Raw Packets

Who Am I?

- Jim O’Gorman
- Jameso@elwood.net
- Jogorman@gmail.com
- http://www.elwood.net/

What is This?

- What is a “raw” packet?
  - Packet Sniffer
    - Ethereal (http://www.ethereal.com/)
    - TCPDump (http://www.tcpdump.org/)
  - Protocol Processors
    - Ethereal has many
    - TCPDump has a few (ex DNS)

What Does a Raw Packet Look Like?

- What does a raw (hex) packet look like?

```
IP 10.10.80.73.49951 >
   64.233.167.99.80: . ack 2913739186
   win 65535

4500 0028 e9f7 4000 4006 0e39 0a0a
   5049

40e9 a763 c31f 0050 befb 759a adac
   21b2

5010 ffff a5d0 0000
```

Why Do I Care?

- Tools do the work for me. Why should I waste my time with this?
- In school, did you learn to do math by hand? Or with a calculator?

What is Hex?

- Base Ten (Decimal) - Fingers and Toes
- Binary - 1s and 0s
- Hex (Hexadecimal) - 1 through 15
  - 0-9 then a for 10, b for 11, etc up to f for 15.

0-F

- A = 10
- B = 11
- C = 12
- D = 13
- E = 14
- F = 15

Powers

- Base Ten is 0-9, so if you need something higher then 9, you use powers:
  - 18 is nothing more then 1^{10} plus 8
  - $1^{10000}1^{1000}1^{100}1^{10}1^{1}$

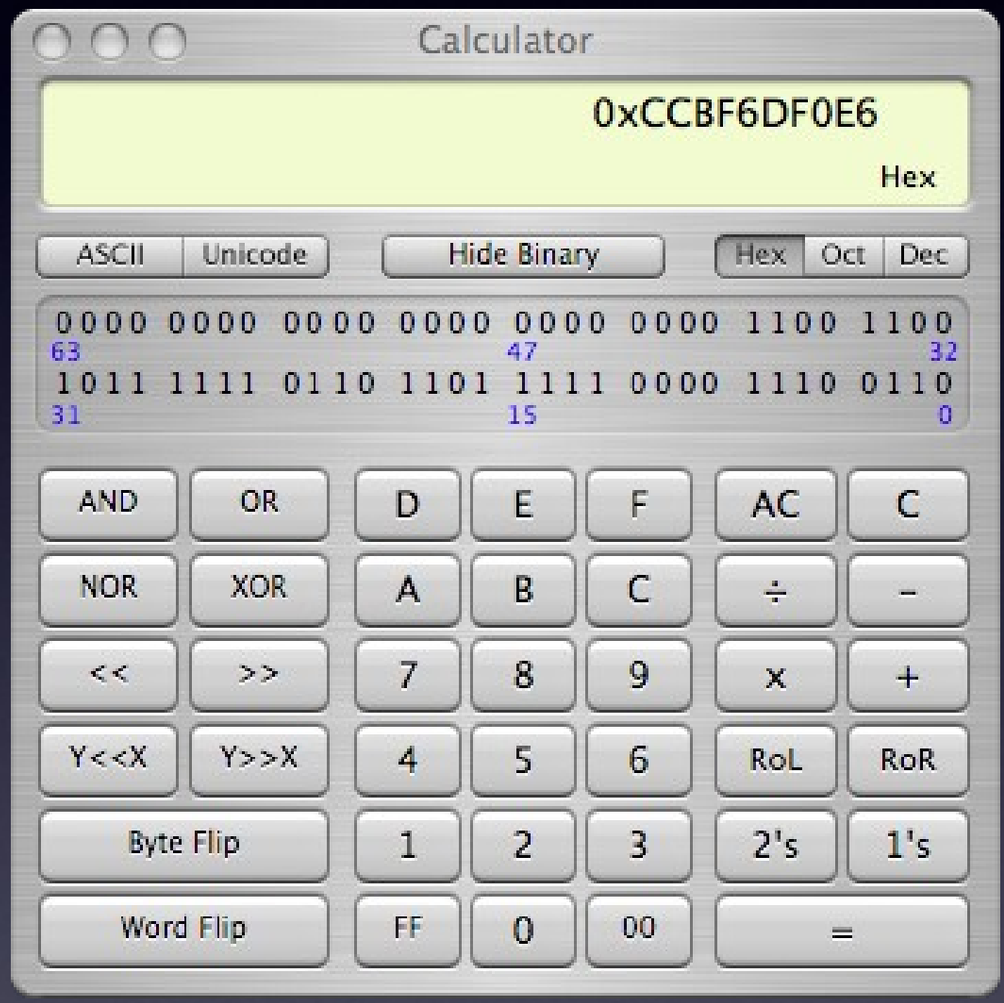
- Hex is 0-F, so when you need more than 15, you use powers:
  - 18 in hex is 0x12 = 1^{16} plus 2
  - 27 in hex is 0x1B = 1^{16} plus 11
  - $1^{65536}1^{4096}1^{256}1^{16}1^{1}$

Binary

- Quick (very quick) binary overview
  - 0 and 1
  - Powers: $1^{128}1^{64}1^{32}1^{16}1^{8}1^{4}1^{2}1^{1}$
  - 2 decimal is 10 binary ($1^{2}+0^{1}$)
  - 15 decimal is 1111 binary ($1^{8}+1^{4}+1^{2}+1^{1}$)
  - Hard to work with as small numbers take quite a bit to write out

