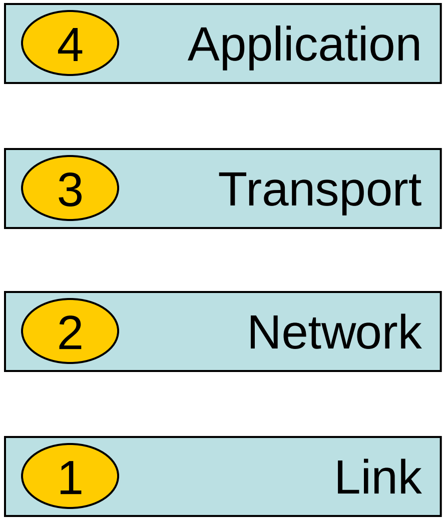
internet protocol suite

javed shaikh

/agenda

- Introduction
  - Networking technology: Ethernet
  - IP addressing
  - Address Resolution Protocols
- Transport layer
  - Transmission Control Protocol (TCP)
  - User Datagram Protocol (UDP)
- Network layer
  - Internet Protocol (IPv4)
  - Internet Control Message Protocol (ICMP)



4 Application

3 Transport

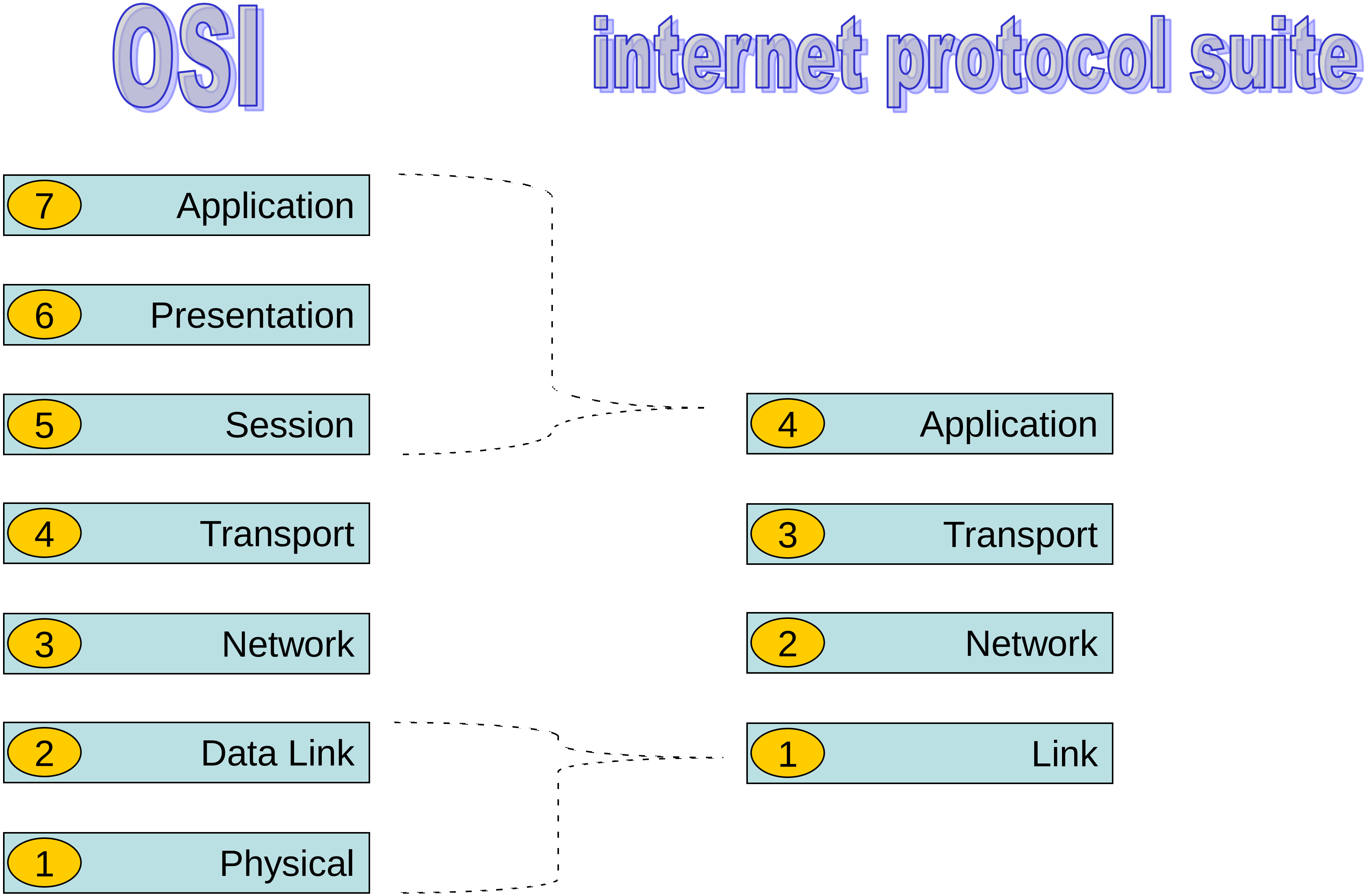
2 Network

1 Link

/references

- TCP/IP – Behrouz Forouzan, *McGraw Hill*
- TCP/IP Guide – Charles M. Kozierok, *No Starch Press* *www.tcpipguide.com*
- TCP/IP Illustrated Volume 1 (The Protocols) – W. Richard Stevens, *Addison-Wesley*

/overview/network layers



OSI

7 Application

6 Presentation

5 Session

4 Transport

3 Network

2 Data Link

1 Physical

internet protocol suite

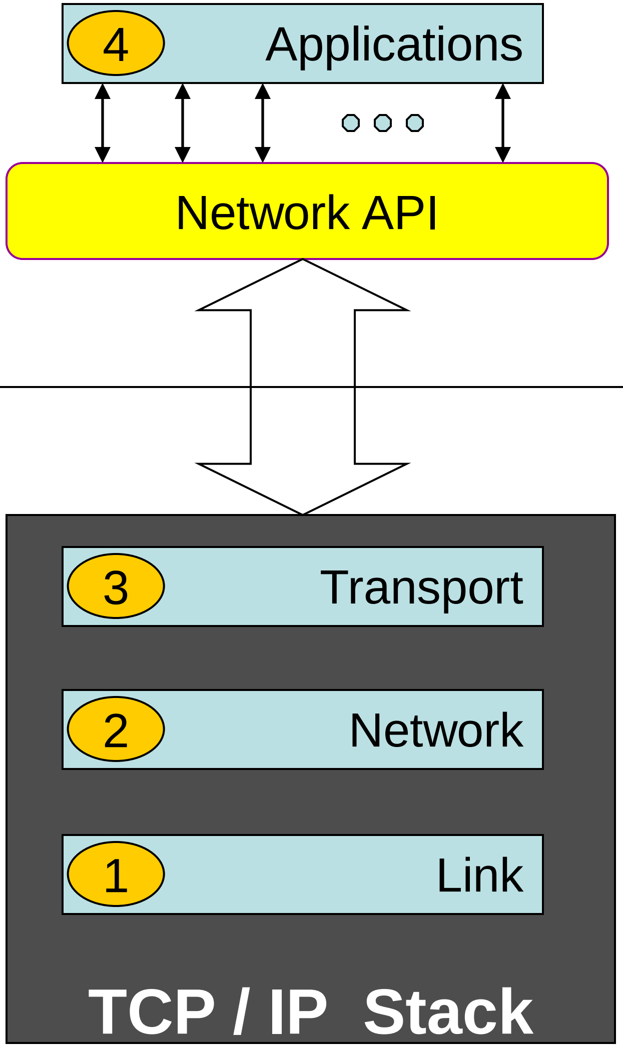
4 Application

3 Transport

2 Network

1 Link

/overview/stack



telnet, ftp, http, e-mail, chat, etc.

AT&T Transport Layer Interface, Berkeley sockets

user

kernel

TCP, UDP → protocol drivers

IP, ICMP → protocol drivers

Network cable, Network interface card, device driver

/overview/applications

User Applications	Administrative Utilities
telnet	hostname
ftp	ping
talk	traceroute
rlogin	arp
rsh	ifconfig
mozilla, iexplorer (http)	netstat
mail	

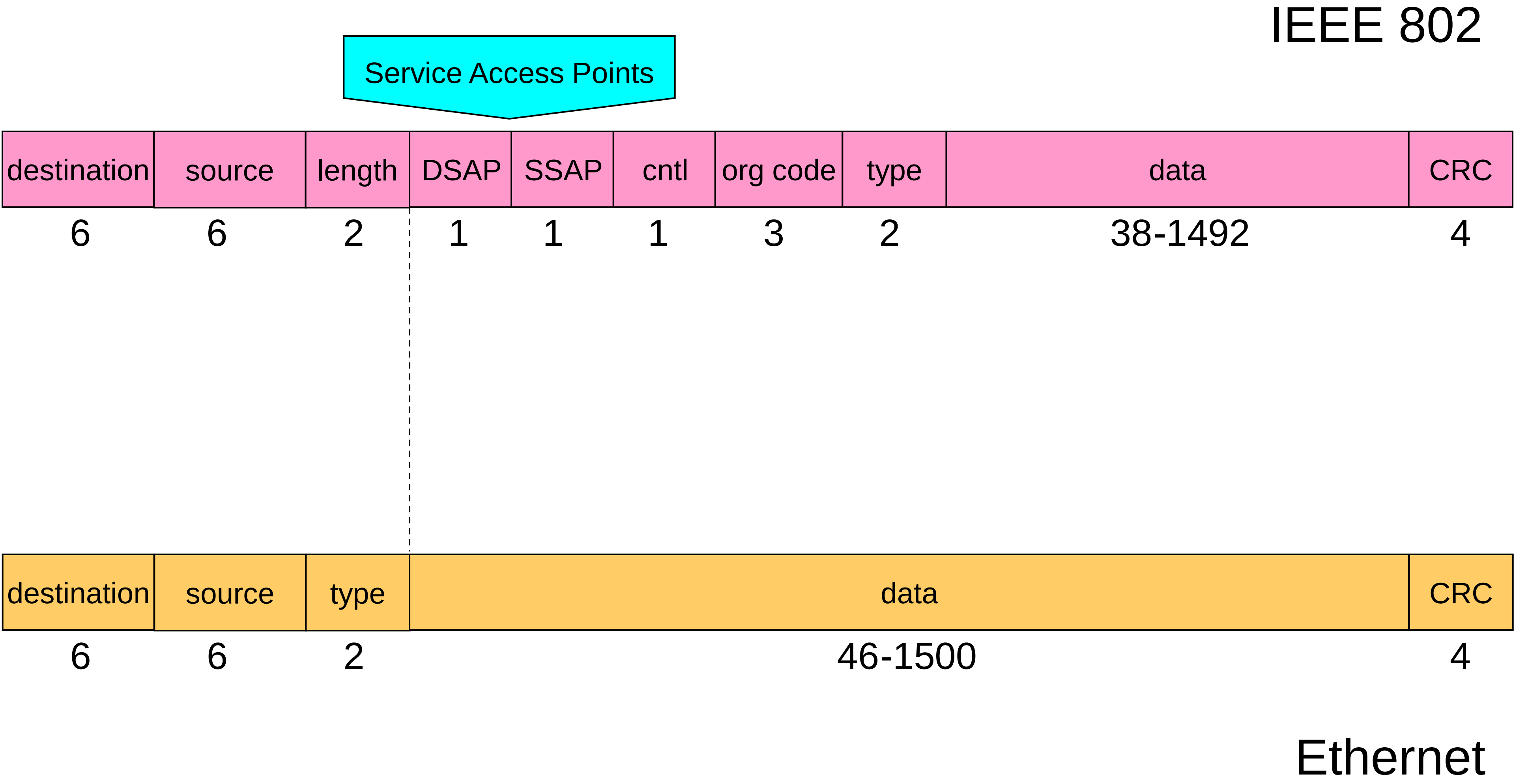
Link Layer

- Networking technologies
  - Token ring
  - ATM
  - FDDI
  - **Ethernet**
  - RS-232 serial line

/link layer/ethernet

- Inventors – DEC, Intel, Xerox
- Access method – CSMA/CD
- Speed – 10 (E) / 100 (FE) /
  1000 (GigE) / 10000 (10G)
- Address – 48 bits
- RFC 894 (Ethernet) *Vs* RFC 1042 (IEEE 802)
- Requirements from Internet host:
  - Must tx/rx Ethernet packets (default)
  - Should be able to rx IEEE 802 packets intermixed with Ethernet
  - May be able to send IEEE 802 packets. If sending both types, the type of packet sent must be configurable

/link layer/ethernet/differences



IEEE 802

Service Access Points

destination	source	length	DSAP	SSAP	cntl	org code	type	data	CRC
6	6	2	1	1	1	3	2	38-1492	4

destination	source	type	data	CRC
6	6	2	46-1500	4

Ethernet

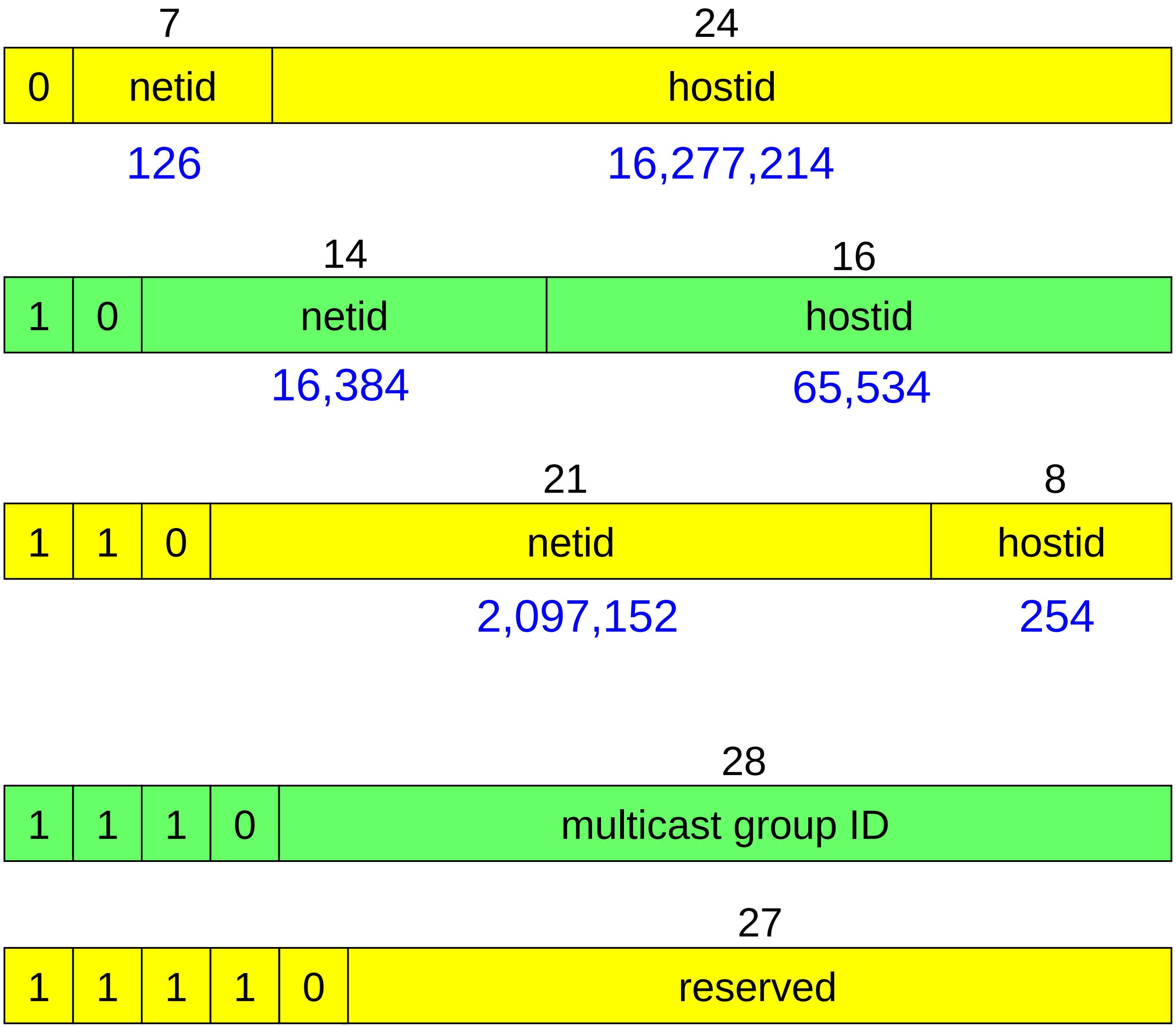
Minimum 64 byte frame required…

/network layer/ip/addressing

- Every interface must have a unique IP address
- 32 bit, dotted-decimal notation
- 3 types
  - *Unicast, broadcast, multicast*
- 5 classes

A	0.0.0.0	127.255.255.255
B	128.0.0.0	191.255.255.255
C	192.0.0.0	223.255.255.255
D	224.0.0.0	239.255.255.255
E	240.0.0.0	247.255.255.255

