

Client ▾	Microsoft Windows ▴	OS X ▴	Linux ▴	BSDs ▴	Solaris ▴	Other ▴
Xplico	No	No	Yes	No	No	No
Wireshark (formerly Ethereal)	Yes	Yes	Yes	Yes	Yes	AIX, HP-UX, IRIX, Tru64 UNIX
tcpdump	Yes (WinDump)	Yes	Yes	Yes	Yes	AIX, HP-UX, IRIX, Tru64 UNIX
SteelCentral Transaction Analyzer	Yes	Version 3.5 capture agents on PowerPC only	GUI, plus version 3.5 capture agents	No	Version 3.5 capture agents on SPARC only	Version 3.5 capture agents on AIX and PA-RISC HP-UX only
snoop	No	No	No	No	Yes	No
OmniPeek (formerly AiroPeek, EtherPeek)	Yes	No	No	No	No	No
Observer	Yes	No	No	No	No	No
ngrep	Yes	Yes	Yes	Yes	Yes	AIX, BeOS, HP-UX, IRIX, Tru64 UNIX
netsniff-ng	No	No	Yes	No	No	No
Microsoft Network Monitor	Yes	No	No	No	No	No
LANMeter	No	No	No	No	No	Fluke proprietary hardware
Kismet	Yes	Yes	Yes	Yes	?	?
justniffer	No	Yes	Yes	Yes	Yes	?
Ettercap	Yes	Yes	Yes	Yes	Yes	?
EtherApe	No	Yes	Yes	Yes	Yes	?
IdSniff	?	Yes	Yes	Yes	Yes	?
CommView	Yes	No	No	No	No	No
Clusterpoint Network Traffic Surveillance System	Yes	Yes	Yes	Yes	No	Any virtual-machine compatible OS
Clarified Analyzer	Yes	Yes	Yes	No	No	?
Carnivore	Yes	No	No	No	No	No
Capsa Free Edition	Yes	No	No	No	No	No
Cain and Abel	Yes	No	No	No	No	No

