Network Traffic Analysis

Objectives

- Discuss how computers abstract network connections into different layers and protocols.
- Explore tools for connecting to different networks and protocols.
- Explore different methods and reasons for spoofing different packets.

References

- https://python3-pwntools.readthedocs.io/
- https://netcat.sourceforge.net
- https://www.tcpdump.org
- https://nmap.org/book/tcpip-ref.html

What are packets ?

A packet is a unit of data that is transmitted between devices over a network. It contains two parts

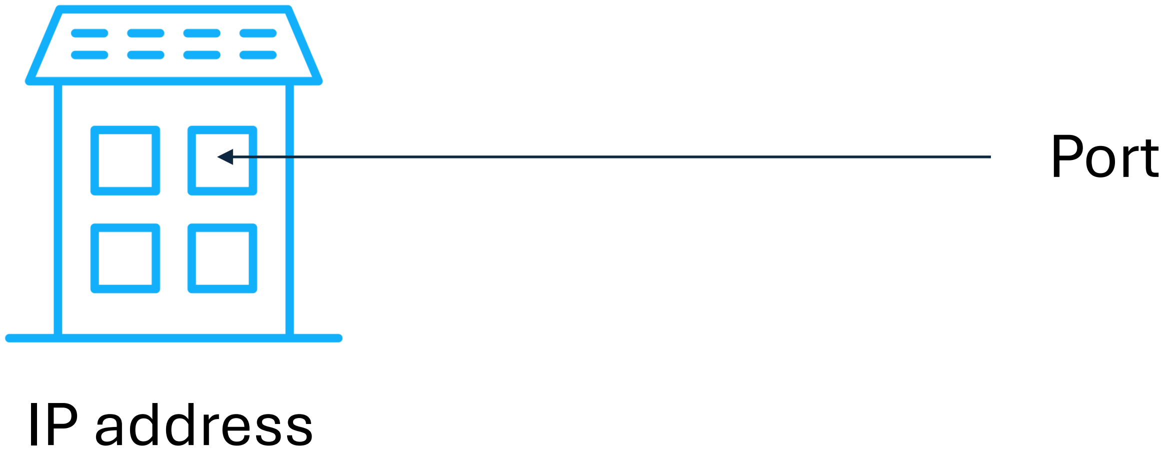
- **Header**: contains information about how to deliver the packet
  - Source address and port
  - Destination address and port
  - Protocol and options
- **Payload**
  - Actual data being transmitted

Header: IP addresses

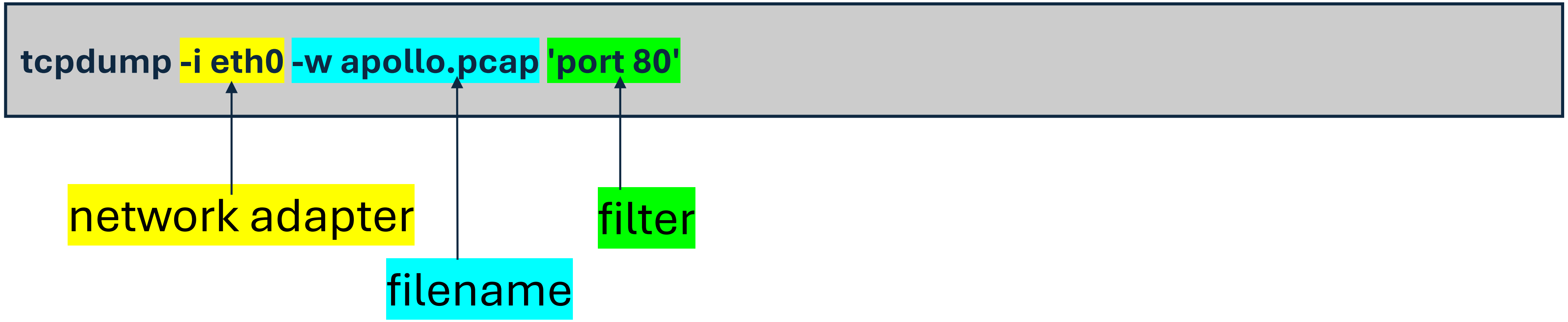
- An IP address allows us to describe the network address of a computer (kind of like a street address)
- For IPv4, it follows the format of 4 numbers (0-255) , separated by periods.