/network layer/ip/addressing



Class A

7 24

0 | netid | hostid

126 16,277,214

Class B

14 16

1 | 0 | netid | hostid

16,384 65,534

Class C

21 8

1 | 1 | 0 | netid | hostid

2,097,152 254

Class D

28

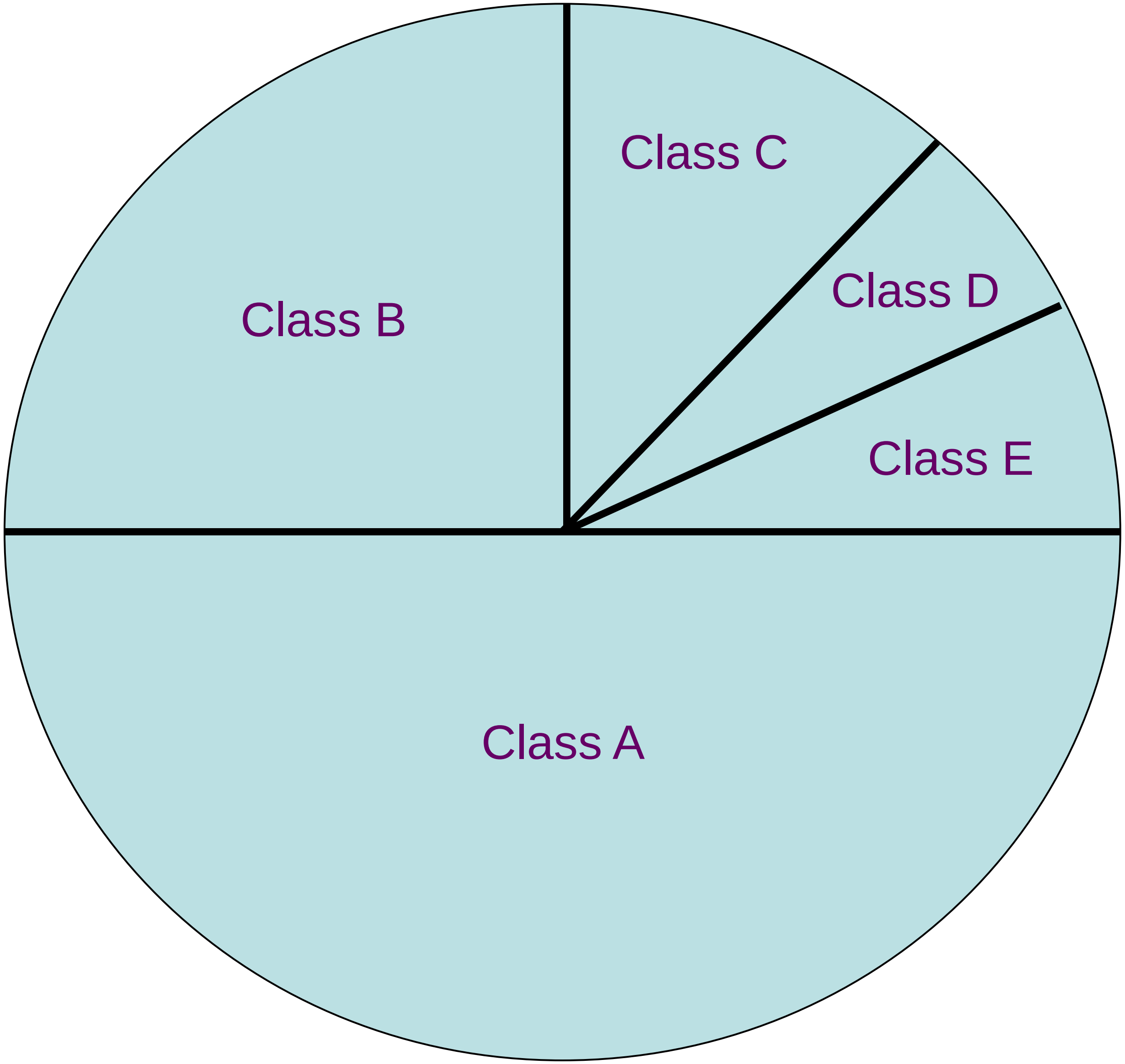
1 | 1 | 1 | 0 | multicast group ID

Class E

27

1 | 1 | 1 | 1 | 0 | reserved

/network layer/ip/addressing

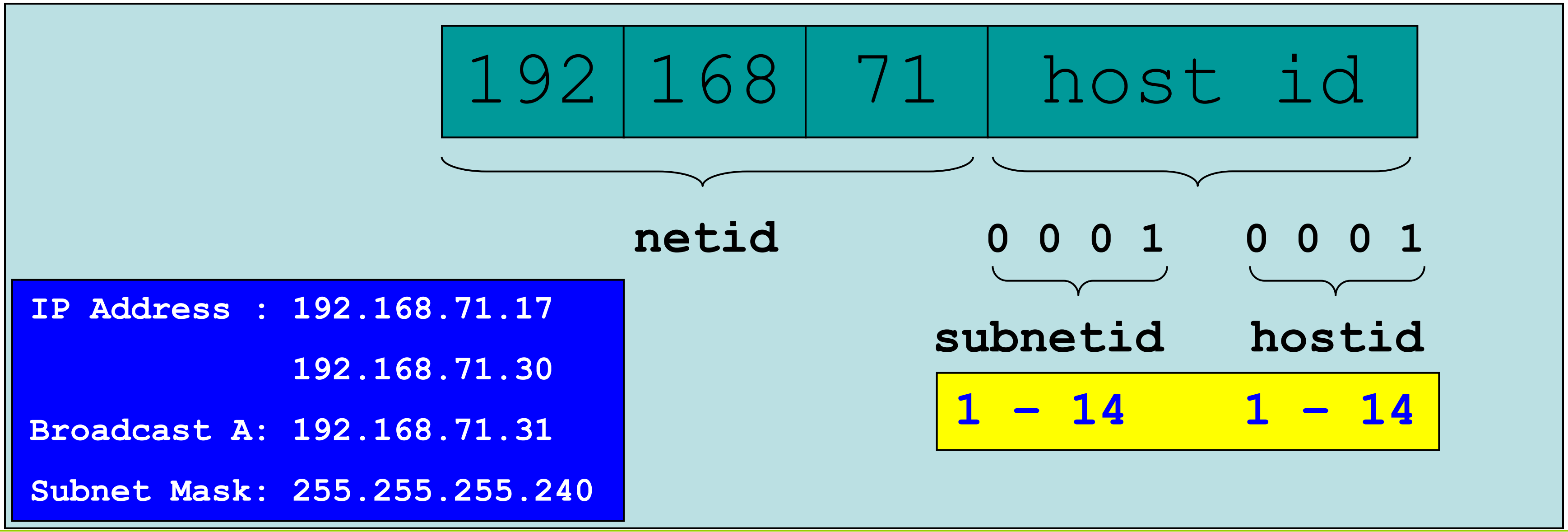


/network layer/ip/addressing/reserved

Start	End	Comment
0.0.0.0	0.255.255.255	-
10.0.0.0	10.255.255.255	Class A private addr block
127.0.0.0	127.255.255.255	Loopback addr block
128.0.0.0	128.0.255.255	-
169.254.0.0	169.254.255.255	Class B private addr block for auto addr allocation
172.16.0.0	172.31.255.255	Private addr block
191.255.0.0	191.255.255.255	-
192.0.0.0	192.0.0.255	-
192.168.0.0	192.168.255.255	Class C private addr block
223.255.255.0	223.255.255.255	-

/network layer/ip/addressing/net & subnet

- Class C IP address → 192.168.71 (254 hosts possible)
- Network Mask → 255.255.255.0
  Network Mask specifies the netid
- Valid hostid → 1 to 254
- Broadcast Address → 192.168.71.255



192 | 168 | 71 | host id

netid

0 0 0 1 0 0 0 1

subnetid hostid

1 – 14 1 – 14

```
IP Address : 192.168.71.17
             192.168.71.30
Broadcast A: 192.168.71.31
Subnet Mask: 255.255.255.240
```

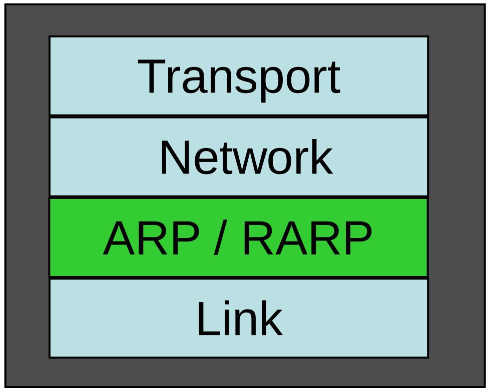
/network layer/ip/addressing/special addrs

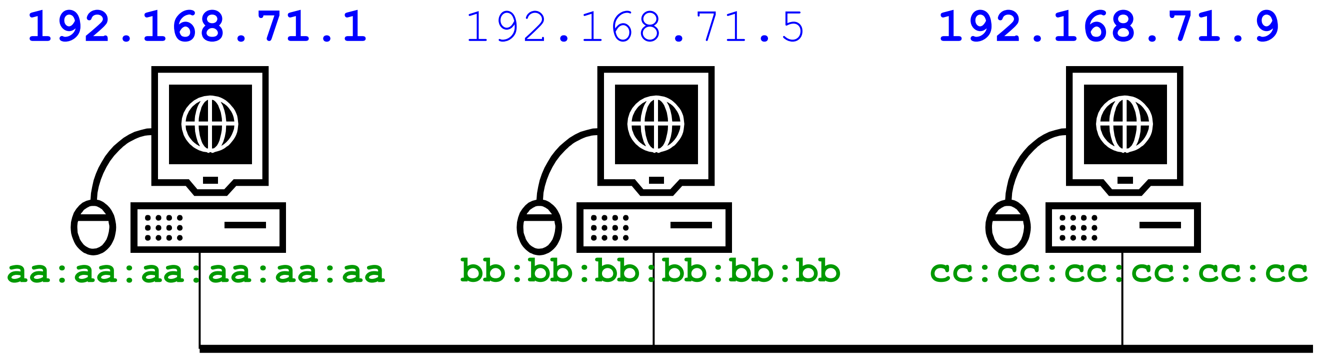
Netid	Subnetid	Hostid	Valid as
0		0	src
0		>0	src
127		**any**	**src / dest**
-1		-1	dest
netid		-1	dest
netid	subnetid	-1	dest
netid	-1	-1	dest

/network → link layer/address mapping/introduction

- A link can be used by any network layer protocol
- An Ethernet link has its own addressing scheme and uses the same to send a frame to destination host
- IP addresses make sense only to network layer and above
- Hence a mechanism to map IP address to a MAC address is required
  - ARP : Address Resolution Protocol
- Similarly a mechanism to obtain an IP address for a MAC address is
  - RARP : Reverse Address Resolution Protocol

/network → link layer/arp





source	destination	type	data	CRC
6	6	2	46-1500	4

R/ARP Packet

hard type	proto type	HAL	PAL	op	src ether addr	src IP addr	dest ether addr	dest IP addr
2	2	1	1	2	6	4	6	4

aa:aa:aa:	**ff:ff:ff:…**	0806	1	800	6	4	1	aa:aa:aa…	192.168.71.1	ff:ff:ff:…	192.168.71.9	CRC

cc:cc:cc:	**aa:aa:aa:**	0806	1	800	6	4	2	cc:cc:cc:..	192.168.71.9	aa:aa:a…	192.168.71.1	CRC

/network → link layer/arp/details

- ARP request is broadcast and reply is unicast
- ARP cache
  - Timeout = 1 to 20 mins. For Linux check:

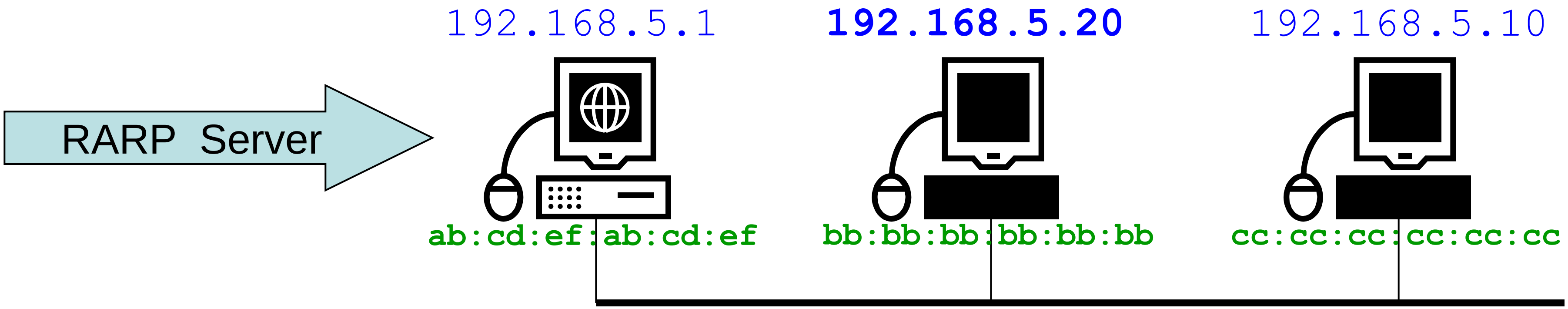
    /proc/sys/net/ipv4/neigh/eth0/gc_stale_time
- ARP request to non-existent host
- Proxy ARP (*For hosts on different networks at the data link layer level, but on the same IP network or subnet*)
- Gratuitous ARP
  - Detects duplication of IP address
  - Updating of older hardware address
- ARP is part of the kernel’s TCP/IP implementation

/network → link layer/arp/assignments

- Use *arp* command on linux system with following options:
  - a : to display all the entries in ARP cache
  - d : to delete an entry from ARP cache
  - s : to add an entry in ARP cache
- Trap and display ARP packets using *tcpdump* (requires root privilege)
  - For a host which does not have its entry in the arp cache
  - For a host which has an entry in the arp cache

/network → link layer/rarp

- A computer acquires its IP address from a file stored on it during bootstrapping procedure. What about a computer that is diskless?



bb:bb:bb:	**ff:ff:ff:…**	8035	1	800	6	4	3	bb:bb:bb…	-	ff:ff:ff…	-	CRC

ab:cd:ef:	**bb:bb:bb..**	8035	1	800	6	4	4	ab:cd:ef…	192.168. 5.1	bb:bb:bb…	192.168. 5.20	CRC

/network → link layer/rarp/details

- RARP request is broadcast and reply is unicast
- RARP requests are sent as hardware-level broadcasts, hence they are not forwarded by routers
- Multiple RARP servers need to be maintained to provide redundancy
- RARP server is implemented as user process. Also this implementation is tied to the system
- Made obsolete by Bootstap Protocol (BOOTP) which is now replaced by Dynamic Host Configuration Protocol (DHCP)

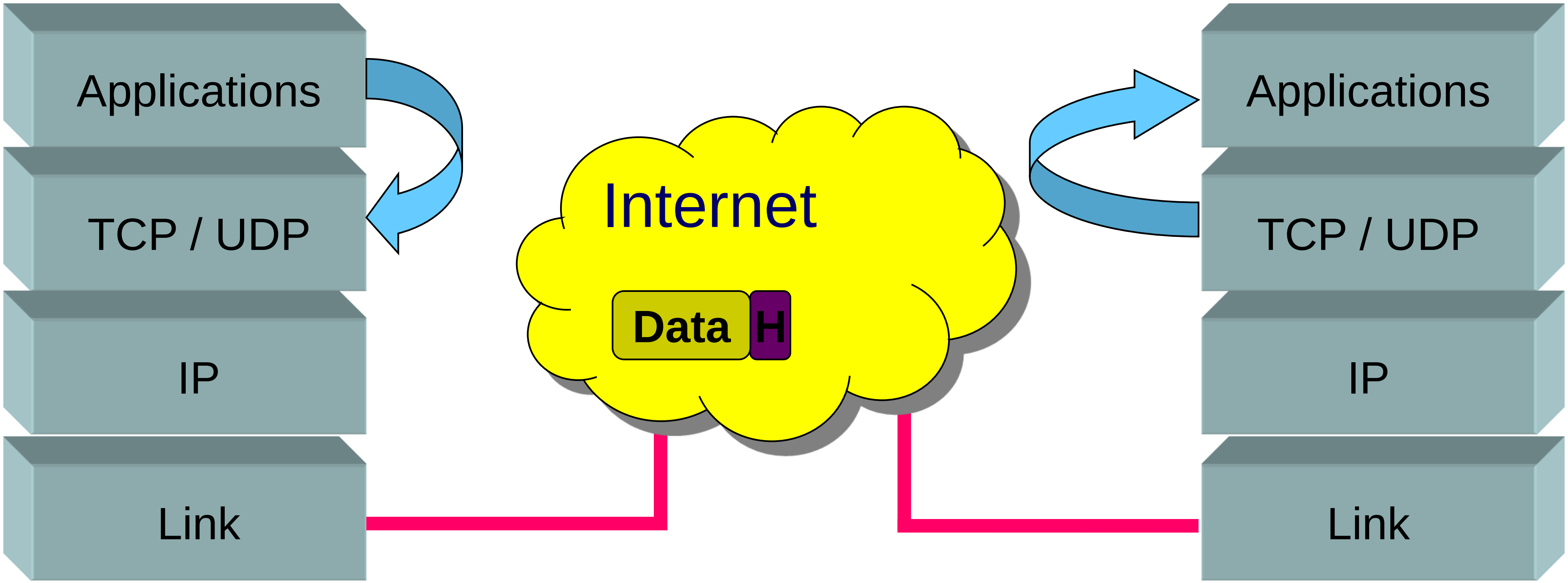
Terminology

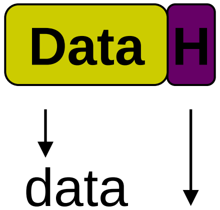
/terms

- Protocol – A *networking protocol* defines a set of rules and messages that enable software and hardware in networked devices to communicate effectively
- Mbps, MBps
- Frame, Segment, Datagram, Packet, Data