# Binary (base 2) • Hexidecimal (HEX / base 16) • Decimal (base 10)

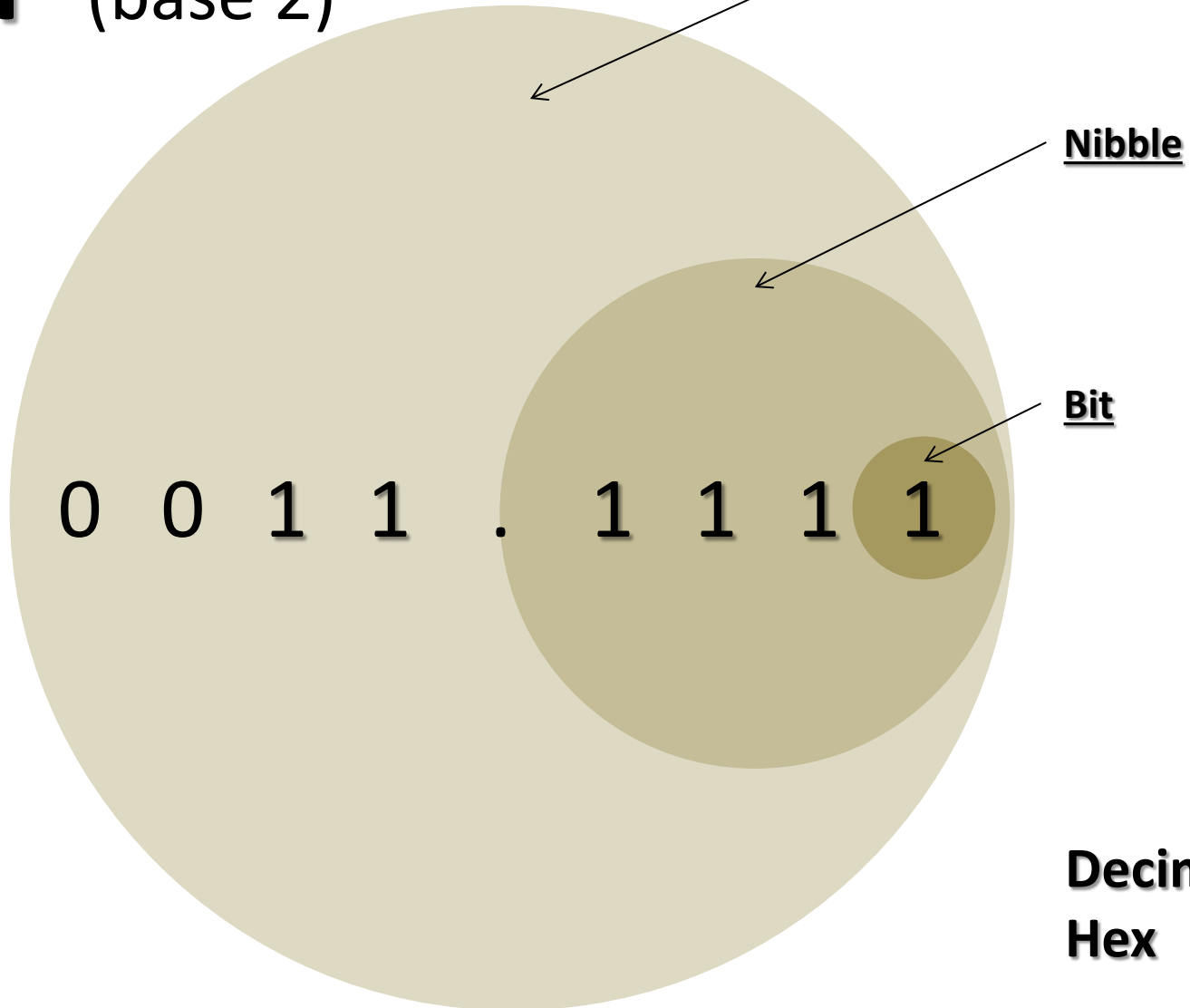
Binary	Hex	Decimal
0000	0	0
0001	1	1
0010	2	2
0011	3	3
0100	4	4
0101	5	5
0110	6	6
0111	7	7
1000	8	8
1001	9	9
1010	A	10
1011	B	11
1100	C	12
1101	D	13
1110	E	14
1111	F	15