Its Easy

- Converting hex is very easy
  - Let your calculator do it for you



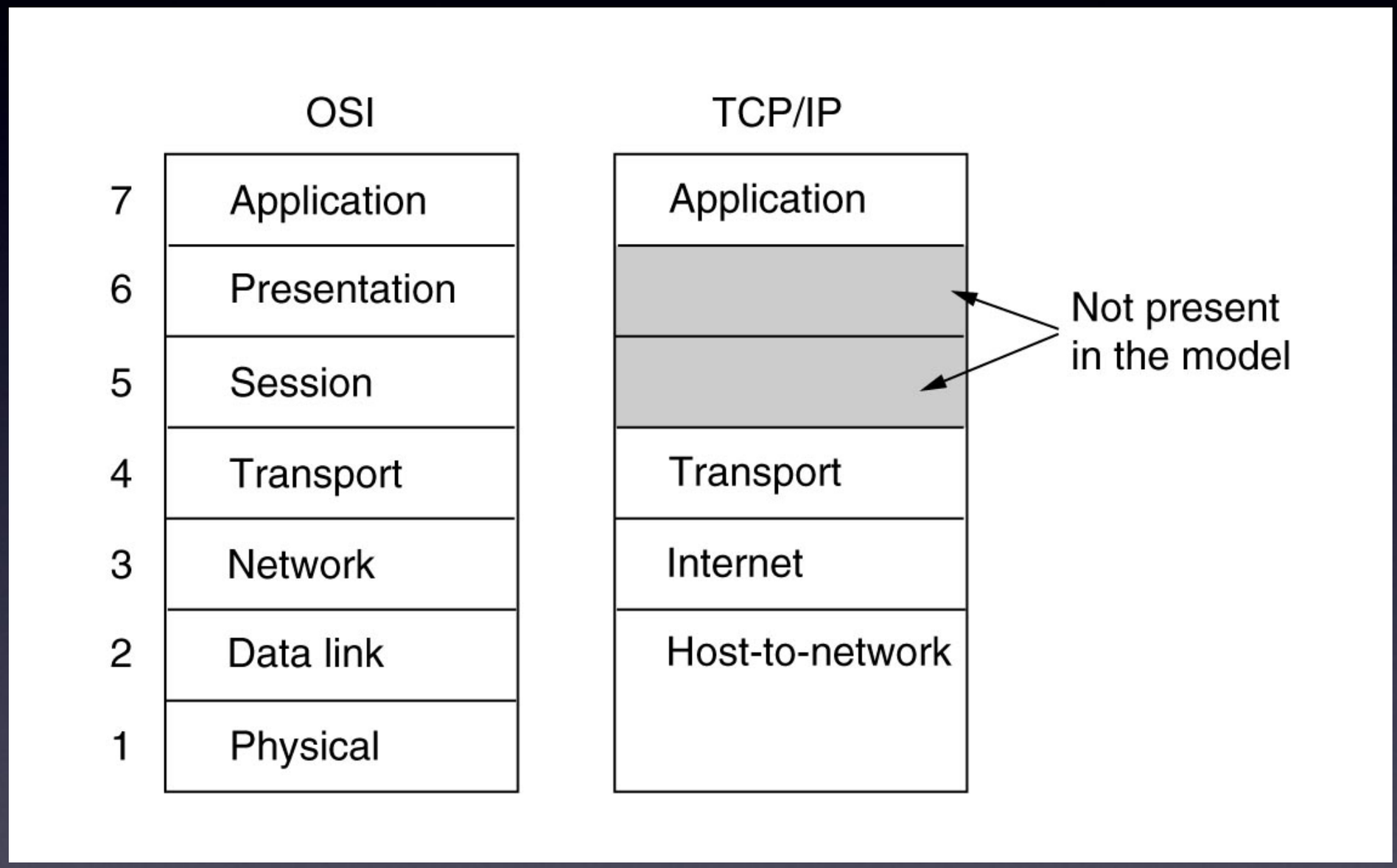
Bit, Nibble, Byte

- Bit - Smallest unit - 0 or 1
- Nibble - Four Bits, Half a Byte
- Byte - Eight Bits, Two Nibbles
- One hex digit is a nibble, two hex digits is a byte

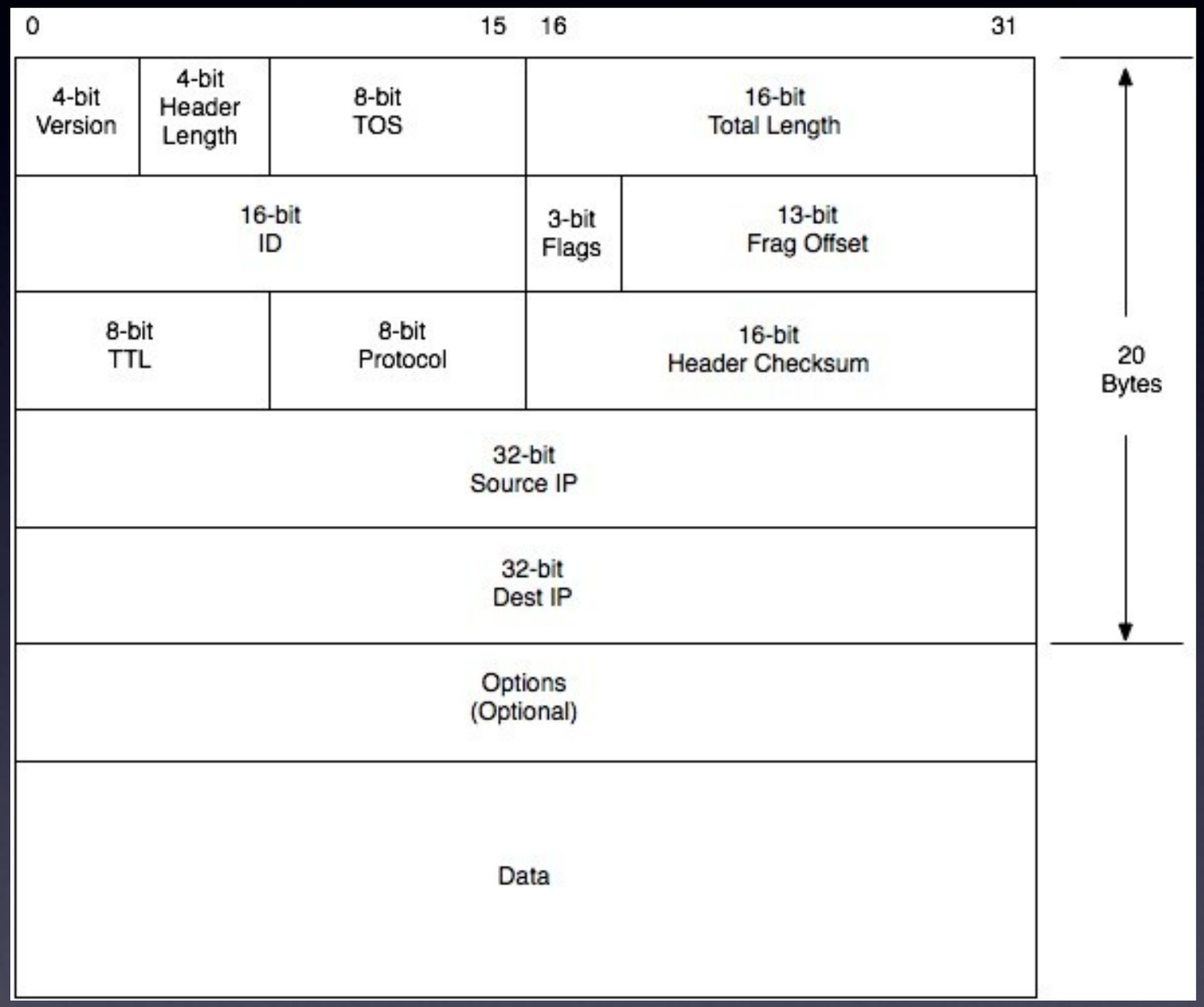
IP Networking

- Most of us know at least something about IP networking
  - At this point we are going to review the encapsulation used by IP
- Packets are broken up into various fields, each serving a different purpose

