TCP/IP

EN.600.444/644

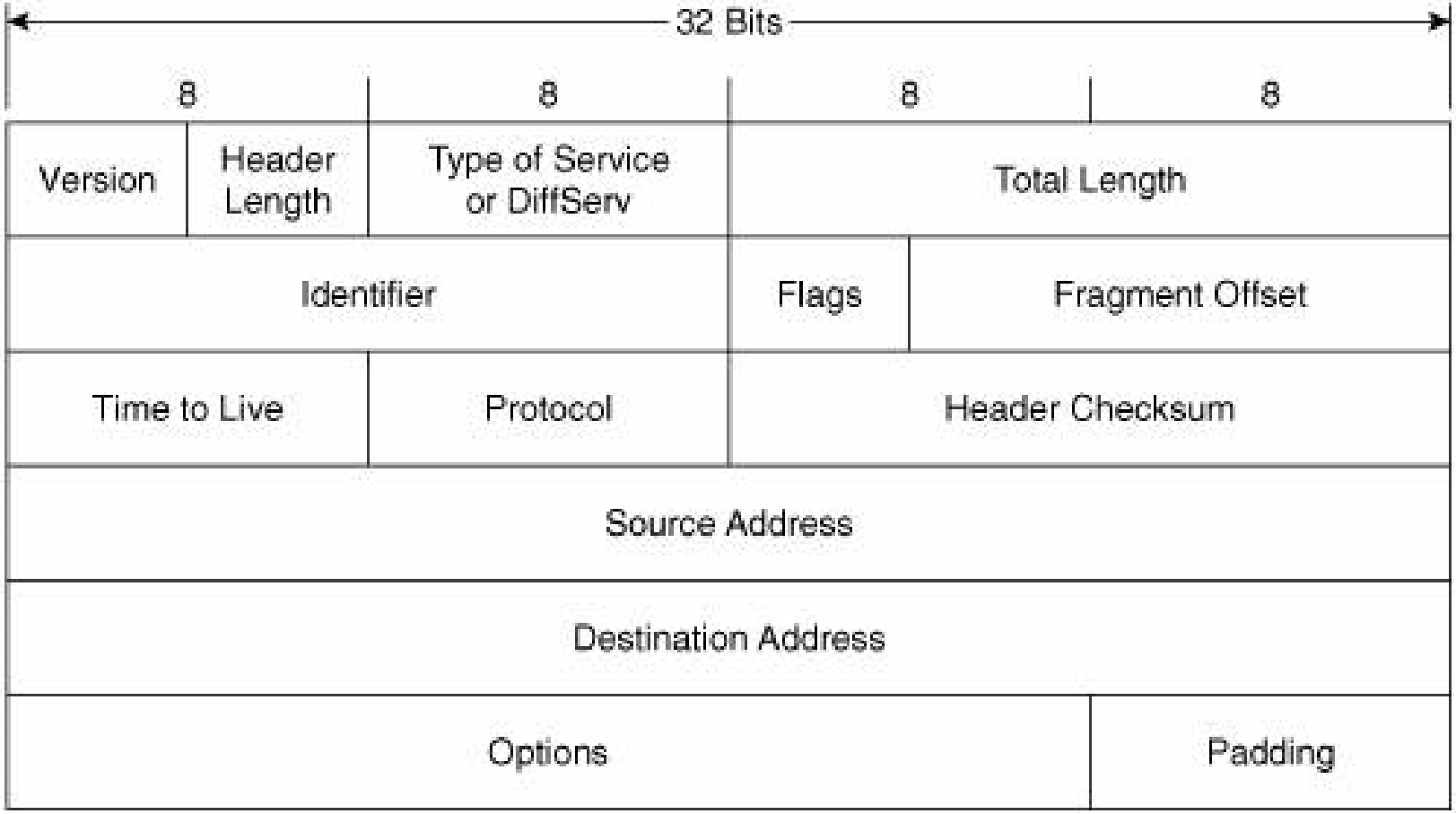
Fall 2019

Dr. Seth James Nielson

IPV4 PROTOCOL

- Layer 3 Protocol
- Handles fragmentation and reassembly
  - Assumed that across multiple LANS, multiple MAC protocols
  - Each MAC protocol might have its own MTU
- Also, of course, includes the global IP address
  - Kind of global…

IPV4 HEADER


32 Bits
8
8
8
8
Version
Header Length
Type of Service or DiffServ
Total Length
Identifier
Flags
Fragment Offset
Time to Live
Protocol
Header Checksum
Source Address
Destination Address
Options
Padding

