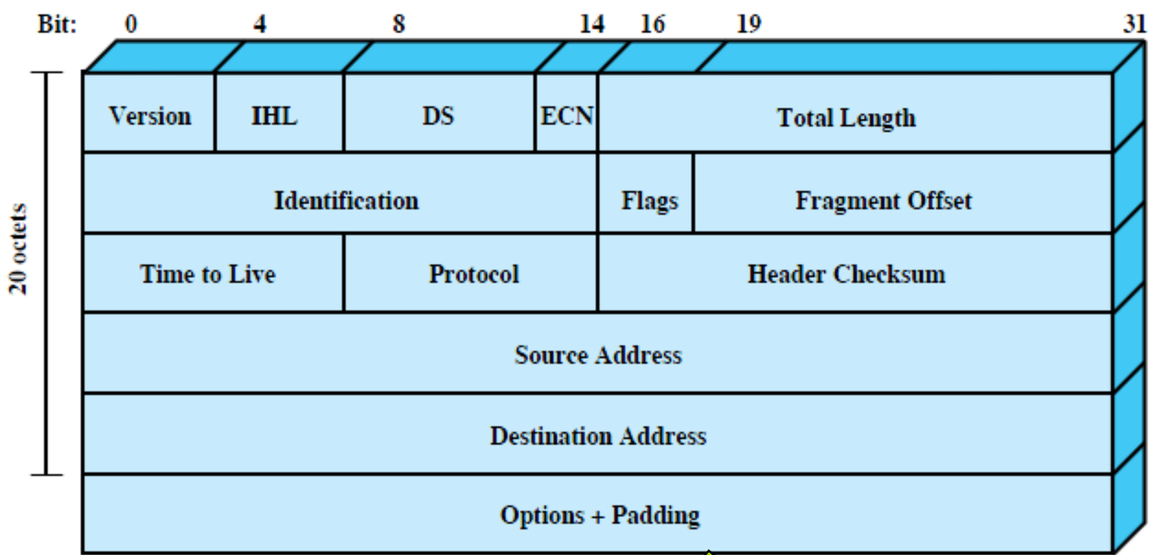
- A receiver cannot know if an incoming fragment is the result of one router fragmenting a datagram or multiple routers fragmenting fragments
- Fragmenting fragments results in subfragments
- Having subfragments requires
  - The receiver to perform reassembly for subfragments first
  - More processing is required (more CPU time)

Why Change IPv4 Addressing?

- Address space exhaustion
  - Two level addressing (network and host) wastes space
  - Network addresses used even if not connected to Internet
  - Growth of networks and the Internet
  - Extended use of TCP/IP
  - Single address per host
- Requirements for new types of service

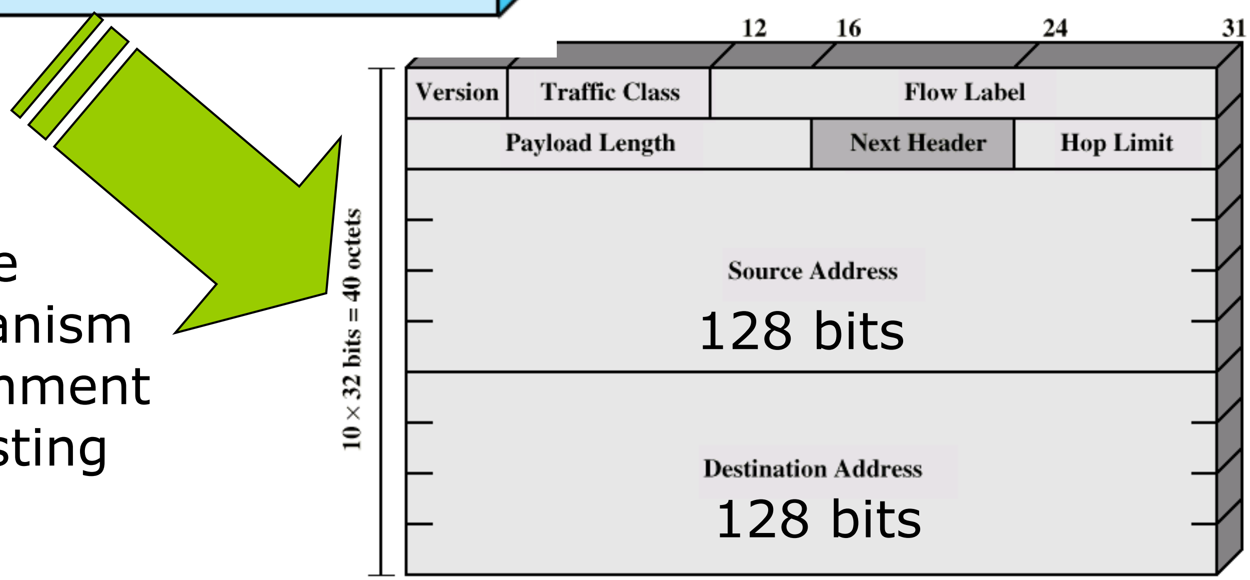
http://www.changeipaddress.org/how-to-change-ip-address.php