Transport Layer

/transport layer





TCP Segment / UDP Datagram

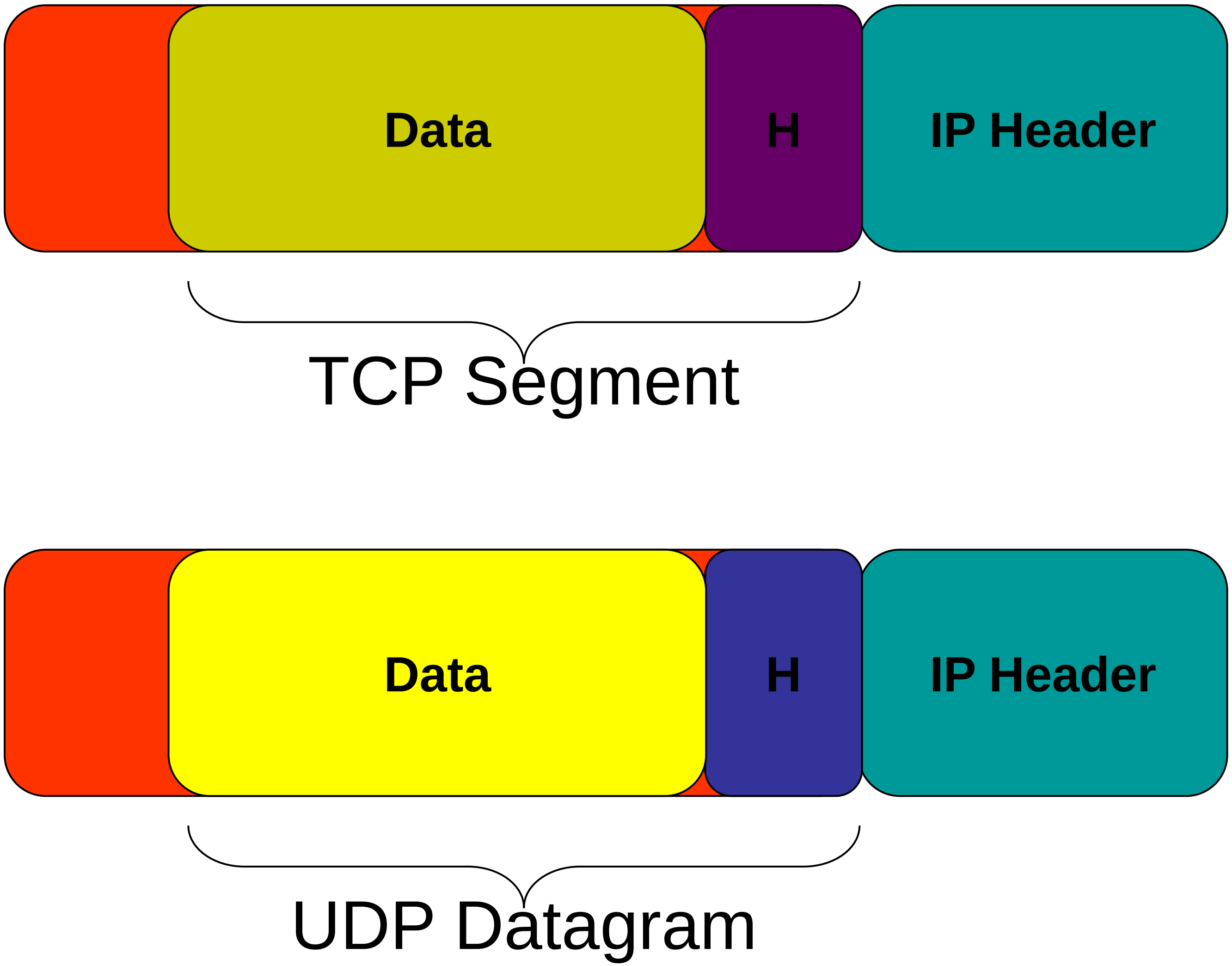
data

header – Contains control and addressing information

/transport layer

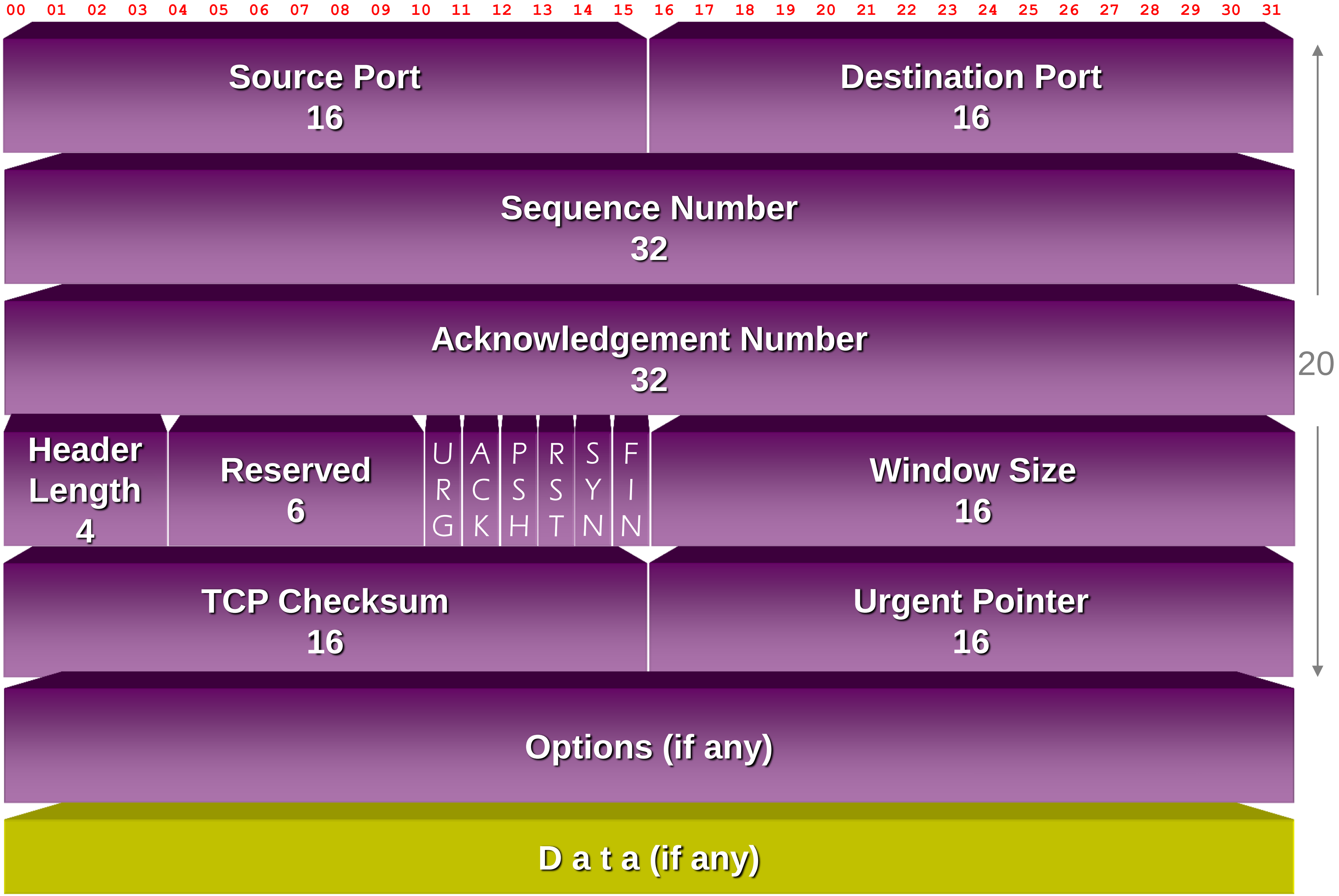
- Provides a flow of data between two hosts for the application layer above
- ***TCP:***
  - Provides reliable flow of data between two hosts
  - Divides data passed to it from application into appropriate sized chunks for the network layer below
  - Acknowledges received packets
  - Sets timeouts for acks, etc.
- ***UDP:***
  - Sends packets of data called **datagrams** from one host to another, but gives no guarantee that it will reach
  - Reliability if desired must be added by the application layer

/transport layer/encapsulation



- Unit of information sent by TCP is called a *segment*
- Application data is broken into best sized chunks to send
- TCP maintains a timer for receiving an ack from the receiver
- Receiver sends an ack to the sender on receipt of data
- A checksum is maintained on header as well as data
- A receiving TCP re-sequences the segments if they arrive out of order
- Duplicate segment is discarded
- It also provides flow control

/transport layer/tcp/header



00 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31

Source Port 16

Destination Port 16

Sequence Number 32

Acknowledgement Number 32

Header Length 4

Reserved 6

URG ACK PSH RST SYN FIN

Window Size 16

TCP Checksum 16

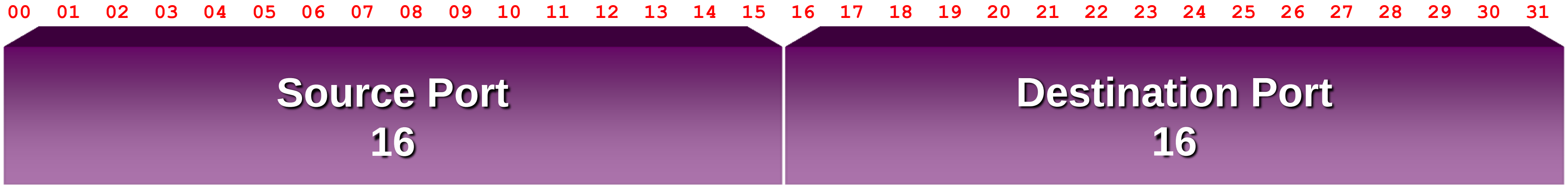
Urgent Pointer 16

20

Options (if any)

D a t a (if any)

/transport layer/tcp/header/port



- A port number identifies sending and receiving application
- System port numbers: 1 – 1023 reserved for root (*Managed by IANA*)
- User port numbers 1024 – 49151 are available for user applications (Can be registered with IANA)
- 49152 – 65535 Private/Dynamic port numbers

/transport layer/tcp/header/port

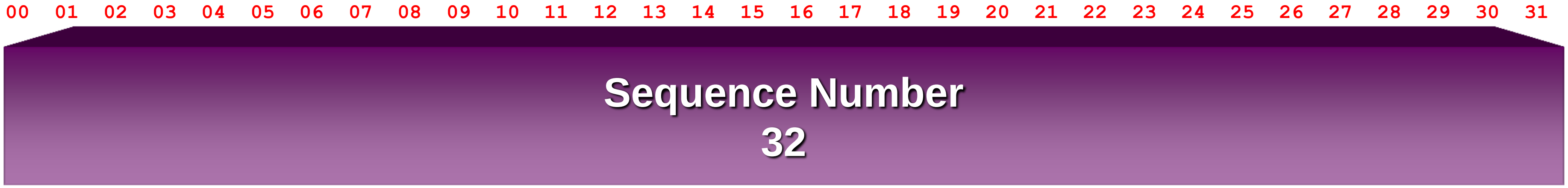
00 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31

Source Port 16	Destination Port 16

telnet	23
ftp	21
smtp	25
tftp	69

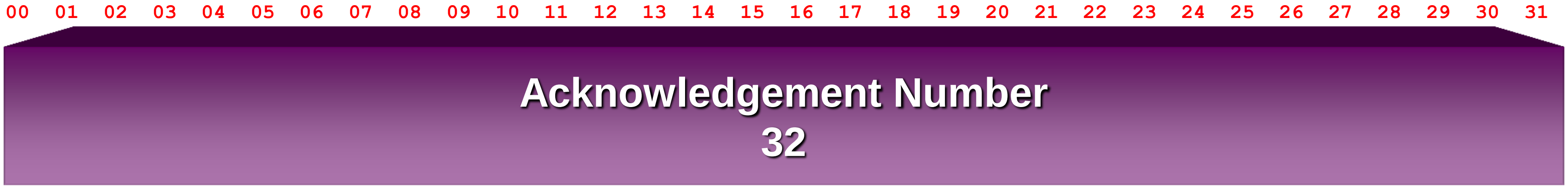
- An IP addre
- Port numbe ion
- The src/ds nd src/dst IP addresses ntify each connectio
- *Socket:* co P addr, port number
- Well known applications and their port numbers are kept in /etc/services

/transport layer/tcp/header/sequence number



- TCP numbers each byte with a sequence number
- “Sequence Number” field specifies the first byte number from the data being sent to the receiving TCP
- A random *initial sequence number* (ISN) is chosen by the host when a new connection is established
- Sequence number wraps back to 0 after reaching $2^{32} – 1$

/transport layer/tcp/header/acknowledgement number



00 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31

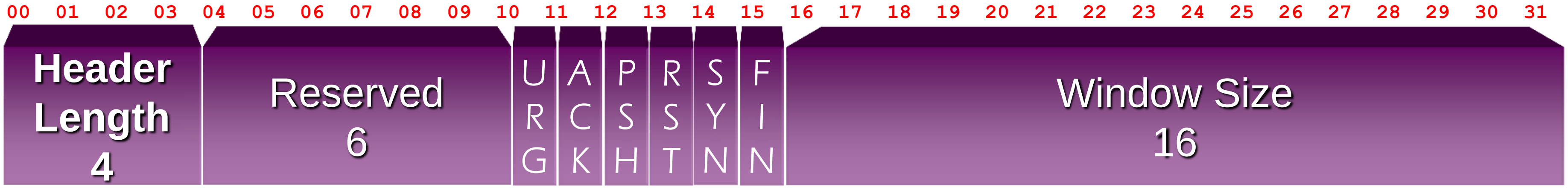
Acknowledgement Number
32

- Used by receiver to acknowledge the number of bytes received so far by specifying the next byte number it expects

- This field is valid only if ACK flag is set

- Each end of connection maintains sequence number of the data flowing in each direction

/transport layer/tcp/header/length



00 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31

Header Length 4

Reserved 6

URG

ACK

PSH

RST

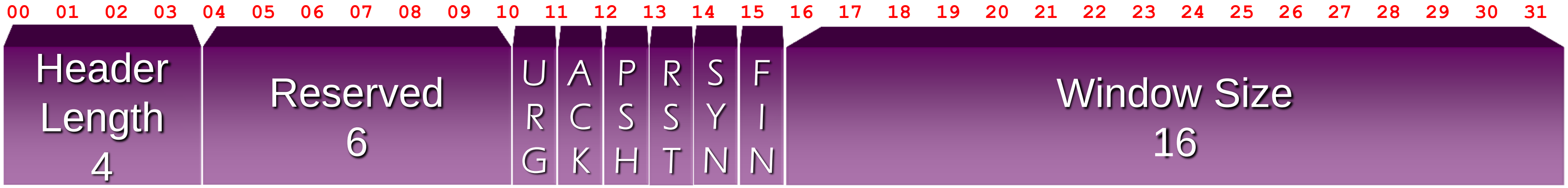
SYN

FIN

Window Size 16

- Header length is in 32-bit words
- Length of options field is variable
- Max length = 60, Min = 20 bytes
- Header always has to be in multiple of 32-bit words:
  - If its not, options have to be padded

/transport layer/tcp/header/flags

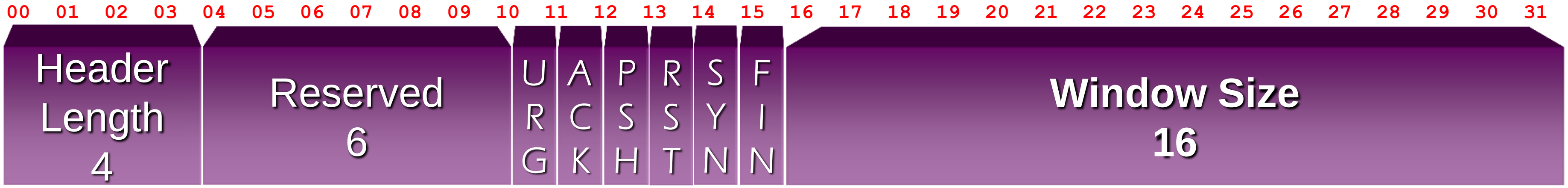


00 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31

Header Length 4	Reserved 6	URG	ACK	PSH	RST	SYN	FIN	Window Size 16

- URG – *The urgent pointer is valid (usage: interrupting data transfer)*
- ACK – *Acknowledgement number is valid*
- PSH – *The receiver should pass this data to the application as soon as possible (eg. telnet)*
- RST – *Reset the connection*
- SYN – *Synchronize sequence number to initiate connection*
- FIN – *The sender is finished sending data*