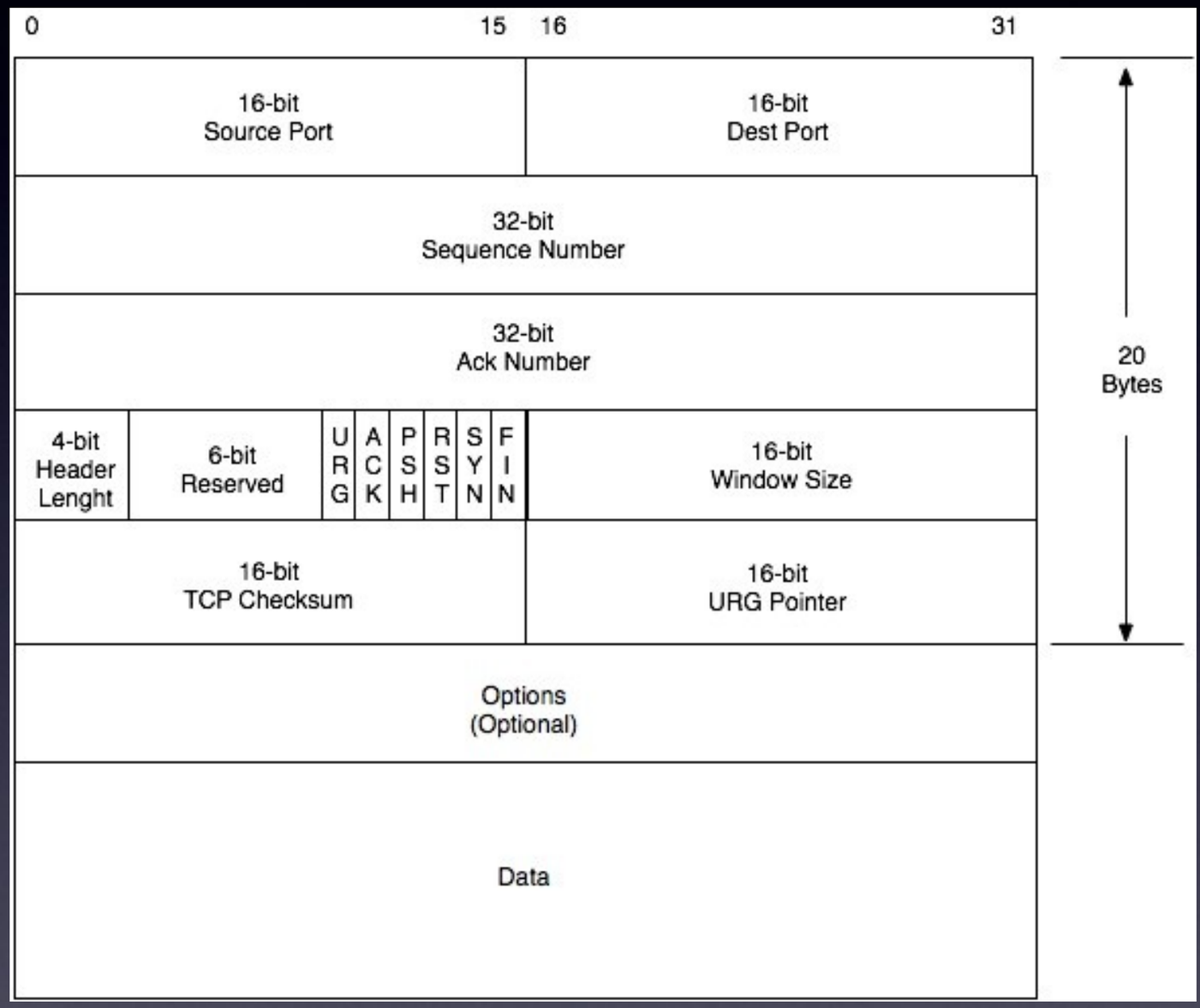
Stacks


OSI
TCP/IP
7
Application
6
Presentation
5
Session
4
Transport
3
Network
2
Data link
1
Physical
Application
Transport
Internet
Host-to-network
Not present
in the model

IP Datagram


0
15
16
31
4-bit Version
4-bit Header Length
8-bit TOS
16-bit Total Length
16-bit ID
3-bit Flags
13-bit Frag Offset
8-bit TTL
8-bit Protocol
16-bit Header Checksum
32-bit Source IP
32-bit Dest IP
Options (Optional)
Data
20 Bytes

TCP Packet

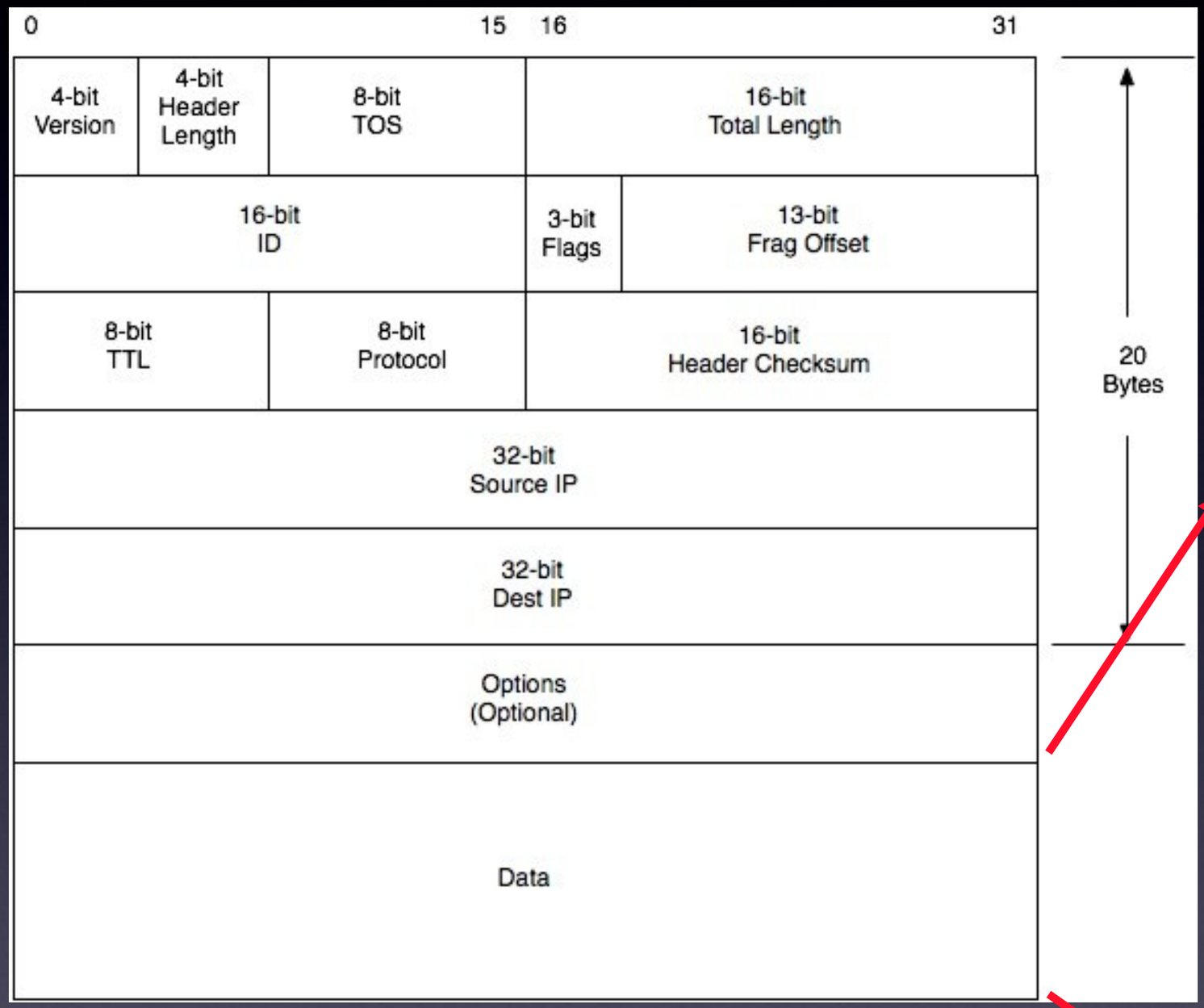

0
15
16
31
16-bit
Source Port
16-bit
Dest Port
32-bit
Sequence Number
32-bit
Ack Number
4-bit
Header
Lenght
6-bit
Reserved
U
R
G
A
C
K
P
S
H
R
S
T
S
Y
N
F
I
N
16-bit
Window Size
16-bit
TCP Checksum
16-bit
URG Pointer
Options
(Optional)
Data
20
Bytes