IP v6 Header vs. IPV4

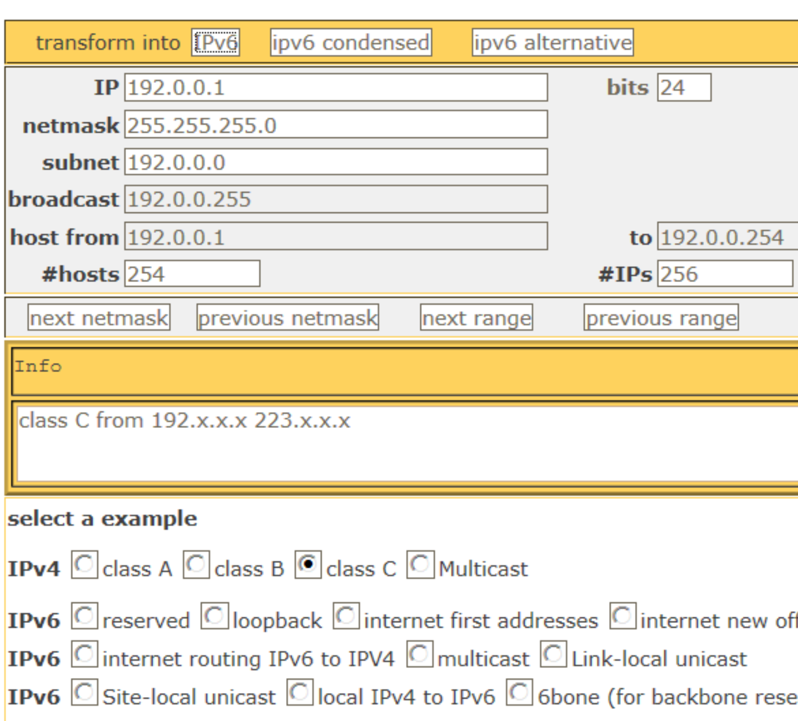


Note:
IPv5 used for Stream Protocol- IP-layer protocol that provides end-to-end guaranteed service across a network.

Features:
Extended address space
Improved option mechanism
Dynamic address assignment
Multicasting and anycasting
Flow routing