/transport layer/tcp/header/window size



00 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31

Header Length 4

Reserved 6

URG

ACK

PSH

RST

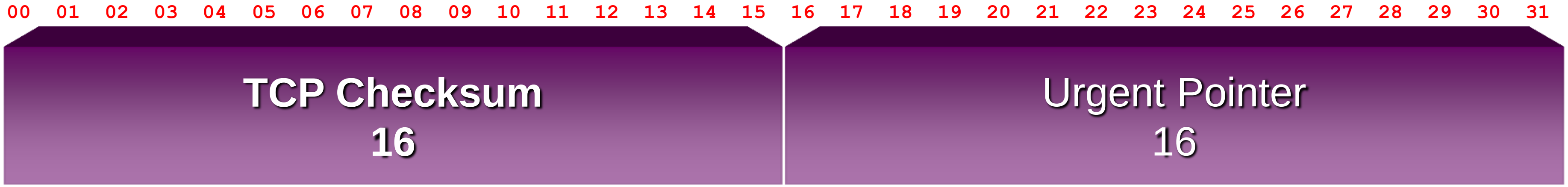
SYN

FIN

Window Size 16

- TCP uses this window to advertise how many bytes it can receive at a time
- Useful for flow control
- Sliding window protocol based on this

/transport layer/tcp/header/checksum



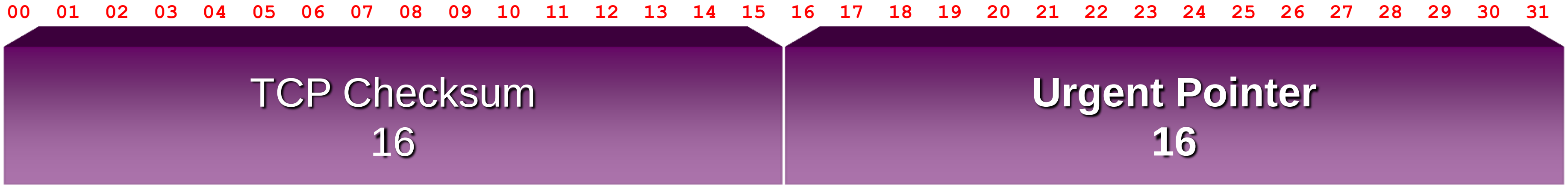
00 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31

TCP Checksum
16

Urgent Pointer
16

- Mandatory field
- Covers header and data
- Uses a pseudo-header for calculation:
  - IP Source and Destination Address fields
  - IP Protocol field
  - TCP Length field

/transport layer/tcp/header/urgent pointer

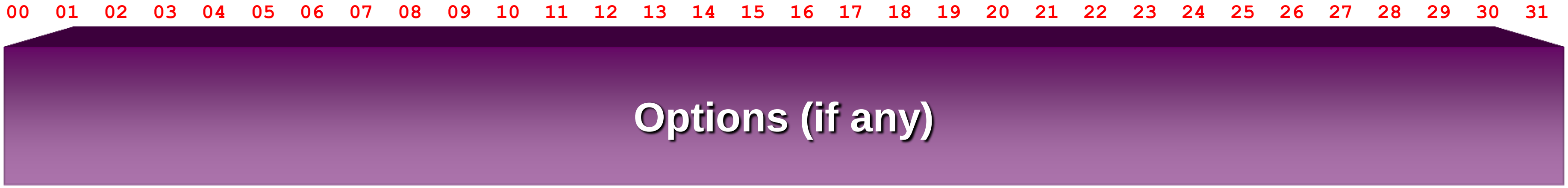


00 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31

TCP Checksum 16	**Urgent Pointer** **16**

- Urgent pointer is a positive offset that must be added to sequence number, to yield the sequence number of the last byte of urgent data
- Used for transmitting emergency data
- Up to the application as to how the urgent data is to be used
- Urgent pointer is valid only if URG flag is set to 1

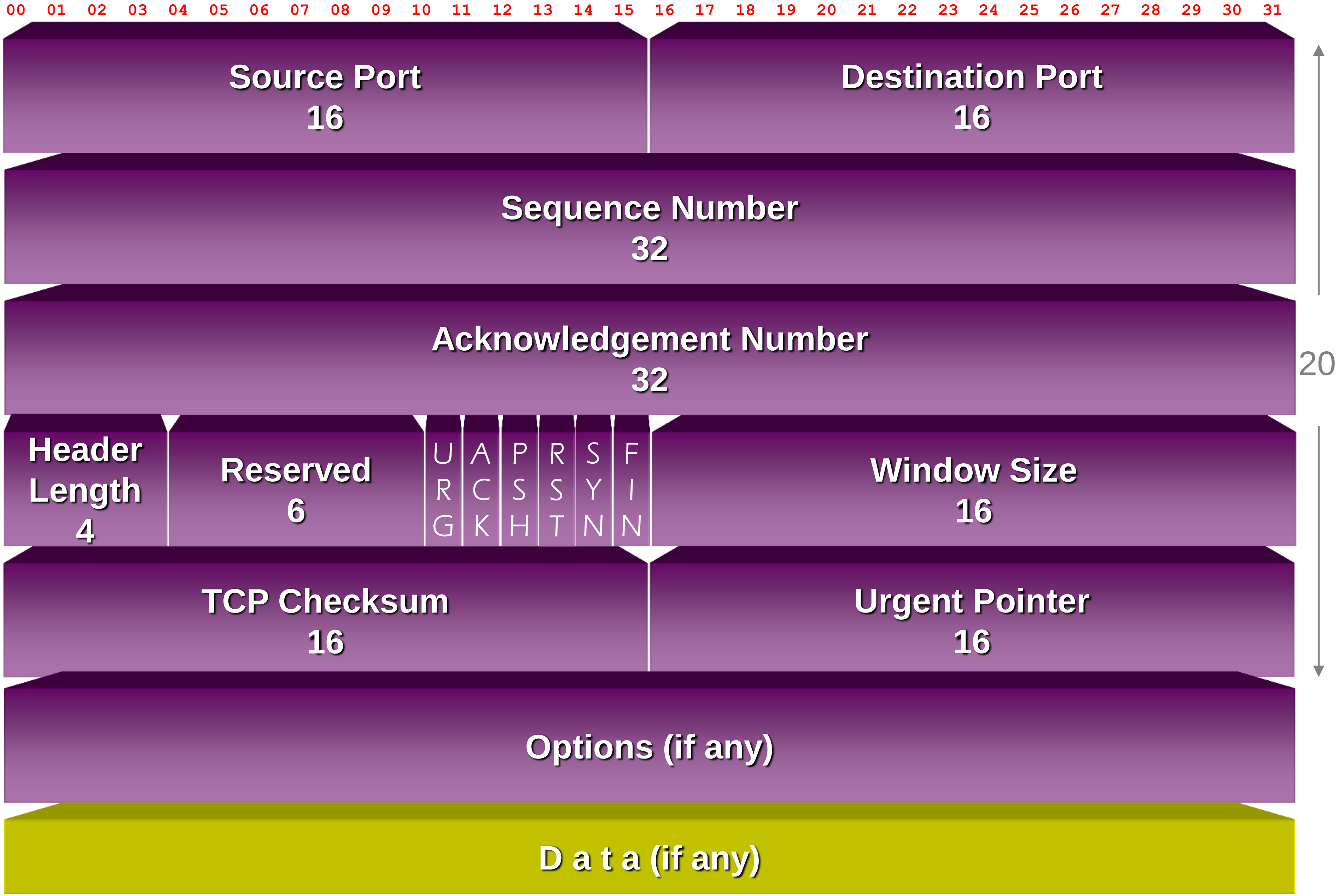
/transport layer/tcp/header/options



- Maximum segment size
- Window scale factor
- Timestamp
- End of option list
- No operation

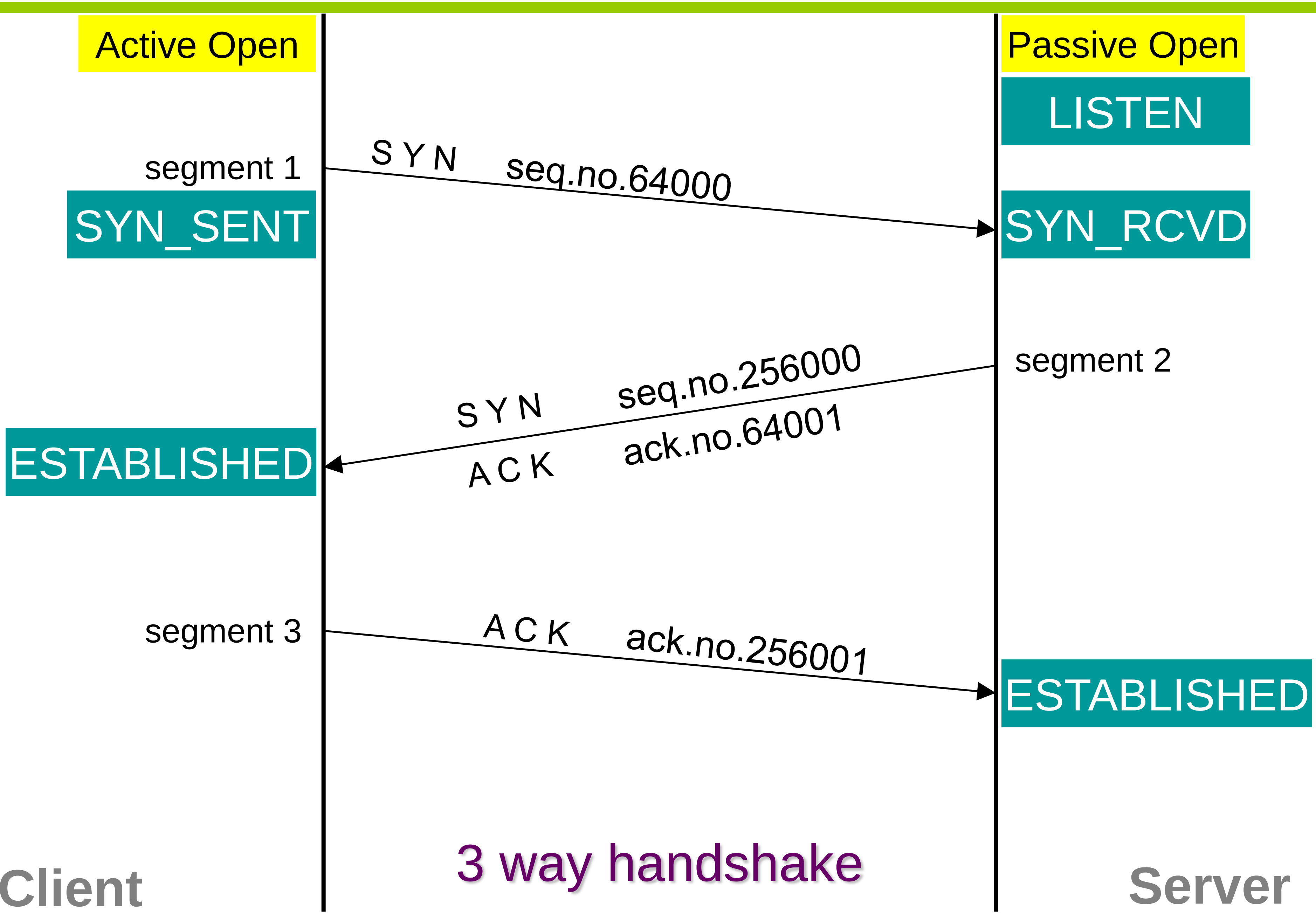
Option-Kind	Option-Length	Option-Data
(1)	(1)	(var)

/transport layer/tcp/header/data



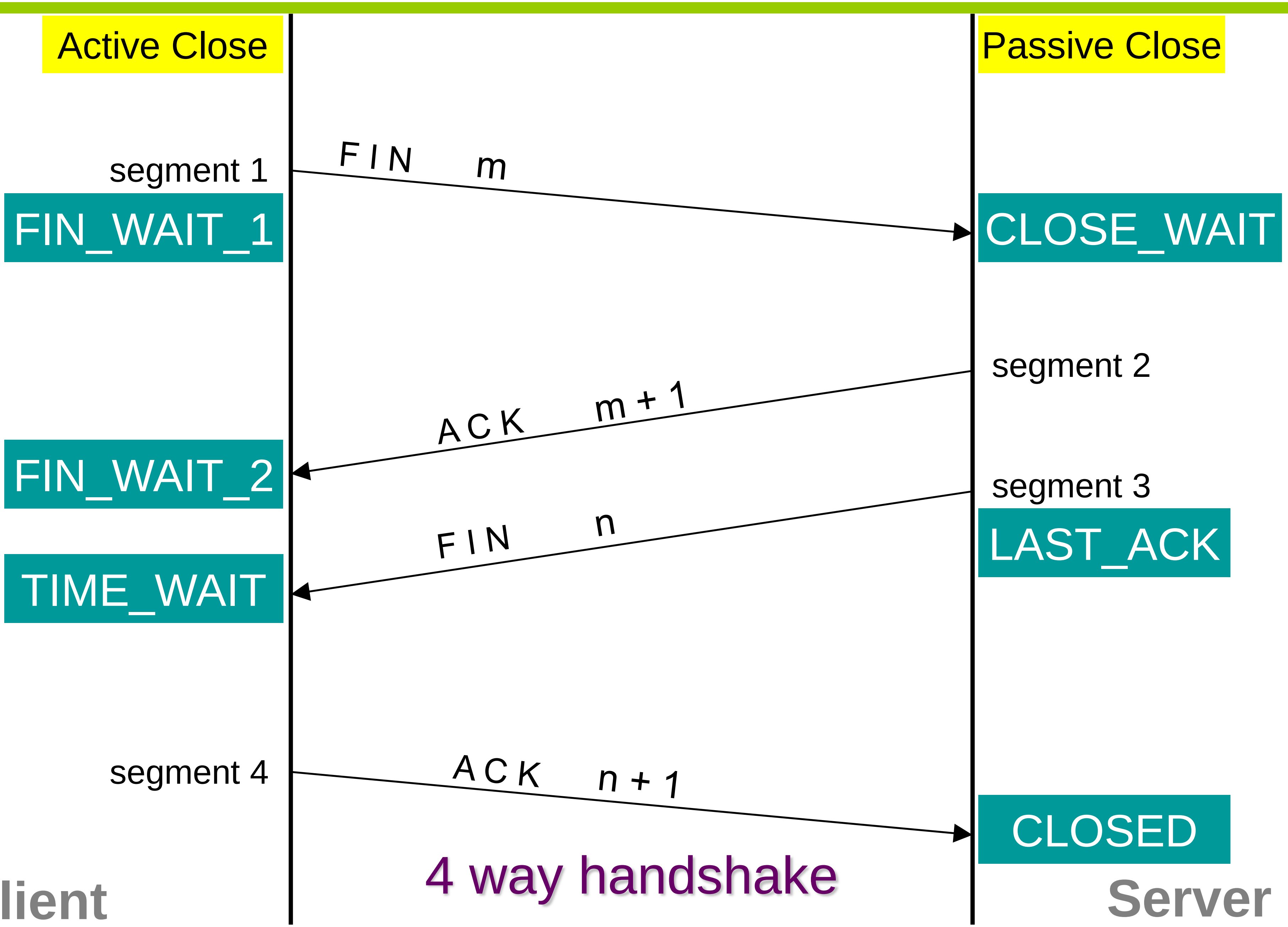
Source Port 16								Destination Port 16
Sequence Number 32								
Acknowledgement Number 32								
Header Length 4	Reserved 6	URG	ACK	PSH	RST	SYN	FIN	Window Size 16
TCP Checksum 16								Urgent Pointer 16
Options (if any)								
Data (if any)								