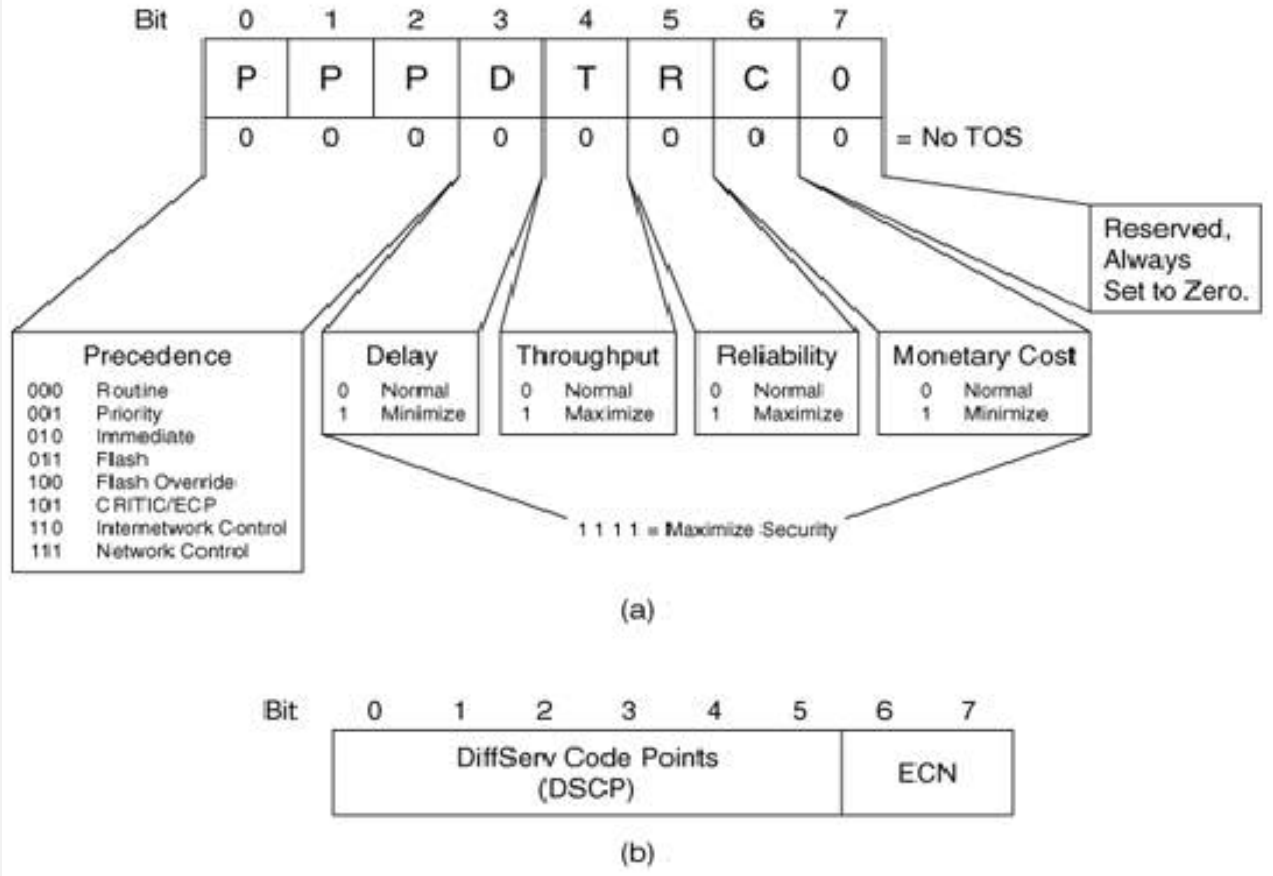
IPV4 HEADER FIELDS

- Version – 0100 (binary 4)
- Header Length – Length of header in 4-byte increments
- Total Length – Size of header and data in bytes (max 65535)
- Identification – for recognizing fragments
- Flags
  - Reserved. Always 0
  - Don't fragment
  - More fragments