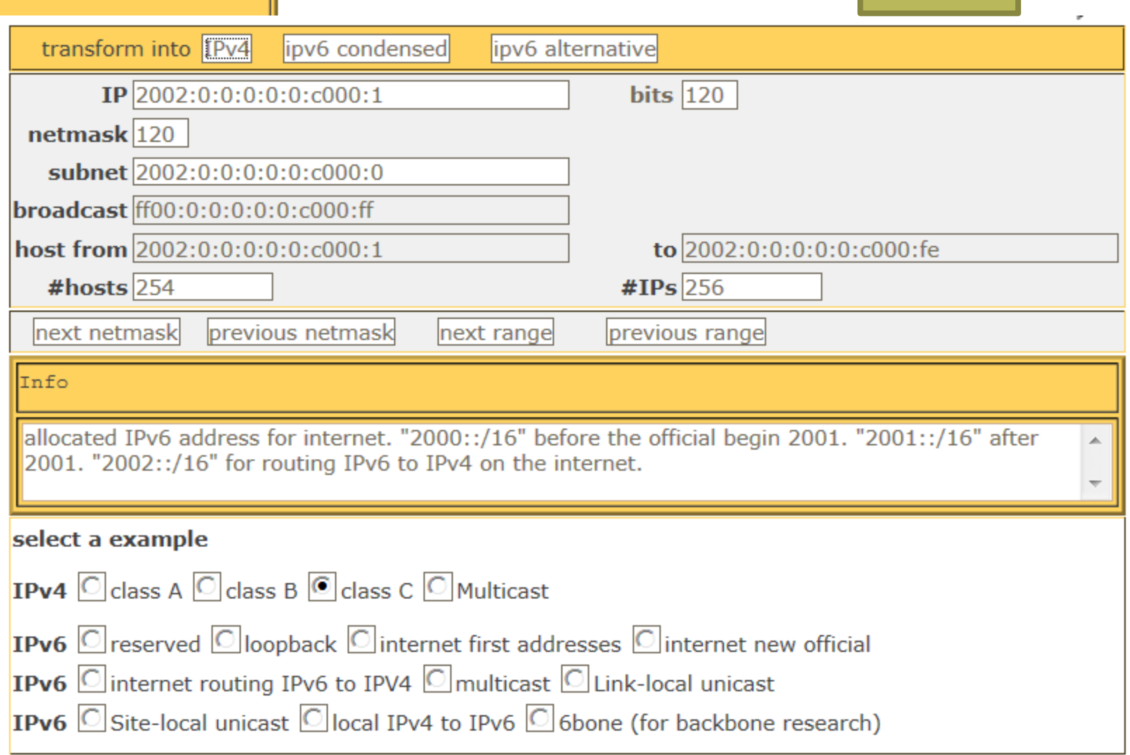


Converting IPv4 to IPv6



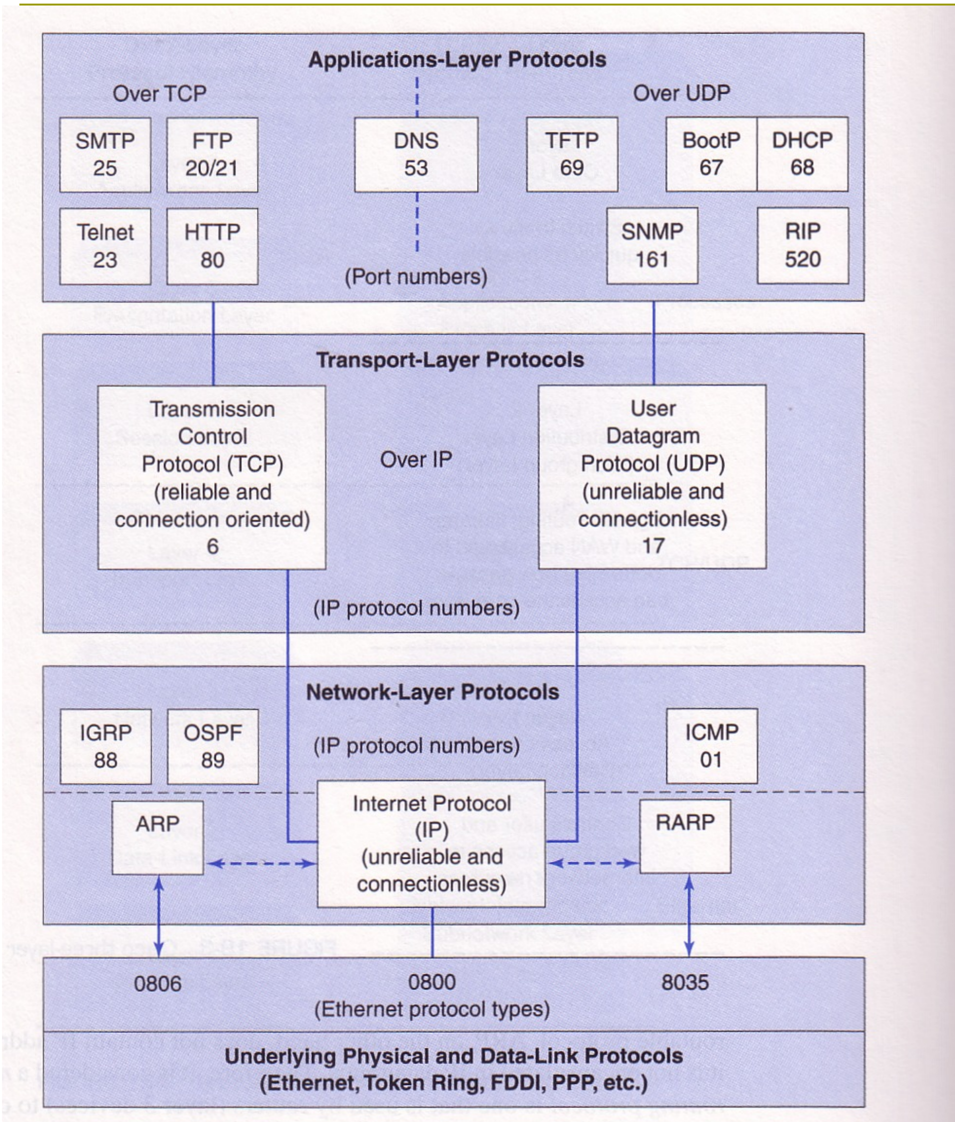
IPv4

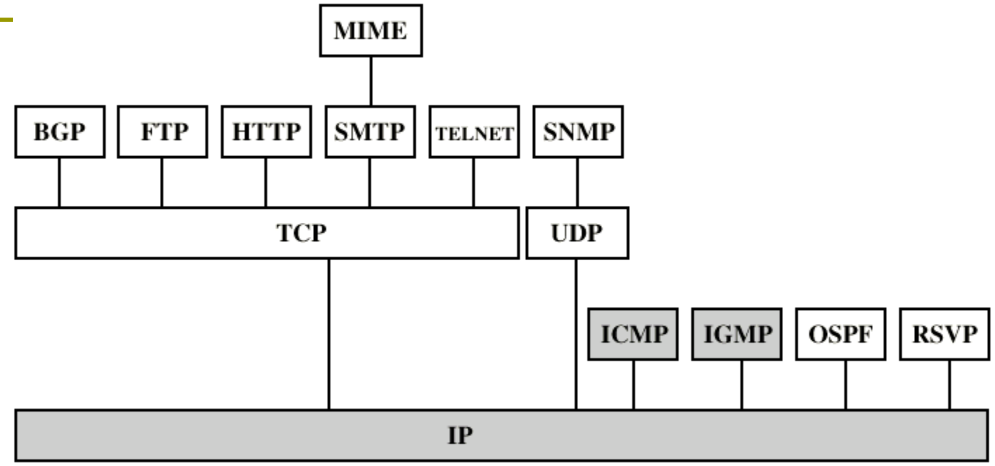
IPv6



http://www.subnetonline.com/pages/subnet-calculators/ipv4-to-ipv6-converter.php

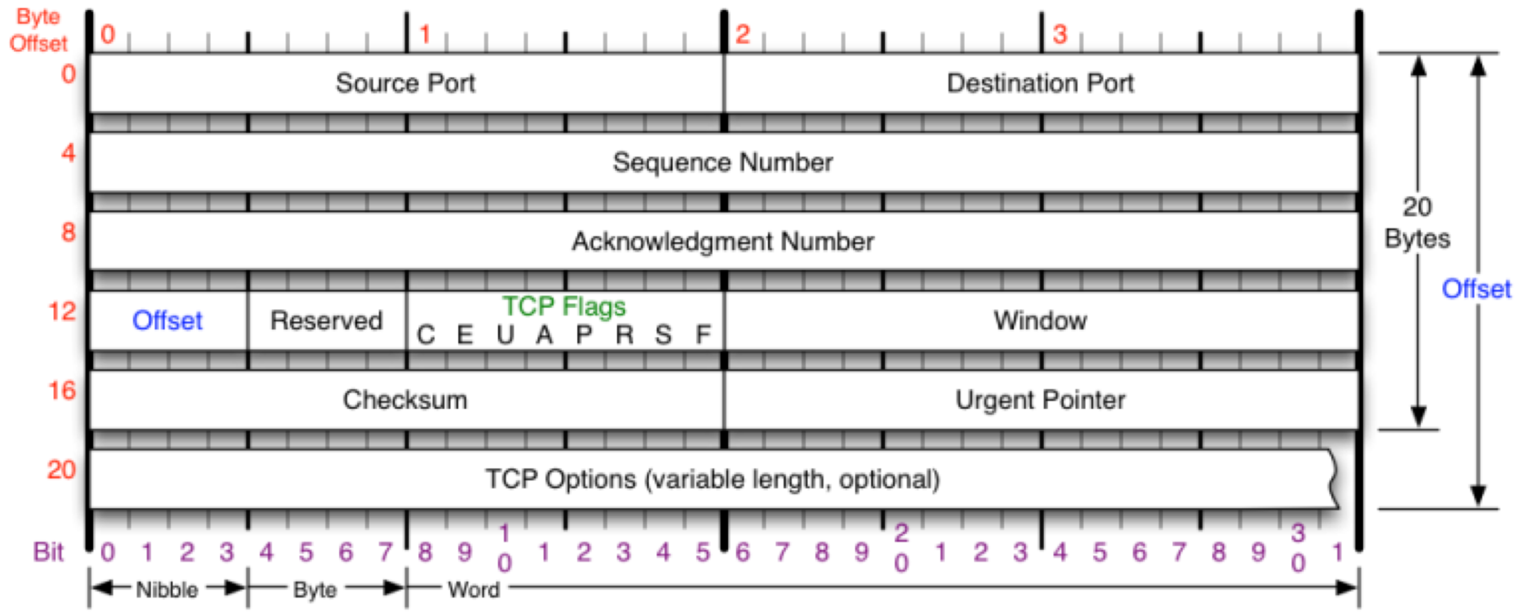
TCP/IP Stack Protocol





- Bridge
  - IS used to connect two LANs using similar LAN protocols
  - Address filter passing on packets to the required network only
  - OSI layer 2 (Data Link)
- Router
  - Connects two (possibly dissimilar) networks
  - Uses internet protocol present in each router and end system
  - OSI Layer 3 (Network)

TCP Header



TCP Flags

C E U A P R S F

Congestion Window
- C 0x80 Reduced (CWR)
- E 0x40 ECN Echo (ECE)
- U 0x20 Urgent
- A 0x10 Ack
- P 0x08 Push
- R 0x04 Reset
- S 0x02 Syn
- F 0x01 Fin

Congestion Notification

ECN (Explicit Congestion Notification). See RFC 3168 for full details, valid states below.

Packet State	DSB	ECN bits
Syn	0 0	1 1
Syn-Ack	0 0	0 1
Ack	0 1	0 0
No Congestion	0 1	0 0
No Congestion	1 0	0 0
Congestion	1 1	0 0
Receiver Response	1 1	0 1
Sender Response	1 1	1 1

TCP Options

0 End of Options List
1 No Operation (NOP, Pad)
2 Maximum segment size
3 Window Scale
4 Selective ACK ok
8 Timestamp

Checksum

Checksum of entire TCP segment and pseudo header (parts of IP header)

