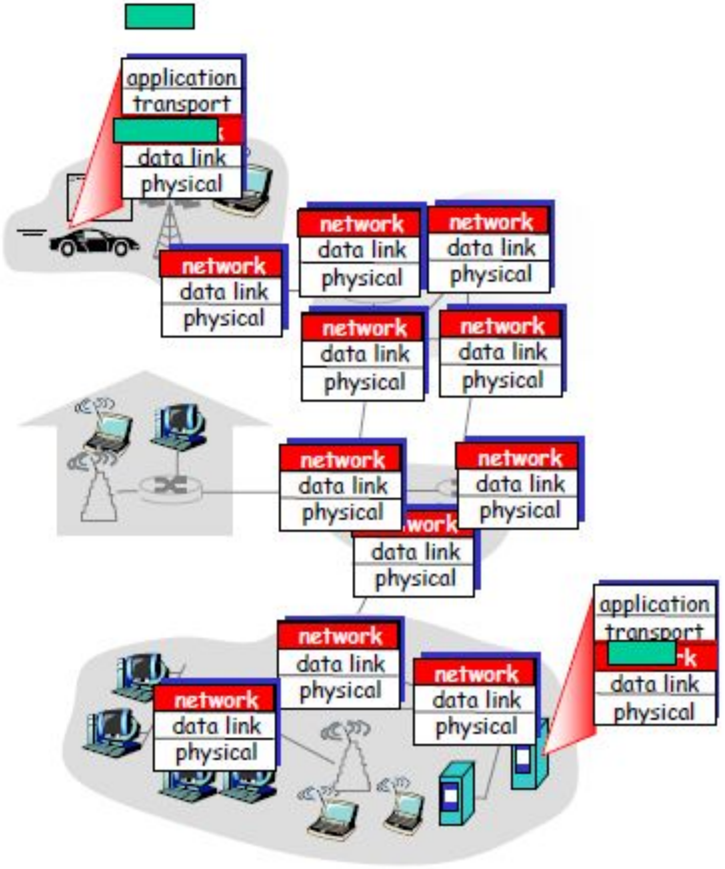
Computer Networks & IOT (18B11CS311)

Even Semester_2024

Network Layer

Network layer

- transport segment from sending to receiving host
- on sending side encapsulates segments into datagrams
- on rcving side, delivers segments to transport layer
- network layer protocols in *every* host, router
- router examines header fields in all IP datagrams passing through it



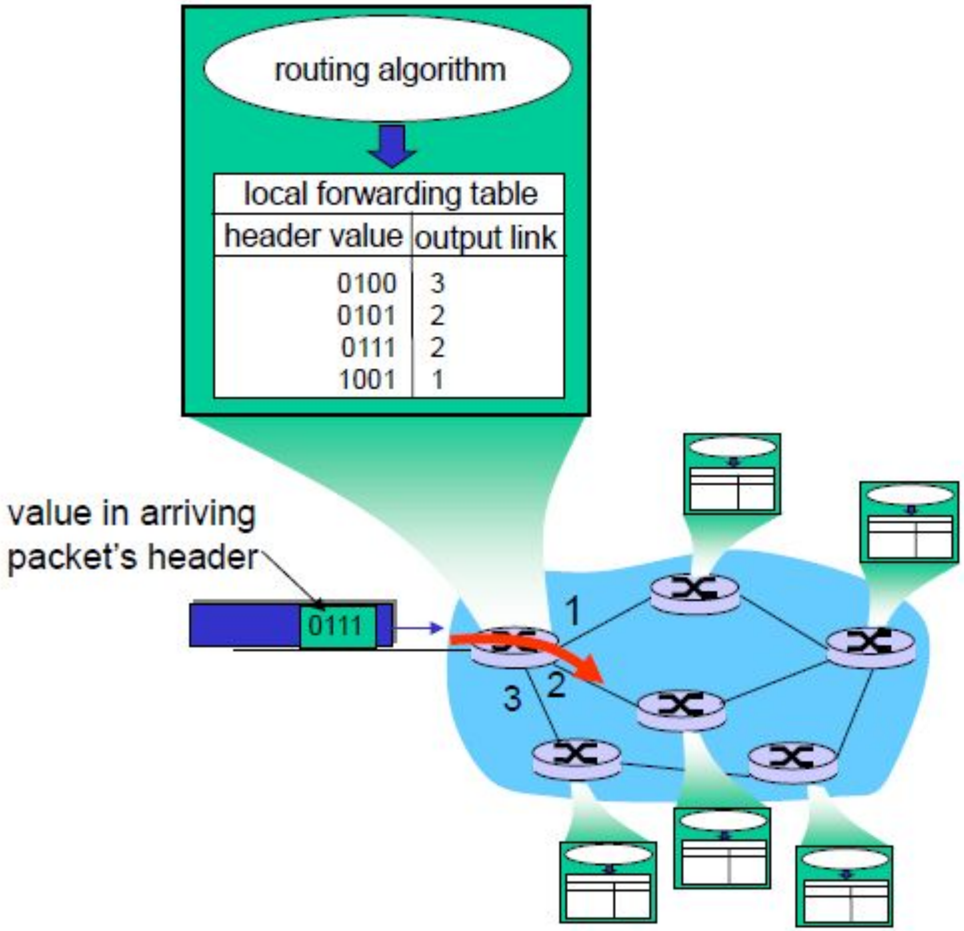
Two Key Network-Layer Functions

- *forwarding:* move packets from router's input to appropriate router output
- *routing:* determine route taken by packets from source to dest.
  - *routing algorithms*

analogy:

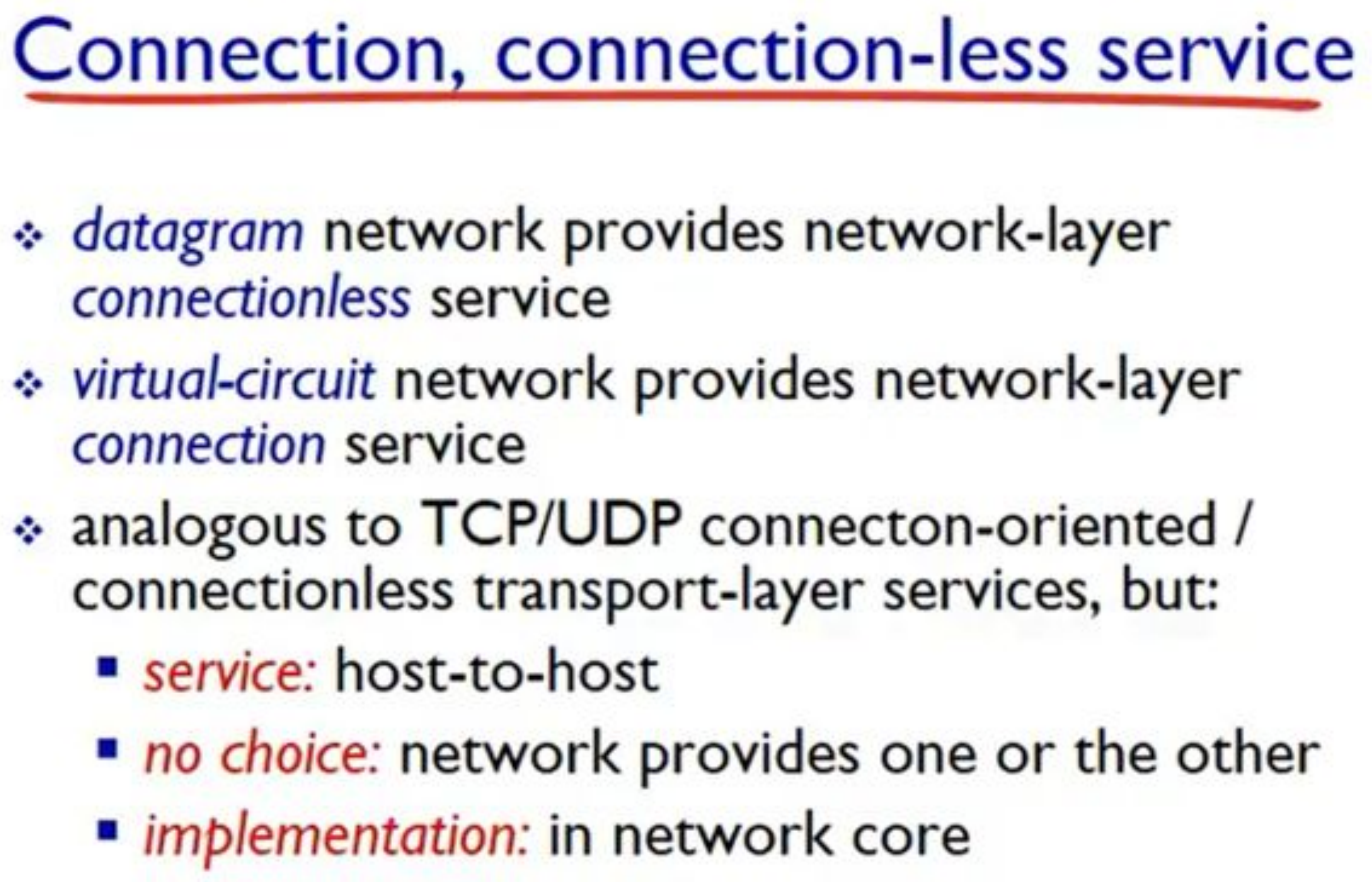
- routing: process of planning trip from source to dest
- forwarding: process of getting through single interchange

Interplay between routing and forwarding



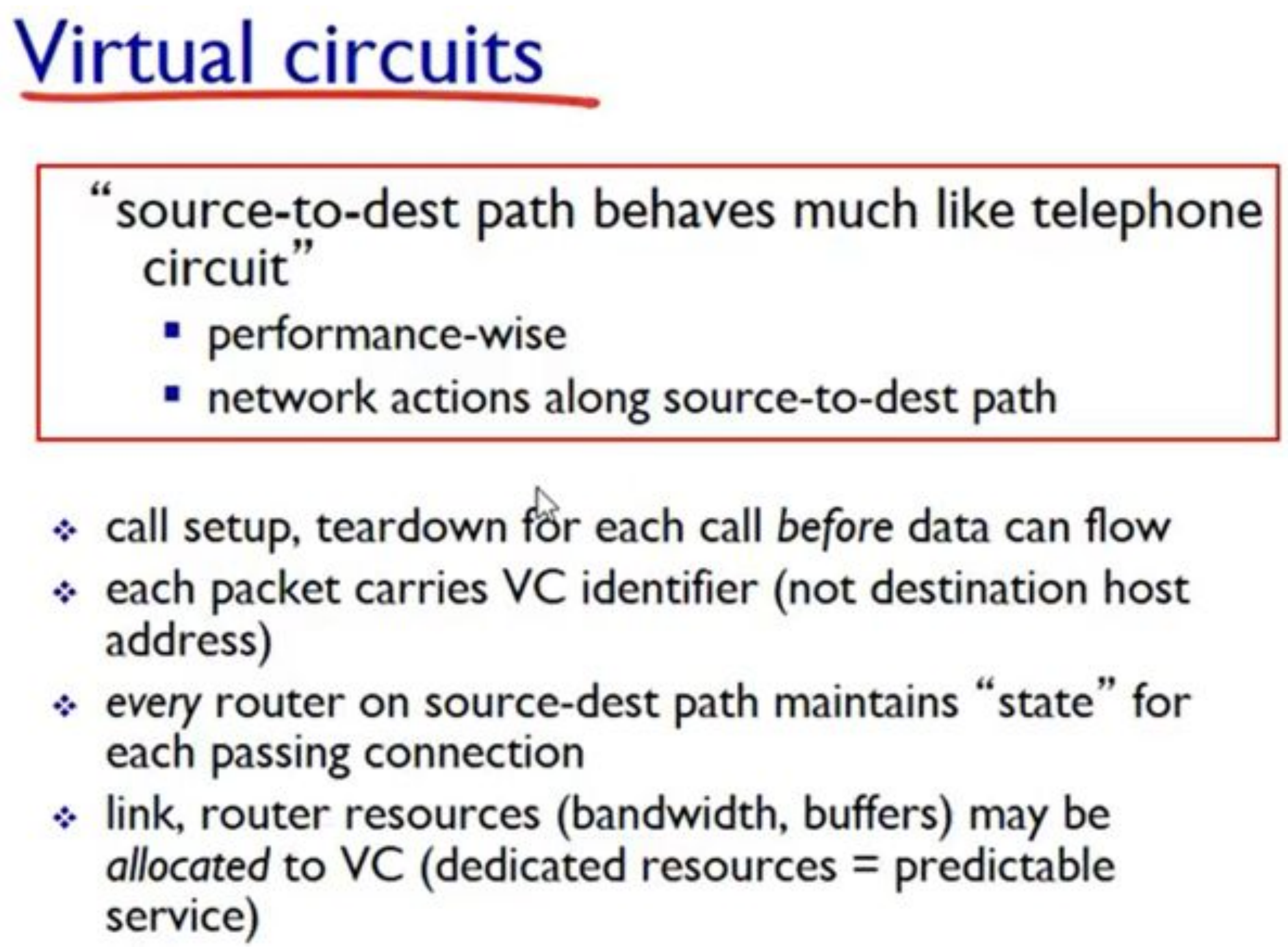
local forwarding table

header value	output link
0100	3
0101	2
0111	2
1001	1



Connection, connection-less service

- *datagram* network provides network-layer *connectionless* service
- *virtual-circuit* network provides network-layer *connection* service
- analogous to TCP/UDP connecton-oriented / connectionless transport-layer services, but:
  - *service:* host-to-host
  - *no choice:* network provides one or the other
  - *implementation:* in network core

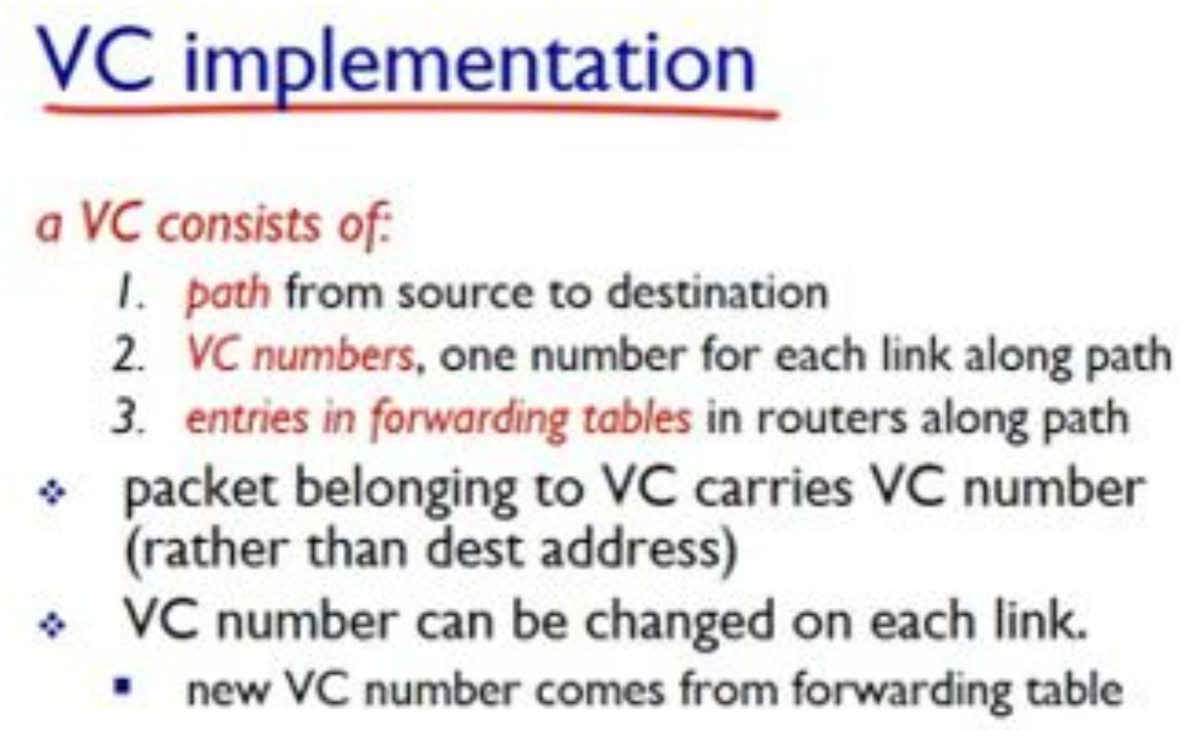


Virtual circuits

"source-to-dest path behaves much like telephone circuit"