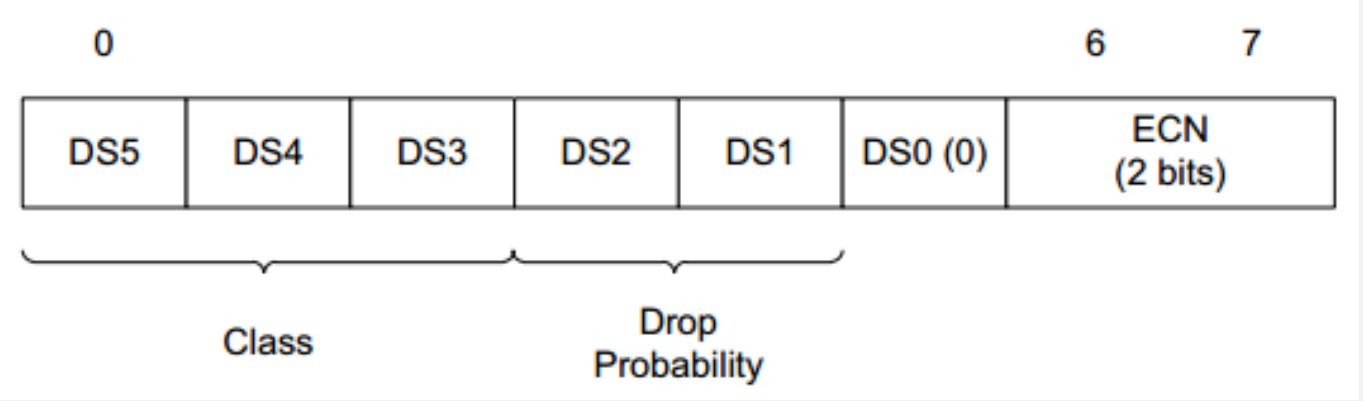
MORE IPV4 HEADER FIELDS

- Fragment Offset – Offset in original datagram
- TTL – Counter to prevent infinite routing
- Protocol – Information about upper layer (17=UDP, 6=TCP)
- Header Checksum
  - Recomputed each hop (because of TTL changes)
  - Checksum field itself always presumed to be 0

ORIGINAL TOS



DIFFERENTIATED SERVICES



ECN

- Explicit Congestion Notification
- When enabled, indicates congestion without dropping
- Not all hardware/software supports ECN

IPV4 ADDRESSES

- IPv4 Addresses are a.b.c.d where each is between 0-255
- In actuality, just a 32-bit number (“four octets”)
  - 192.0.2.235
  - 3221226219
  - 0xC00002EB (0xC0.0x00.0x02.0xEB)
- Private Networks:
  - 10.0.0.0
  - 172.16.0.0
  - 192.168.0.0