Header: Ports

- At the transport layer, we use the abstraction of a port to explain the location of a specific service (kind of like a window describes a specific room in a house.)
- For TCP and UDP there are 65,535 available ports.



Storing Capture In A PCAP



```
tcpdump -i eth0 -w apollo.pcap 'port 80'
```

network adapter

filename

filter

Let us store a pcap network capture our our traffic on the nginx webserver so we can download it and examine it.

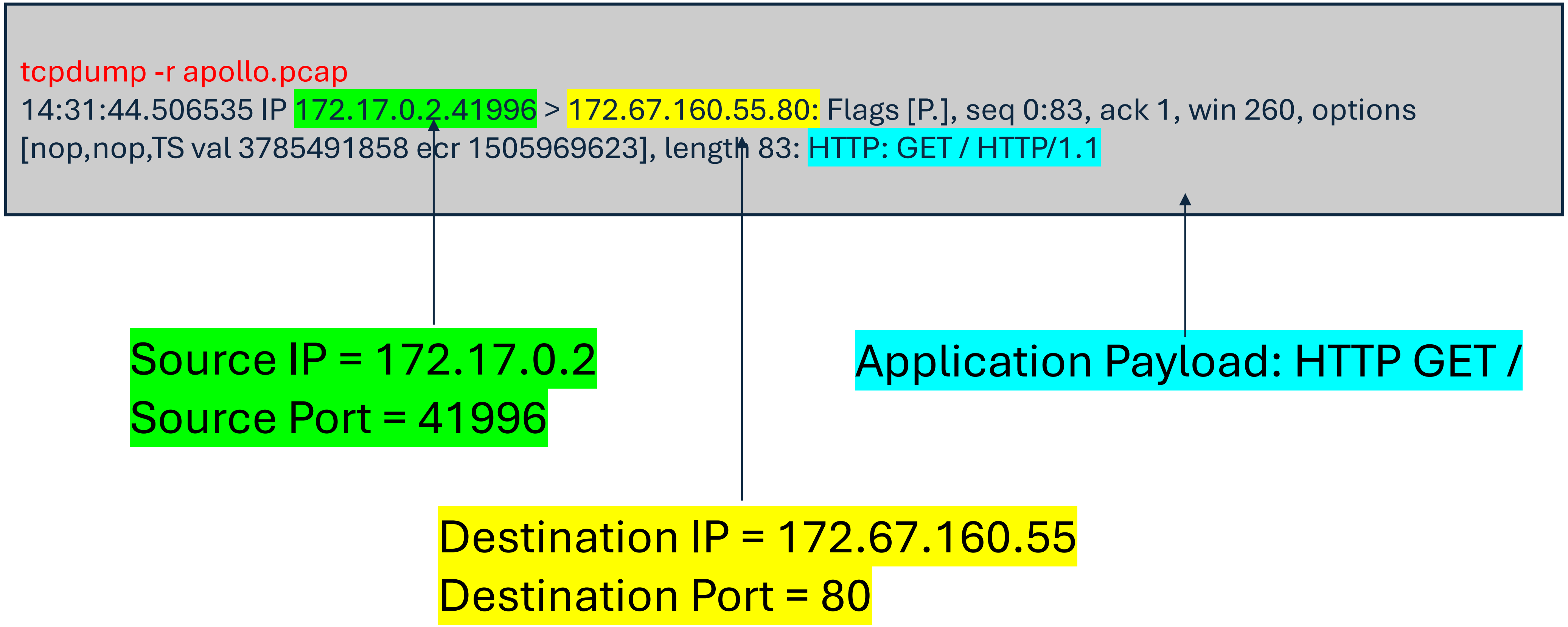
Captures: Berkley Packet Filters

```
tcpdump -i eth0 -w capture.pcap 'port 80'
```

filter

- Filters a stream of packets using primitives
  - type: **host**, **net**, **port,** and portrange
  - dir: direction of traffic (**dst, src**)
  - proto: matches a particular protocol (**tcp, udp, ip, dns, http)**
- *man pcap-filter* for more information about BPF

Examining Our First Packet



```
tcpdump -r apollo.pcap
14:31:44.506535 IP 172.17.0.2.41996 > 172.67.160.55.80: Flags [P.], seq 0:83, ack 1, win 260, options
[nop,nop,TS val 3785491858 ecr 1505969623], length 83: HTTP: GET / HTTP/1.1
```