- performance-wise
- network actions along source-to-dest path

- call setup, teardown for each call *before* data can flow
- each packet carries VC identifier (not destination host address)
- *every* router on source-dest path maintains "state" for each passing connection
- link, router resources (bandwidth, buffers) may be *allocated* to VC (dedicated resources = predictable service)



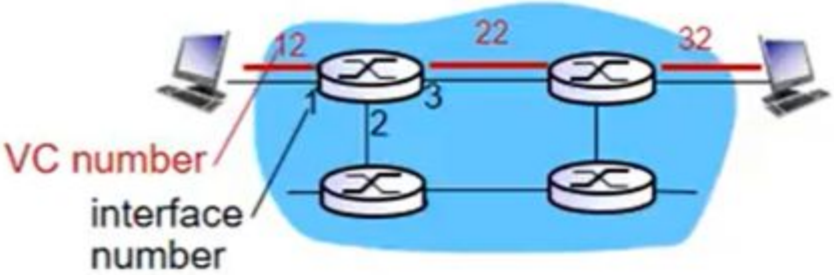
VC implementation

a VC consists of:

1. *path* from source to destination
2. *VC numbers*, one number for each link along path
3. *entries in forwarding tables* in routers along path

- packet belonging to VC carries VC number (rather than dest address)
- VC number can be changed on each link.
  - new VC number comes from forwarding table

VC forwarding table



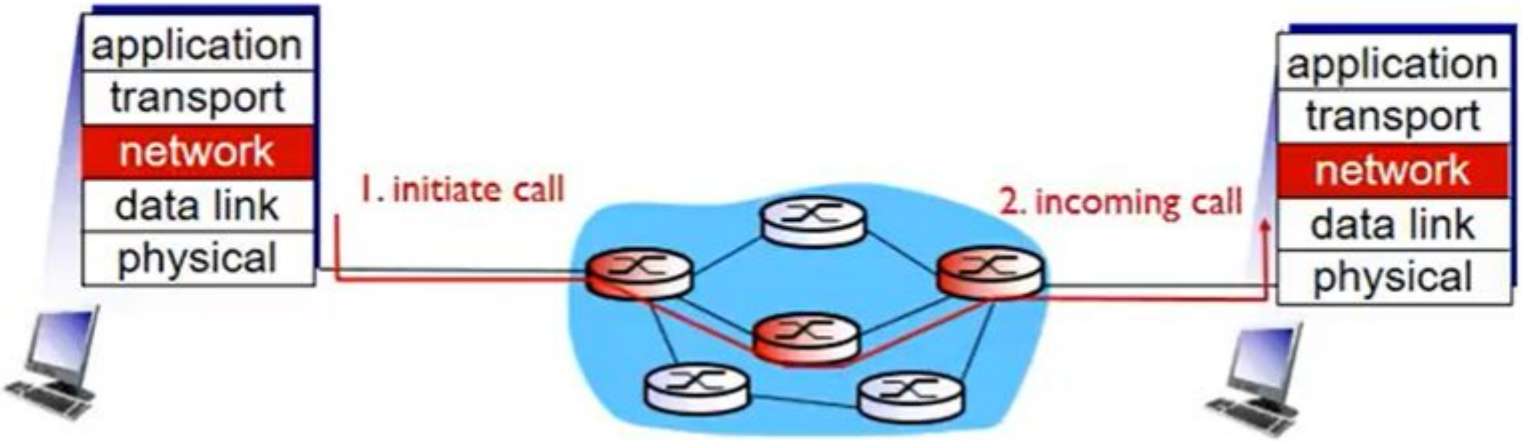
forwarding table in northwest router:

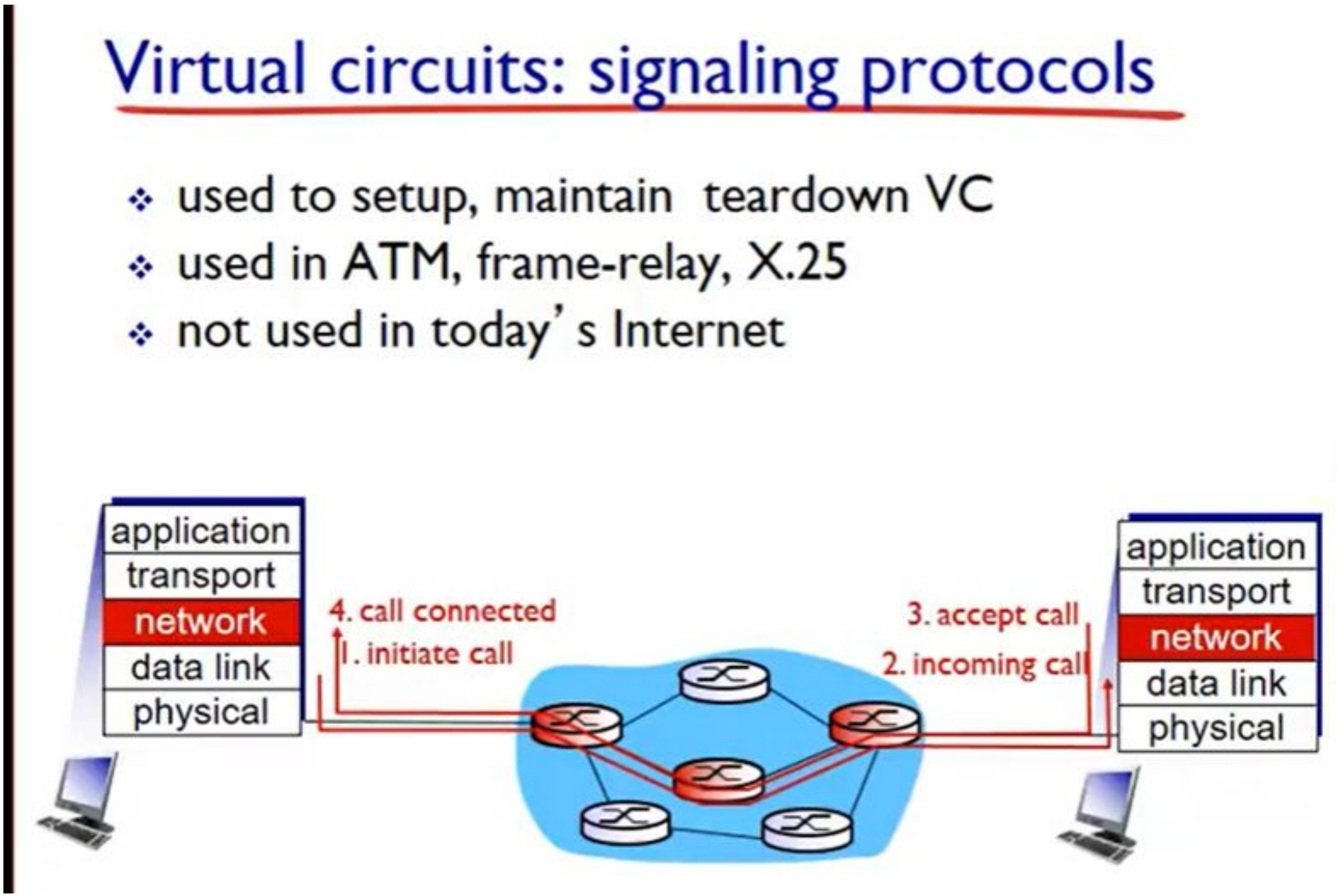
Incoming interface	Incoming VC #	Outgoing interface	Outgoing VC #
1	12	3	22
2	63	1	18
3	7	2	17
1	97	3	87
…	…	…	…

VC routers maintain connection state information!

Virtual circuits: signaling protocols

- used to setup, maintain teardown VC
- used in ATM, frame-relay, X.25
- not used in today' s Internet



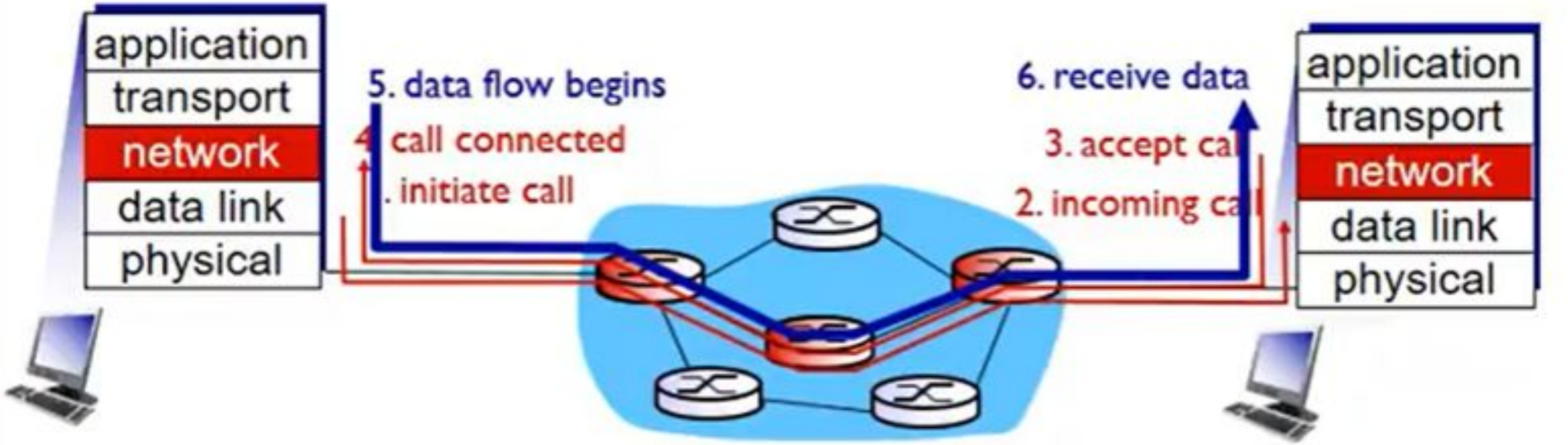


Virtual circuits: signaling protocols

- used to setup, maintain teardown VC
- used in ATM, frame-relay, X.25
- not used in today' s Internet

Virtual circuits: signaling protocols

- used to setup, maintain teardown VC
- used in ATM, frame-relay, X.25
- not used in today' s Internet



Datagram networks

- no call setup at network layer
- routers: no state about end-to-end connections
  - no network-level concept of "connection"
- packets forwarded using destination host address

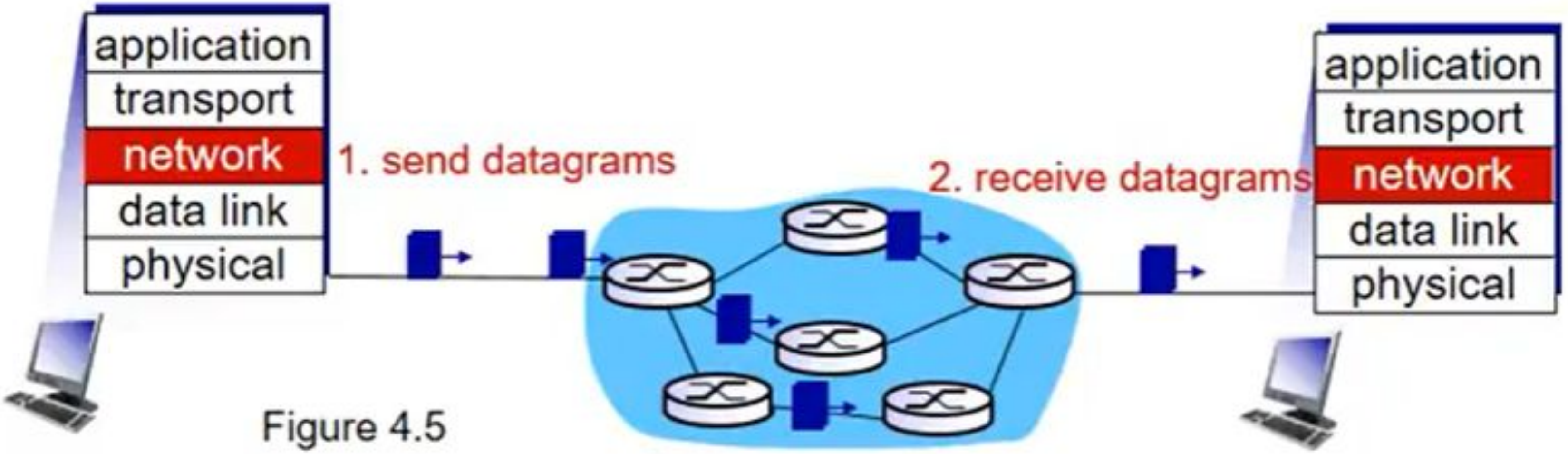
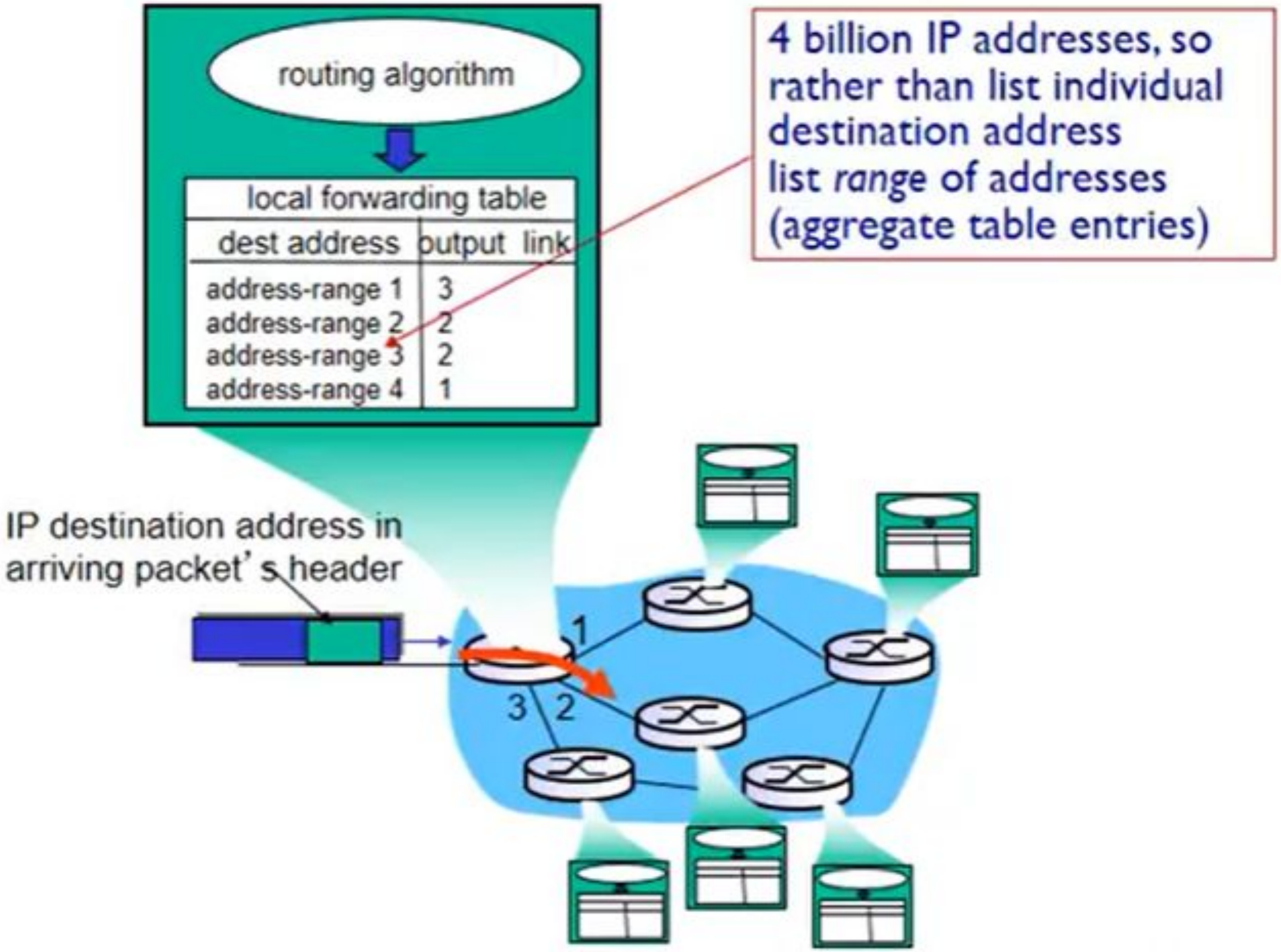