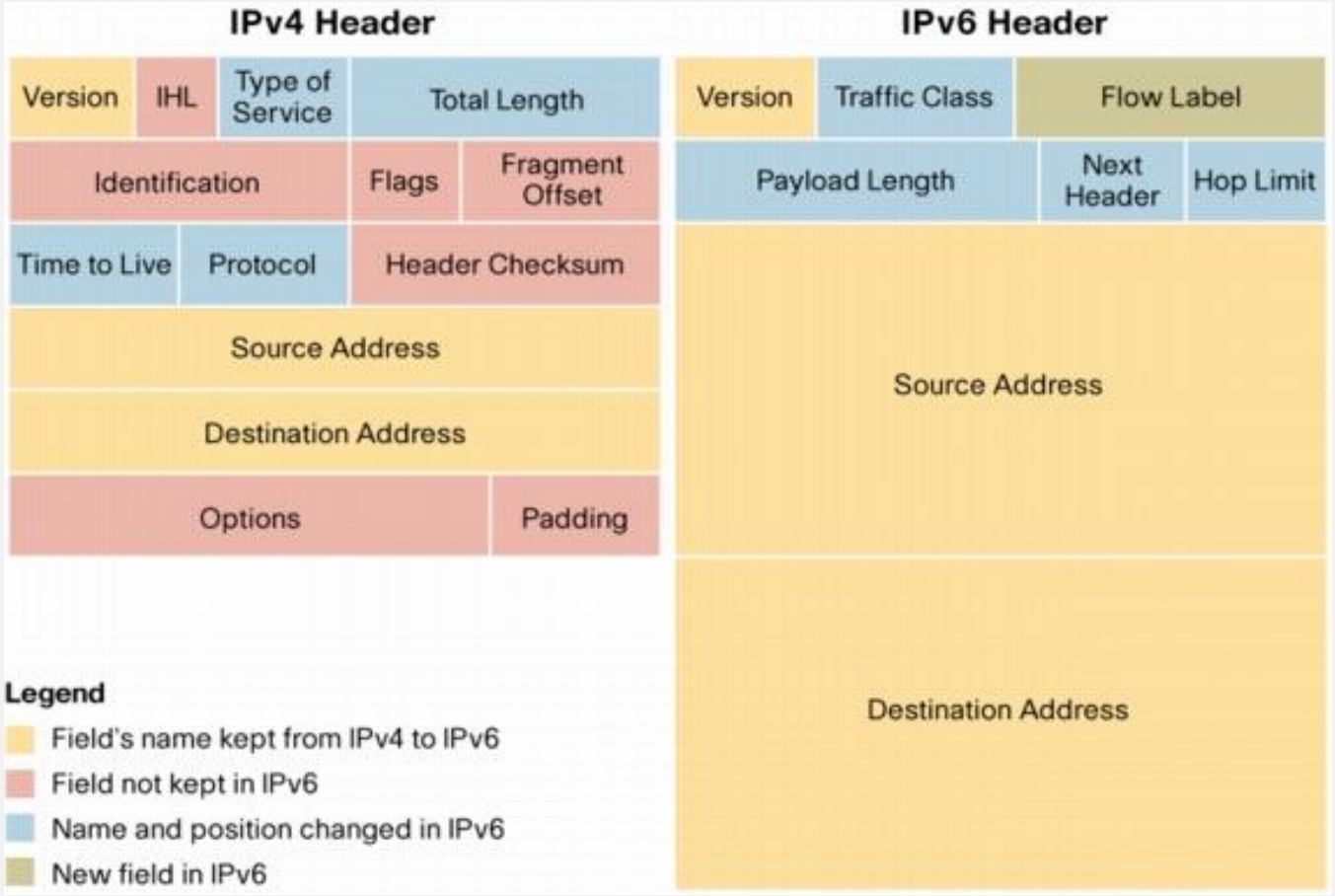
FRAGMENTATION

- If packet size > MTU, and DNF is 0
  - Each fragment gets its own size
  - Each fragment except the last gets MF set to 1
  - Fragment offset is location in the original packet
  - Identification field is unique identifier of original datagram
- Reassembly
  - Use src, dst, protocol, and identification to identify fragments
  - Use offset to store data in reassembly buffer
  - Use MF = 0 to recognize end of reassembly

FRAGMENTATION ISSUES

- IPv4 Fragmentation had security issues
  - IP Fragmentation Overlap (overwrite a fragment)
  - Buffer full (too many incomplete fragments)
  - Fragment overrun
  - (Note that most of these are DoS, but some evasion)

IPV6

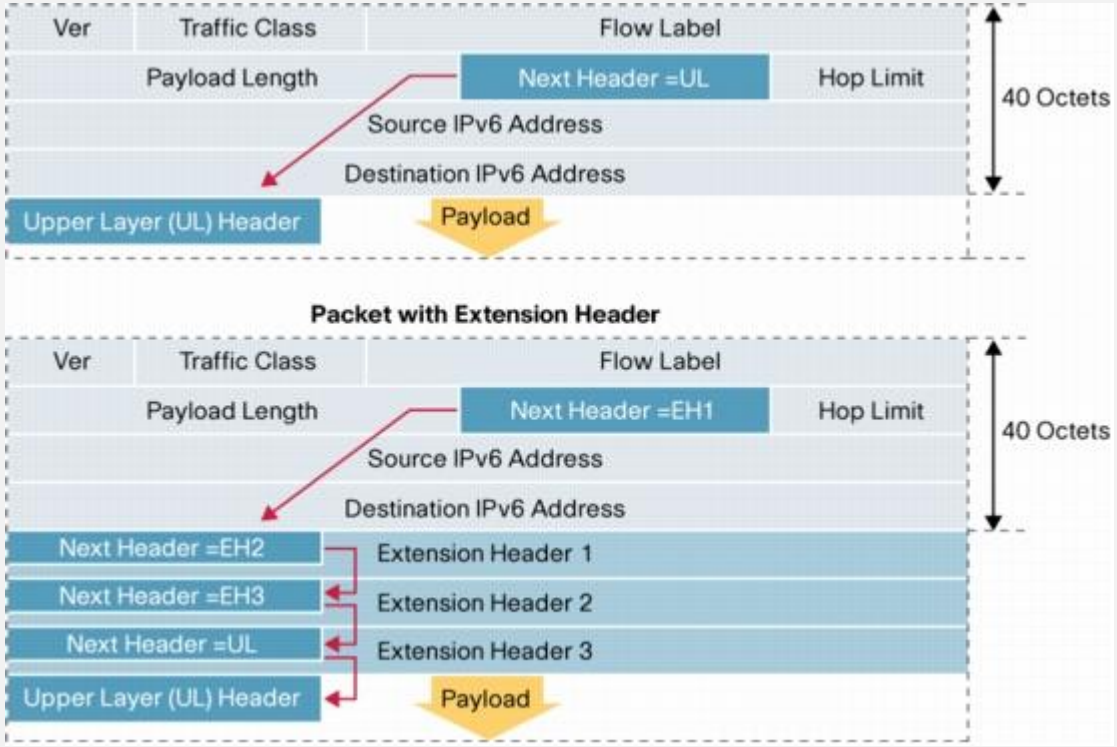

IPv4 Header
Version
IHL
Type of Service
Total Length
Identification
Flags
Fragment Offset
Time to Live
Protocol
Header Checksum
Source Address
Destination Address
Options
Padding
IPv6 Header
Version
Traffic Class
Flow Label
Payload Length
Next Header
Hop Limit
Source Address
Destination Address
Legend
Field's name kept from IPv4 to IPv6
Field not kept in IPv6
Name and position changed in IPv6
New field in IPv6