/transport layer/tcp/connection/open



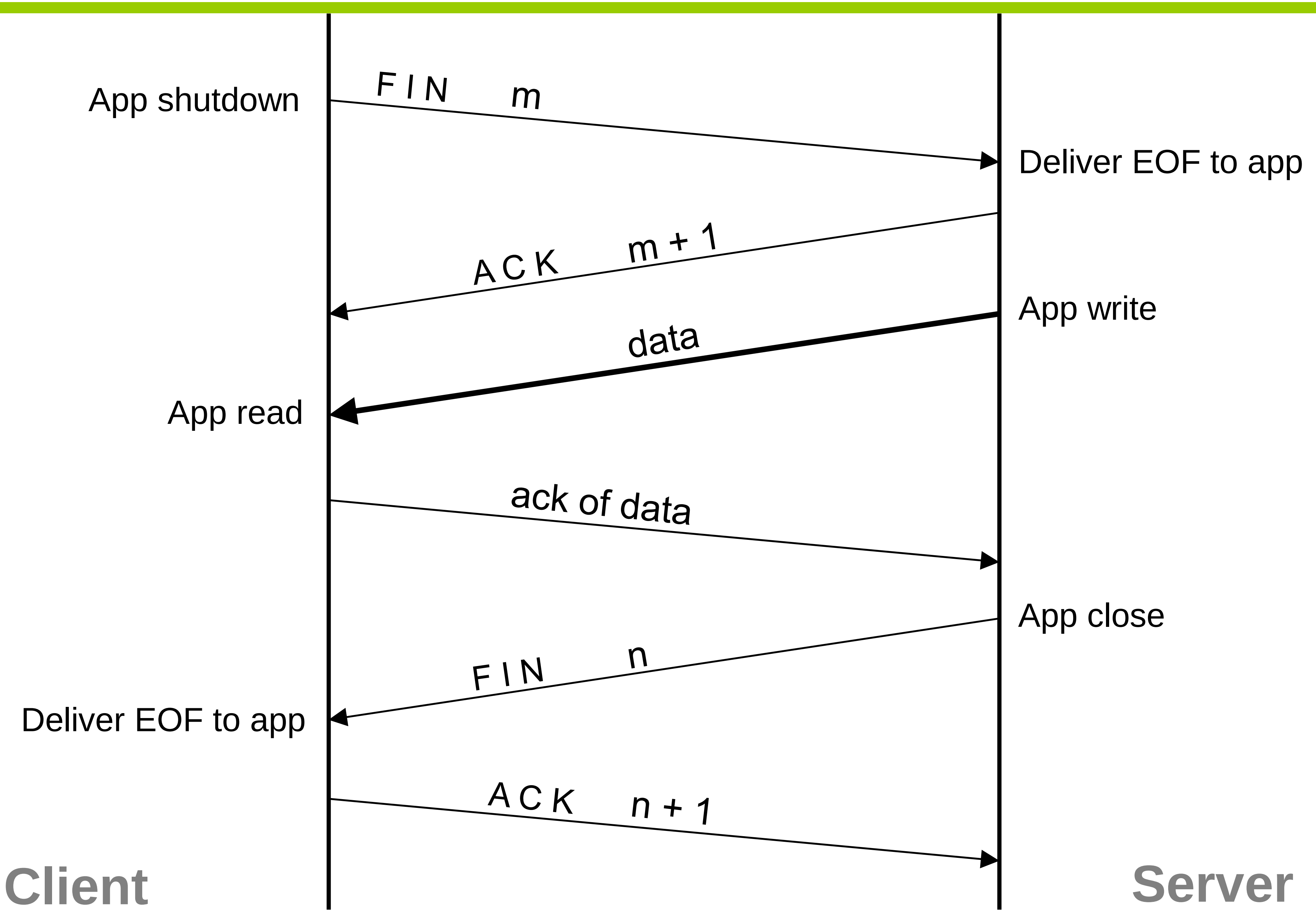
3 way handshake

/transport layer/tcp/connection/close

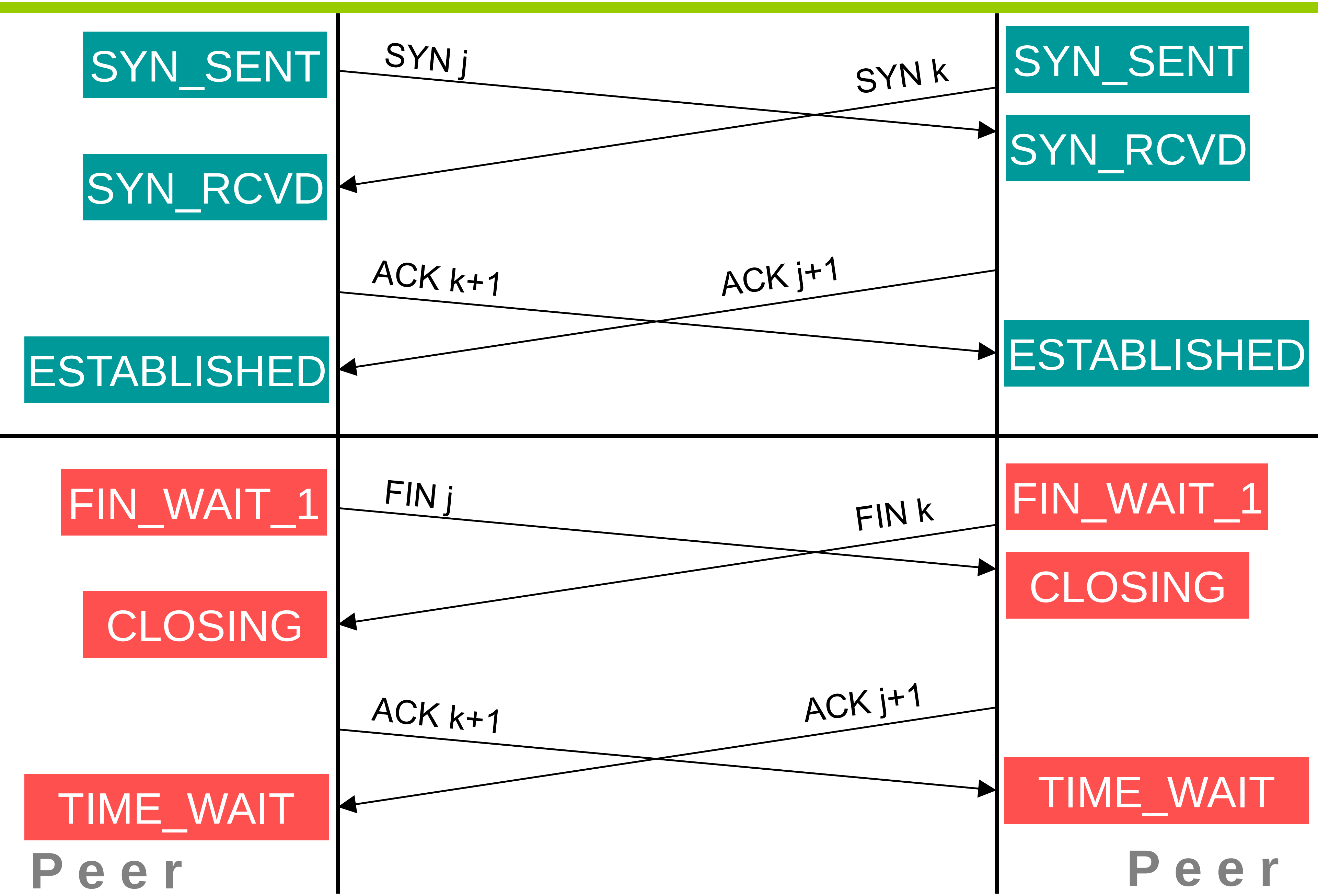


4 way handshake

/transport layer/tcp/connection/half close



/transport layer/tcp/connection/simultaneous



- Connection can’t be established if remote host is down
- There is a timeout set for a retry
- Maximum segment lifetime (MSL): *It is the max amount of time any segment can exist in the network before being discarded*
- A connection is reset:
  - If request arrives and no process is listening on the destination port
  - By sending a reset (*abortive release*)
- A TCP connection is said to be *half-open* if one end has closed or aborted the connection without the knowledge of the other end

/transport layer/tcp/connection/queue

- There's a fixed length queue of connection
- Backlog is between 0 – 5
- Connections in a queue are already accepted by TCP, they are waiting to be accepted by the application
- If there is no room in the queue, TCP simply ignores the SYN from incoming connection

- Interactive (*E.g. rlogin, telnet*)
  - – Normally transmitted in segments smaller than the max segment size
  - – Delayed acks are piggybacked by receiver along with data going back to sender over WAN
- Bulk
  - – Sliding window protocol: Receiver does not have to acknowledge every received packet
  - – The acks are cumulative
  - – Window is advertised by receiver

/transport layer/tcp/data flow factors

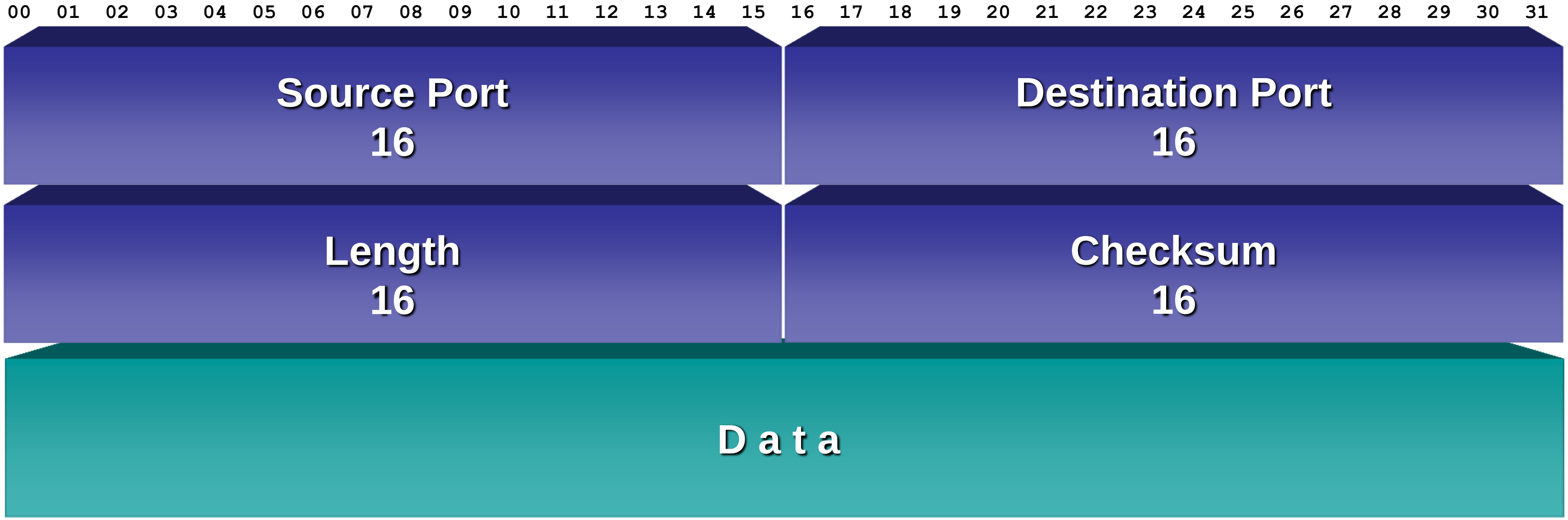
- PUSH Flag
- Slow start
- URGENT Flag
- Congestion

/transport layer/tcp/assignment

1. Trap TCP packets using *tcpdump*
2. Trap TCP packets belonging to specific application (telnet, ftp, etc.) between any two hosts
3. Use *netstat* to find how many TCP sockets are open and what their states are

- Created as an alternative transport protocol for applications that don’t need the features of TCP
- Proposed in RFC 768 in 1980
- Serves as an interface between application processes and IP
- Simple and fast

/transport layer/udp/header



00 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31

Source Port 16	Destination Port 16
Length 16	Checksum 16
Data	

- Checksum field is optional
- Checksum is computed for actual header + pseudo header comprising of:
  - IP Source and Destination Address fields
  - IP Protocol field
  - UDP Length field

/transport layer/udp/does not

- Establish connections before sending data. It packages the data and sends it off

- Provide acks
- Provide guarantee of reception

- Detect lost messages and retransmit them

- Ensure data ordering
- Provide any mechanism to handle congestion or manage the flow of data between devices

/transport layer/udp/applications

Port#	Keyword	Protocol	Comments
53	domain	Domain Name Server	Uses a simple request / reply messaging system
67 / 68	bootps / bootpc	Bootstrap protocol & DHCP	Host configuration protocols
69	tftp	Trivial File Transfer Protocol	For quick and easy transfer of small files
161 / 162	snmp	Simple Network Management Protocol	An administrative protocol
520 / 521	router / ripng	Routing Information Protocol (RIP-1, RIP-2, RIPng)	Routing protocols
2049	nfs	Network File System	Used UDP earlier. New versions use TCP

/transport layer/udp/assignment

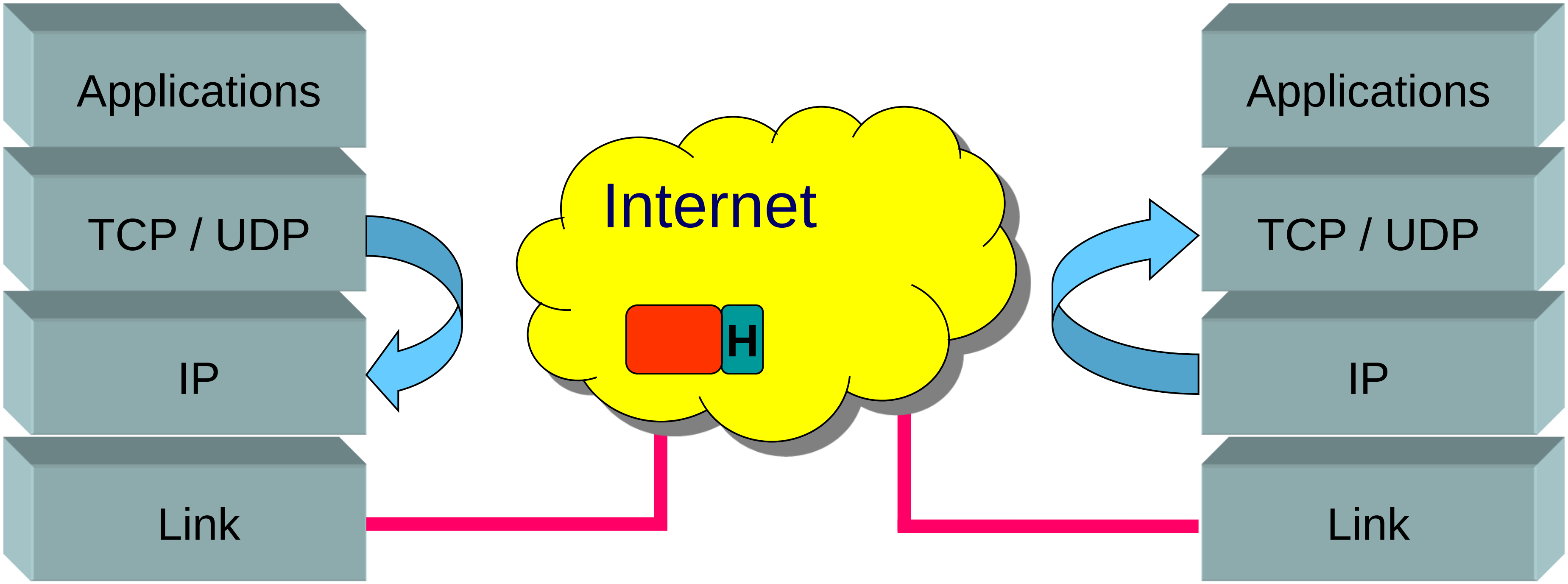
1. Trap UDP datagrams using *tcpdump*

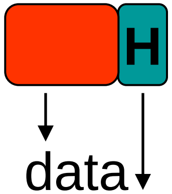
2. Trap UDP datagrams and dump the contents for analysis of its header

Network Layer

- This layer is responsible for routing messages through networks
- IP is a connectionless protocol that doesn't provide reliability, flow control or error recovery (These functions must be provided at a higher level)
- It offers a best effort service. If something goes wrong, IP discards the datagram and tries to send an ICMP message to the source host
- A message unit in an IP network is called an IP *datagram*. This is the basic unit of information transmitted across TCP/IP networks