Figure 4.5

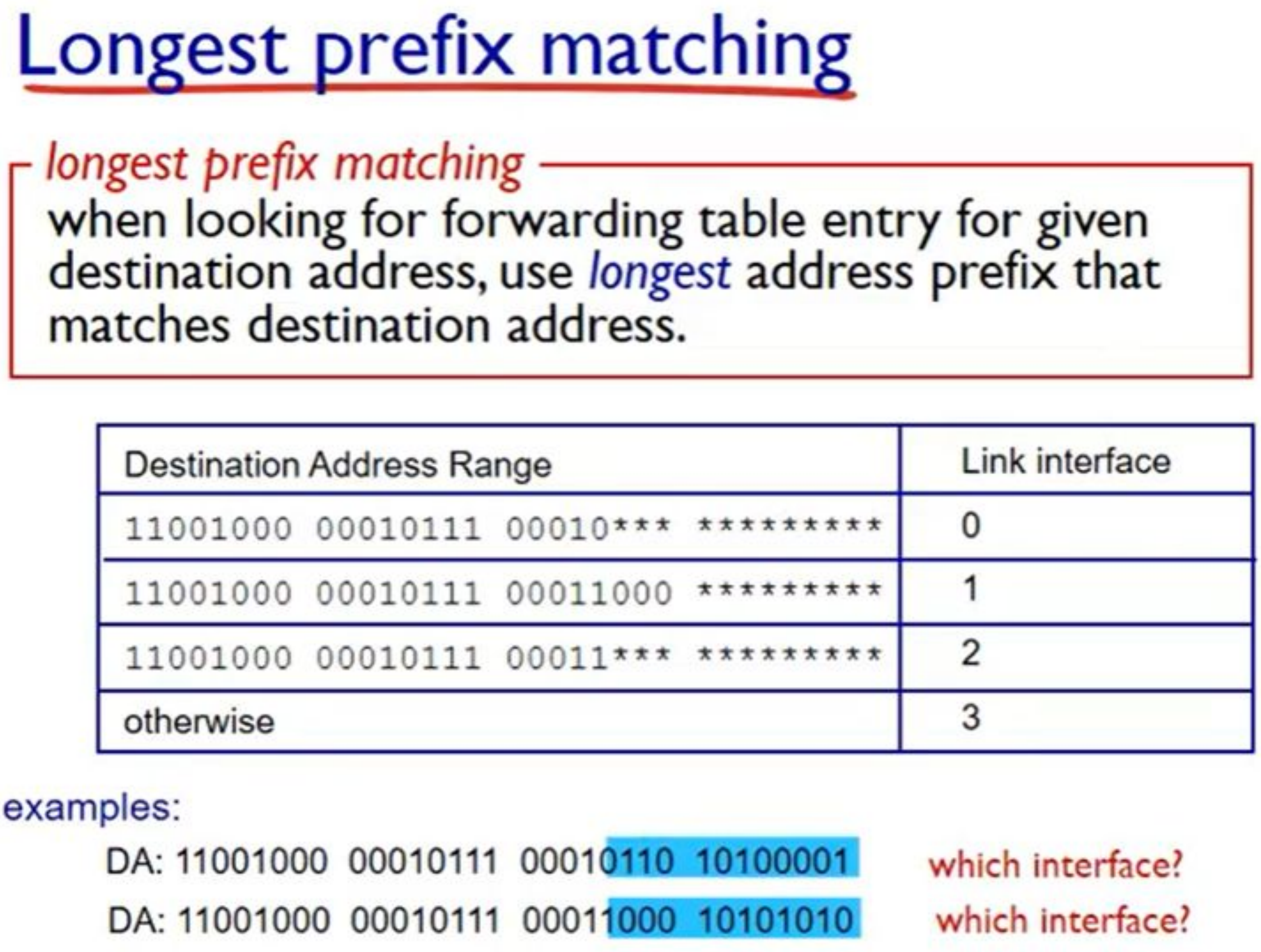
Datagram forwarding table



local forwarding table

dest address	output link
address-range 1	3
address-range 2	2
address-range 3	2
address-range 4	1

4 billion IP addresses, so rather than list individual destination address list *range* of addresses (aggregate table entries)



Longest prefix matching

longest prefix matching

when looking for forwarding table entry for given destination address, use *longest* address prefix that matches destination address.

Destination Address Range	Link interface
11001000 00010111 00010*** *********	0
11001000 00010111 00011000 *********	1
11001000 00010111 00011*** *********	2
otherwise	3

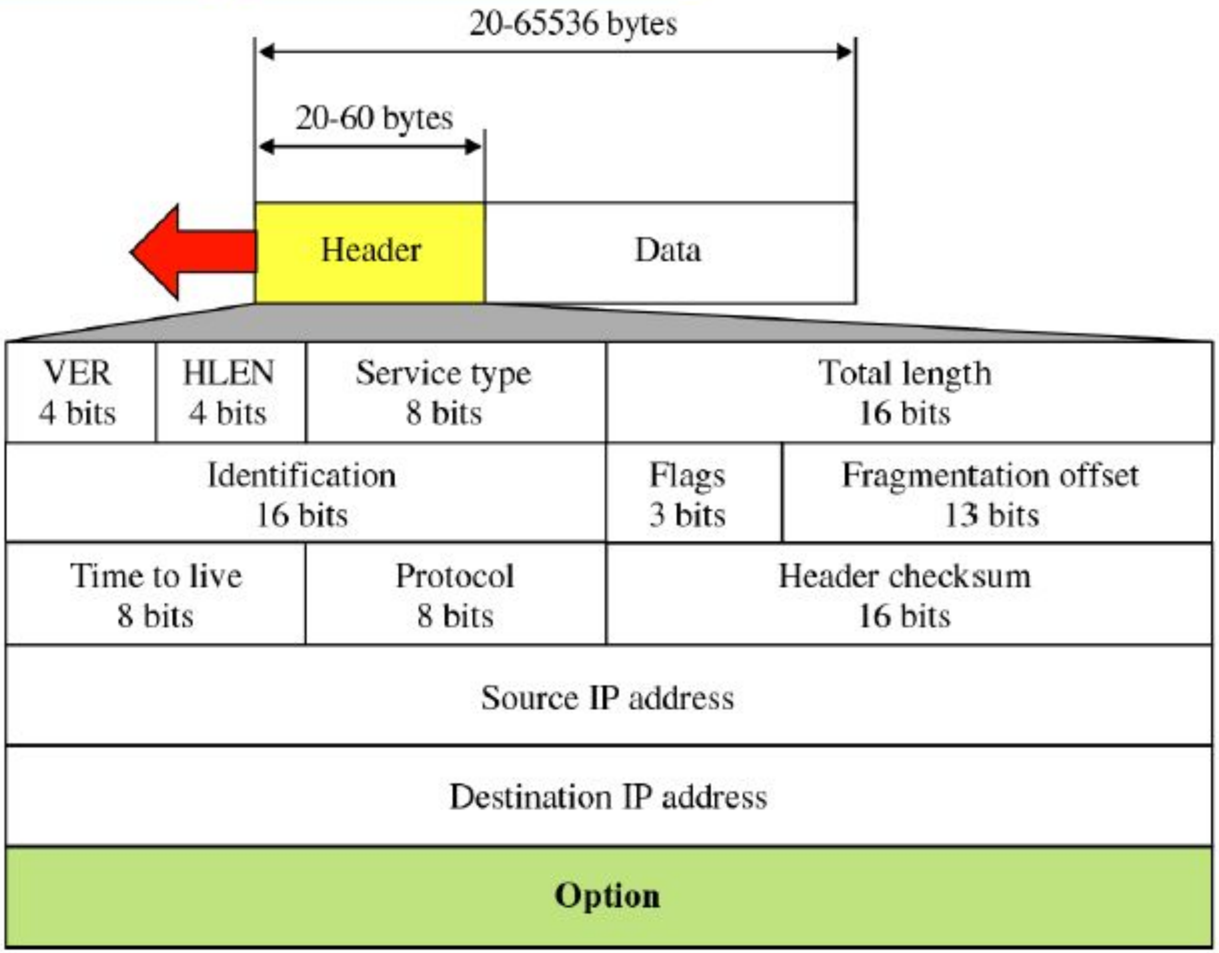
examples:

DA: 11001000 00010111 00010110 10100001 which interface?

DA: 11001000 00010111 00011000 10101010 which interface?

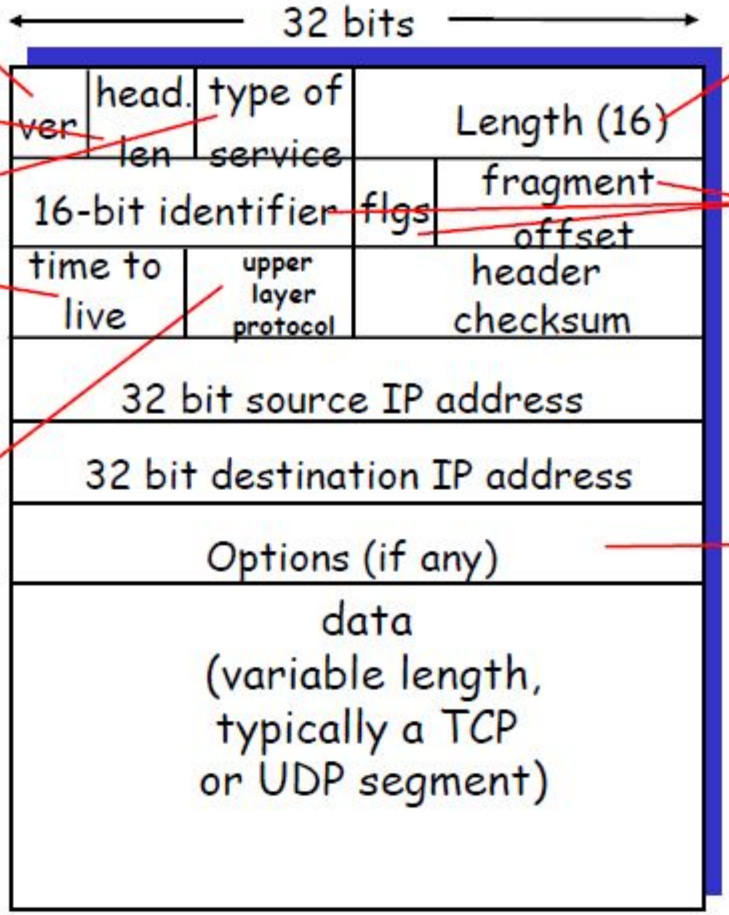
Internet Protocol

IPv4 Datagram Format



<table>
<tr><td>VER
4 bits</td><td>HLEN
4 bits</td><td colspan="2">Service type
8 bits</td><td colspan="2">Total length
16 bits</td></tr>
<tr><td colspan="4">Identification
16 bits</td><td>Flags
3 bits</td><td>Fragmentation offset
13 bits</td></tr>
<tr><td colspan="2">Time to live
8 bits</td><td colspan="2">Protocol
8 bits</td><td colspan="2">Header checksum
16 bits</td></tr>
<tr><td colspan="6">Source IP address</td></tr>
<tr><td colspan="6">Destination IP address</td></tr>
<tr><td colspan="6"><b>Option</b></td></tr>
</table>

IP datagram format



32 bits

ver

head. len

type of service

Length (16)

16-bit identifier

flgs

fragment offset

time to live

upper layer protocol

header checksum

32 bit source IP address

32 bit destination IP address

Options (if any)

data (variable length, typically a TCP or UDP segment)

IP protocol version number

header length (bytes)