128 • 64 • 32 • 16 • 8 • 4 • 2 • 1

$$8 + 4 + 2 + 1 = 15$$

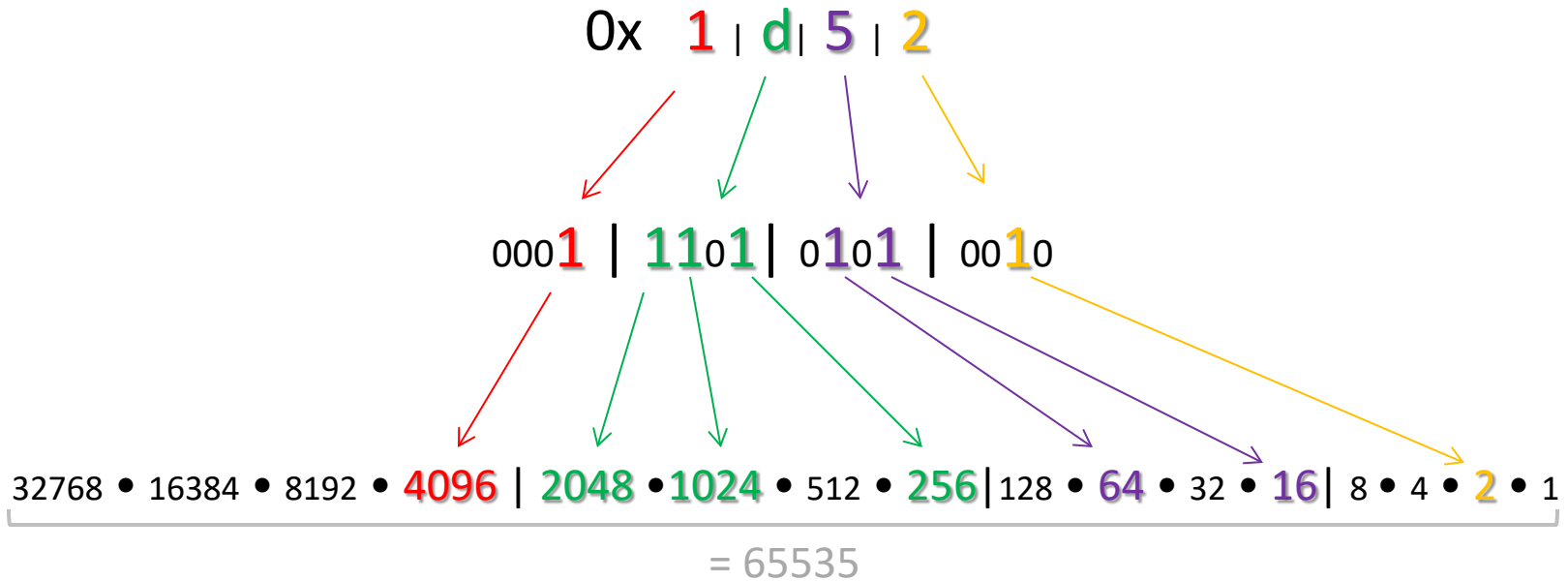
→ 1111 = F = 15

**$n^2$**  (base 2)



**Decimal** = 63

**Hex** = 3F



$$= 4096 + 2048 + 1024 + 256 + 64 + 16 + 2$$

$$= 7479$$

#### Alternate Method:

0x 1 d 5 2

0001 1101 0101 0010

$$= (1 \cdot 16^3) + (13 \cdot 16^2) + (5 \cdot 16^1) + (2 \cdot 1)$$

$$= 4096 + 3328 + 80 + 2$$

$$= 7479$$

# Decoding an IP Header

IHL (Header Length):

$N \times 4 =$

$5 \times 4 = 20$

= 20 bytes length

Type Of Service:

= 00

= Offset of 0

= located at beginning of packet header and extends for 4 bits.

Packet Length:

= 0034

= 0000.0000.0011.0100

= 52 bytes length

IPID:

= c9e7

= 1100 . 1001 . 1110 . 0111

= 49152 + 2304 + 224 + 7

= 51687

IP Ver:

4=IPv4

6=IPv6

Fragmentation

= 0x4000

= 0100 . 0000

RES	DF	MF	OF	OF	OF	OF	OF
R	D	M	O	O	O	O	O
0	1	0	0	0	0	0	0

4500 0034 c9e7 4000 3d06 178c d823 d9ba  
 ac10 00b7 0017 12f5 729a 2105 145c db4f  
 6018 16d0 a7cb 0000 0204 05b4 6c73 202d  
 6c61 0000

ICMP TYPE:

00 = ECHO Reply

08 = ECHO Request

03 = DEST Unreachable

05 = Redirect

11 (0b) = Time Exceeded

TTL(Hop Limit-IPv6):

= 3d

= 0011 . 1101

= 48 + 13

= 61

Protocol:

01= ICMP

06= TCP

11= UDP

CHECKSUM:

= 178c

= 0001 . 0111 . 1000 . 1100

= 4096 + 1792 + 128 + 12

= 6028

Destination Address:

= ac10 = 1010 . 1100 . 0001 . 0000 = 172.16

= 00b7 = 0000 . 0000 . 1011 . 0111 = 0.183

= 172.16.0.183

Source Address:

= d823 = 1101 . 1000 . 0010 . 0011 = 216.35

= d9ba = 1101 . 1001 . 1011 . 1010 = 217.186

= 216.35.217.186

# Decoding a TCP Header

Window Size:

= 0x16d0  
 = 0001.0110.1101.0000  
 = 4096 + 1536 + 208 + 0  
 = 5840 bytes length

Source Port:

= 0x17  
 = 23 (Telnet)

Destination

Port:  
 = 0x12f5  
 = 4853

Sequence #:

729a 2105  
 = 1922703621

ACK #:

145c db4f  
 = 341629775

Header Length:

= n x 4 = x  
 = 6 x 4 = 24  
 x - 20 = z  
 24 - 20 = 4

TCP Options  
 Length = 4 bytes

FLAGS:

= 0x18  
 = 0001.1000

= CWR	= ECN	= URG	= ACK	= PSH	= RST	= SYN	= FIN
C	E	U	A	P	R	S	F
0	0	0	1	1	0	0	0

4500 0034 c9e7 4000 3d06 178c d823 d9ba  
 ac10 00b7 0017 12f5 729a 2105 145c db4f  
 6018 16d0 a7cb 0000 0204 05b4 6c73 202d  
 6c61 0000

CHECKSUM: = 0xa7cb

= 1010 . 0111 . 1100 . 1011  
 = 40960 + 1792 + 192 + 11  
 = 42955

Urgent Pointer

= 0000  
 = Not Set

TCP Options:

= 0204 05b4  
 = 4 Bytes

Payload / Data Length Formula:

= 6c73 2026 6c61 0000

= IP Total Length - [ IHL + TCP Header Length ] = Payload Length

= 52 - [ (5\*4) + (6\*4) ] = 8

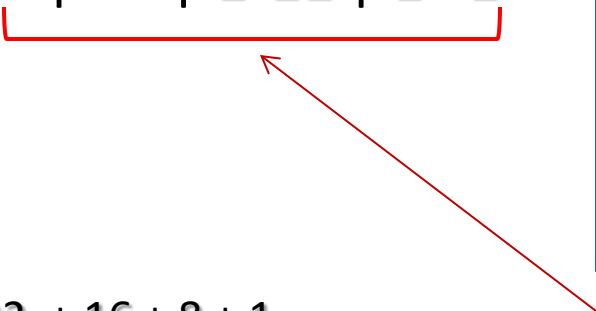
= 52 - (20+24) = 8 bytes

## COMPUTING FRAGMENT OFFSET VALUE

4500 0030 002a[00b9]4001 43a6 0101 0101  
0202 0202 9988 9988 9988 9988 9988 9988  
9988 9988 9988 9988 9988 9988 9988 9988

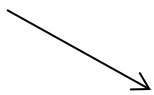
0x 0 | 0 | B | 9

0000 | 0000 | 1011 | 1001



$$= 128 + 32 + 16 + 8 + 1$$

$$= 185$$


$$185 * 8 = 1480$$

### Why Multiply By "8" ?

Possible total IP datagram value = 65536

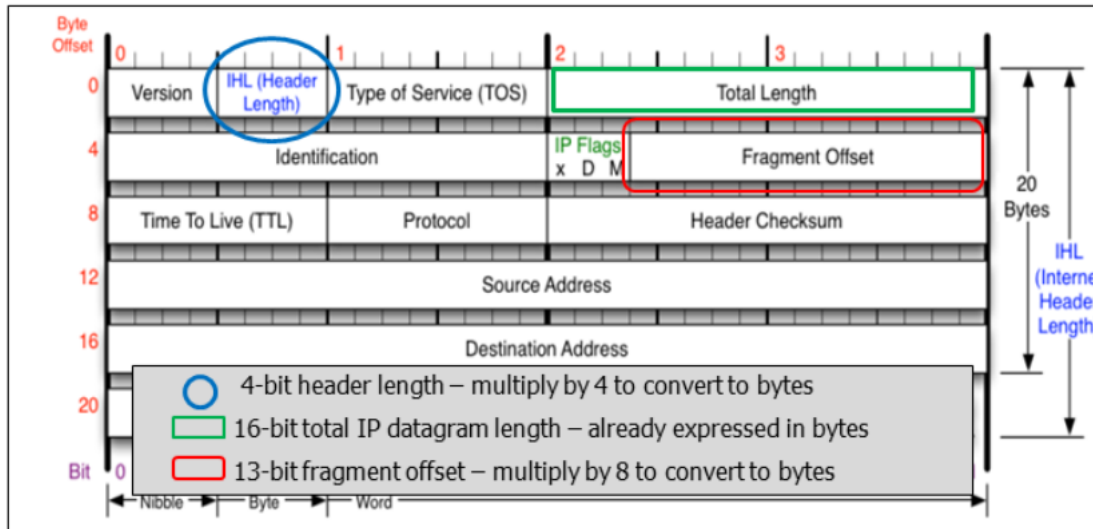
Fragment offset length  $2^{13} = 8192$

$$65536 / 8192 = 8$$

Which is why you multiply fragment offset by 8 to get the number of bytes of transport Header and payload data that came before this fragment in the associated fragment train.