/network layer/ip



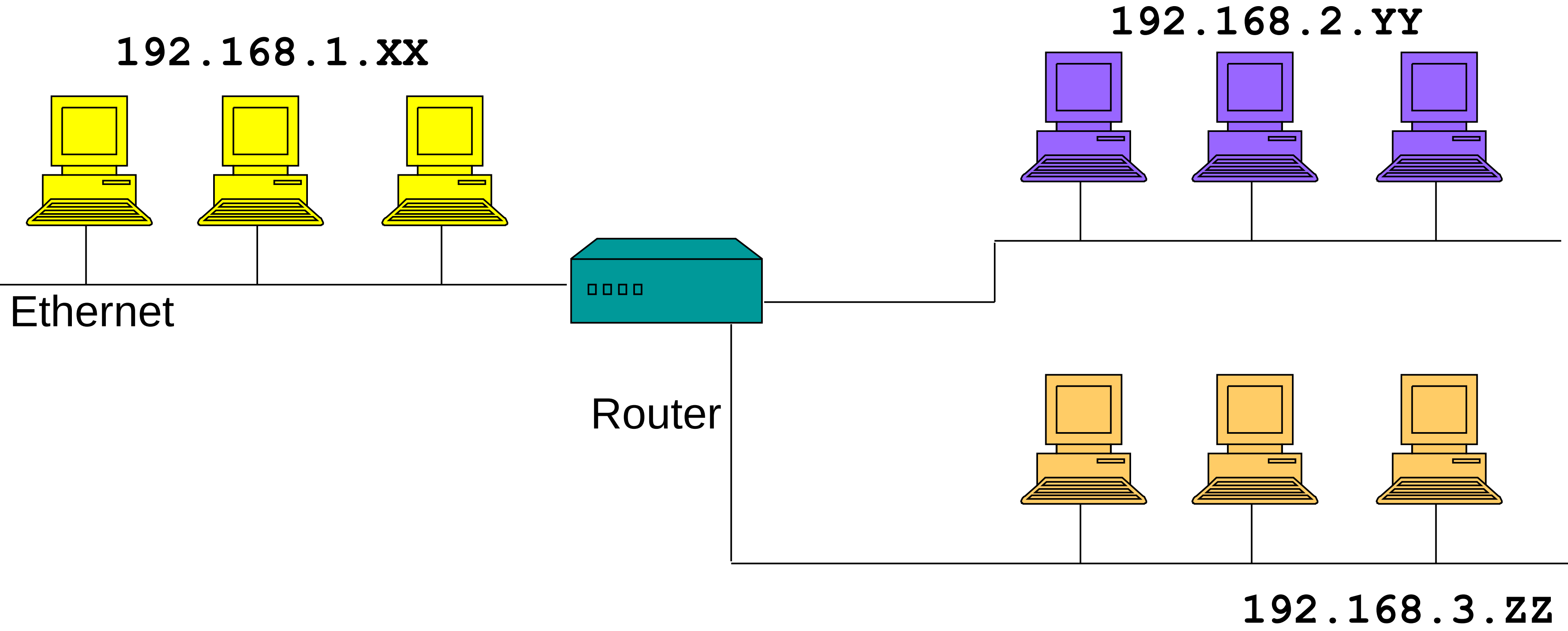


IP Datagram

data

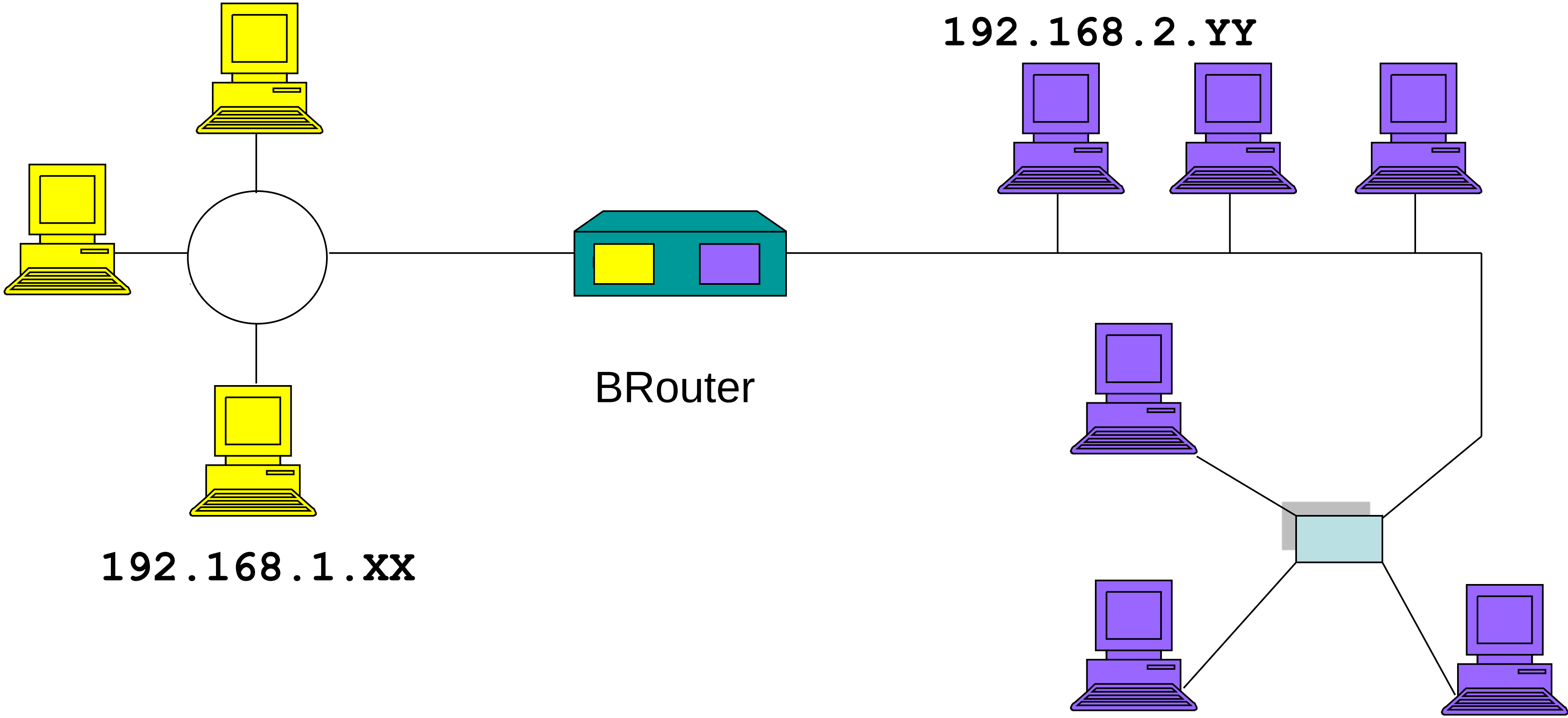
header – Contains control and addressing information

/network layer/ip/devices



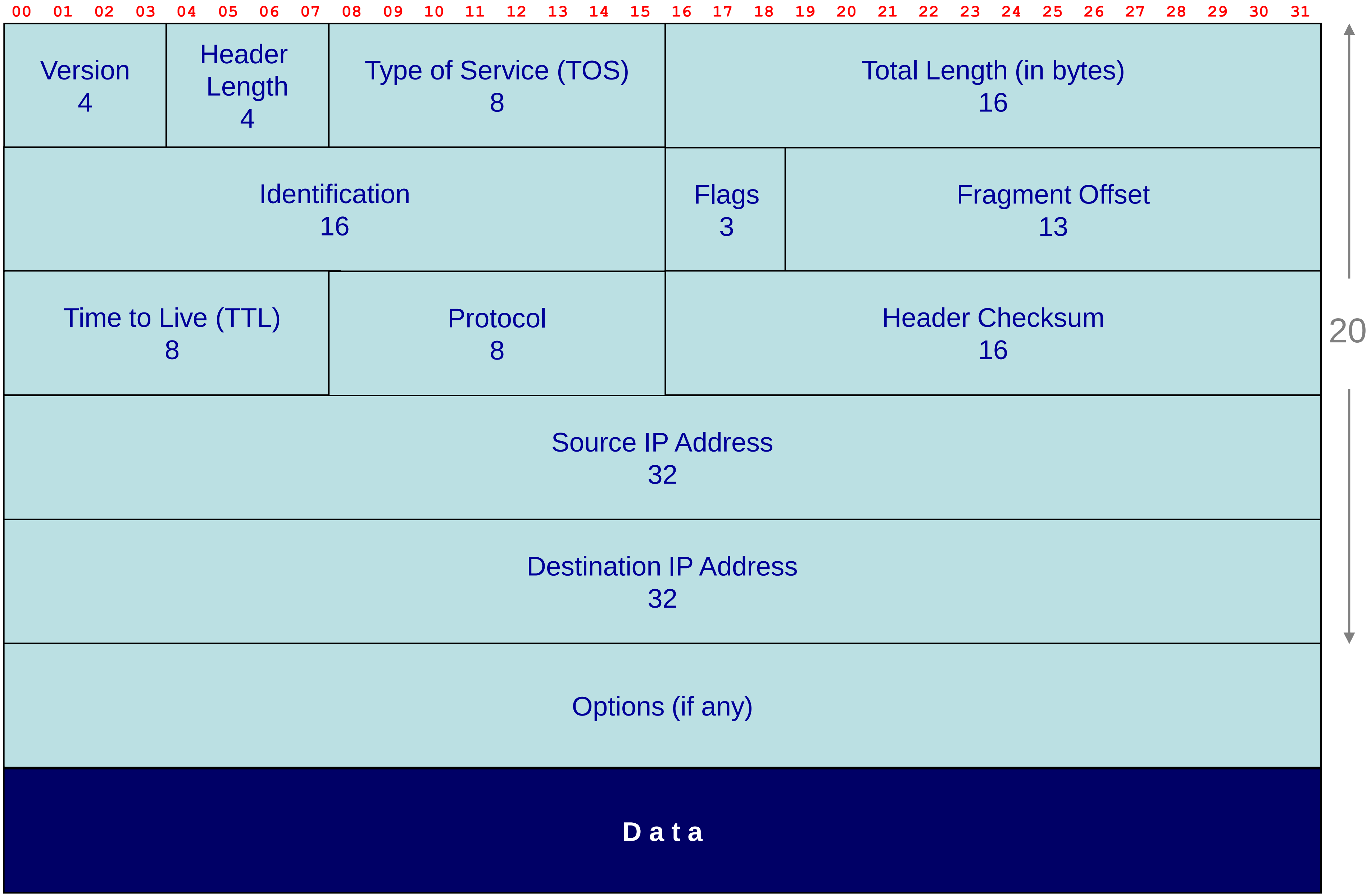
- A router provides interface between two networks. Also called as *Gateway*
- It routes the *datagrams* leaving and entering the network to enable them to get nearer to their destination

/network layer/ip/devices



- Wherever necessary the router will translate the network access protocols used by one network into the protocols used by the other

/network layer/ip/header



00 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31

Version
4

Header Length
4

Type of Service (TOS)
8

Total Length (in bytes)
16

Identification
16

Flags
3

Fragment Offset
13

Time to Live (TTL)
8

Protocol
8

Header Checksum
16

Source IP Address
32

Destination IP Address
32

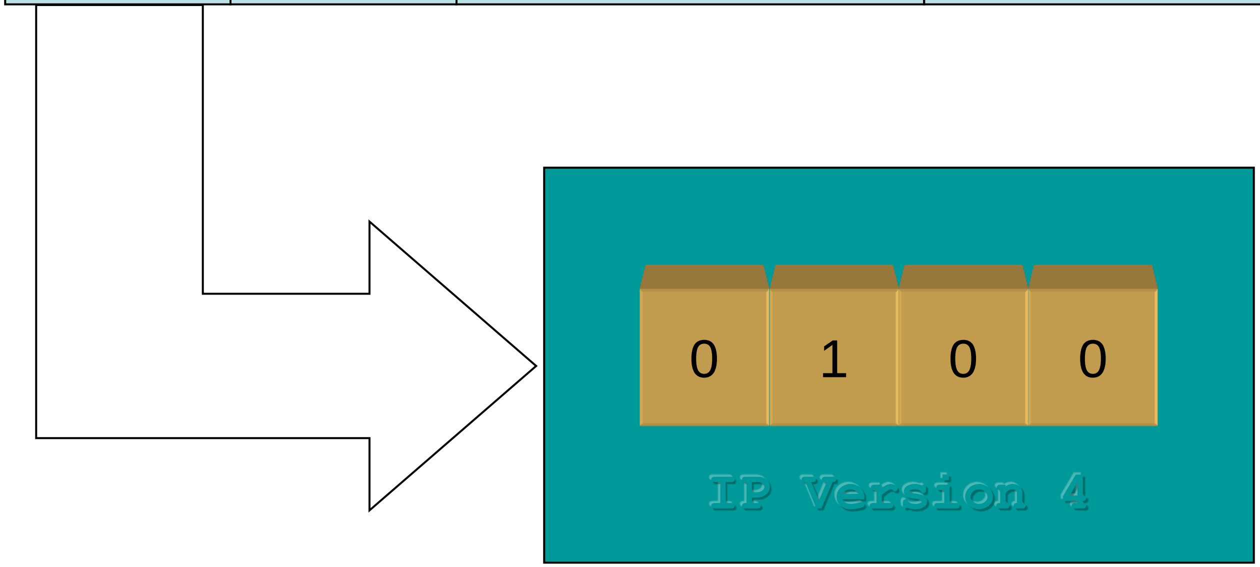
Options (if any)

Data

20

/network layer/ip/header/version

00 01 02 03	04 05 06 07	08 09 10 11 12 13 14 15	16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
Version **4**	Header Length 4	Type of Service (TOS) 8	Total Length (in bytes) 16

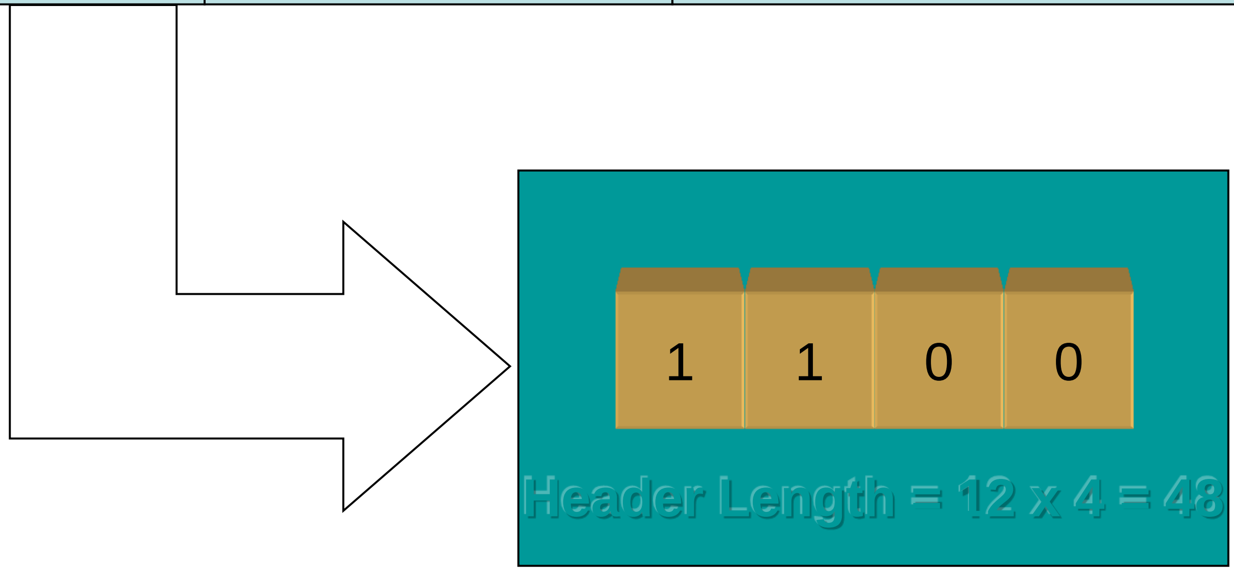


- Version field is 4 bits long
- It is the release version of IP

/network layer/ip/header/length

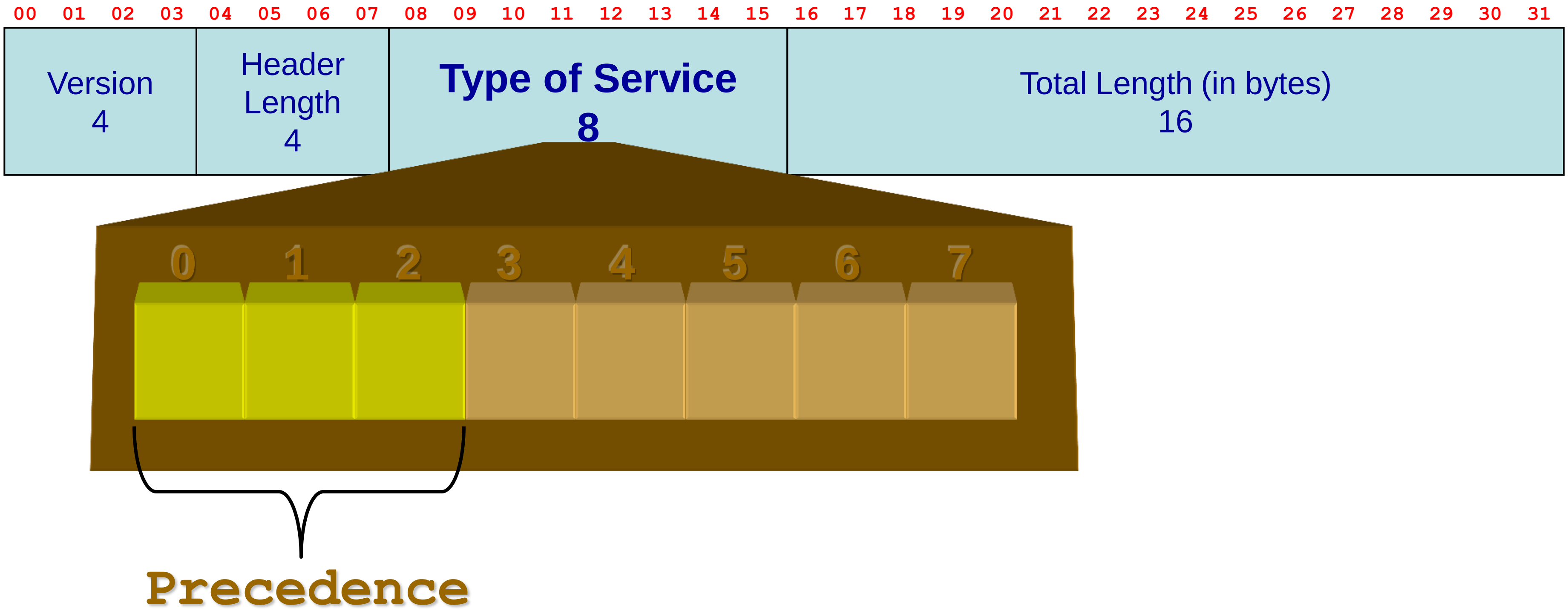
00 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31