"type" of data

max number remaining hops (decremented at each router)

upper layer protocol to deliver payload to 6→TCP;17→udp

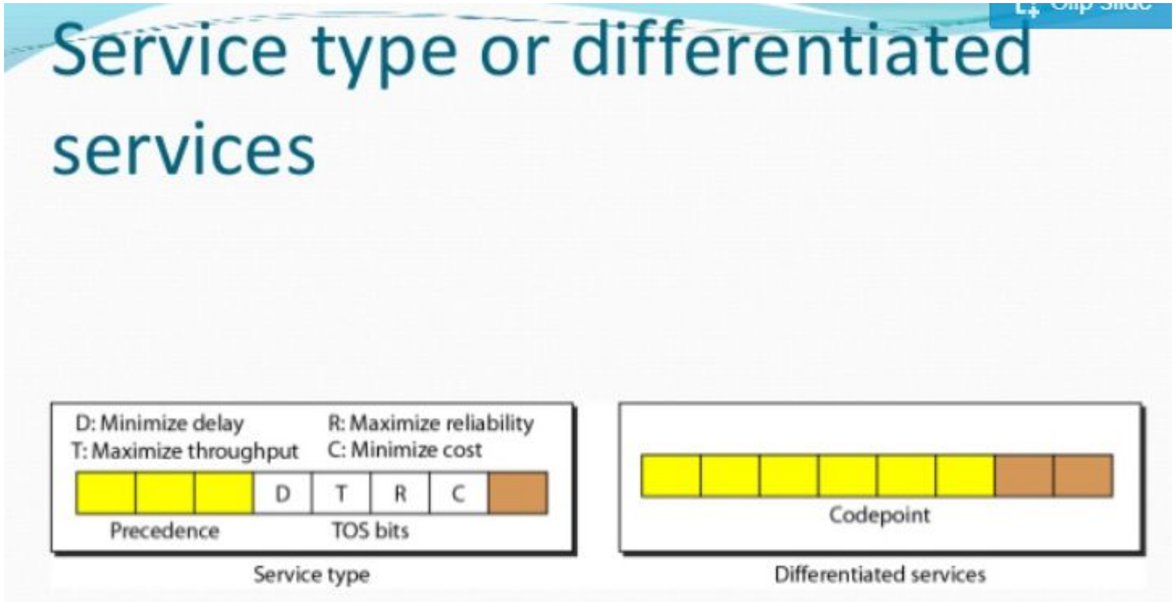
total datagram length (bytes)

for fragmentation/ reassembly

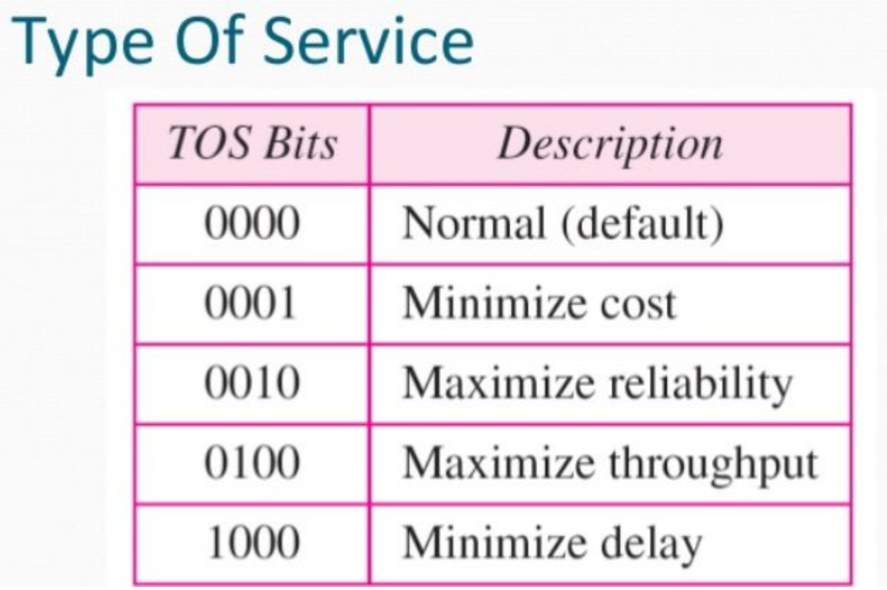
E.g. timestamp, record route taken, specify list of routers to visit.

how much overhead with TCP?

- 20 bytes of TCP
- 20 bytes of IP
- = 40 bytes + app layer overhead

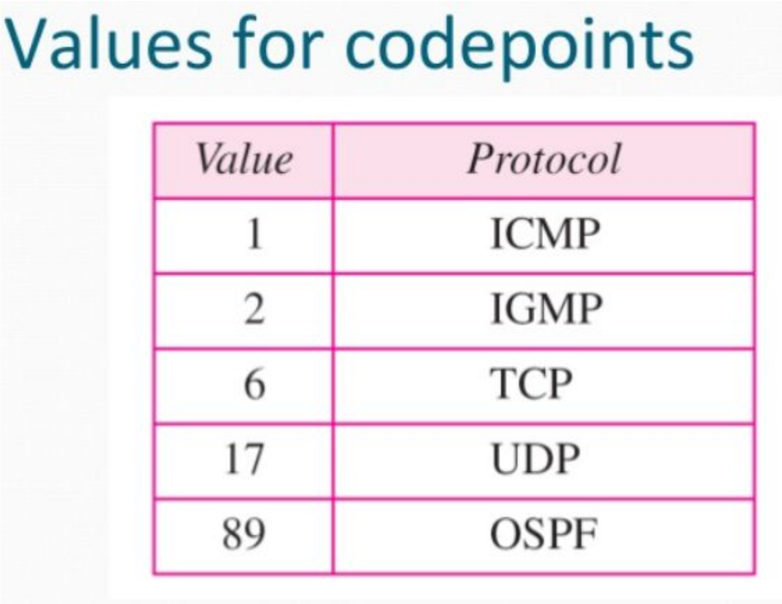


Service type or differentiated services



Type Of Service

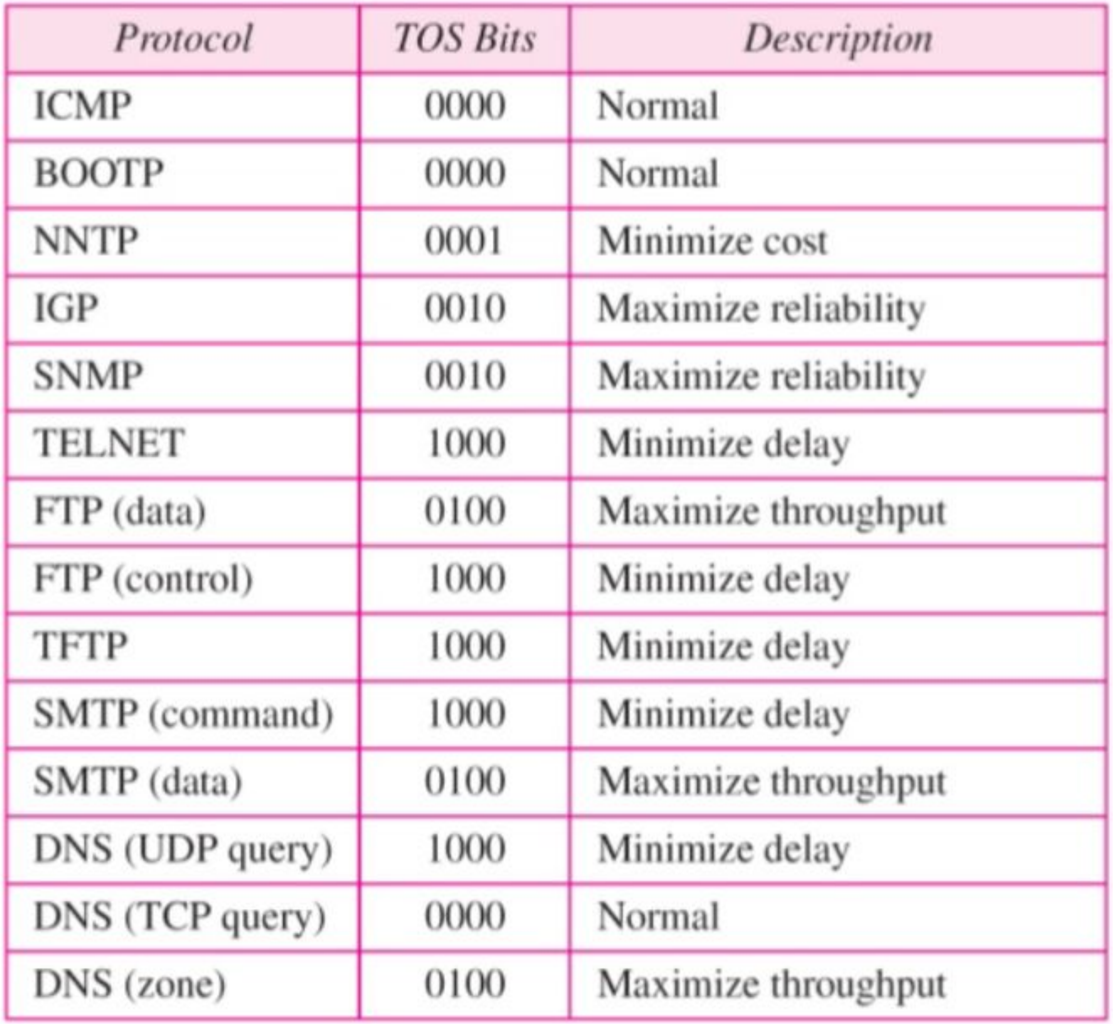
TOS Bits	*Description*
0000	Normal (default)
0001	Minimize cost
0010	Maximize reliability
0100	Maximize throughput
1000	Minimize delay



Values for codepoints

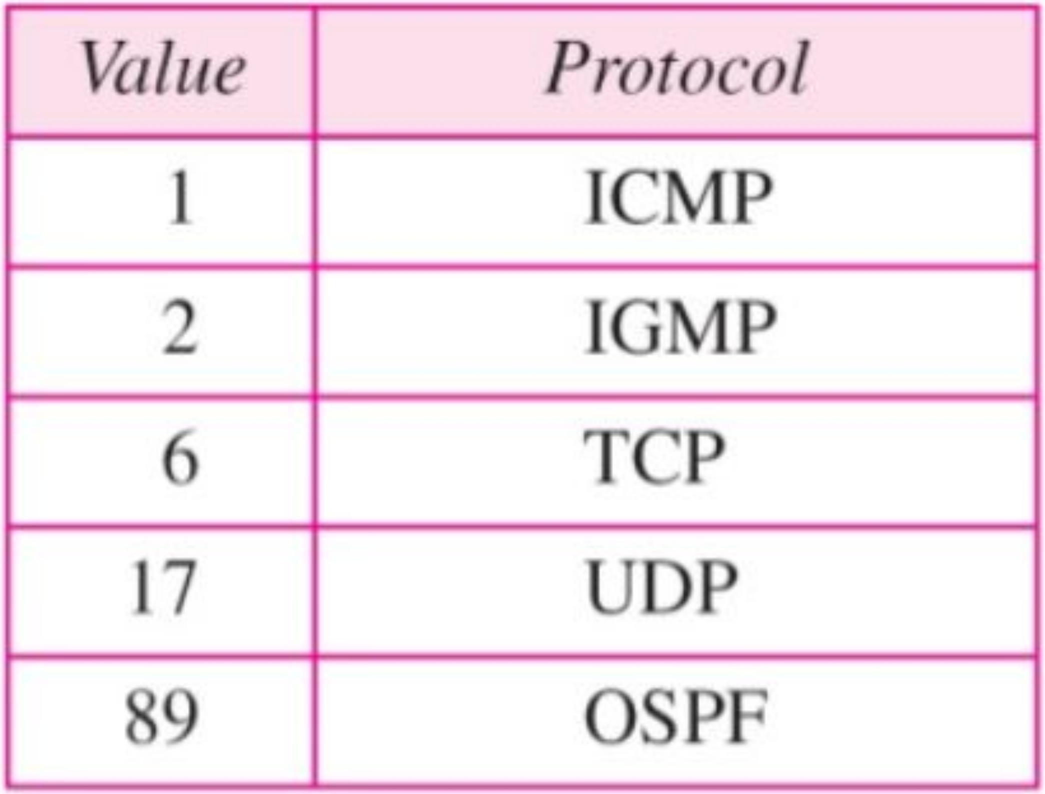
Value	*Protocol*
1	ICMP
2	IGMP
6	TCP
17	UDP
89	OSPF

Default TOS

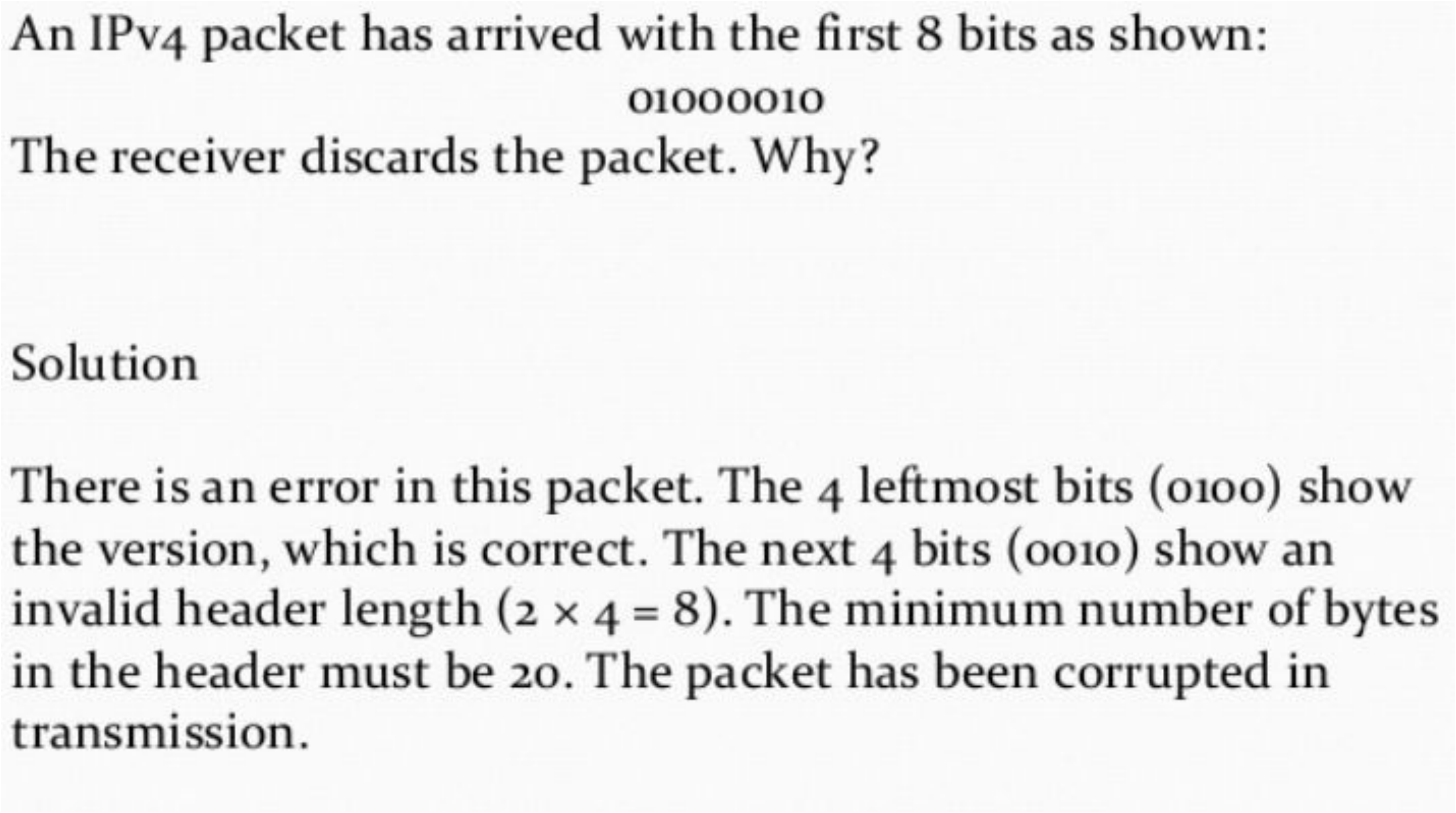


Protocol	*TOS Bits*	*Description*
ICMP	0000	Normal
BOOTP	0000	Normal
NNTP	0001	Minimize cost
IGP	0010	Maximize reliability
SNMP	0010	Maximize reliability
TELNET	1000	Minimize delay
FTP (data)	0100	Maximize throughput
FTP (control)	1000	Minimize delay
TFTP	1000	Minimize delay
SMTP (command)	1000	Minimize delay
SMTP (data)	0100	Maximize throughput
DNS (UDP query)	1000	Minimize delay
DNS (TCP query)	0000	Normal
DNS (zone)	0100	Maximize throughput