**13 low order bits**

# FRAGMENT = FIRST, LAST, MIDDLE, NOT



## Not a Fragment

```
0x0000: 4510 003c 0000 [4000] 3106 dfc4 c73b 962e
0x0010: 0aff 017f 01bb c5cc c2da 982d a6e4 875a
```

010:0 | 0000 | 0000 | 0000

## First Fragment

```
0x0000: 4510 003c 0000 [2000] 3106 dfc4 c73b 962e
0x0010: 0aff 017f 01bb c5cc c2da 982d a6e4 875a
```

001:0 | 0000 | 0000 | 0000

## Middle Fragment

```
0x0000: 4510 003c 0000 [2100] 3106 dfc4 c73b 962e
0x0010: 0aff 017f 01bb c5cc c2da 982d a6e4 875a
```

001:0 | 0001 | 0000 | 0000

## Middle Fragment

```
0x0000: 4510 003c 0000 [3000] 3106 dfc4 c73b 962e
0x0010: 0aff 017f 01bb c5cc c2da 982d a6e4 875a
```

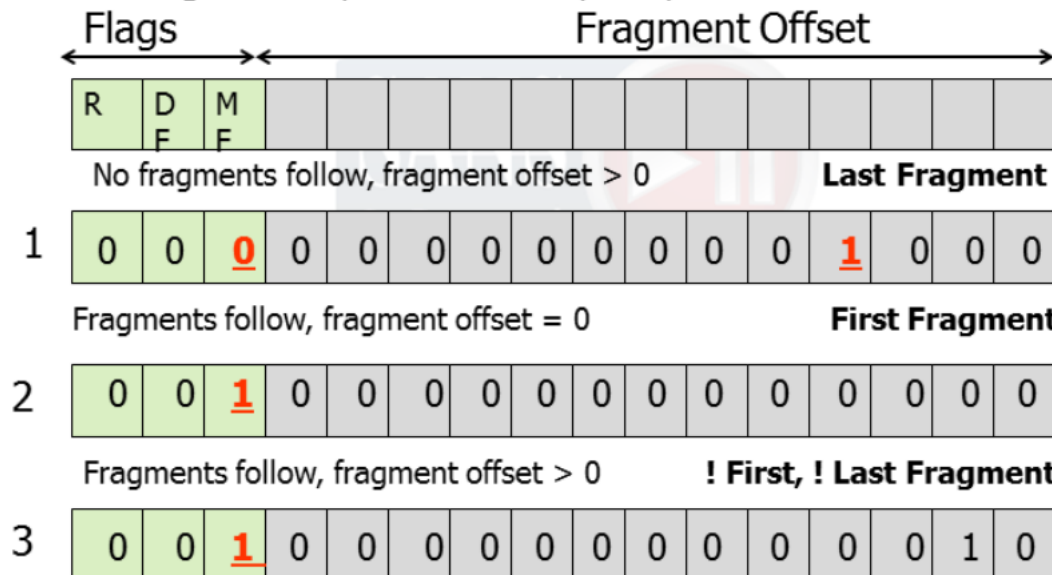
001:1 | 0000 | 0000 | 0000

## Last Fragment

```
0x0000: 4510 003c 0000 [1000] 3106 dfc4 c73b 962e
0x0010: 0aff 017f 01bb c5cc c2da 982d a6e4 875a
```

000:1 | 0000 | 0000 | 0000

If it is fragmented, is it the first, last, or neither first nor last?



RFC 3956 describes the R flag