Version 4	**Header Length 4**	Type of Service (TOS) 8	Total Length (in bytes) 16



- Header length is number of 32 bit words
- It includes options
- Max = 60, Min = 20, it may need padding

/network layer/ip/header/TOS/precedence

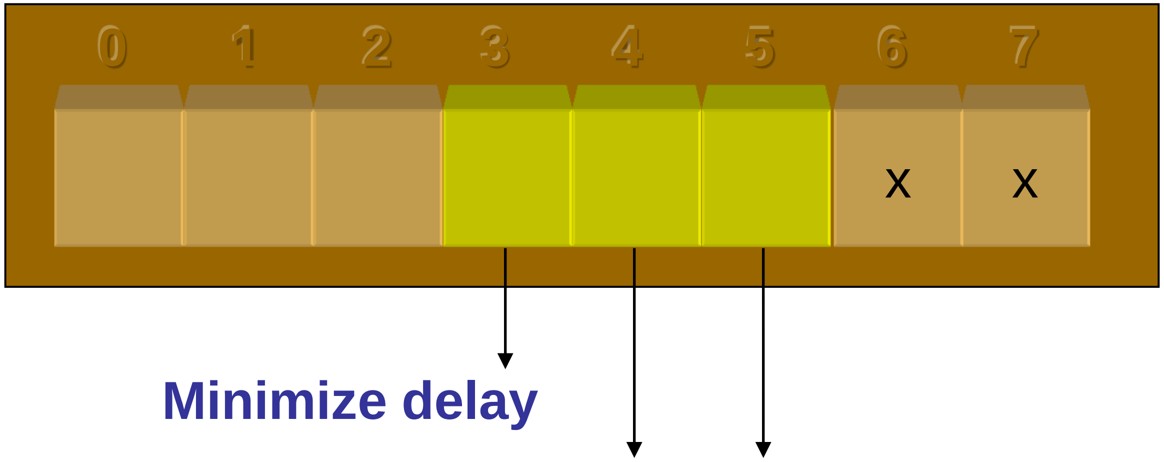


- A busy network can discard datagrams on the basis of its precedence
- 8 levels
- This field is used by router to handle congestion

/network layer/ip/header/TOS/bits

00 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31

Version 4	Header Length 4	**Type of Service** **8**	Total Length (in bytes) 16



Minimize delay

Maximize throughput **Maximize reliability**

- Only 1 of these 3 bits can be turned on at a time
- All 3 bits set to 0 implies normal service

/network layer/ip/header/TOS/use

Application	Minimize Delay	Maximize throughput	Maximize reliability	Minimize monetary cost
Telnet / Rlogin	1			
FTP				
Control	1			
Data		1		
SMTP				
Command	1			
Data		1		
SNMP			1	
NNTP				1

/network layer/ip/header/TOS/ds

00 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31

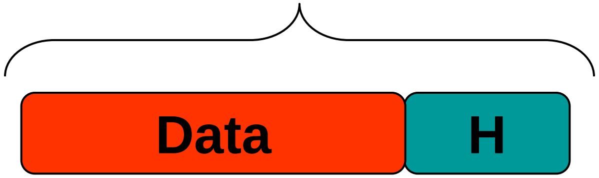
Version 4	Header Length 4	**Type of Service** **8**	Total Length (in bytes) 16

- RFC 2474 redefines the first six bits of the *TOS* field to support a technique called *Differentiated Services (DS)*

/network layer/ip/header/total length

00 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31

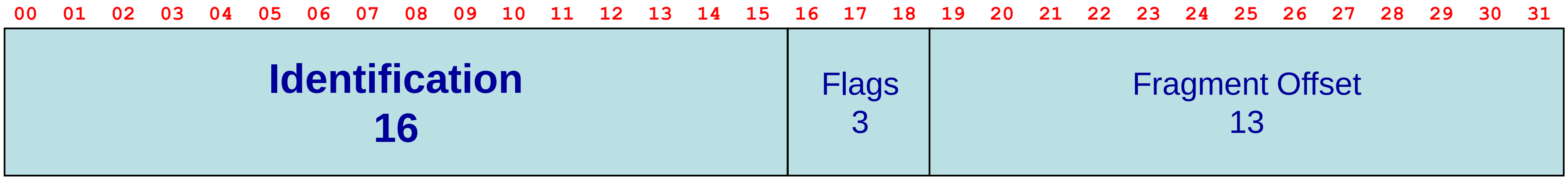
Version 4	Header Length 4	Type of Service 8	**Total Length (in bytes)** **16**



IP Datagram

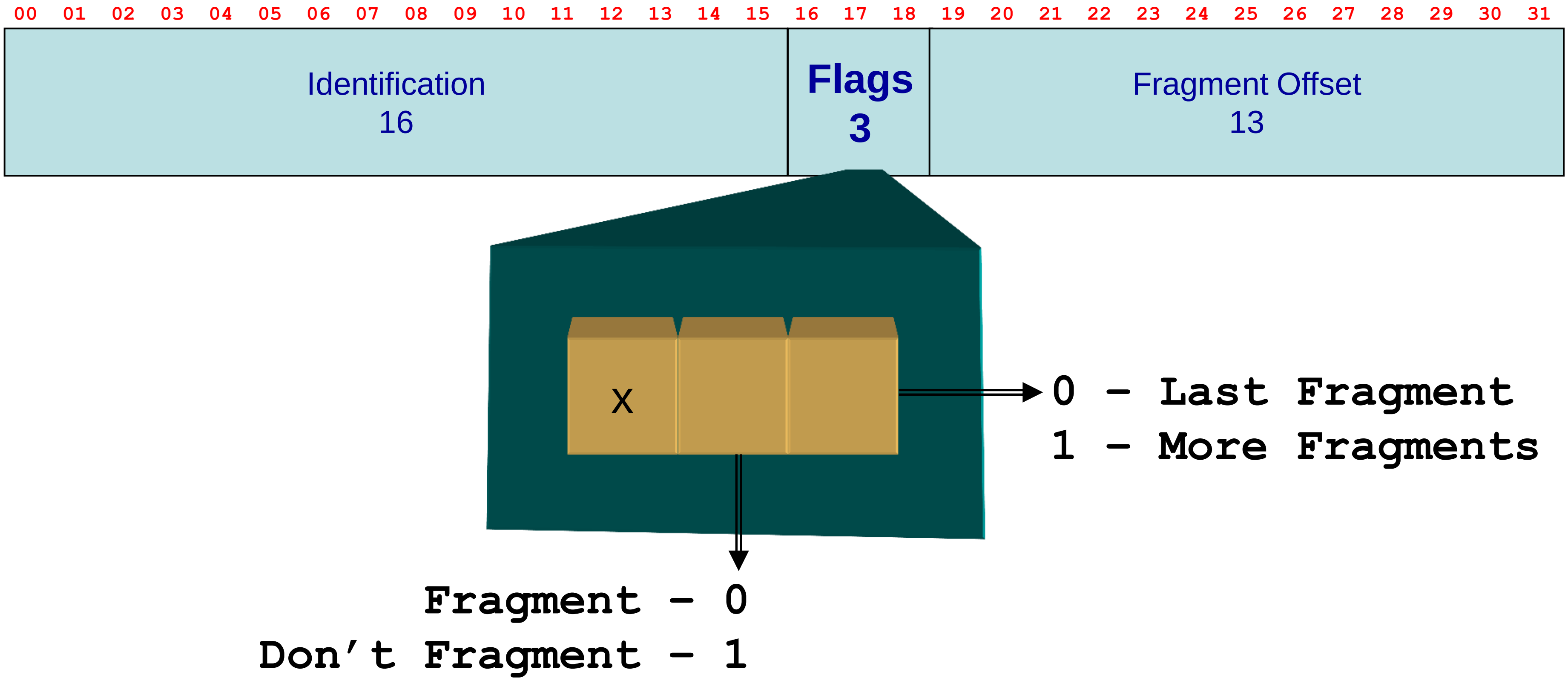
- Max size of an IP datagram = 64k
- Total length can change if a datagram is broken into multiple fragments

/network layer/ip/header/identification



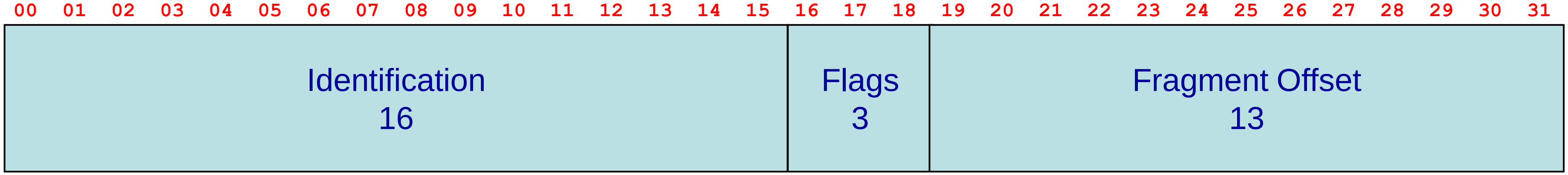
- Contains a unique value for each IP datagram, normally incremented by 1
- If a datagram is broken into multiple fragments, then this number is copied into each of those fragments
- A fragment is a datagram with its own IP header and is routed independently of any other datagrams

/network layer/ip/header/flags



- Fragment offset contains offset of a fragment from the beginning of original datagram. It is specified in units of 8 bytes (64 bits)

/network layer/ip/header/fragmentation



00 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31

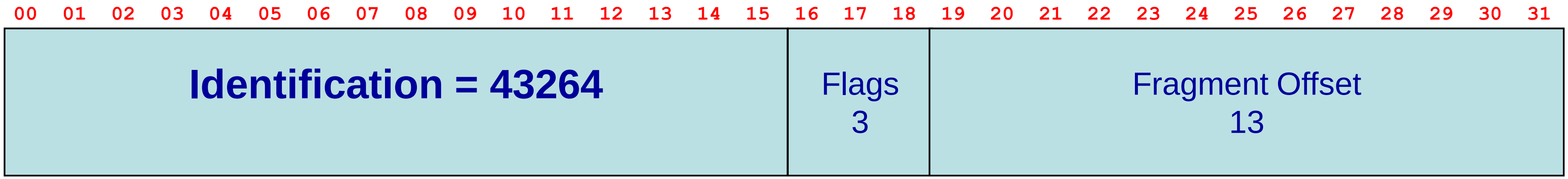
Identification
16

Flags
3

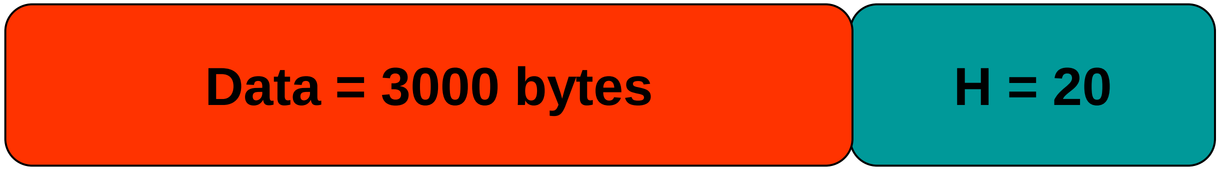
Fragment Offset
13

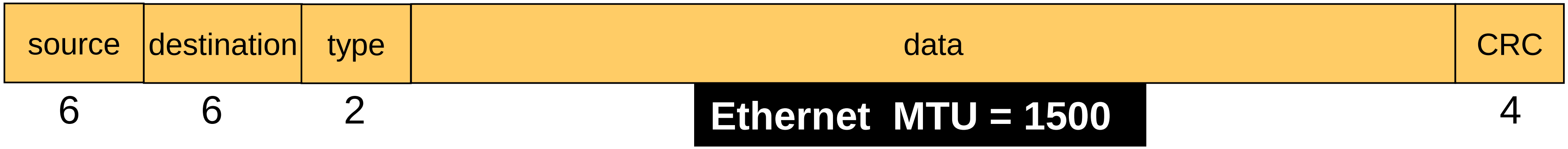
- Link layer imposes an upper limit on the size of the frame that can be transmitted
- IP queries and obtains link layer’s MTU
- IP compares MTU with the datagram size and performs fragmentation if necessary
- Fragmentation may be done either by sending host or by an intermediate router
- Fragments are assembled only at the destination host

/network layer/ip/header/fragmentation/example

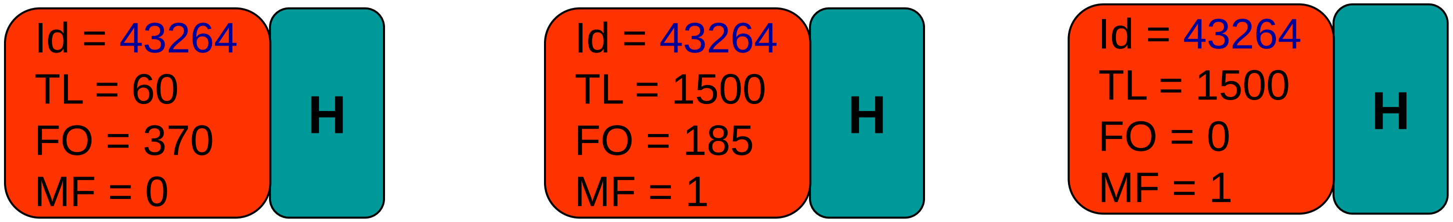


Total Length = 3020 bytes





Data	Hdr
1480	20
1480	20
40	20
3000	**60**



Actual FO: 2960 / 8

Actual FO: 1480 / 8

/network layer/ip/header/fragmentation

<table>
<tr><td>00</td><td>01</td><td>02</td><td>03</td><td>04</td><td>05</td><td>06</td><td>07</td><td>08</td><td>09</td><td>10</td><td>11</td><td>12</td><td>13</td><td>14</td><td>15</td><td>16</td><td>17</td><td>18</td><td>19</td><td>20</td><td>21</td><td>22</td><td>23</td><td>24</td><td>25</td><td>26</td><td>27</td><td>28</td><td>29</td><td>30</td><td>31</td></tr>
<tr><td colspan="16">Identification
16</td><td colspan="3">Flags
3</td><td colspan="13">Fragment Offset
13</td></tr>
</table>

- Fragmentation and reassembly is transparent to transport layer
- If one fragment is lost, entire datagram has to be retransmitted
- If *'don't fragment'* bit is set, IP router will not fragment that datagram
- Fragmentation can cause performance degradation

/network layer/ip/header/TTL

00 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31

Time to Live (TTL) **8**	Protocol 8	Header Checksum 16

- IP does not know the complete route to any destination
- IP routing is done on a hop-by-hop basis
- TTL is an upper limit initialized by the sender, on the number of routers through which a datagram can pass
- TTL is decremented by 1 by every router that handles the datagram
- When TTL = 0, datagram is thrown away and the sender is notified by an ICMP “Time exceeded” message

/network layer/ip/header/protocol

<table>
<tr><td>00</td><td>01</td><td>02</td><td>03</td><td>04</td><td>05</td><td>06</td><td>07</td><td>08</td><td>09</td><td>10</td><td>11</td><td>12</td><td>13</td><td>14</td><td>15</td><td>16</td><td>17</td><td>18</td><td>19</td><td>20</td><td>21</td><td>22</td><td>23</td><td>24</td><td>25</td><td>26</td><td>27</td><td>28</td><td>29</td><td>30</td><td>31</td></tr>
<tr><td colspan="8">Time to Live (TTL)
8</td><td colspan="8"><b>Protocol</b>
<b>8</b></td><td colspan="16">Header Checksum
16</td></tr>
</table>

- Identifies upper layer protocol that gave the data to IP to send or is the intended recipient