Protocol Values



Value	*Protocol*
1	ICMP
2	IGMP
6	TCP
17	UDP
89	OSPF



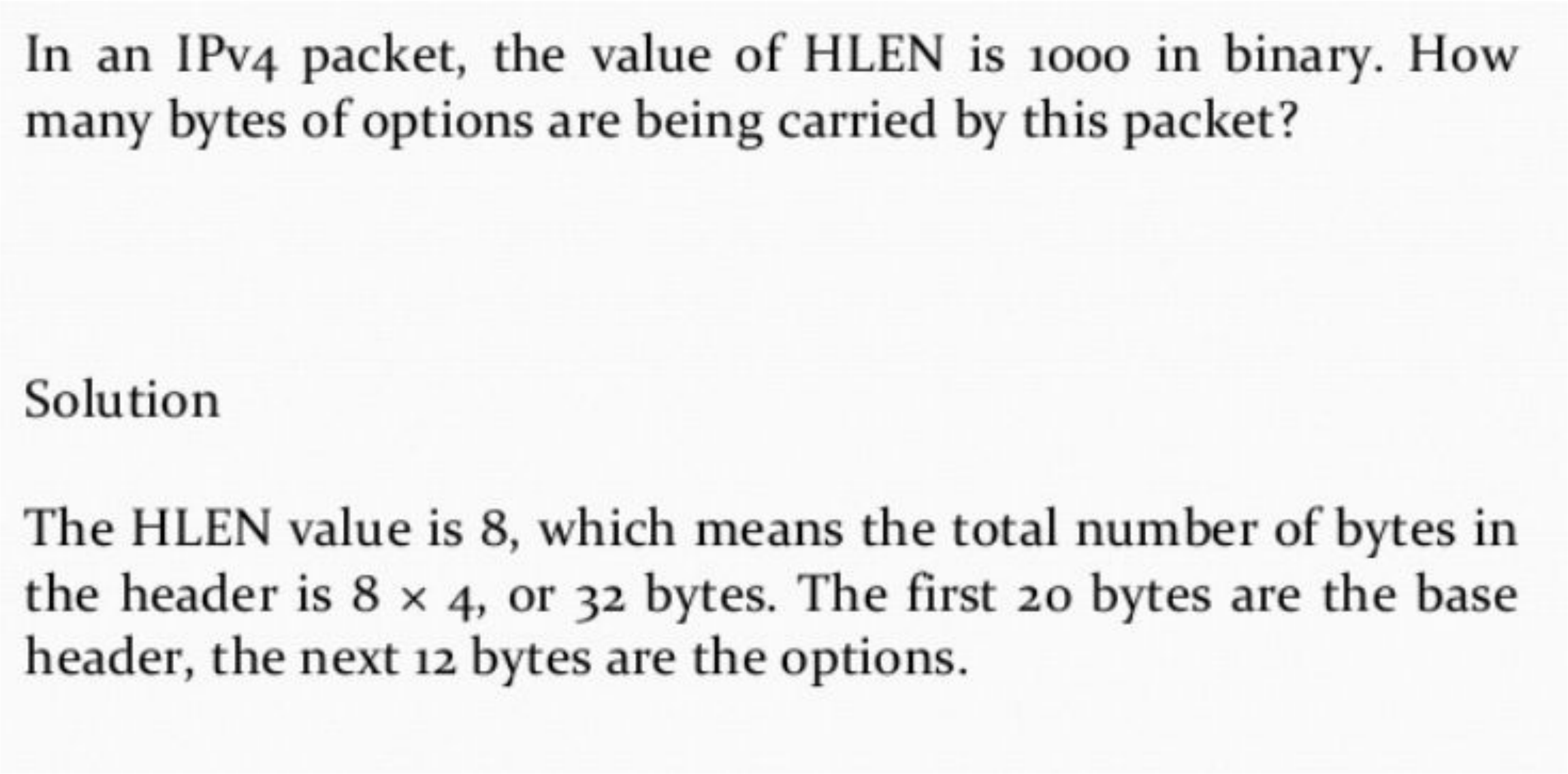
An IPv4 packet has arrived with the first 8 bits as shown:

01000010

The receiver discards the packet. Why?

Solution

There is an error in this packet. The 4 leftmost bits (0100) show the version, which is correct. The next 4 bits (0010) show an invalid header length (2 × 4 = 8). The minimum number of bytes in the header must be 20. The packet has been corrupted in transmission.



In an IPv4 packet, the value of HLEN is 1000 in binary. How many bytes of options are being carried by this packet?

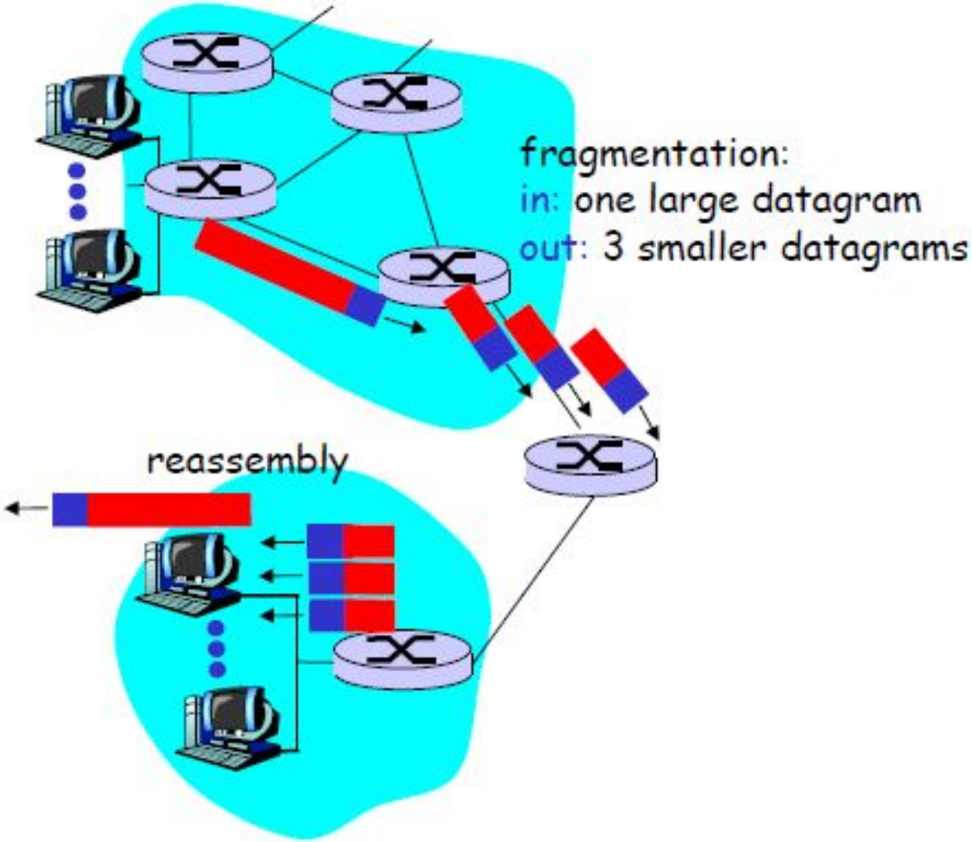
Solution

The HLEN value is 8, which means the total number of bytes in the header is 8 × 4, or 32 bytes. The first 20 bytes are the base header, the next 12 bytes are the options.

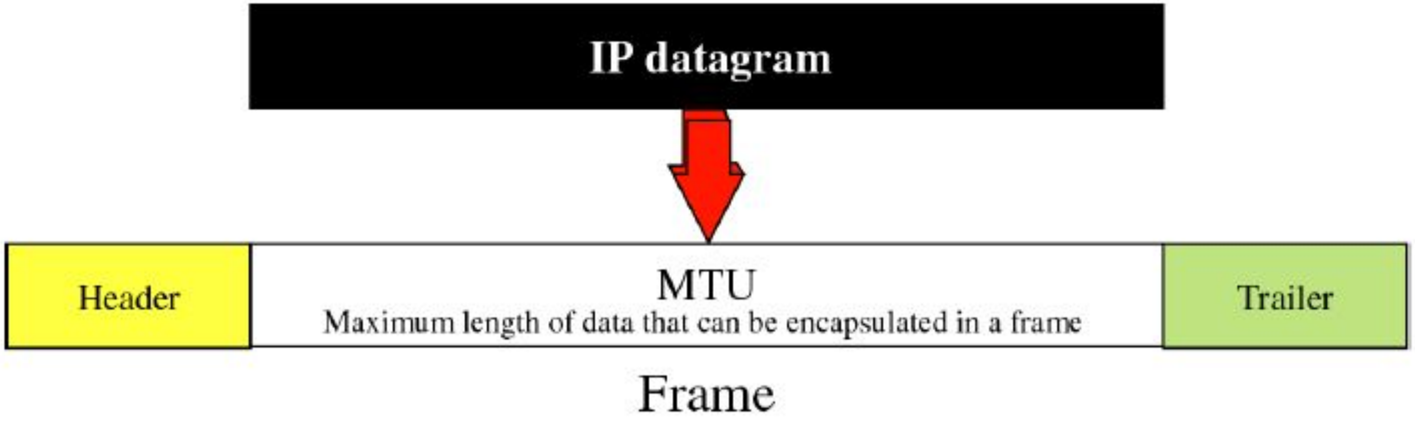
Fragmentation and NAT

IP Fragmentation & Reassembly

- network links have MTU (max.transfer size) - largest possible link-level frame.
  - different link types, different MTUs
- large IP datagram divided ("fragmented") within net
  - one datagram becomes several datagrams
  - "reassembled" only at final destination
  - IP header bits used to identify, order related fragments



- MTU: Maximum Transfer Unit



IP fields related to fragmentation:

Identification (16 bits):

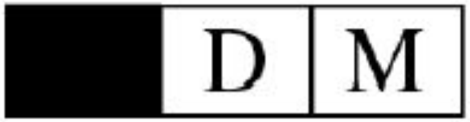
- Provides uniqueness of each datagram.
- Copied into all fragments.

Flags (3 bits):

- First bit is reserved.
- If *do not fragment* bit is set, and datagram does not fit the frame, datagram is discarded and ICMP error message is sent.
- If *more fragments* bit is set, there are other fragments following.