IPV6 HEADER FIELDS

- Version = 0110 (binary 6)
- Traffic Class: Differentiated services plus ECN
- Flow label: Hint for multiple outbound paths
- Payload length: Includes extension headers, in bytes
- Next header: Type of next extension or transport header
- Hop limit: Decrement by 1, discard if 0

EXTENSION HEADERS

- IPv6 is always 40 bytes for main header
- Can have additional headers:



COMMON EXTENSION HEADERS

Order	Header Type	Next Header Code
1	**Basic IPv6 Header**	-
2	**Hop-by-Hop Options**	0
3	**Destination Options (with Routing Options)**	60
4	**Routing Header**	43
5	**Fragment Header**	44
6	**Authentication Header**	51
7	**Encapsulation Security Payload Header**	50
8	**Destination Options**	60
9	**Mobility Header**	135

IPV6 FRAGMENTATION

- To deal with IPv4 Frag issues, ONLY SENDER can frag in IPv6
- Ergo, ***sender must know smallest MTU of path!***
- Path MTU Discovery (PMTUD)
  - If too big, send an ICMPv6 "packet too big" to sender
- Otherwise, max IPv6 packet size is 1,280 bytes.
- Uses a fragmentation extension header
  - Identification
  - MF, etc

IPV6 FRAG PROBLEMS

- Studies from 2014-present indicate IPv6 fragmentation fails
- About one-third of IPv6 hosts could not receive frags
- Many are concluding that IPv6 fragmentation is deprecated
- Maximum IPv6 packet size between 1280 and 1350
- See,
  - https://blog.apnic.net/2016/05/19/fragmenting-ipv6/
  - https://labs.apnic.net/?p=1033