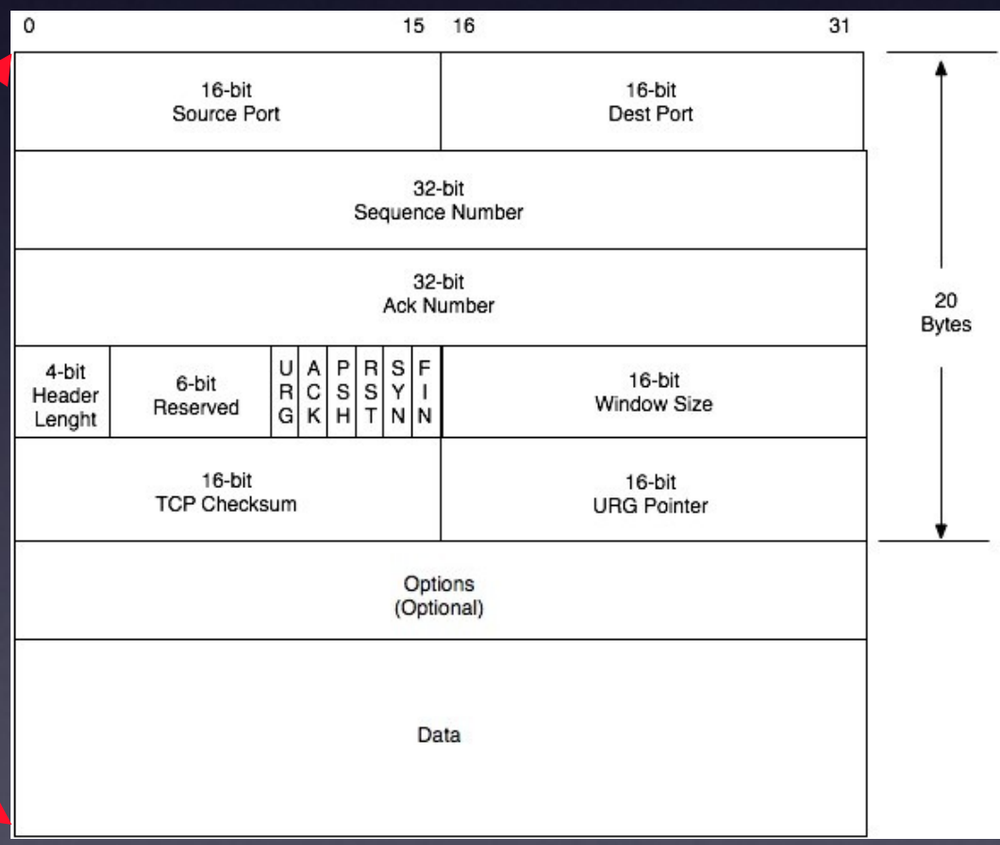
Other Protocols

- UDP
- ICMP
- IGMP
- etc

Russian Dolls

IP/TCP Dolls

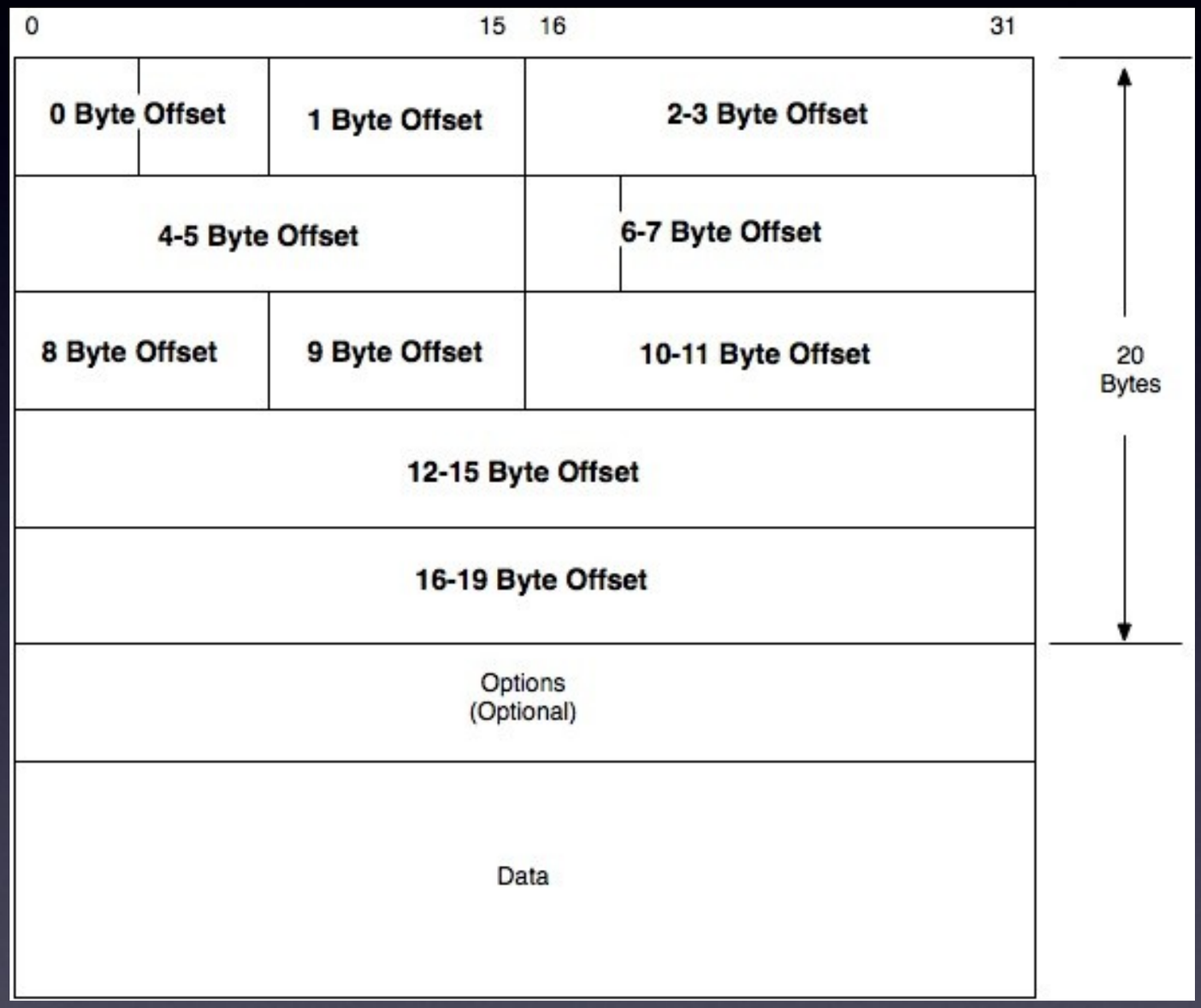

0
15
16
31
4-bit Version
4-bit Header Length
8-bit TOS
16-bit Total Length
16-bit ID
3-bit Flags
13-bit Frag Offset
8-bit TTL
8-bit Protocol
16-bit Header Checksum
32-bit Source IP
32-bit Dest IP
Options (Optional)
Data
20 Bytes


0
15
16
31
16-bit Source Port
16-bit Dest Port
32-bit Sequence Number
32-bit Ack Number
4-bit Header Lenght
6-bit Reserved
URG
ACK
PSH
RST
SYN
FIN
16-bit Window Size
16-bit TCP Checksum
16-bit URG Pointer
Options (Optional)
Data
20 Bytes

IP Encapsulation

- Physical -> Internet -> Transport -> Application
  - Ethernet -> IP -> TCP -> HTTP
  - Ethernet -> IP -> UDP -> DNS
  - Ethernet -> IP -> TCP -> SSH