D: Do not fragment

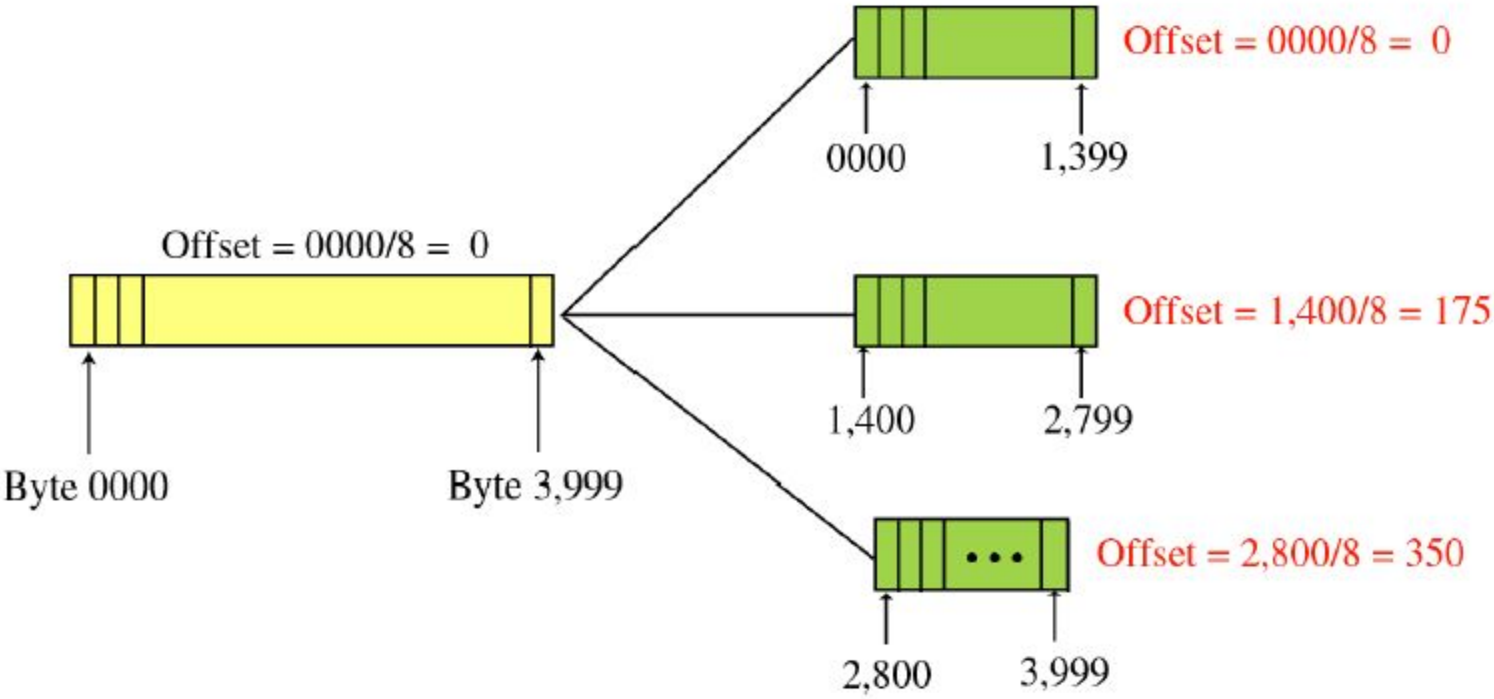
M: More fragments



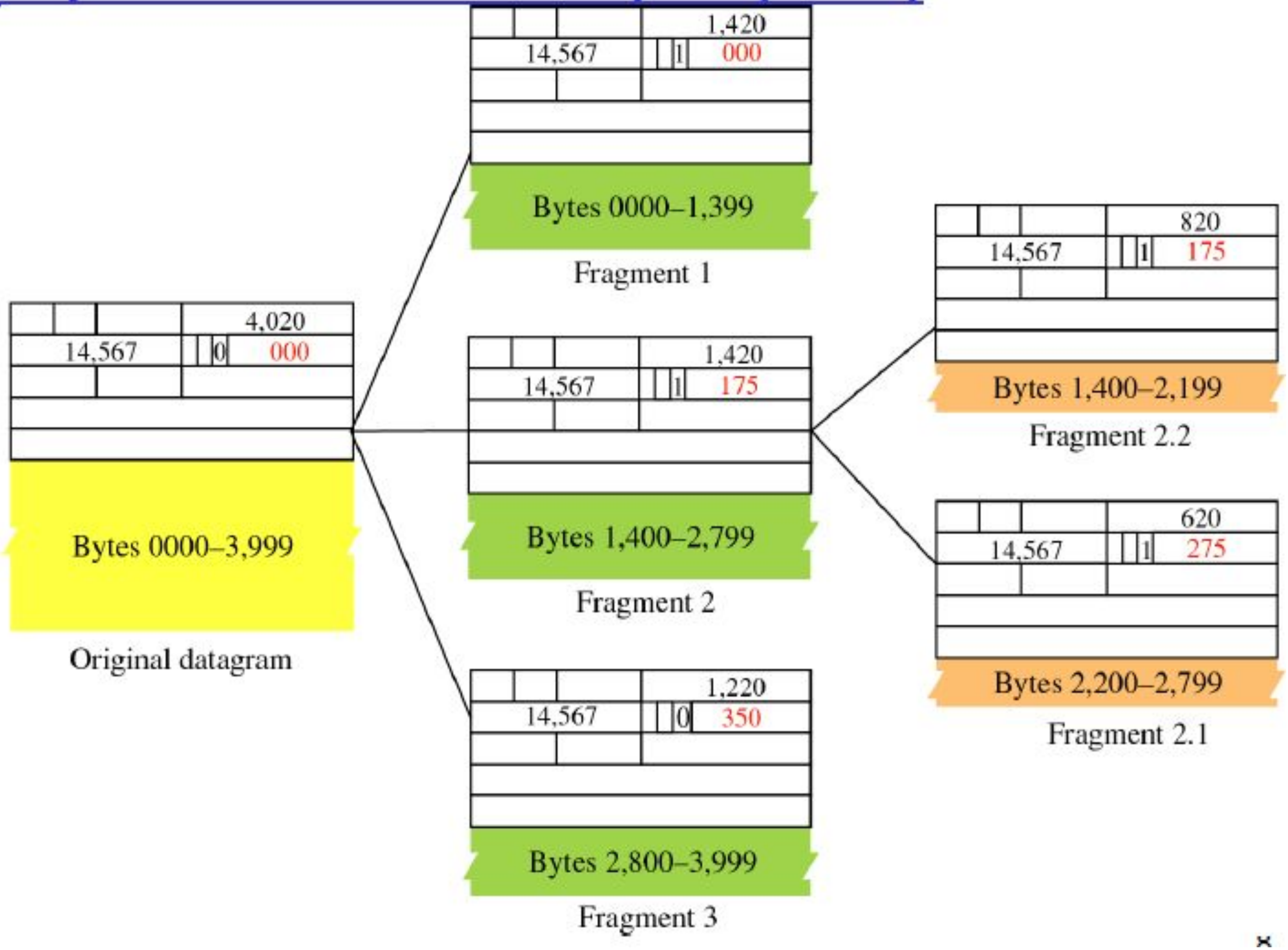
Fragmentation offset (13 bits):

- Offset of data in the original datagram.
- Specified in units of 8 bytes.

Fragmentation Example



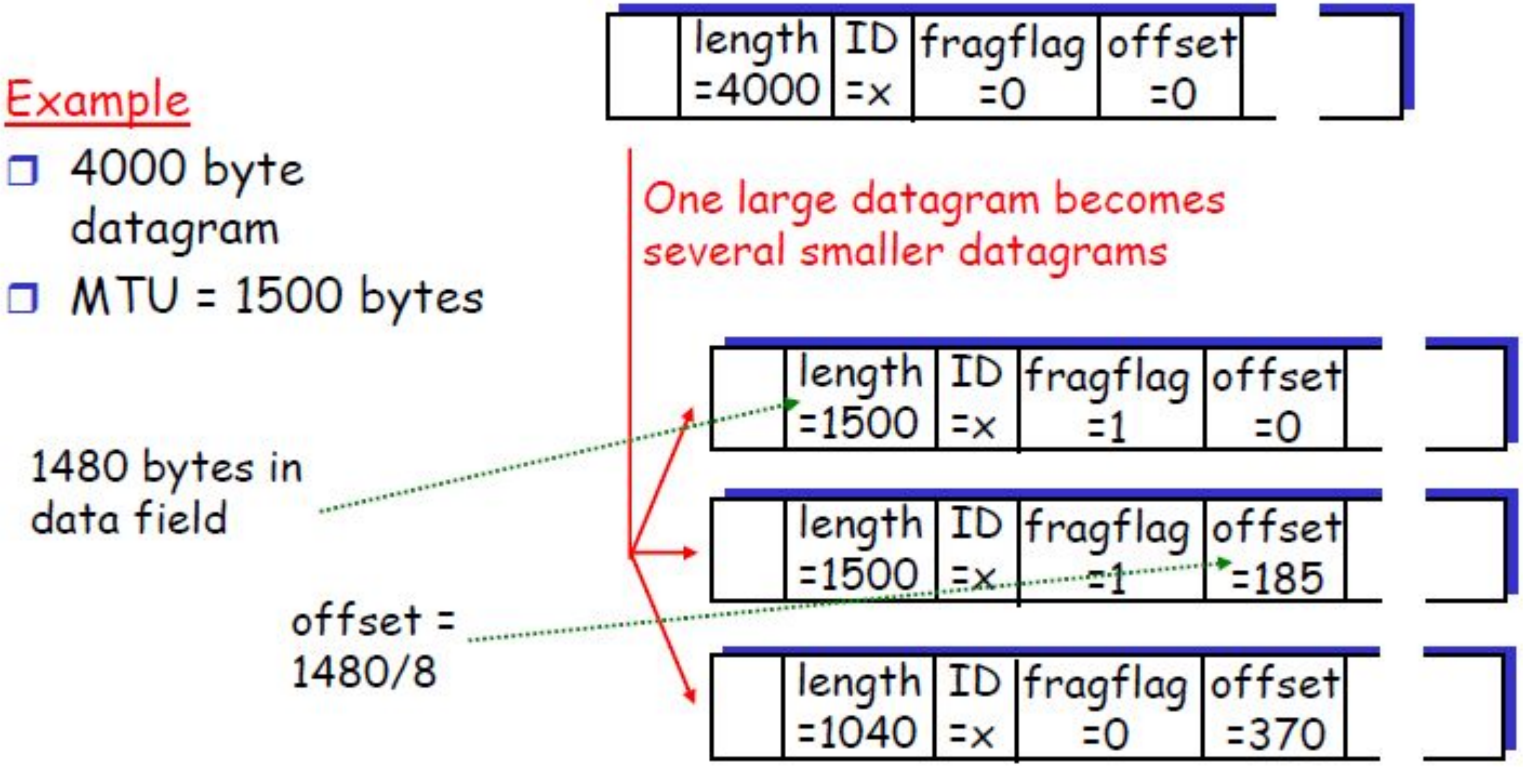
Fragmentation Example (cont'd)



IP Fragmentation and Reassembly

Example

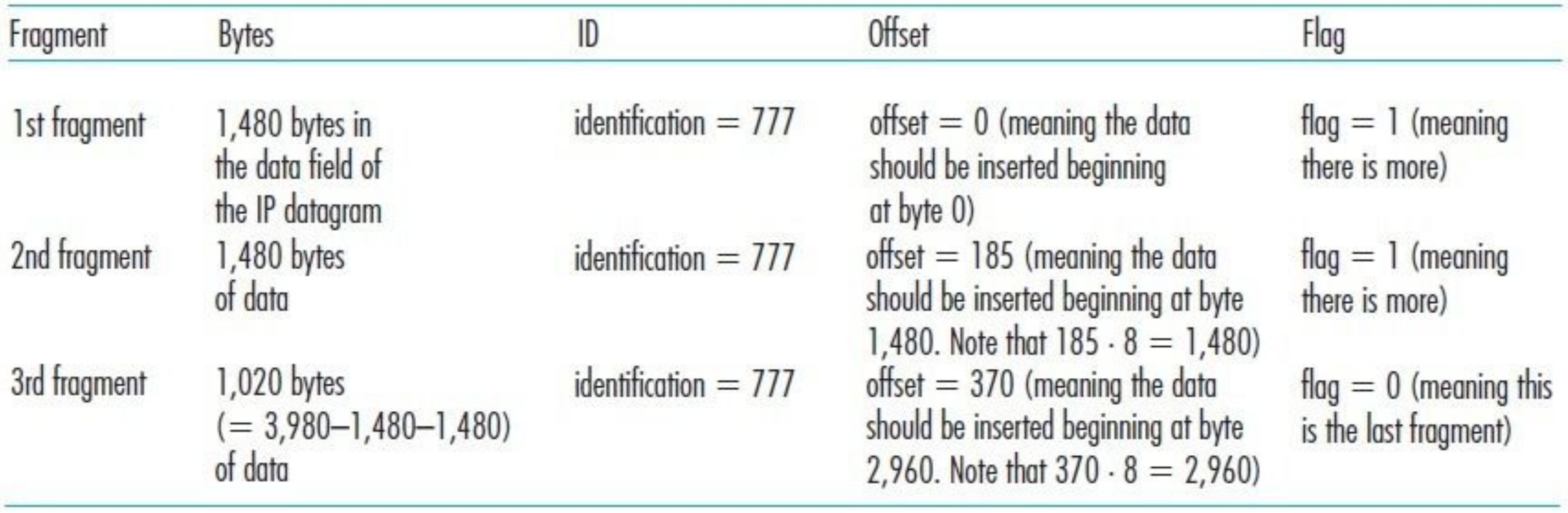
- 4000 byte datagram
- MTU = 1500 bytes



length	ID	fragflag	offset
=4000	=x	=0	=0

One large datagram becomes several smaller datagrams

length	ID	fragflag	offset
=1500	=x	=1	=0
=1500	=x	=1	=185
=1040	=x	=0	=370



Fragment	Bytes	ID	Offset	Flag
1st fragment	1,480 bytes in the data field of the IP datagram	identification = 777	offset = 0 (meaning the data should be inserted beginning at byte 0)	flag = 1 (meaning there is more)
2nd fragment	1,480 bytes of data	identification = 777	offset = 185 (meaning the data should be inserted beginning at byte 1,480. Note that 185 · 8 = 1,480)	flag = 1 (meaning there is more)
3rd fragment	1,020 bytes (= 3,980–1,480–1,480) of data	identification = 777	offset = 370 (meaning the data should be inserted beginning at byte 2,960. Note that 370 · 8 = 2,960)	flag = 0 (meaning this is the last fragment)

Some Points

- A packet has arrived with an M bit value of 0.
- A packet has arrived with an M bit value of 1.
    - Is this the first fragment, the last fragment, or a middle fragment?
    - Do we know if the packet was fragmented?

Example 7.5

A packet has arrived with an M bit value of 0. Is this the first fragment, the last fragment, or a middle fragment? Do we know if the packet was fragmented?

Solution

If the M bit is 0, it means that there are no more fragments; the fragment is the last one. However, we cannot say if the original packet was fragmented or not. A nonfragmented packet is considered the last fragment.

Cont..

- A packet has arrived with an M bit value of 1 and a fragmentation offset value of zero.
  - Is this the first fragment, the last fragment, or a middle fragment?
- A packet has arrived in which the offset value is 100.
  - What is the number of the first byte?
  - Do we know the number of the last byte?

Example 7.6

A packet has arrived with an *M* bit value of 1. Is this the first fragment, the last fragment, or a middle fragment? Do we know if the packet was fragmented?

Solution

If the *M* bit is 1, it means that there is at least one more fragment. This fragment can be the first one or a middle one, but not the last one. We don't know if it is the first one or a middle one; we need more information (the value of the fragmentation offset). See also the next example.

Example 7.7

A packet has arrived with an *M* bit value of 1 and a fragmentation offset value of zero. Is this the first fragment, the last fragment, or a middle fragment?

Solution

Because the *M* bit is 1, it is either the first fragment or a middle one. Because the offset value is 0, it is the first fragment.

Example 7.8

A packet has arrived in which the offset value is 100. What is the number of the first byte? Do we know the number of the last byte?

Solution

To find the number of the first byte, we multiply the offset value by 8. This means that the first byte number is 800. We cannot determine the number of the last byte unless we know the length of the data.

Example 7.9

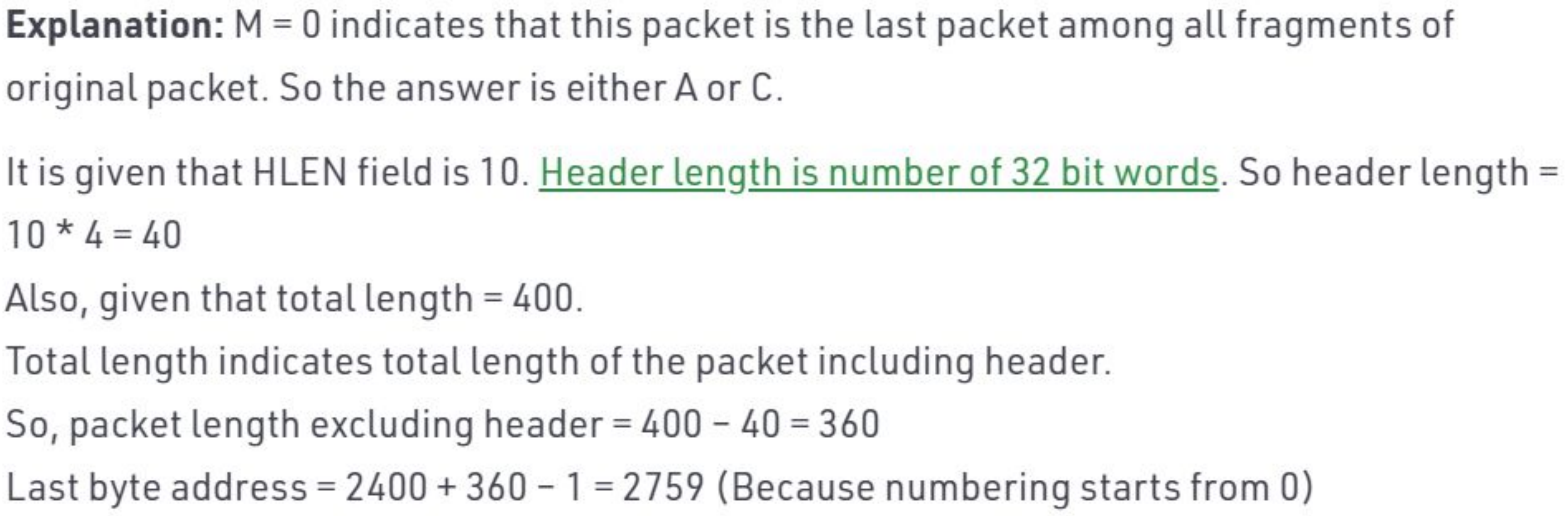
A packet has arrived in which the offset value is 100, the value of HLEN is 5 and the value of the total length field is 100. What is the number of the first byte and the last byte?

Solution

The first byte number is $100 \times 8 = 800$. The total length is 100 bytes and the header length is 20 bytes (5×4), which means that there are 80 bytes in this datagram. If the first byte number is 800, the last byte number must be 879.

In an IPv4 datagram, the M bit is 0, the value of HLEN is 10, the value of total length is 400 and the fragment offset value is 300. The position of the datagram, the sequence numbers of the first and the last bytes of the payload, respectively are

(A) Last fragment, 2400 and 2789
(B) First fragment, 2400 and 2759
(C) Last fragment, 2400 and 2759
(D) Middle fragment, 300 and 689



Explanation: M = 0 indicates that this packet is the last packet among all fragments of original packet. So the answer is either A or C.