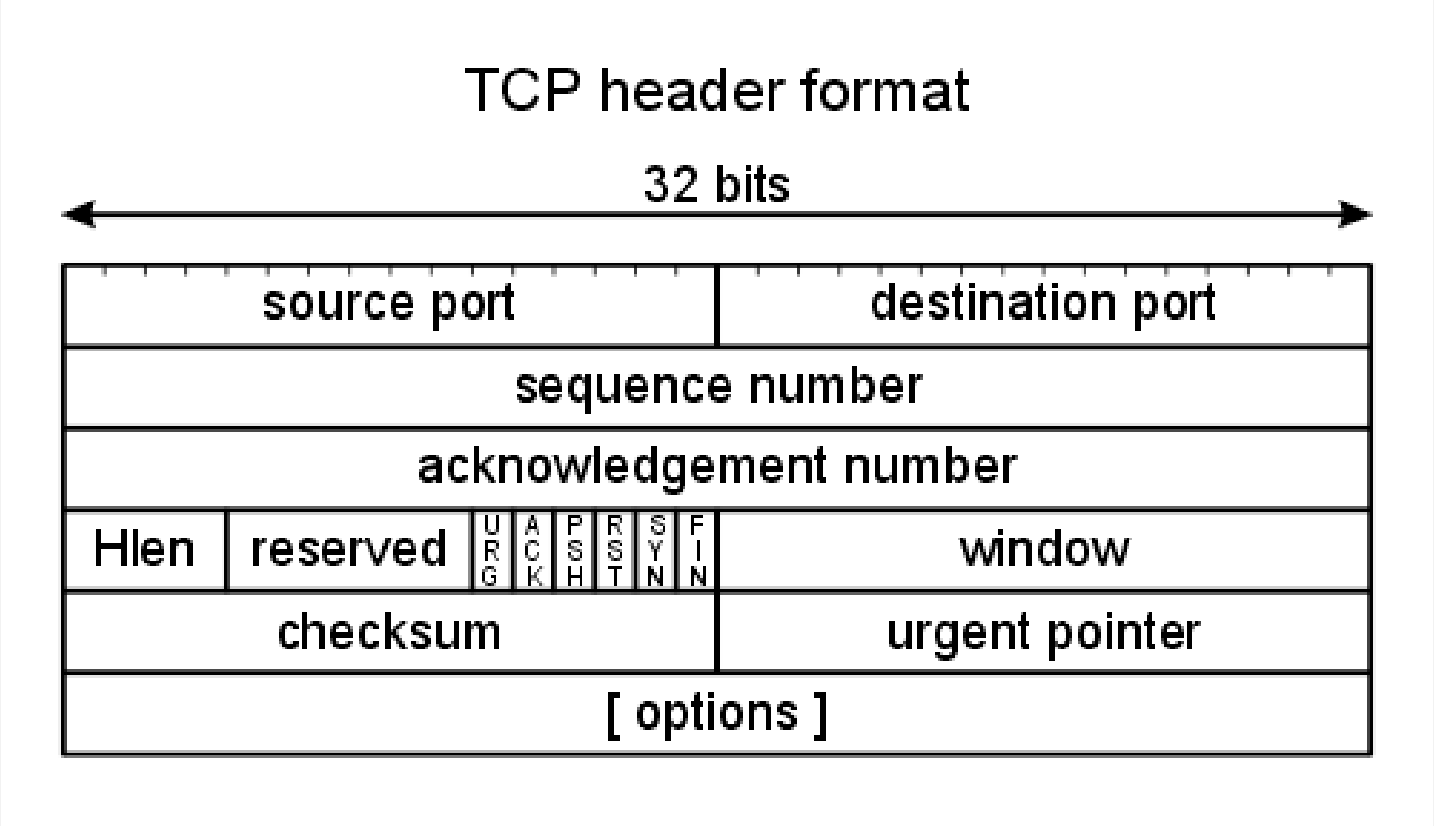
TCP PROTOCOL

- Layer 4 Protocol
- Multiplexing (ports)
- Reliable Delivery (ack/resend)
- Congestion control
- Units called *segments*

REVIEW

- Ethernet ***frame***
- IP ***packet***
- TCP ***segment***

TCP HEADER


TCP header format
32 bits
source port
destination port
sequence number
acknowledgement number
Hlen
reserved
URG
ACK
PSH
RST
SYN
FIN
window
checksum
urgent pointer
[options]