Byte Offsets



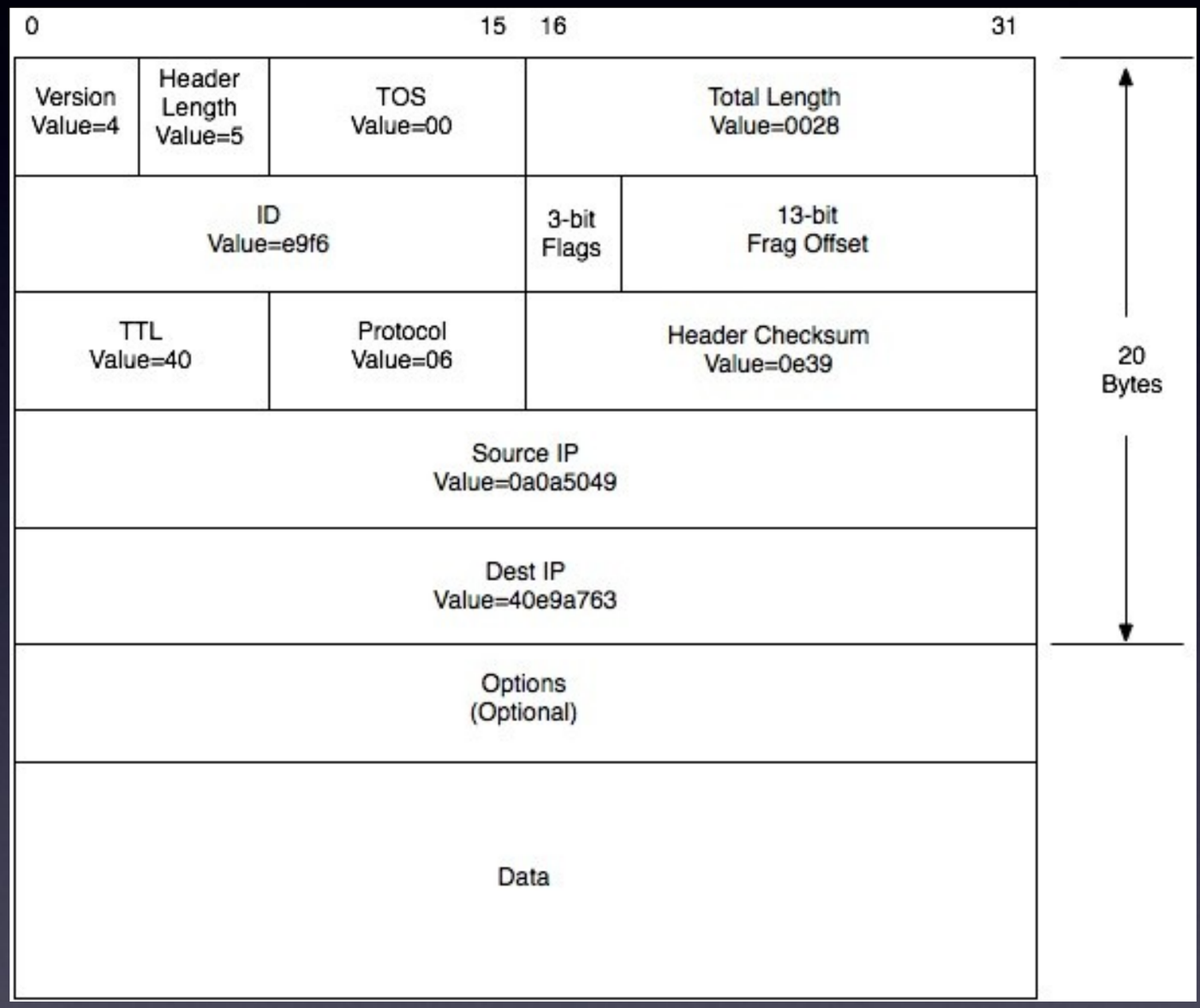
TCPDump

- -i Interface to listen on (ex. -i fxp0)
- -s Snaplen, or how much of the packet to capture (ex. -s 0 (capture whole packet))
- -X Print each packet in both hex and ASCII.
- -n Don’t convert address to hostnames

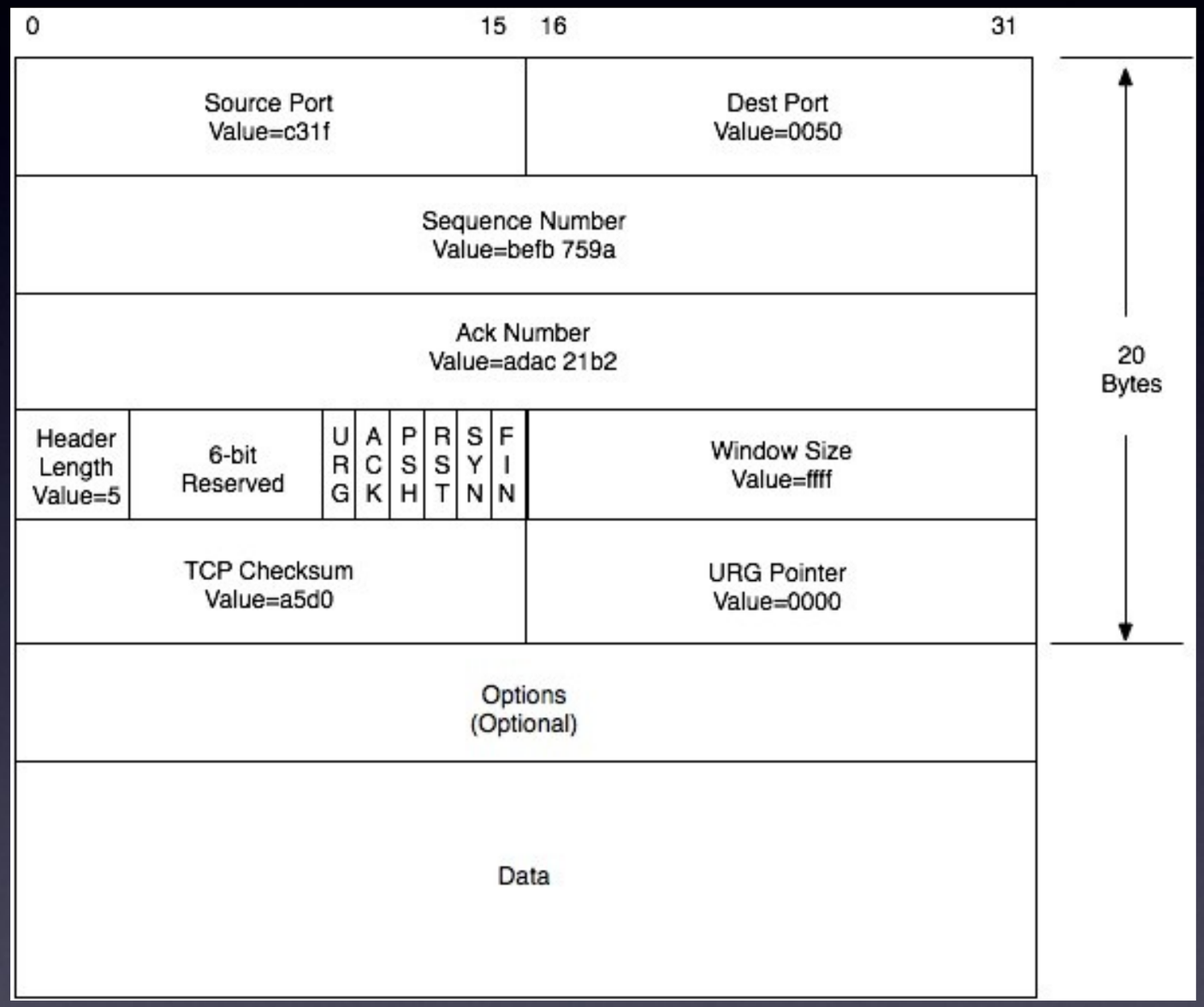
Our Packet

```
4500 0028 e9f7 4000 4006 0e39 0a0a 5049
40e9 a763 c31f 0050 befb 759a adac 21b2
5010 ffff a5d0 0000
```

IP Field Breakdown



TCP Field Breakdown



Packet Crafter

- Nemesis (http://nemesis.sourceforge.net/)
- Hping (http://www.hping.org/)
- Manually put together packets for various purposes