It is given that HLEN field is 10. Header length is number of 32 bit words. So header length = 10 * 4 = 40

Also, given that total length = 400.

Total length indicates total length of the packet including header.

So, packet length excluding header = 400 – 40 = 360

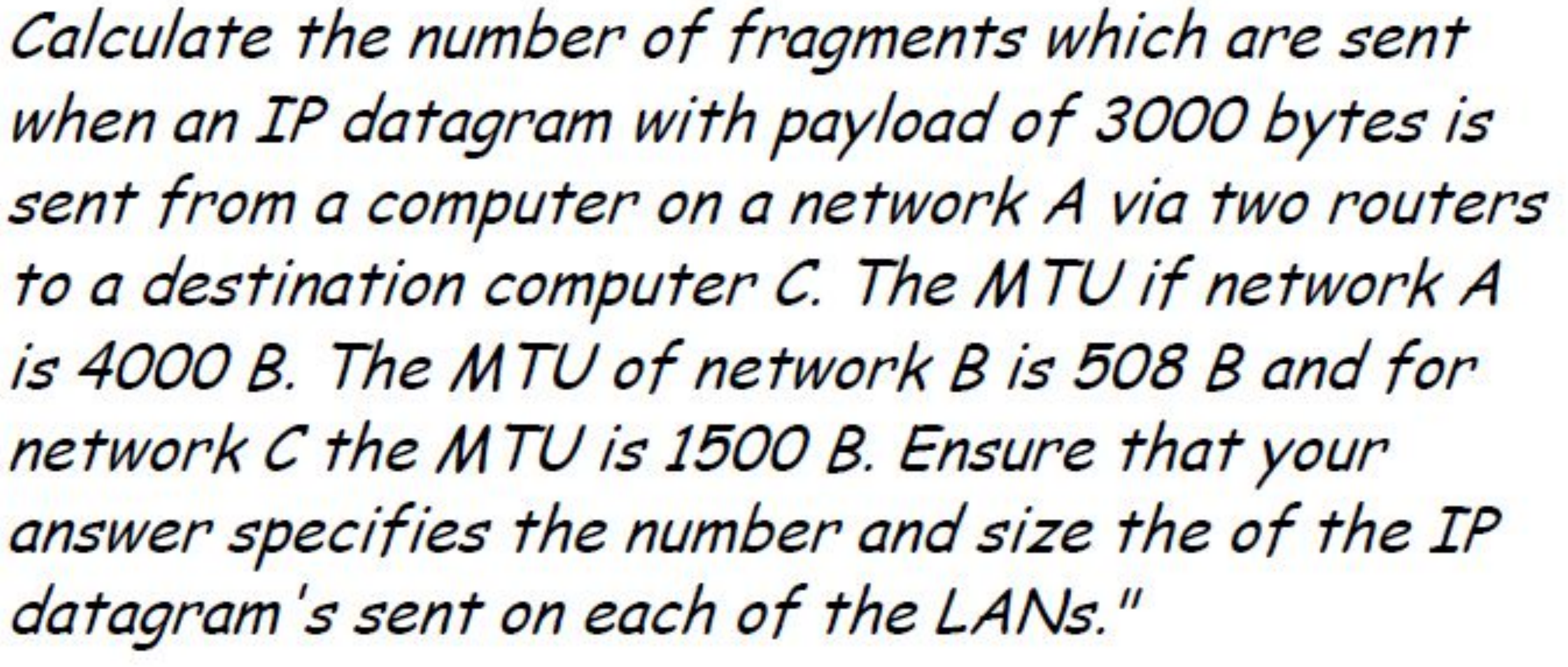
Last byte address = 2400 + 360 – 1 = 2759 (Because numbering starts from 0)

Consider sending a 3000 byte datagram into a link that has an MTU of 500 bytes. How many fragments are generated? What are their characteristics (i.e. what are the flags and offset values for each

Consider sending a 3000 byte datagram into a link that has an MTU of 500 bytes. How many fragments are generated? What are their characteristics (i.e. what are the flags and offset values for each

Assume that the DF flag was not set :)
Assume that no optional fields of the IP header are in use (i.e. IP header is 20 bytes)
The original datagram was 3000 bytes, subtracting 20 bytes for header, that leaves 2980 bytes of data.
Assume the ID of the original packet is 'x'
With an MTU of 500 bytes, 500 - 20 = 480 bytes of data may be transmitted in each packet
Therefore, ceiling(2980 / 480) = 7 packets are needed to carry the data.
The packets will have the following characteristics (NOTE: offset is measured in 8 byte blocks, you don't need to specify Total_len)

Packet 1: ID=x, Total_len=500, MF=1, Frag_offset=0
Packet 2: ID=x, Total_len=500, MF=1, Frag_offset=60
Packet 3: ID=x, Total_len=500, MF=1, Frag_offset=120
Packet 4: ID=x, Total_len=500, MF=1, Frag_offset=180
Packet 5: ID=x, Total_len=500, MF=1, Frag_offset=240
Packet 6: ID=x, Total_len=500, MF=1, Frag_offset=300
Packet 7: ID=x, Total_len=120, MF=0, Frag_offset=360



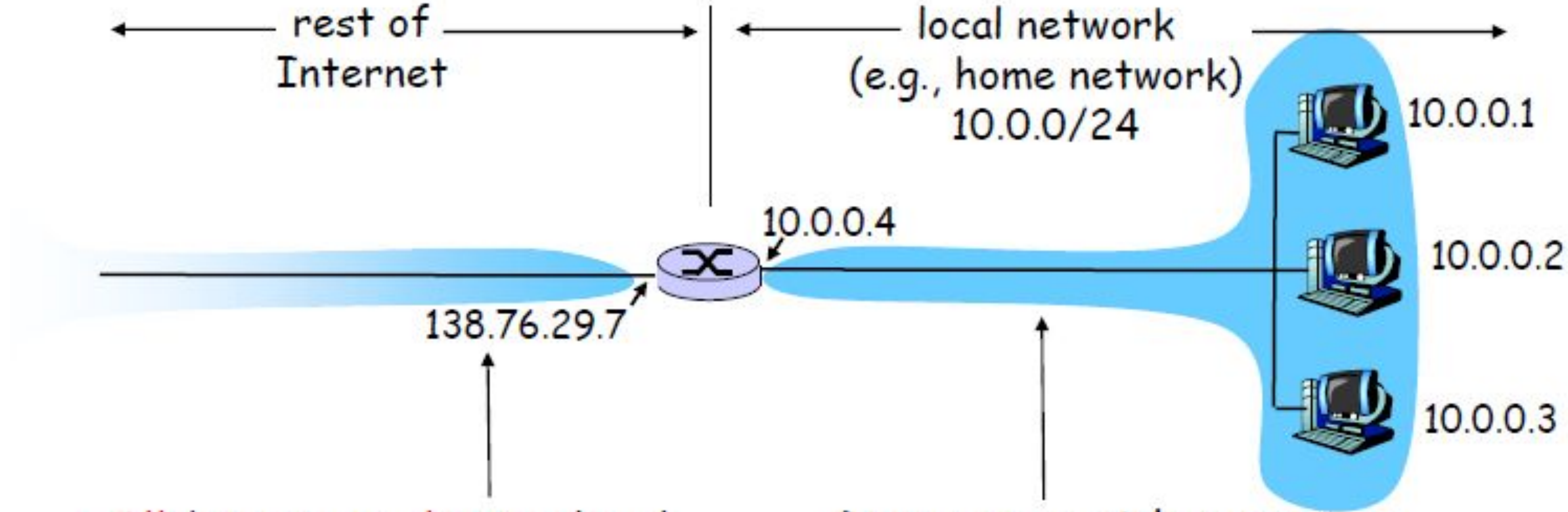
Calculate the number of fragments which are sent when an IP datagram with payload of 3000 bytes is sent from a computer on a network A via two routers to a destination computer C. The MTU if network A is 4000 B. The MTU of network B is 508 B and for network C the MTU is 1500 B. Ensure that your answer specifies the number and size the of the IP datagram's sent on each of the LANs."

Solution

- **On First LAN**
- Total size of initial PDU = 3000 + 20 B (PCI) = 3020 B.
- Network A: MTU 4000 B > 3020 B - therefore one packet is sent.
- (packet sent with offset=0, more=FALSE
- **On Second LAN**
- Network B: MTU 508 B < 3020 B - therefore fragmentation is required.
- IP fragment payload size = 508- 20 B = 488 B. (Note this is aligned to an 8-byte boundary)
- Total number of packets sent via network B = round(3000/488) = 7 packets.
- First six packets of size 508 B, each with 488B of IP packet payload
- The last packet has 48 B of data, and therefore of size 20 B + 72 B=92 B.
- (all packets except first sent with offset>0,)
- (packets 1-6 have more=TRUE, last packet has more=FALSE)
- (This last fragment does not need to have a length that is divisible by 8)

- **On Final LAN**
- Fragments are not reassembled by a router - i.e. at router C.
- Router C therefore receives 7 packets, 6 of size 508 B, 1 of size 92 B.
- Network C: MTU 1500 B > 508 B - No further fragmentation is therefore needed.
- i.e. There are 7 packets, as in network C, 6 of size 508 B and one of size 92 B.

NAT: Network Address Translation

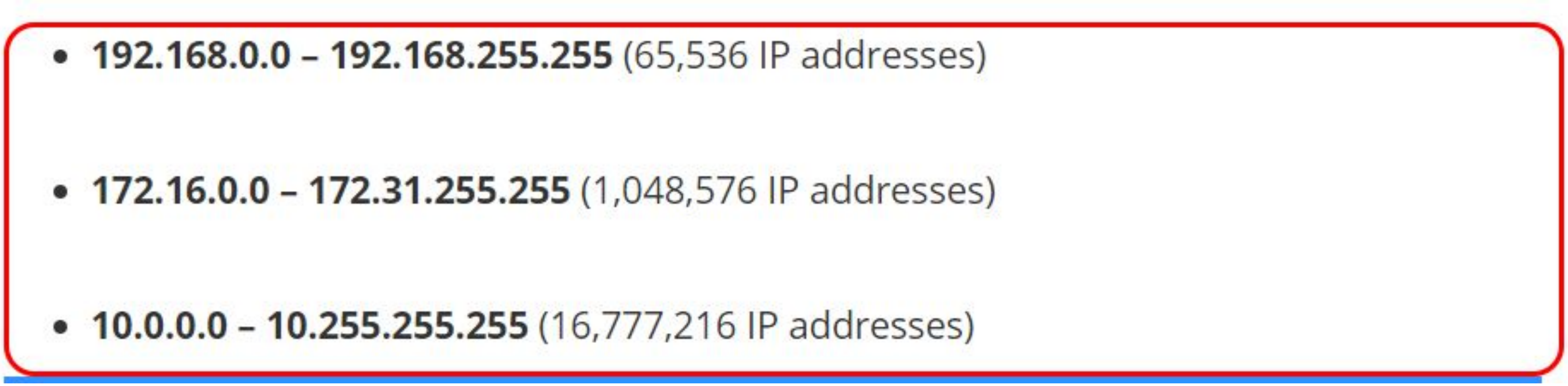


All datagrams *leaving* local network have **same** single source NAT IP address: 138.76.29.7, different source port numbers

Datagrams with source or destination in this network have 10.0.0/24 address for source, destination (as usual)

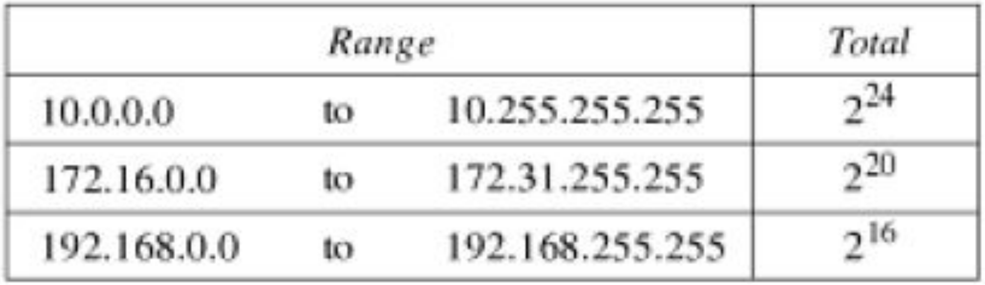
Reserved for private networks

- The organizations that distribute IP addresses to the world reserves a range of IP addresses for *private networks*.
- Your simple home network, with its router at the center and computers connected to it—wired or wireless—classifies as one of those networks.

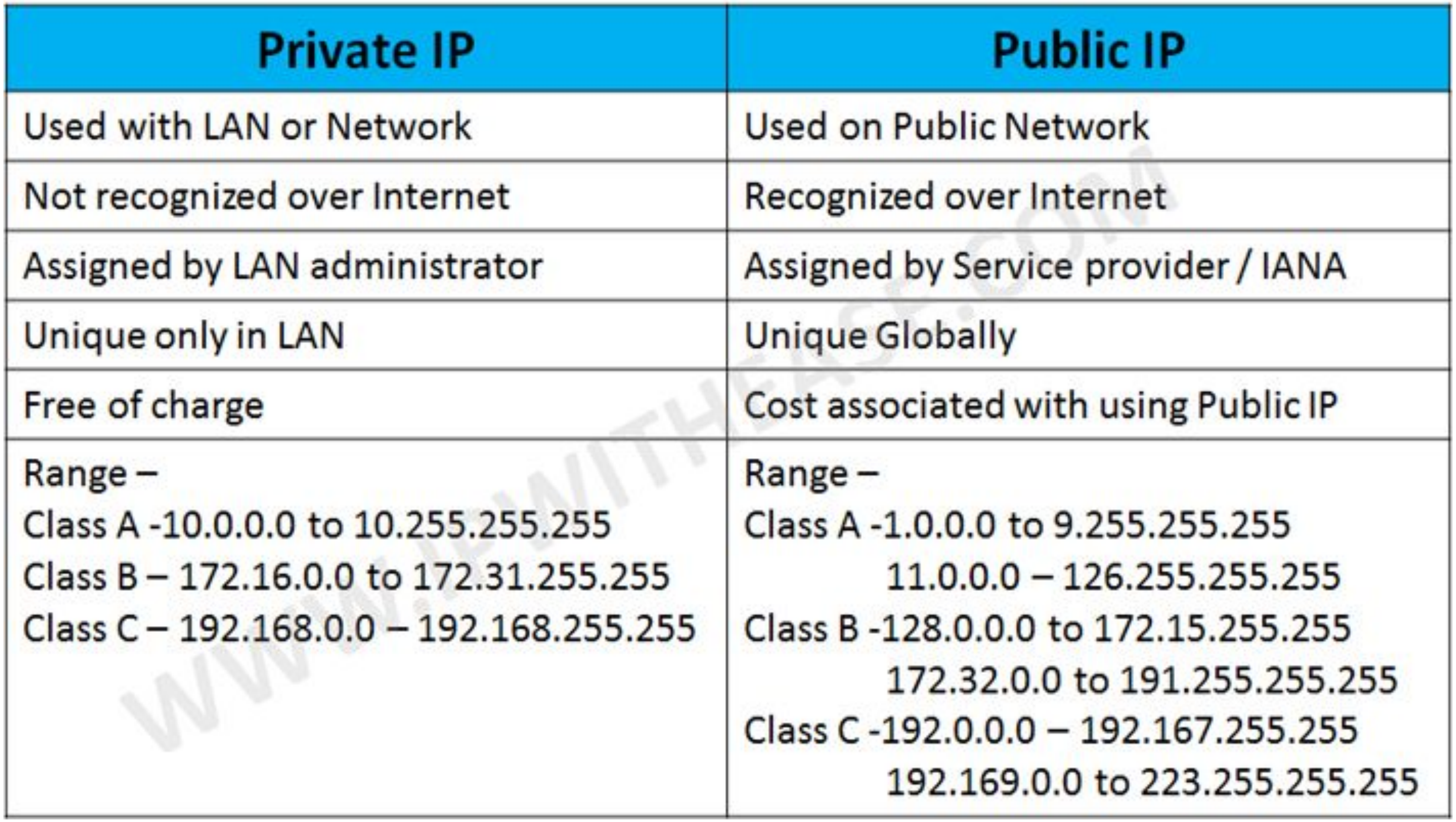


- **192.168.0.0 – 192.168.255.255** (65,536 IP addresses)
- **172.16.0.0 – 172.31.255.255** (1,048,576 IP addresses)
- **10.0.0.0 – 10.255.255.255** (16,777,216 IP addresses)