Source IP = 172.17.0.2
Source Port = 41996

Application Payload: HTTP GET /

Destination IP = 172.67.160.55
Destination Port = 80

Header: Packet Encapsulation

IP Header (From 172.17.0.2; To 172.67.160.55)

172.17.0.2 -> 172.67.160.55
The IP Header contains the IP Source and Destination

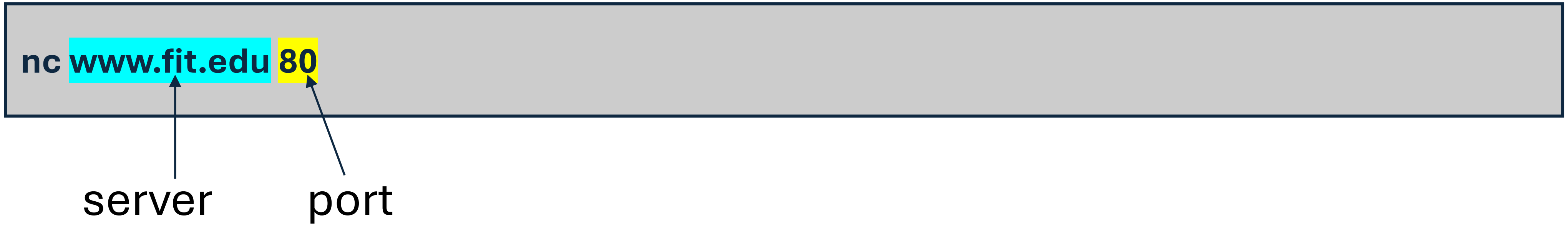
TCP Header

Port 41996 -> Port 80
The TCP Header contains the Source and Dest. ports

TCP Data (HTTP Application)

The specific application data
“HTTP GET /”

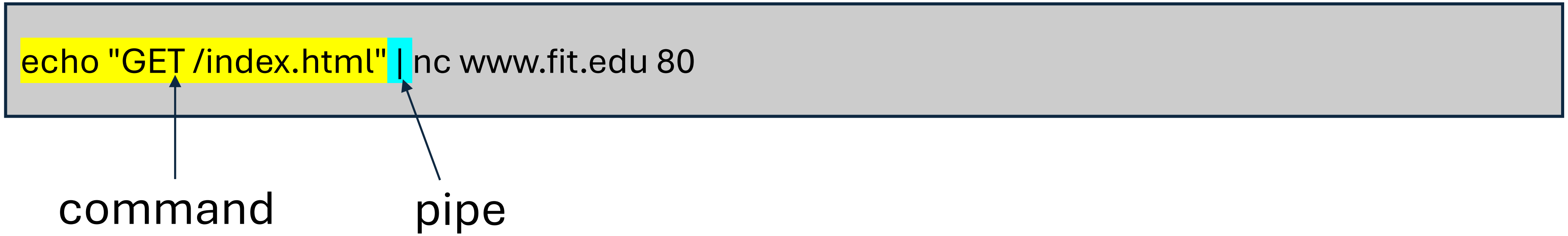
Netcat: The Swiss Army Networking Tool



```
nc www.fit.edu 80
```

- Netcat is a very versatile tool that allows us to connect to a server
- Allows us to interact with a network server by sending/receiving data
- The “swiss army knife” of networking tools

Piping Input Into Netcat



```
echo "GET /index.html" | nc www.fit.edu 80
```

command

pipe

- Remember we can use pipes to connect the output of one command to the input of the next command

- Here we echo GET /index.html so that when netcat connects to the server, it sends the command and reports the response.