Initial Nemesis Command Line

- `nemesis tcp -d en0 -D 64.233.167.99 -S 10.10.80.73`

- Run nemesis in tcp mode, send the packet out interface en0, destination IP of 64.233.167.99, source IP of 10.10.80.73

```
00:04:19.851510 IP 10.10.80.73.30680 > 64.233.167.99.42024: S
   1945339175:1945339175(0) win 4096

        0x0000:  4500 0028 b643 0000 ff06 c2ec 0a0a 5049
   E..(.C.........PI

        0x0010:  40e9 a763 77d8 a428 73f3 8527 6630 8b06
   @..cw..(s..'f0..

        0x0020:  5002 1000 56f0 0000
   P...V...
```

IP Version

- 4 bit length (one nibble)
- High-order nibble of the 0 byte offset
- Common values are “4” and “6” for IPv4 and IPv6
- 4500

IP Header Length (1)

- 4 bit length (one nibble)
- Low-order nibble of the 0 byte offset
- Value is expressed in number of 32-bit words. (Byte has 8 bits, so a word is 4 bytes.) To determine value, the number must be multiplied by 4
- 4500
- 5*4=20
- Calculated by the stack, no reason to set in nemesis

IP Header Length (2)

```
4500 0028 e9f7 4000 4006 0e39 0a0a 5049
40e9 a763 c31f 0050 befb 759a adac 21b2
5010 ffff a5d0 0000
```

IP Type Of Service

- 8 bit length (one byte)
- 1st byte offset from 0 in the IP header
- A method to inform intermediate routers of the type of quality of service desired
- Empty in our example, so no reason to set in nemesis

```
4500 0028 e9f7 4000 4006 0e39 0a0a 5049

40e9 a763 c31f 0050 befb 759a adac 21b2

5010 ffff a5d0 0000
```

IP Total Length

- 16 bit field (two bytes)
- 2nd byte offset from 0 in the IP header
- Reports the total length of the IP datagram
- Maximum value of 65,535 (0xffff)
- Calculated by the stack, no reason to set in nemesis

```
4500 0028 e9f7 4000 4006 0e39 0a0a 5049

40e9 a763 c31f 0050 befb 759a adac 21b2

5010 ffff a5d0 0000
```

IP Identification (1)

- 16 bit field (two bytes)
- 4th byte offset from 0 in the IP header
- Unique ID for the packet - used often with fragments

```
4500 0028 e9f7 4000 4006 0e39 0a0a 5049

40e9 a763 c31f 0050 befb 759a adac 21b2

5010 ffff a5d0 0000
```

- 0xe9f7 = 59,895

IP Identification (2)

- nemesis tcp -d en0 -D 64.233.167.99 -S 10.10.80.73 -I 59895

```
4500 0028 e9f7 0000 ff06 8f38 0a0a 5049
40e9 a763 d33e 154b 159b 11d8 204a 107e
5002 1000 1c7e 0000
```

VS.

```
4500 0028 e9f7 4000 4006 0e39 0a0a 5049
40e9 a763 c31f 0050 befb 759a adac 21b2
5010 ffff a5d0 0000
```

IP Flags (1)

- 3 bit field
- First 3 bits of the high order nibble of the 6th byte offset from 0 in the IP header
- A series of single bit flags to set various options for the datagram
  - If the datagram is a fragment, if it is not allowed to be fragmented, if it is the last fragment, or if there are more fragments.
  - First bit is always 0

IP Flags (2)

Reserved	0=Frag if neded 1=Do not Frag	0=Last Frag 1=More Frags

```
4500 0028 e9f7 4000 4006 0e39 0a0a 5049

40e9 a763 c31f 0050 befb 759a adac 21b2

5010 ffff a5d0 0000
```

- 0x4 = 0100 binary
- 3 bit field = 010 = Do not Frag

IP Flags (3)

- nemesis tcp -d en0 -D 64.233.167.99 -S 10.10.80.73 -I 59895 -FD

```
4500 0028 e9f7 4000 ff06 4f38 0a0a 5049

40e9 a763 5aa3 f934 5637 b5a6 7cff 9f7a

5002 1000 e112 0000
```

vs.

```
4500 0028 e9f7 4000 4006 0e39 0a0a 5049

40e9 a763 c31f 0050 befb 759a adac 21b2

5010 ffff a5d0 0000
```

IP Fragment Offset

- 13 bit field
- Starts in the last bit of the low order nibble of the 6th byte offset from 0 in the IP header
- Used in fragment reassembly
- Only used on the 2nd fragment on in a fragment stream

```
4500 0028 e9f7 4000 4006 0e39 0a0a 5049

40e9 a763 c31f 0050 befb 759a adac 21b2

5010 ffff a5d0 0000
```

IP Time To Live (1)

- 8 bit field (one byte)
- 8th byte offset from 0 in the IP header
- Used to prevent routing loops. Each router on the path decrements this field by 1 before forwarding it on
- Set to different values based on the operating system
- Can be used to estimate how many hops a remote site is located from you

IP Time To Live (2)

- 0x40 = 64

```
4500 0028 e9f7 4000 4006 0e39 0a0a 5049
40e9 a763 c31f 0050 befb 759a adac 21b2
5010 ffff a5d0 0000
```

- nemesis tcp -d en0 -D 64.233.167.99 -S 10.10.80.73 -I 59895 -FD -T 64

```
4500 0028 e9f7 4000 4006 0e39 0a0a 5049
40e9 a763 e84e 559f 7627 17c0 3400 2af9
5002 1000 3274 0000
```

IP Protocol

- 8 bit field
- 9th byte offset from 0 in the IP header
- Indicates the embedded protocol
  - 1=ICMP
  - 6=TCP
  - 17=UDP (0x11)

```
4500 0028 e9f7 4000 4006 0e39 0a0a 5049
40e9 a763 c31f 0050 befb 759a adac 21b2
5010 ffff a5d0 0000
```

IP Header Checksum

- 16 bit field
- 10th byte offset from 0 in the IP header
- RFC791: “The 16-bit ones compliment of the ones compliment sum of all 16-bit words in the header.”
- Allows the stack to detect and discard corrupted packets

```
4500 0028 e9f7 4000 4006 0e39 0a0a 5049

40e9 a763 c31f 0050 befb 759a adac 21b2

5010 ffff a5d0 0000
```

IP Source Address

- 32 bit field
- 12th byte offset from 0 in the IP header
- IP address of the sender of the datagram

```
4500 0028 e9f7 4000 4006 0e39 0a0a 5049
40e9 a763 c31f 0050 befb 759a adac 21b2
5010 ffff a5d0 0000
```

- 0x0a=10, 0x50=80, 0x49=79
- 10.10.80.79

IP Destination Address

- 32 bit field
- 16th byte offset from 0 in the IP header
- IP address of recipient of the IP datagram

```
4500 0028 e9f7 4000 4006 0e39 0a0a 5049

40e9 a763 c31f 0050 befb 759a adac 21b2

5010 ffff a5d0 0000
```

- 0x40=64, 0xe9=233, 0xa7=167, 0x63=99
- 64.233.167.99

IP Options and Padding

- Only if the IP header length is greater then 20
- Options depend on implementation of the IP stack
- Header must end on a multiple of 32 bits, so may be padded with 0s

TCP Source Port (1)

- 16 bit field
- 0 byte offset
- Sending port
- Max value of 65,535
- 0xc31f=49,951

4500 0028 e9f7 4000 4006 0e39 0a0a 5049

40e9 a763 c31f 0050 befb 759a adac 21b2

5010 ffff a5d0 0000

TCP Source Port (2)

- nemesis tcp -d en0 -D 64.233.167.99 -S 10.10.80.73 -I 59895 -FD -T 64 -x 49951

4500 0028 e9f7 4000 4006 0e39 0a0a 5049

40e9 a763 c31f 1f28 2f47 0d09 4103 48f3

5002 1000 b4b4 0000

VS.