Offset

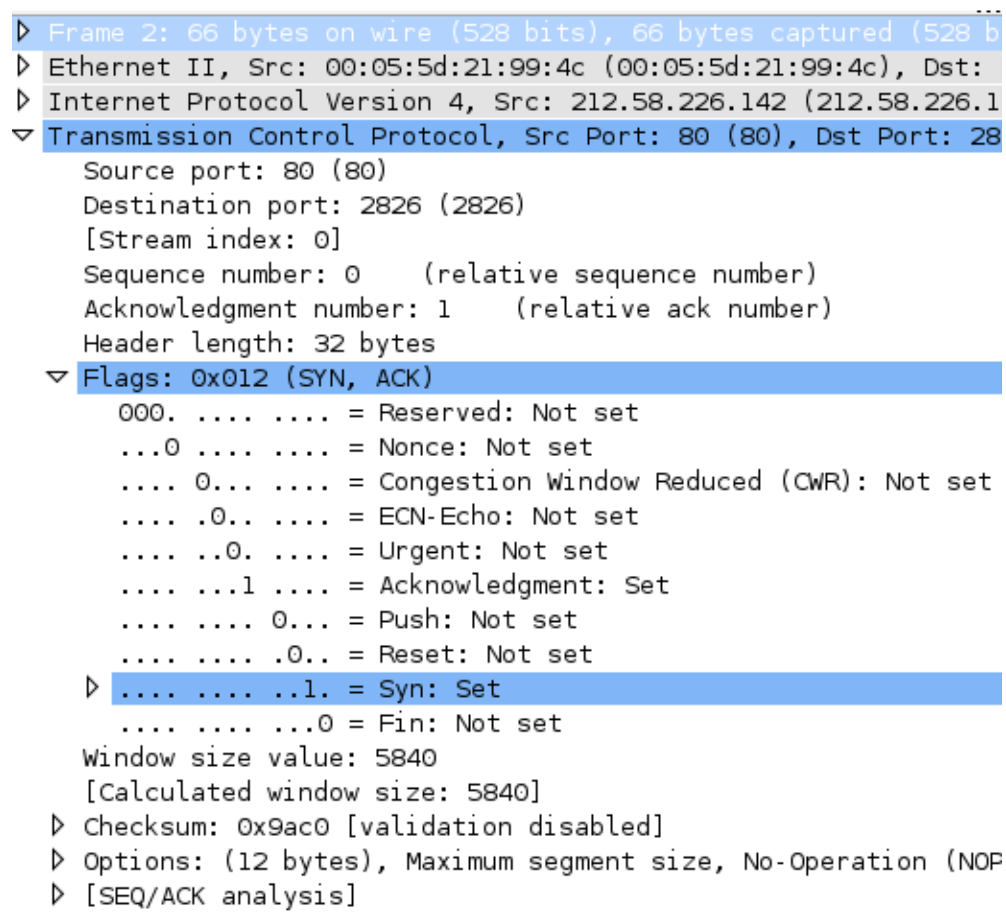
Number of 32-bit words in TCP header, minimum value of 5. Multiply by 4 to get byte count.

RFC 793

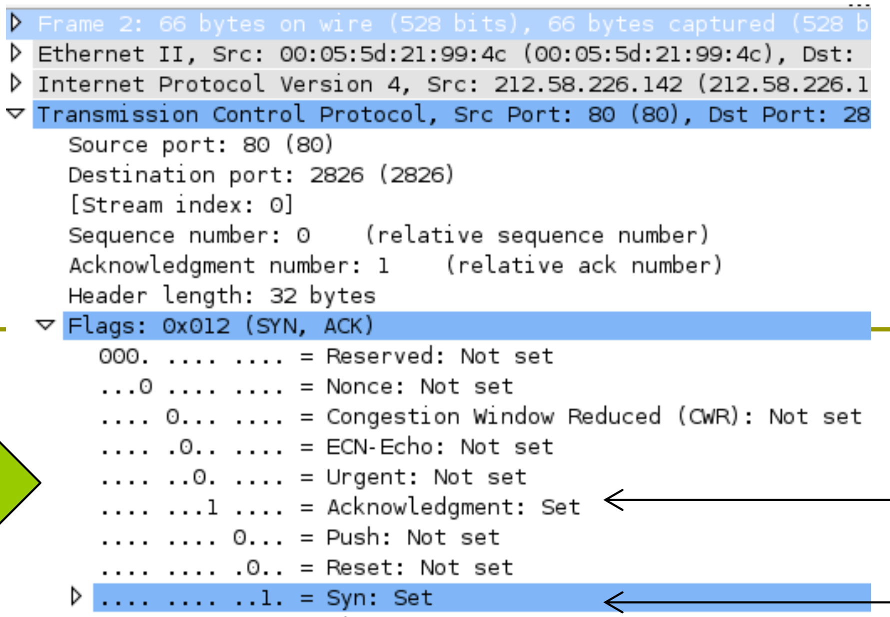
Please refer to RFC 793 for the complete Transmission Control Protocol (TCP) Specification.