PwnTools: Automating Our Connections

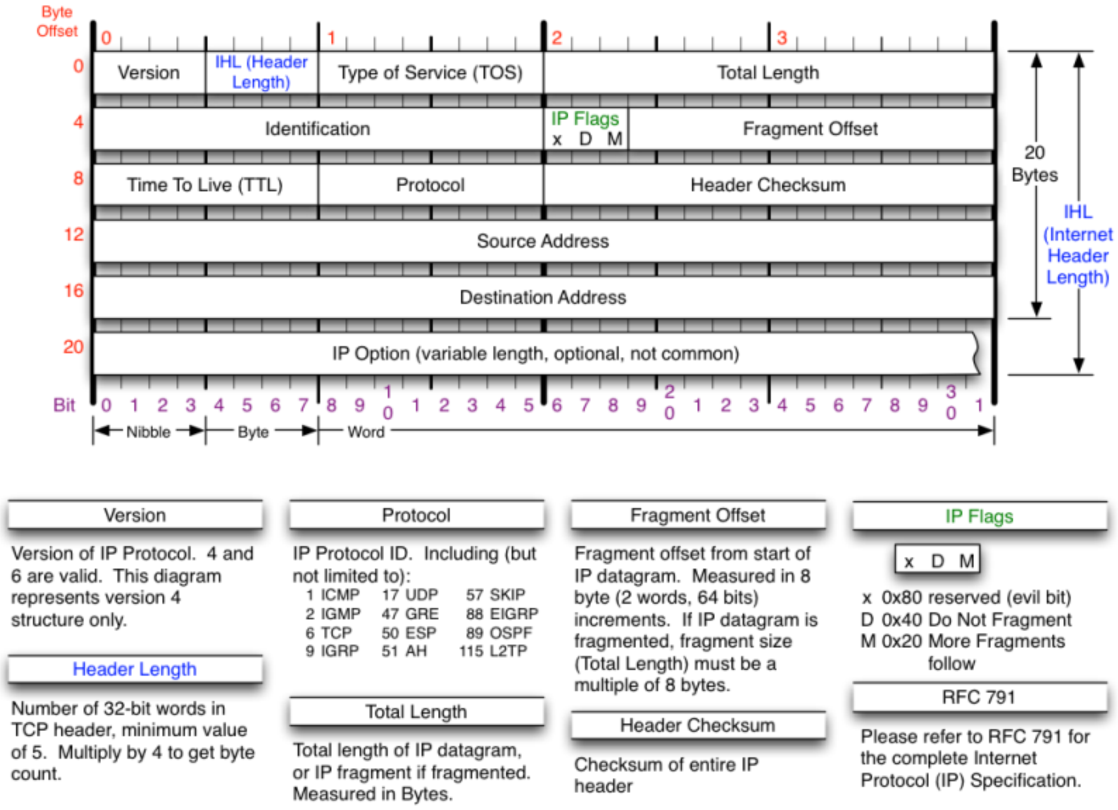
```
from pwn import *

p = remote('127.0.0.1',1984)

for _ in range(0,10):
  equation = p.recv()
  parts = equation.split()

p.interactive()
```

- Pwntools is a python3 framework that provides a lot of hacker functionality
- We can use the remote(server,port) function to interact with servers
- recv() and send() allows us to receive and send data
- Interactive() allows us to connect output and input to the connection

Protocol Header: IP

- Host-to-host protocol
- Specified in RFC 791
- 12 fields of varying sizes

- Handles
  - Routing
  - Fragmentation
  - Message Integrity
  - Data encapsulation



Version

Version of IP Protocol. 4 and 6 are valid. This diagram represents version 4 structure only.

Header Length

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

Protocol

IP Protocol ID. Including (but not limited to):

1 ICMP	17 UDP	57 SKIP
2 IGMP	47 GRE	88 EIGRP
6 TCP	50 ESP	89 OSPF
9 IGRP	51 AH	115 L2TP

Total Length

Total length of IP datagram, or IP fragment if fragmented. Measured in Bytes.

Fragment Offset

Fragment offset from start of IP datagram. Measured in 8 byte (2 words, 64 bits) increments. If IP datagram is fragmented, fragment size (Total Length) must be a multiple of 8 bytes.