ICMP = 1

TCP = 6

UDP = 17

/network layer/ip/header/checksum

00 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31

Time to Live (TTL) 8	Protocol 8	**Header Checksum** **16**

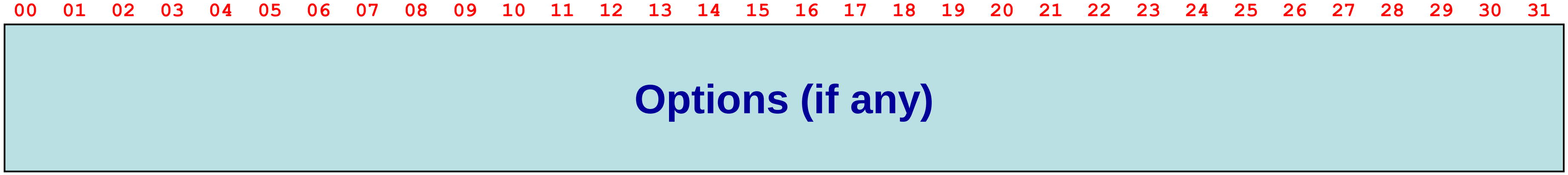
- Header checksum is calculated by 16-bit one's complement sum of the header
- The receiver of the datagram cross-checks integrity of the header by re-computing the checksum of the header and comparing it with the stored checksum
- If it does not match, IP discards the received datagram
- No error message is generated

/network layer/ip/header/address

<table>
<tr><th>00</th><th>01</th><th>02</th><th>03</th><th>04</th><th>05</th><th>06</th><th>07</th><th>08</th><th>09</th><th>10</th><th>11</th><th>12</th><th>13</th><th>14</th><th>15</th><th>16</th><th>17</th><th>18</th><th>19</th><th>20</th><th>21</th><th>22</th><th>23</th><th>24</th><th>25</th><th>26</th><th>27</th><th>28</th><th>29</th><th>30</th><th>31</th></tr>
<tr><td colspan="32">Source IP Address
32</td></tr>
<tr><td colspan="32">Destination IP Address
32</td></tr>
</table>

- 32 bit *valid* IP addresses

/network layer/ip/header/options



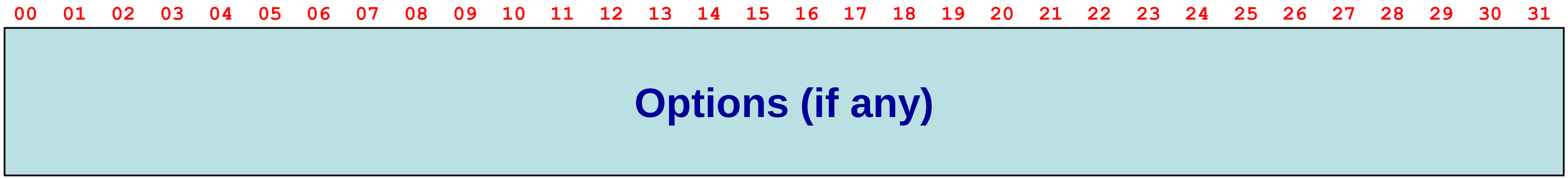
- It's a variable-length list of optional information for the datagram
- Options can't exceed 40 bytes
- Each option field has either 1 or 3 parts

  Type: 8 bits, identifies type of option

  Length: 8 bits, length of total option

  Data: variable length, applicable to option

/network layer/ip/header/options/list

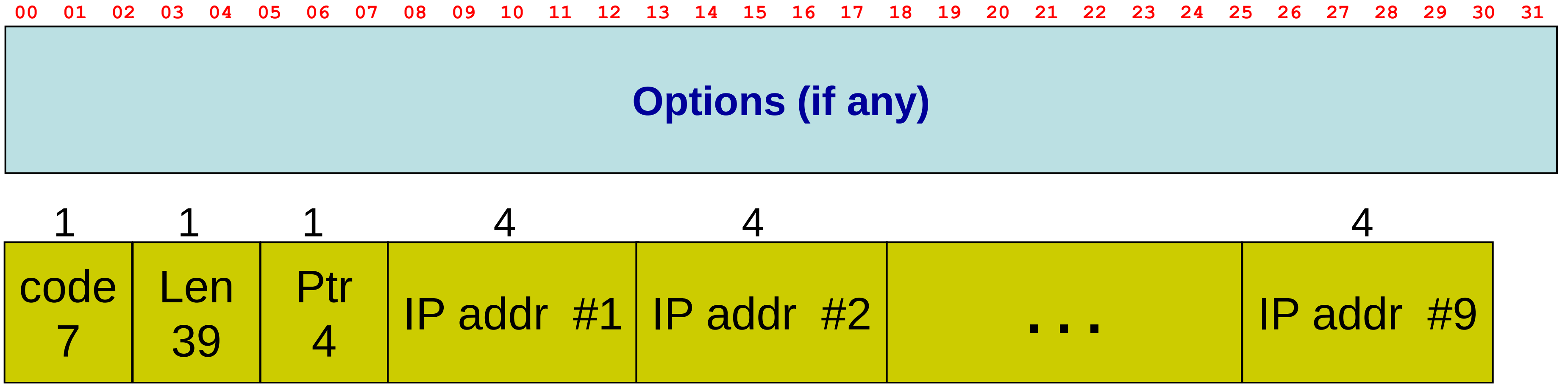


00 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31

Options (if any)

- Security and handling restrictions (for US military)
- Record route (have each router record its IP address)
- Timestamp (have each router record its IP address and time)
- Loose source routing (specifying a list of IP addrs that must be traversed by datagram)
- Strict source routing (only the addrs in the list can be traversed)

/network layer/ip/header/options/record route



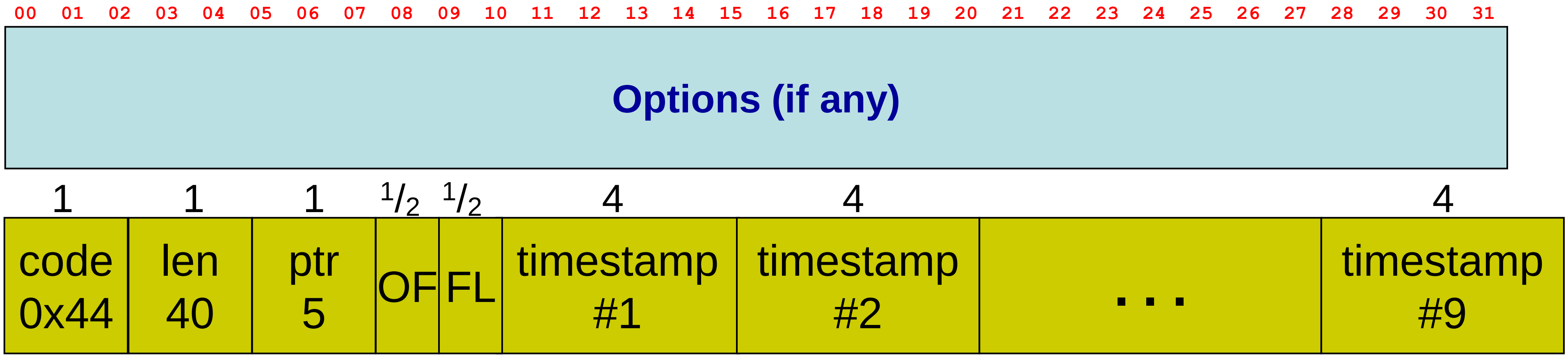
00 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31

Options (if any)

1	1	1	4	4		4
code 7	Len 39	Ptr 4	IP addr #1	IP addr #2	. . .	IP addr #9

- Every router that handles the datagram with above option set, adds its own IP addr to a list in *options* field
- This feature is used for knowing the path (addrs of all the routers) through which the datagram passed on its way to the destination host
- Used by *ping* utility when used with –r parameter

/network layer/ip/header/options/timestamp



00 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31

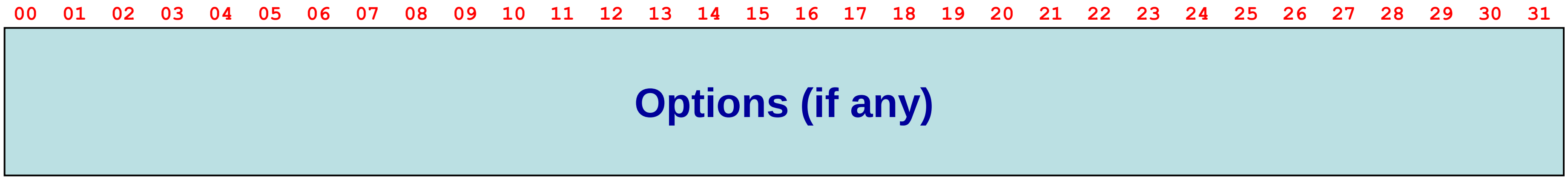
Options (if any)

1	1	1	1/2	1/2	4	4		4
code 0x44	len 40	ptr 5	OF	FL	timestamp #1	timestamp #2	. . .	timestamp #9

- Timestamp is the number of milliseconds past midnight of a system (can also be some other format)
- If a router can’t add timestamp due to shortage of space, it increments *overflow* field by 1

flags	description
0	Record only timestamp
1	Each router records IP Address + Timestamp
3	A router records its timestamp only if its IP addr is in the list initialized by the sender

/network layer/ip/header/options/source routing

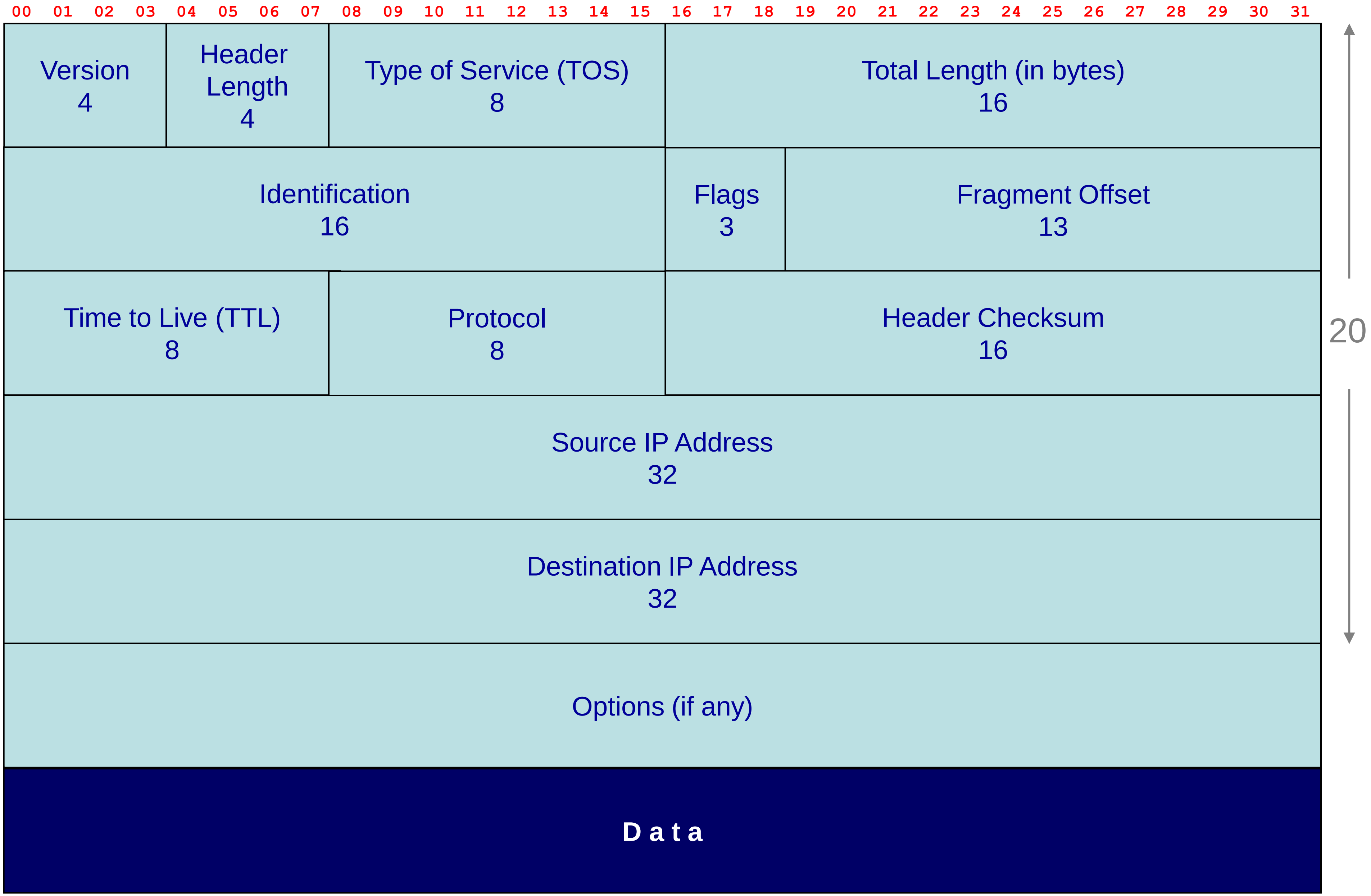


- *Strict:* The sender specifies exact path that the IP datagram must follow. Code = 0x83

- *Loose:* As above, except that the datagram can also pass through other routers between any two addresses in the list. Code = 0x89

code	len	ptr	IP addr #1	IP addr #2	. . .	IP addr #9
1	1	1	4	4		4

/network layer/ip/header



Version 4

Header Length 4

Type of Service (TOS) 8

Total Length (in bytes) 16

Identification 16

Flags 3

Fragment Offset 13

Time to Live (TTL) 8

Protocol 8

Header Checksum 16

Source IP Address 32

Destination IP Address 32

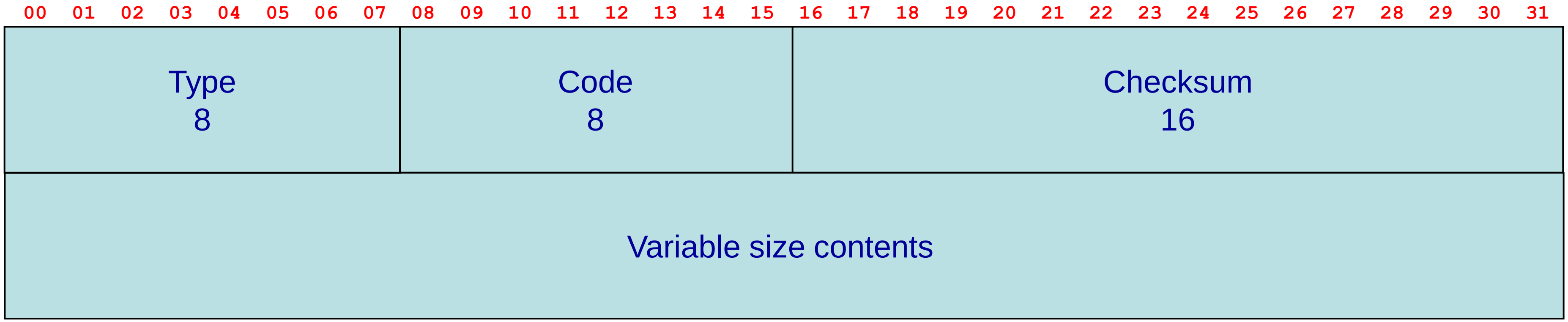
Options (if any)

Data

- ifconfig
  - Study and understand its output for various interfaces on the host
- netstat
  - Print MTU of each interface on your host
  - Print routing table
- tcpdump
  - Trap and display IP packets
  - Display only header part of IP packet. Study this header
- traceroute
  - Study program operation
  - Find path between any two hosts using traceroute
  - Explain the output

/network layer/icmp

- Communicates error messages and other conditions that require attention
- These messages are used either by IP or by TCP/UDP
- All ICMP messages are encapsulated in an IP datagram



Type
8

Code
8

Checksum
16

Variable size contents

- ICMP error message always contains the IP header and first 8 bytes of the IP datagram that caused this error to be generated

/network layer/icmp/message types

Type	Code	Description	Query	Error
0	0	echo reply	•	
3	0	network unreachable		•
	1	host unreachable		•
	2	protocol unreachable		•
	3	port unreachable		•
	4	fragmentation needed…		•
4	0	source quench		•
8	0	echo request	•	
11	0	TTL = 0 during transit		•

/network layer/icmp/message types

Type	Code	Description	Handled by/msg
0	0	echo reply	user process
3	0	network unreachable	“no route to host”
	1	host unreachable	“no route to host”
	2	protocol unreachable	“connection refused”
	3	port unreachable	“connection refused”
	4	fragmentation needed…	“message too long”
4	0	source quench	kernel for TCP
8	0	echo request	kernel generates reply
11	0	TTL = 0 during transit	“Time exceeded”

/network layer/icmp/rules

- An ICMP error message is never generated in response to:
  - An ICMP error message
  - A datagram destined to an IP broadcast address
  - A datagram sent as a link-layer broadcast
  - A fragment (other than first) of a datagram
  - A datagram whose source address does not specify a single host

1. Use *ping* to learn more about ICMP

2. Find round-trip-time from a host to another host

3. Trap only *ping* echo and reply packets using *tcpdump*

4. Use ping to record route of a datagram from host to destination host
5. Use ping to record timestamps of all routers that a datagram passes through while reaching its destination host

Questions ?