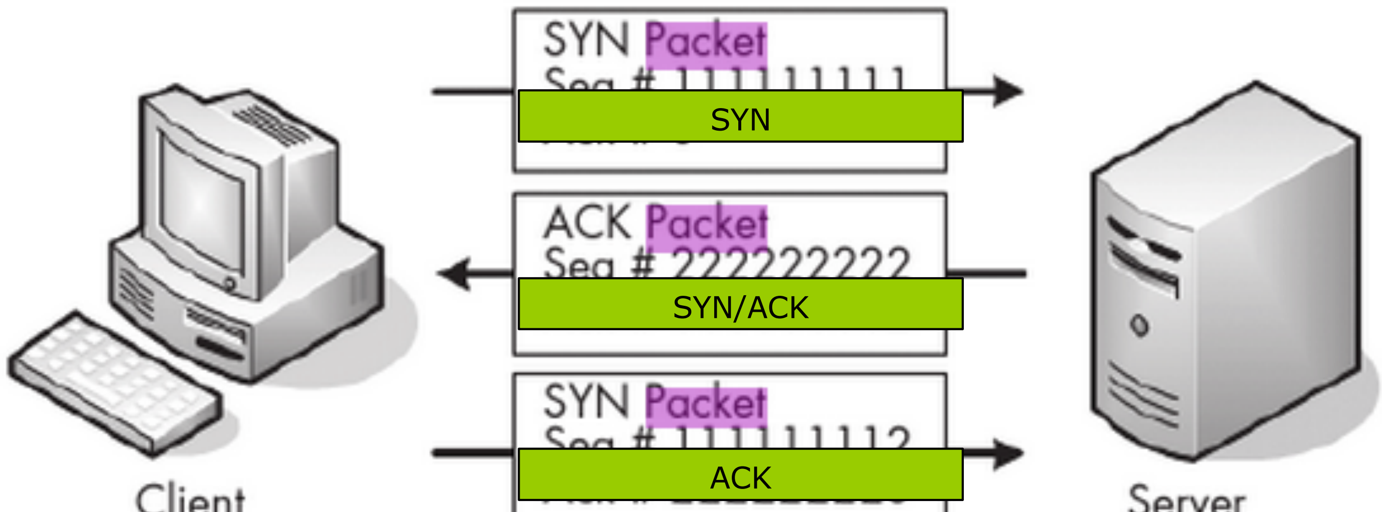
TCP

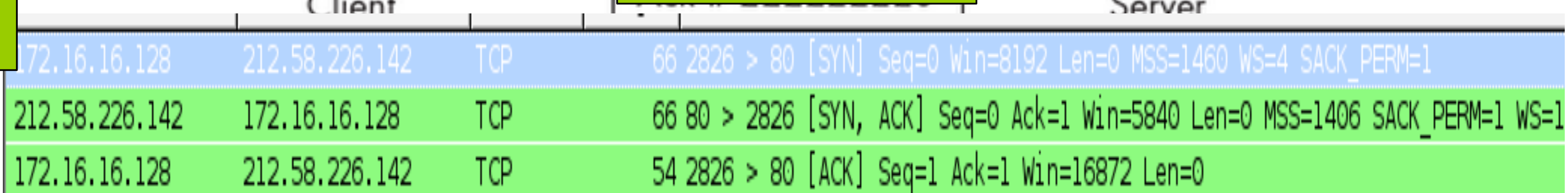
- Used for reliability (RFC 793)
  - Data sequencing
  - Error recovery
  - Built-in error checking
- Layer 4 OSI model
- Contains source/Destination PORT
- Connection Oriented
  - TCP Handshake (3-way setup)
  - TCP Teardown (4-way teardown)
  - TCP Reset



TCP Handshack 3-Way

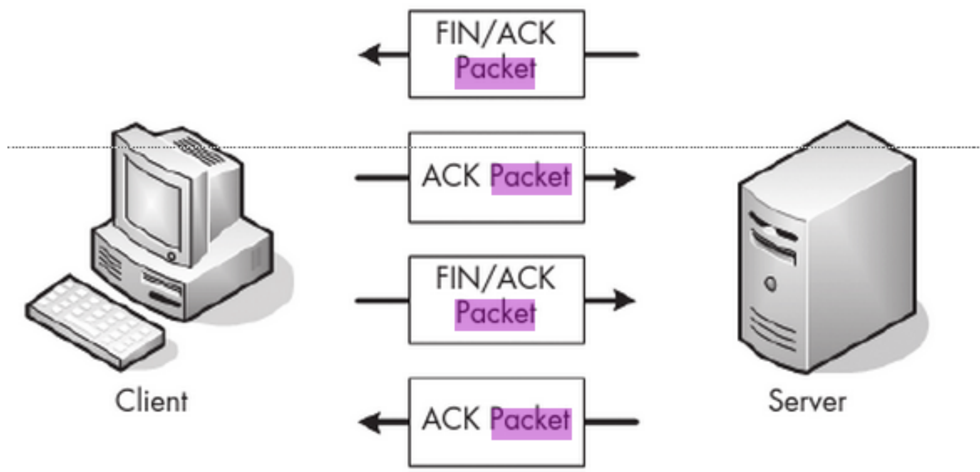






TCP Termination 4-Way

No.	Time	Source	Destination	Protocol	Info
40	17.905747	65.208.228.223	145.254.160.237	TCP	http > 3372 [FIN, ACK] Seq=18365 Ack=480 Win=6432 Len=0
41	17.905747	145.254.160.237	65.208.228.223	TCP	3372 > http [ACK] Seq=480 Ack=18366 Win=9236 Len=0
42	30.063228	145.254.160.237	65.208.228.223	TCP	3372 > http [FIN, ACK] Seq=480 Ack=18366 Win=9236 Len=0
43	30.393704	65.208.228.223	145.254.160.237	TCP	http > 3372 [ACK] Seq=18366 Ack=481 Win=6432 Len=0