Header Checksum

Checksum of entire IP header

IP Flags

x D M

x 0x80 reserved (evil bit)
D 0x40 Do Not Fragment
M 0x20 More Fragments follow

RFC 791

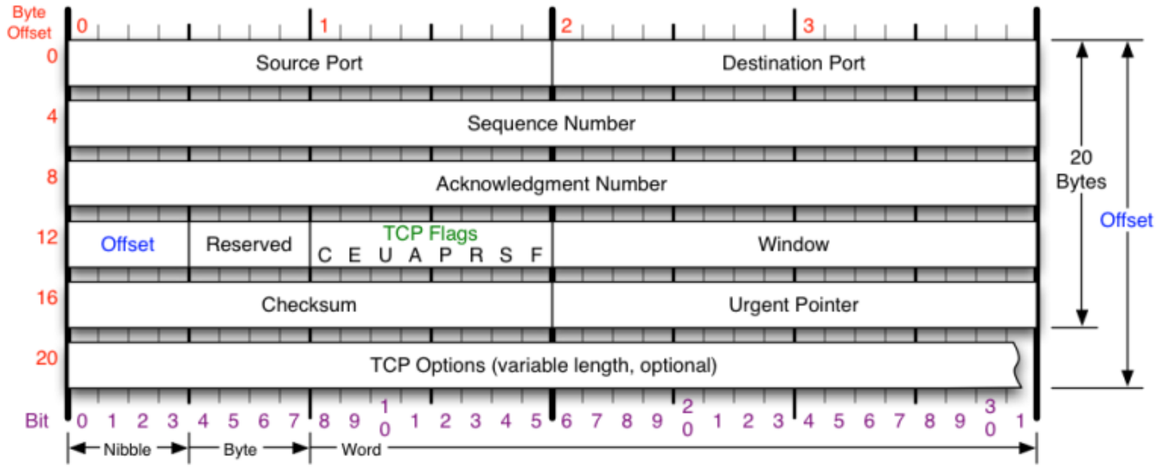
Please refer to RFC 791 for the complete Internet Protocol (IP) Specification.

Image copied from: https://nmap.org/book/tcpip-ref.html

Protocol Header: TCP

- Specified in RFC 793
- Service level protocol
- Connection-oriented

- Handles
  - Basic data transfer
  - Reliability
  - Flow control
  - Multiplexing
  - Connection state
  - In-order delivery



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

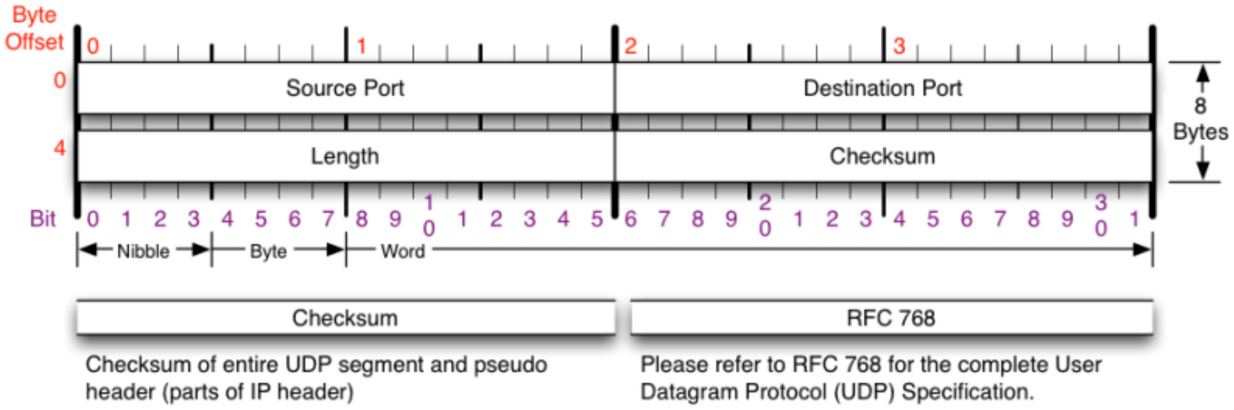
Please refer to RFC 793 for the complete Transmission Control Protocol (TCP) Specification.

Image copied from: https://nmap.org/book/tcpip-ref.html

Protocol Header: UDP

- Specified in UDP 768
- Service level protocol
- Connectionless

- Handles
  - Basic data transfer
  - ~~Reliability~~
  - ~~Flow control~~
  - ~~Multiplexing~~
  - ~~Connection state~~
  - ~~In-order delivery~~