4500 0028 e9f7 4000 4006 0e39 0a0a 5049

40e9 a763 c31f 0050 befb 759a adac 21b2

5010 ffff a5d0 0000

TCP Destination Port (1)

- 16 bit field
- 2nd byte offset from 0 in the TCP header
- Receiving port
- 0x0050=80

4500 0028 e9f7 4000 4006 0e39 0a0a 5049

40e9 a763 c31f 0050 befb 759a adac 21b2

5010 ffff a5d0 0000

TCP Destination Port (2)

- `nemesis tcp -d en0 -D 64.233.167.99 -S 10.10.80.73 -I 59895 -FD -T 64 -x 49951 -y 80`

```
4500 0028 e9f7 4000 4006 0e39 0a0a 5049
40e9 a763 c31f 0050 5e6d ea31 2aa5 4a3b
5002 1000 dc53 0000
```

VS.

```
4500 0028 e9f7 4000 4006 0e39 0a0a 5049
40e9 a763 c31f 0050 befb 759a adac 21b2
5010 ffff a5d0 0000
```

TCP Sequence Number (1)

- 32 bit field
- 4th byte offset from 0 in the TCP header
- The sequence number of the first byte of data in this packet.
- When the SYN flag is present, it is the initial sequence number, and the first byte of data is the initial sequence number + 1

TCP Sequence Number (2)

- 0xbefb759a = 3,204,150,682

4500 0028 e9f7 4000 4006 0e39 0a0a 5049

40e9 a763 c31f 0050 befb 759a adac 21b2

5010 ffff a5d0 0000

- nemesis tcp -d en1 -D 64.233.167.99 -S 10.10.80.73 -I 59895 -FD -T 64 -x 49951 -y 80 -s 3204150682

4500 0028 e9f7 4000 4006 0e39 0a0a 5049

40e9 a763 c31f 0050 befb 759a 3fa5 280c

5002 1000 fd8b 0000

TCP Acknowledgement Number (1)

- 32 bit field
- 8th byte offset from 0 in the TCP header
- When the ACK flag is set, contains the value of the next expected sequence number

TCP Acknowledgement Number (2)

- 0xadac21b2 = 2,913,739,186

4500 0028 e9f7 4000 4006 0e39 0a0a 5049

40e9 a763 c31f 0050 befb 759a adac 21b2

5010 ffff a5d0 0000

- nemesis tcp -d en1 -D 64.233.167.99 -S 10.10.80.73 -I 59895 -FD -T 64 -x 49951 -y 80 -s 3204150682 -a 2913739186

4500 0028 e9f7 4000 4006 0e39 0a0a 5049

40e9 a763 c31f 0050 befb 759a adac 21b2

5002 1000 95de 0000

TCP Header Length (1)

- 4 bit field
- 12th byte offset from 0 in the TCP header
- Size of the TCP header in 32 bit words
- Value is expressed in number of 32-bit words. (Byte has 8 bits, so a word is 4 bytes.) To determine value, the number must be multiplied by 4

TCP Header Length (2)

- 5*4=20

4500 0028 e9f7 4000 4006 0e39 0a0a 5049

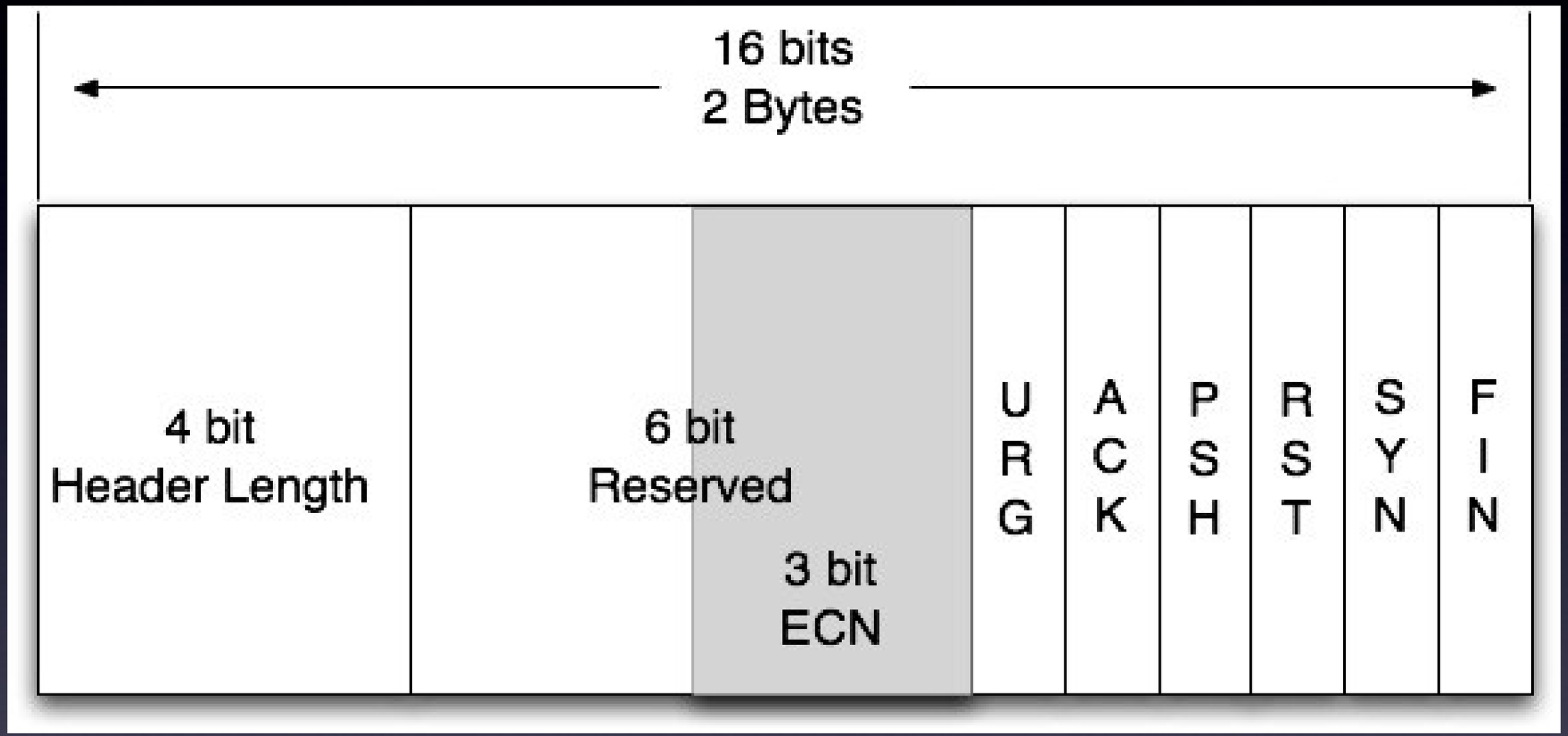
40e9 a763 c31f 0050 befb 759a adac 21b2

5010 ffff a5d0 0000

TCP Reserved and Flags (1)

- 6 bit reserved field for future use
  - Low order 3 bits used for ECN in some stacks
- 6 bit flag field
  - Urgent
  - Acknowledge
  - Push
  - Reset
  - Synchronize
  - Finish
- Does not end on standard boundaries, causing us to have to break this down to binary to read it

TCP Reserved and Flags (2)