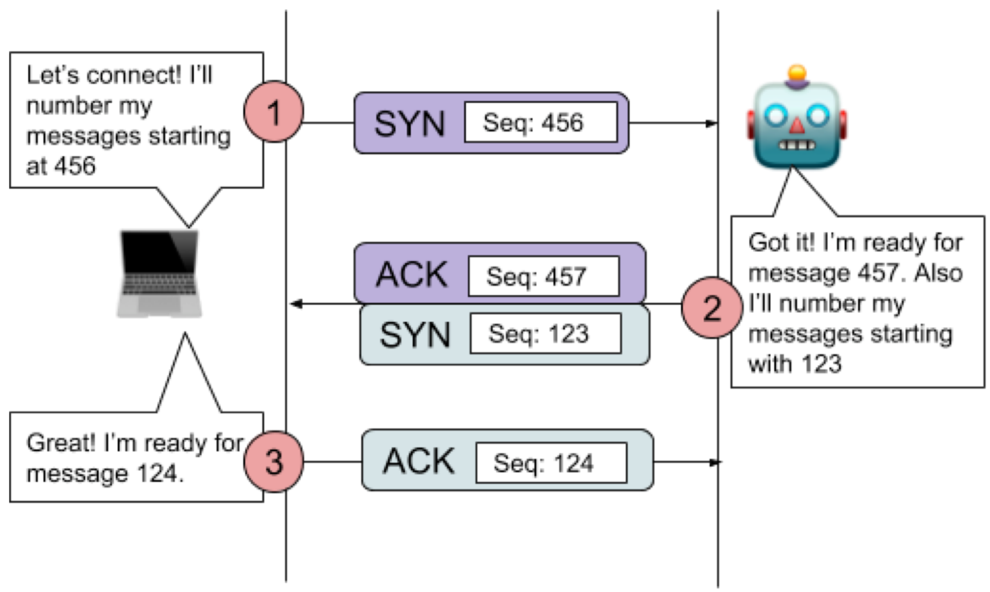


Sequence number and ACK number

Main reason for SEQ and ACK is to make sure packets are not LOST Or DUPLICATED!

TCP Termination
4-Way

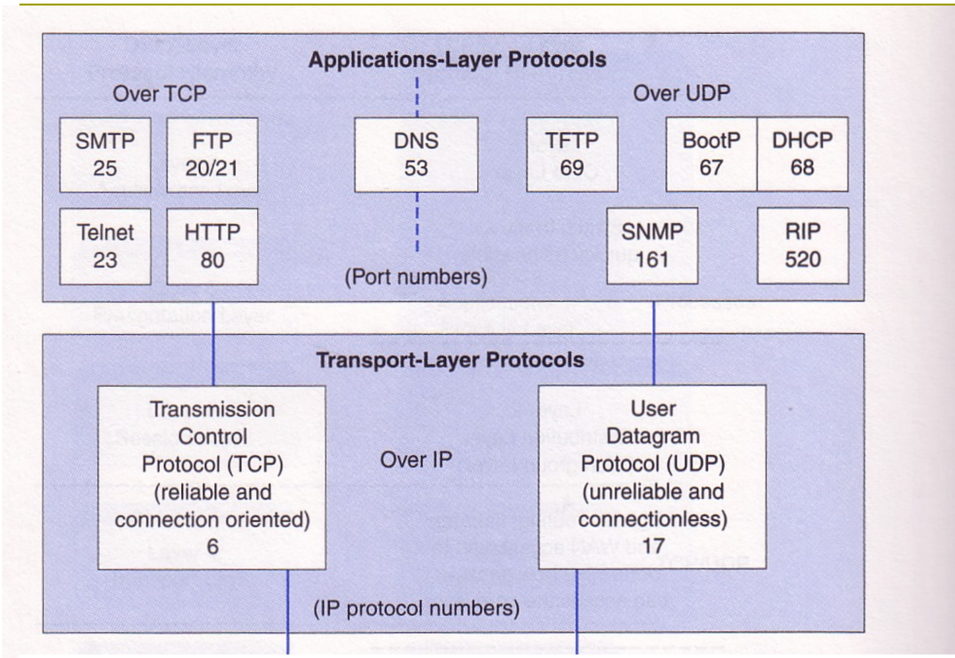
No.	Time	Source	Destination	Protocol	Info
40	17.905747	65.208.228.223	145.254.160.237	TCP	http > 3372 [FIN, ACK] Seq=18365 Ack=480 Win=6432 Len=0
41	17.905747	145.254.160.237	65.208.228.223	TCP	3372 > http [ACK] Seq=480 Ack=18366 Win=9236 Len=0
42	30.063228	145.254.160.237	65.208.228.223	TCP	3372 > http [FIN, ACK] Seq=480 Ack=18366 Win=9236 Len=0
43	30.393704	65.208.228.223	145.254.160.237	TCP	http > 3372 [ACK] Seq=18366 Ack=481 Win=6432 Len=0