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

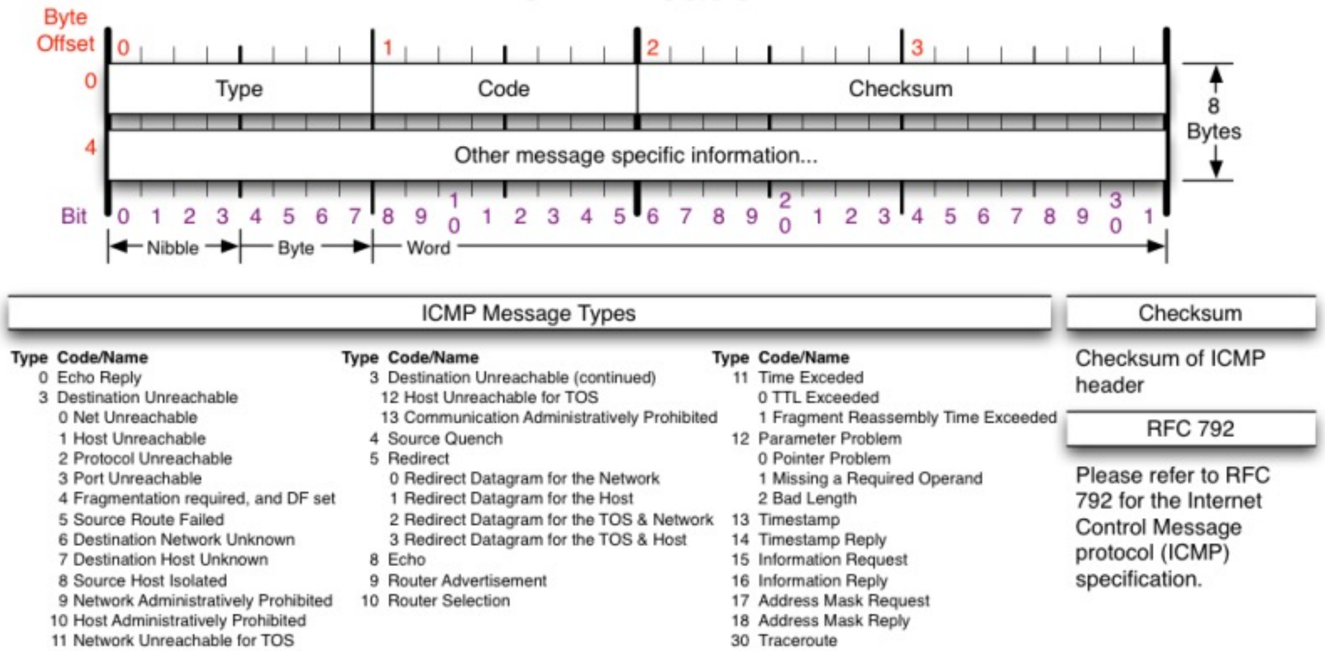
Please refer to RFC 768 for the complete User Datagram Protocol (UDP) Specification.

Image from: https://nmap.org/book/tcpip-ref.html

Protocol Header: ICMP

- Specified in RFC 792
- Error-reporting protocol
- Host-host or
- Gateway-host

- Handles
  - Error Types & Codes
  - Essential to IP



Byte Offset	0	1	2	3
0	Type	Code	Checksum	
4	Other message specific information...			

ICMP Message Types

Type	Code/Name
0	Echo Reply
3	Destination Unreachable
	0 Net Unreachable
	1 Host Unreachable
	2 Protocol Unreachable
	3 Port Unreachable
	4 Fragmentation required, and DF set
	5 Source Route Failed
	6 Destination Network Unknown
	7 Destination Host Unknown
	8 Source Host Isolated
	9 Network Administratively Prohibited
	10 Host Administratively Prohibited
	11 Network Unreachable for TOS
3	Destination Unreachable (continued)
	12 Host Unreachable for TOS
	13 Communication Administratively Prohibited
4	Source Quench
5	Redirect
	0 Redirect Datagram for the Network
	1 Redirect Datagram for the Host
	2 Redirect Datagram for the TOS & Network
	3 Redirect Datagram for the TOS & Host
8	Echo
9	Router Advertisement
10	Router Selection
11	Time Exceded
	0 TTL Exceeded
	1 Fragment Reassembly Time Exceeded
12	Parameter Problem
	0 Pointer Problem
	1 Missing a Required Operand
	2 Bad Length
13	Timestamp
14	Timestamp Reply
15	Information Request
16	Information Reply
17	Address Mask Request
18	Address Mask Reply
30	Traceroute

Checksum

Checksum of ICMP header

RFC 792

Please refer to RFC 792 for the Internet Control Message protocol (ICMP) specification.

Image from: https://nmap.org/book/tcpip-ref.html