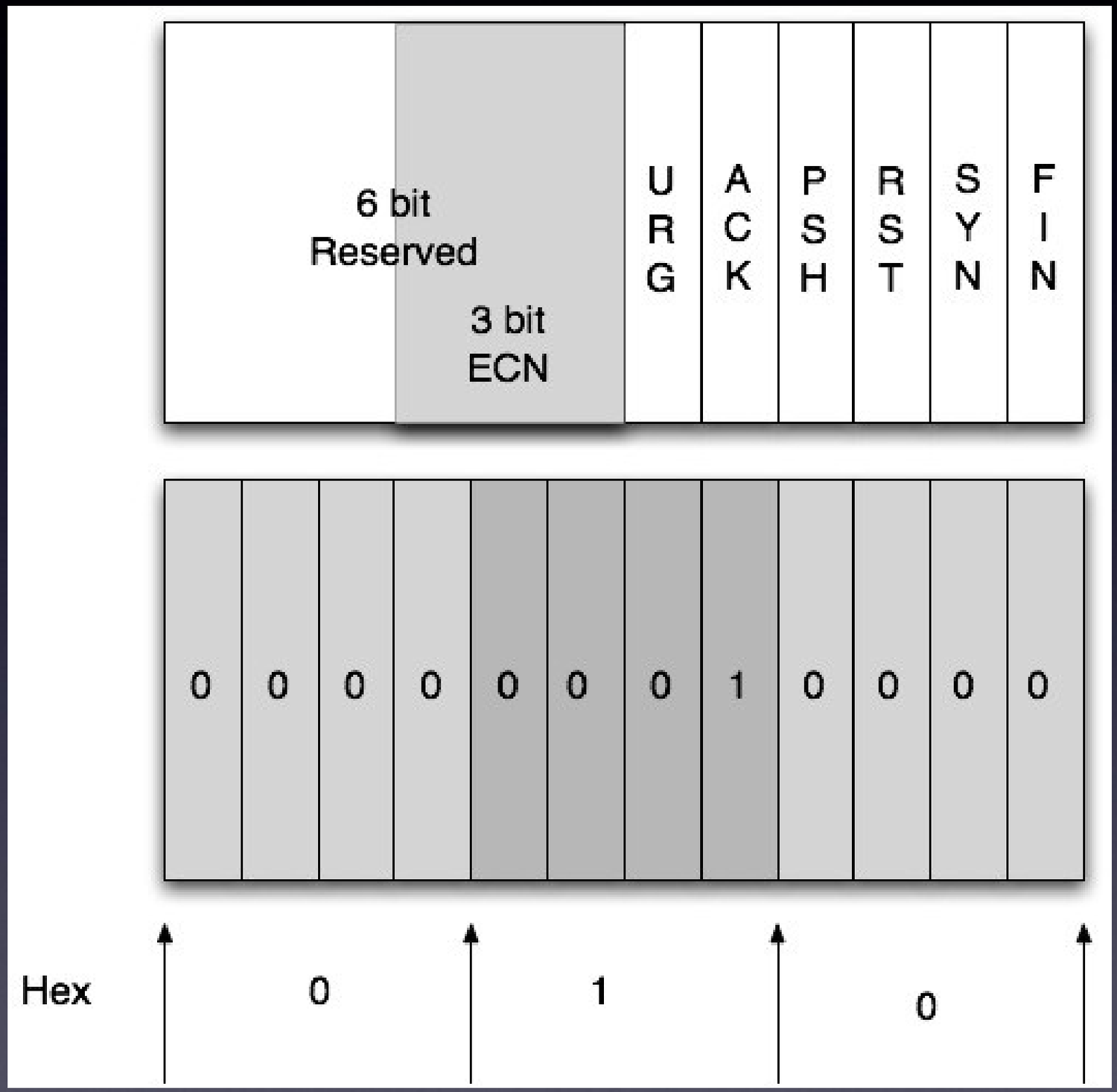


4500 0028 e9f7 4000 4006 0e39 0a0a 5049
40e9 a763 c31f 0050 befb 759a adac 21b2
5010 ffff a5d0 0000

TCP Reserved and Flags (3)

- 0x010 is the value - this includes the reserved and flags. To get the value we must break this apart
- 0000 0001 0000

TCP Reserved and Flags (4)



TCP Reserved and Flags (5)

- One of the most complex sections of the TCP header, is also one of the most important as you need to be able to figure out the flags
- Easiest way to process is to convert the final three nibbles of the 12th byte offset and the 13th byte offset from 0 in the TCP header into binary
- Each flag is true or false
- In our case, only the ACK flag is set

TCP Reserved and Flags (6)

- `nemesis tcp -d en1 -D 64.233.167.99 -S 10.10.80.73 -I 59895 -FD -T 64 -x 49951 -y 80 -s 3204150682 -a 2913739186 -fA`

```
4500 0028 e9f7 4000 4006 0e39 0a0a 5049
40e9 a763 c31f 0050 befb 759a adac 21b2
5010 1000 95d0 0000
```

VS.

```
4500 0028 e9f7 4000 4006 0e39 0a0a 5049
40e9 a763 c31f 0050 befb 759a adac 21b2
5010 ffff a5d0 0000
```

TCP Window Size (1)

- 16 bit field
- 14th byte offset from 0 in the TCP header
- The number of bytes the sender is able to receive, beginning from the ack field value

4500 0028 e9f7 4000 4006 0e39 0a0a 5049

40e9 a763 c31f 0050 befb 759a adac 21b2

5010 ffff a5d0 0000

TCP Window Size (2)

- 0xffff = 65,535
- nemesis tcp -d en1 -D 64.233.167.99 -S 10.10.80.73 -I 59895 -FD -T 64 -x 49951 -y 80 -s 3204150682 -a 2913739186 -fA -w 65535

```
4500 0028 e9f7 4000 4006 0e39 0a0a 5049

40e9 a763 c31f 0050 befb 759a adac 21b2

5010 ffff a5d0 0000
```

TCP Checksum (1)

- 16 bit field
- 16th byte offset from 0 in the TCP header
- RFC793 - “The checksum field is the 16 bit one's complement of the one's complement sum of all 16-bit words in the header and text. If a segment contains an odd number of header and text octets to be checksummed, the last octet is padded on the right with zeros to form a 16-bit word for checksum purposes. The pad is not transmitted as part of the segment. While computing the checksum, the checksum field itself is replaced with zeros.”

TCP Checksum (2)

- Time consuming to compute by hand.
- TCPDump will report invalid checksums

4500 0028 e9f7 4000 4006 0e39 0a0a 5049

40e9 a763 c31f 0050 befb 759a adac 21b2

5010 ffff a5d0 0000

TCP Urgent Pointer

- 16 bit field
- 18th byte offset from 0 in the TCP header
- When the URG flag is true, points to the sequence number of the last byte of a series of urgent data

```
4500 0028 e9f7 4000 4006 0e39 0a0a 5049

40e9 a763 c31f 0050 befb 759a adac 21b2

5010 ffff a5d0 0000
```

TCP Options

- Follow the 20th byte in the TCP header
- Total length must be a multiple of 32 bits - Padded with 0s to fit
- Included in checksum
- Very common
  - Maximum Segment Size
  - Timestamp
  - Window scale factor
  - etc

Packet Comparison

```
4500 0028 e9f7 4000 4006 0e39 0a0a 5049
40e9 a763 c31f 0050 befb 759a adac 21b2
5010 ffff a5d0 0000

4500 0028 e9f7 4000 4006 0e39 0a0a 5049
40e9 a763 c31f 0050 befb 759a adac 21b2
5010 ffff a5d0 0000
```

Final Command Line

- `nemesis tcp -d en1 -D 64.233.167.99 -S 10.10.80.73 -I 59895 -FD -T 64 -x 49951 -y 80 -s 3204150682 -a 2913739186 -fA -w 65535`

Questions?