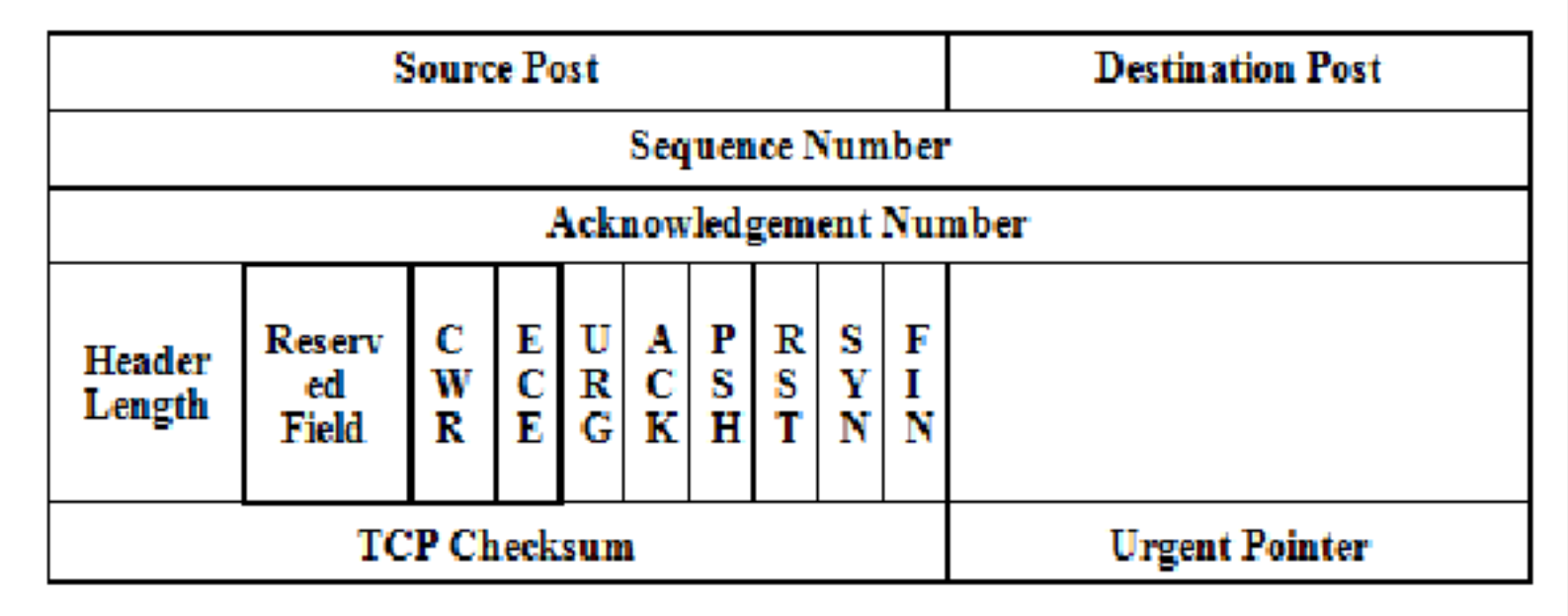
TCP HEADER FIELDS

- Source Port, Destination Port for multiplexing
- Sequence Number of the first data byte
- Acknowledgement Number of next expected seq. no.
- Hlen number of 32-byte words in the TCP header
- Window number of bytes willing to receive
- Checksum over the header and data

TCP FLAGS

- URG The URGENT POINTER field contains valid data
- ACK The acknowledgement number is valid
- PSH The receiver should pass this data to the application as soon as possible
- RST Reset the connection
- SYN Synchronize sequence numbers to initiate a connection.
- FIN Sender is finished sending data

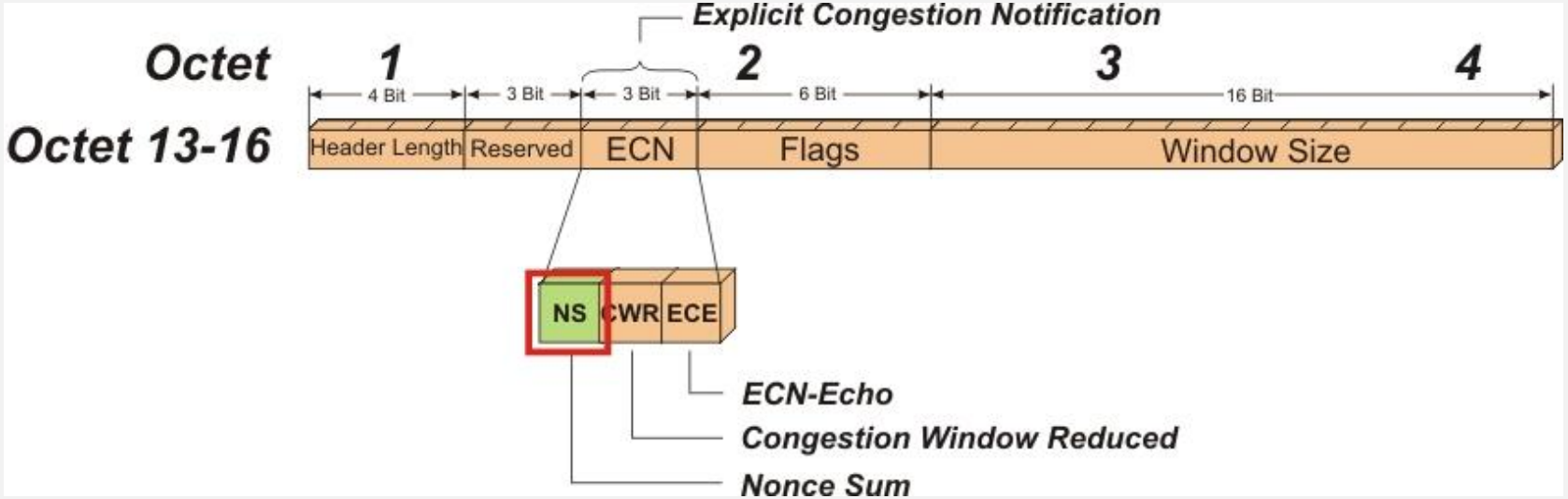
ECN FLAGS ADDED IN RFC 3168



TCP FLAGS (CONT)

- CWR – Acknowledges that congestion notification received
- ECN – indicates congestion notification via IP layer
  - (NOTE! Requires ECN capable IP layer!)
  - Sent until CWR received
- Only used if negotiated using TCP options during handshake

OPTIONAL NS FLAG



ONE-BIT NONCE

- NS is a parity bit used to catch changes to a packet
- Because it's only one bit, a cheater can guess it 50% right
- But, over repeated trials (frequent congestion) will get caught

TCP OPTIONS

- Header can be "extended" with options
- Each option can have up to three fields:
  - Option Type (1 byte)
  - Option Length (1 byte)
  - Option data (variable)
- Examples include
  - Selective acknowledgement
  - ECN