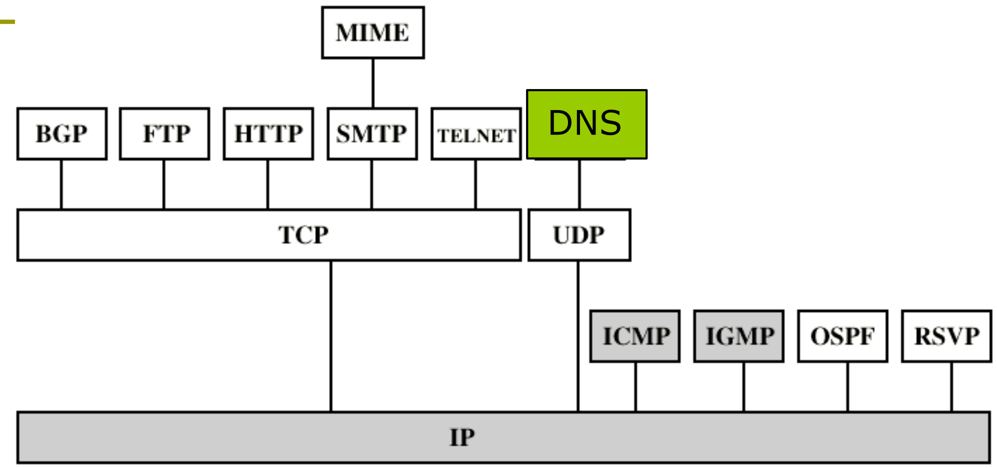


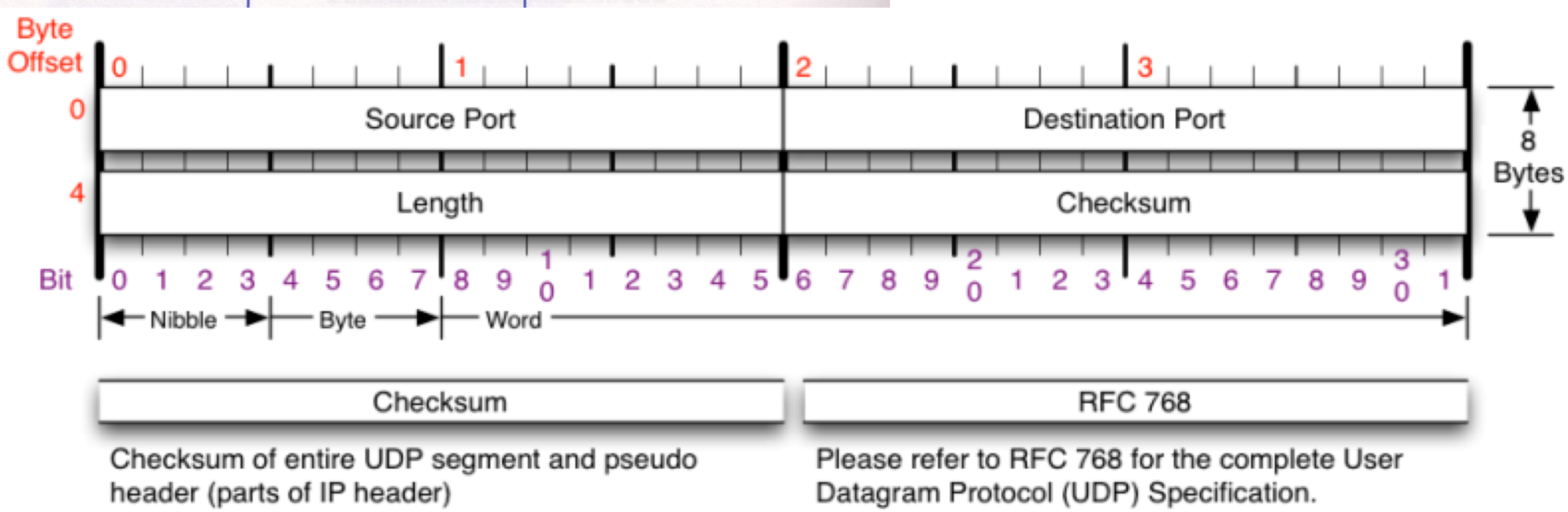
Sequence number and ACK number

Main reason for SEQ and ACK is to make sure packets are not LOST Or DUPLICATED!
Why not 2-way Handshake? Because each side has a different SEQ number!
Why not start with ZERO SEQ number? Then there could be confusion with other connections!

UDP/IP Stack Protocol







Comparing TCP and UDP

UDP v/s TCP		
Characteristics/ Description	**UDP**	**TCP**
General Description	Simple High speed low functionality "wrapper" that interface applications to the network layer and does little else	Full-featured protocol that allows applications to send data reliably without worrying about network layer issues.
Protocol connection Setup	Connection less data is sent without setup	Connection-oriented; Connection must be Established prior to transmission.
Data interface to application	Message base-based is sent in discrete packages by the application.	Stream-based; data is sent by the application with no particular structure
Reliability and Acknowledgements	Unreliable best-effort delivery without acknowledgements	Reliable delivery of message all data is acknowledged.
Retransmissions	Not performed. Application must detect lost data and retransmit if needed.	Delivery of all data is managed, and lost data is retransmitted automatically.
Features Provided to Manage flow of Data	None	Flow control using sliding windows; window size adjustment heuristics; congestion avoidance algorithms
Overhead	Very Low	Low, but higher than UDP
Transmission speed	Very High	High but not as high as UDP
Data Quantity Suitability	Small to moderate amounts of data.	Small to very large amounts of data.

There is no SYNC/ACK in UDP

There is no ACK

Using the requested Window Size

Due to need to have ACK

Comparing UDP and TCP

TCP	UDP
Keeps track of lost packets. Makes sure that lost packets are re-sent	Doesn't keep track of lost packets
Adds sequence numbers to packets and reorders any packets that arrive in the wrong order	Doesn't care about packet arrival order
Slower, because of all added additional functionality	Faster, because it lacks any extra features
Requires more computer resources, because the OS needs to keep track of ongoing communication sessions and manage them on a much deeper level	Requires less computer resources
Examples of programs and services that use TCP: - HTTP - HTTPS - FTP - Many computer games	Examples of programs and services that use UDP: - DNS - IP telephony - DHCP - Many computer games

References/Research

- What is TCP Fast Open?