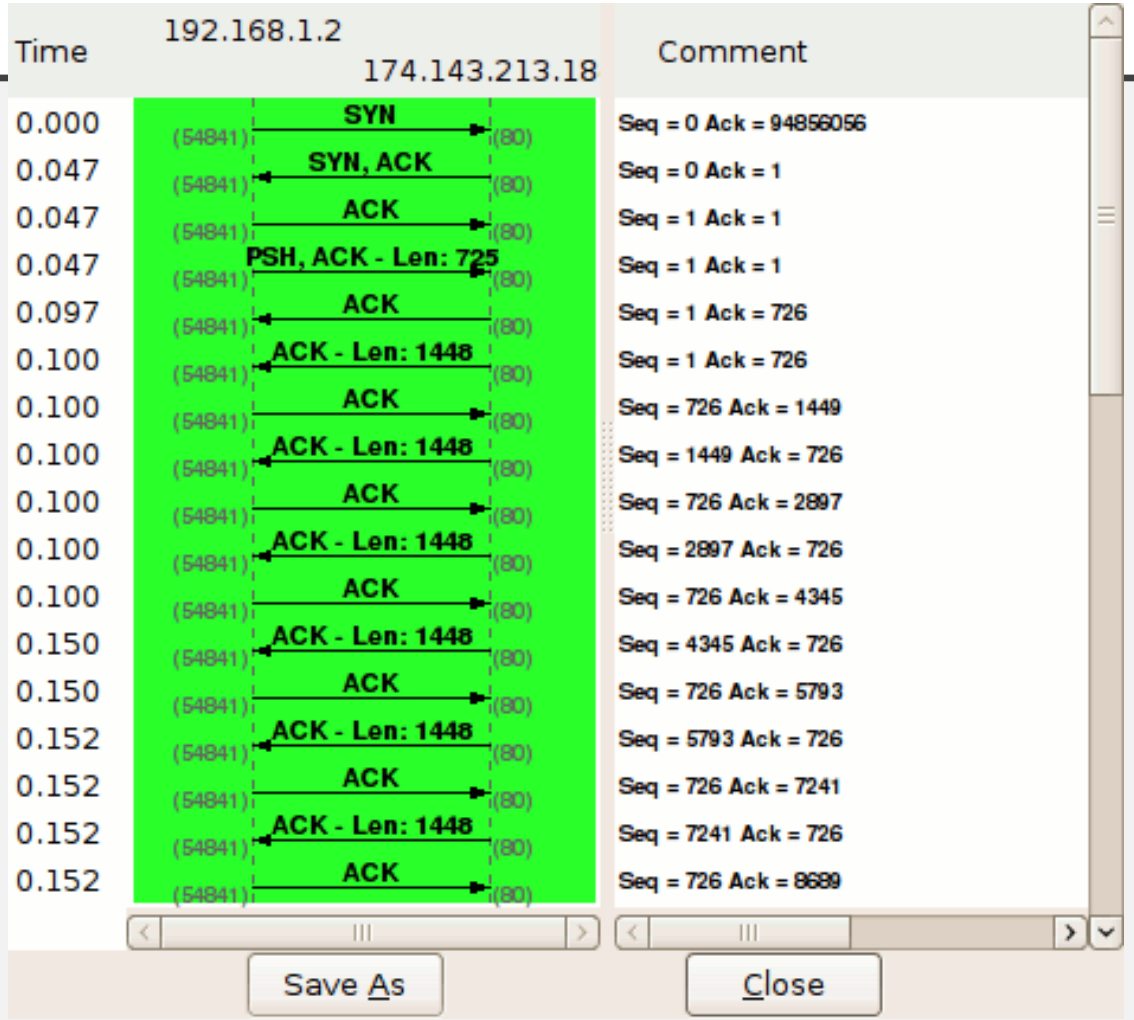
TCP SEQUENCE NUMBERS

- TCP header has a value for seq num and ack num every time
- SYN Sequence Number random between 0-4,294,967,295
- SYN Ack Num should be 0 (but any value should be ignored)
- SYN-ACK Seq Num also random
- SYN-ACK Ack Num is SYN Seq Num + 1
- (SYN-ACK) ACK Seq Num is SYN Seq Num + 1
- (SYN-ACK) ACK Ack Num is SYN-ACK Seq Num + 1

DATA SEQUENCE NUMBERS

- Technically, there is only one packet type in TCP
- Flags simply indicate how the values can be used
- Sequence number is set every time
- But only increased by the length of the data
- (or increased by +1 for SYN and FIN)
- Ack field indicates that the ACK number is valid

WIRESHARK TRACE



TCP SHUTDOWN

SECURITY

- Obviously, TCP is not designed with security in mind
  - No confidentiality!
  - No authentication!
  - No integrity (checksum is not cryptographic)
- No secure availability either!
  - End any connection with RST