Addresses for private networks



Range			*Total*
10.0.0.0	to	10.255.255.255	2^{24}
172.16.0.0	to	172.31.255.255	2^{20}
192.168.0.0	to	192.168.255.255	2^{16}



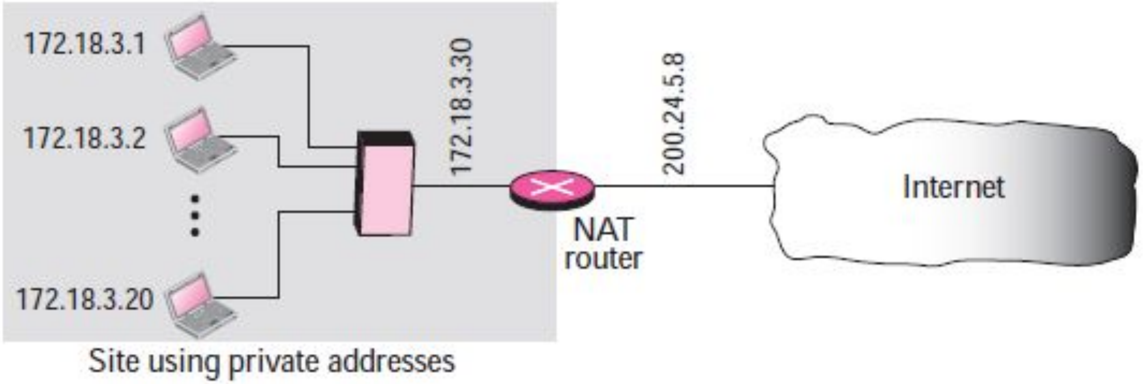
Private IP	Public IP
Used with LAN or Network	Used on Public Network
Not recognized over Internet	Recognized over Internet
Assigned by LAN administrator	Assigned by Service provider / IANA
Unique only in LAN	Unique Globally
Free of charge	Cost associated with using Public IP
Range – Class A -10.0.0.0 to 10.255.255.255 Class B – 172.16.0.0 to 172.31.255.255 Class C – 192.168.0.0 – 192.168.255.255	Range – Class A -1.0.0.0 to 9.255.255.255 11.0.0.0 – 126.255.255.255 Class B -128.0.0.0 to 172.15.255.255 172.32.0.0 to 191.255.255.255 Class C -192.0.0.0 – 192.167.255.255 192.169.0.0 to 223.255.255.255

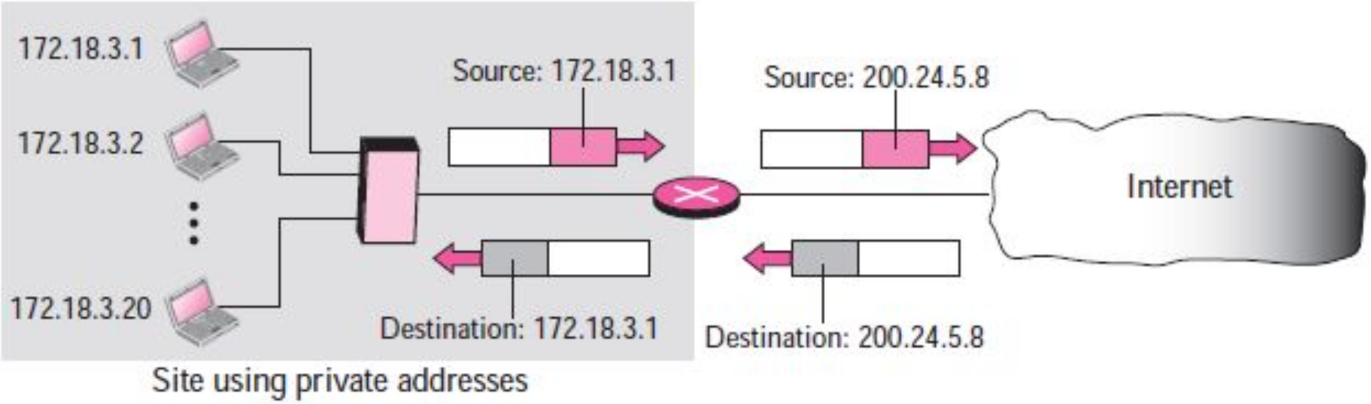
Routers

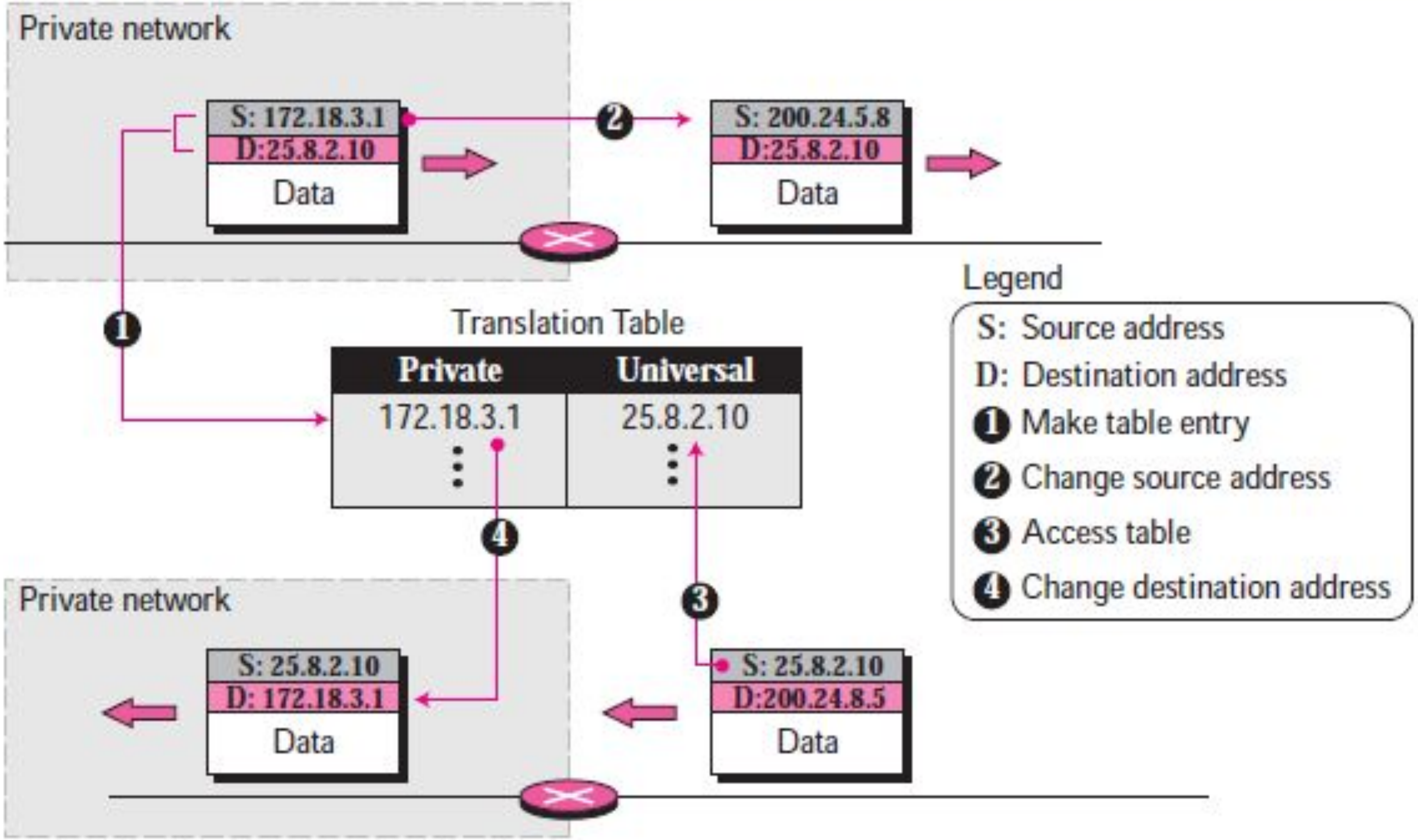
- Here’s a look at the default private (also called “local”) IP addresses for popular brands of routers:
- **Linksys** routers use 192.168.1.1
- **D-Link** and **NETGEAR** routers are set to 192.168.0.1
- **Cisco** routers use either 192.168.10.2, 192.168.1.254 or 192.168.1.1
- **Belkin** and **SMC** routers often use 192.168.2.1

NAT: Network Address Translation

- Motivation: local network uses just one IP address as far as outside world is concerned:
  - range of addresses not needed from ISP: just one IP address for all devices
  - can change addresses of devices in local network without notifying outside world
  - can change ISP without changing addresses of devices in local network
  - devices inside local net not explicitly addressable, visible by outside world (a security plus).


172.18.3.1
172.18.3.2
172.18.3.20
172.18.3.30
200.24.5.8
NAT
router
Internet
Site using private addresses


172.18.3.1
172.18.3.2
172.18.3.20
Source: 172.18.3.1
Source: 200.24.5.8
Internet
Destination: 172.18.3.1
Destination: 200.24.5.8
Site using private addresses



Translation Table

Private	Universal
172.18.3.1	25.8.2.10
⋮	⋮

Legend

S: Source address
D: Destination address
❶ Make table entry
❷ Change source address
❸ Access table
❹ Change destination address

NAT: Network Address Translation

Implementation: NAT router must:

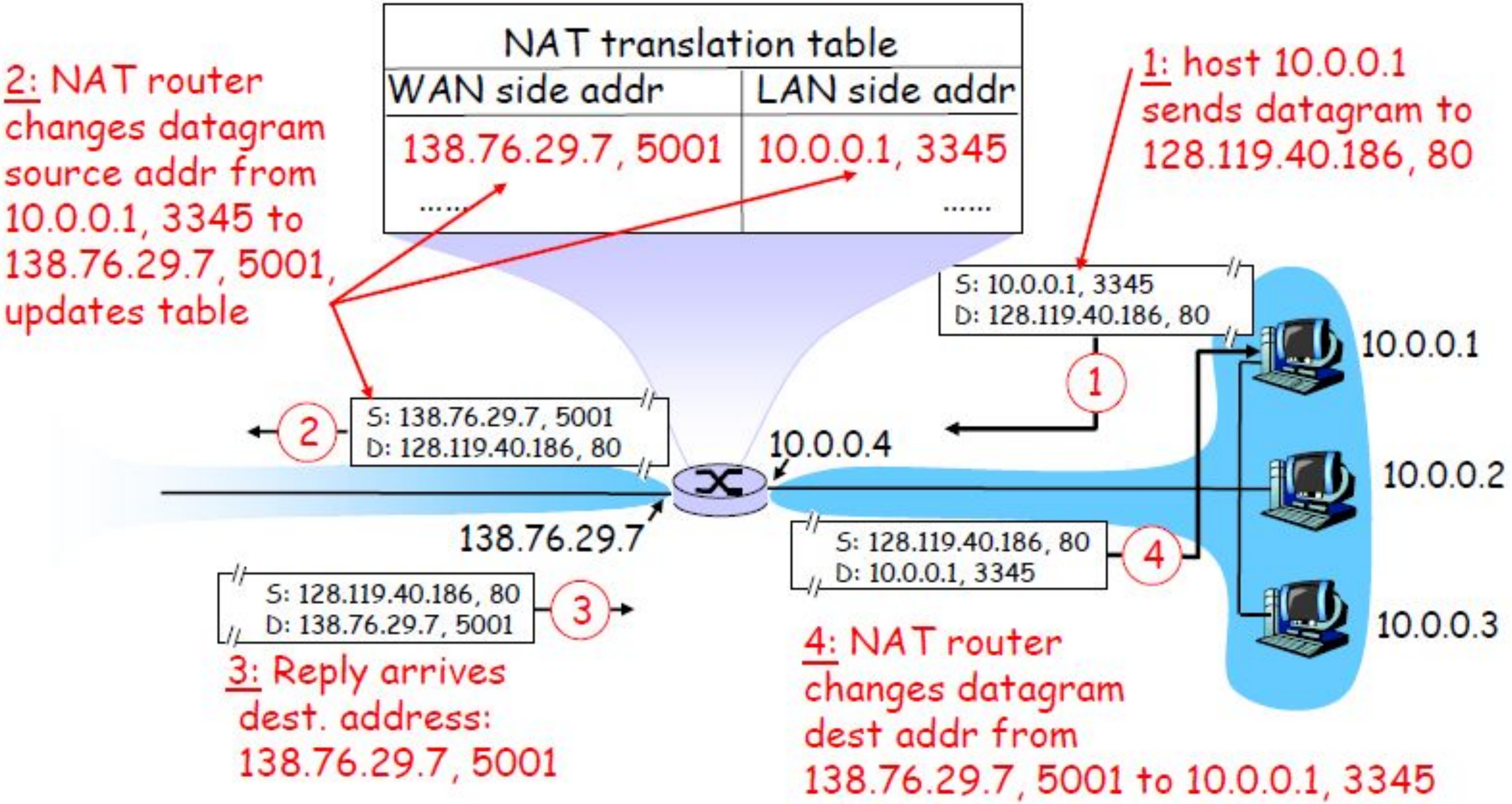
- *outgoing datagrams: replace* (source IP address, port #) of every outgoing datagram to (NAT IP address, new port #)

  . . . remote clients/servers will respond using (NAT IP address, new port #) as destination addr.

- *remember (in NAT translation table)* every (source IP address, port #) to (NAT IP address, new port #) translation pair

- *incoming datagrams: replace* (NAT IP address, new port #) in dest fields of every incoming datagram with corresponding (source IP address, port #) stored in NAT table

NAT: Network Address Translation



NAT translation table

WAN side addr	LAN side addr
138.76.29.7, 5001	10.0.0.1, 3345
……	……

1: host 10.0.0.1 sends datagram to 128.119.40.186, 80

S: 10.0.0.1, 3345
D: 128.119.40.186, 80

2: NAT router changes datagram source addr from 10.0.0.1, 3345 to 138.76.29.7, 5001, updates table

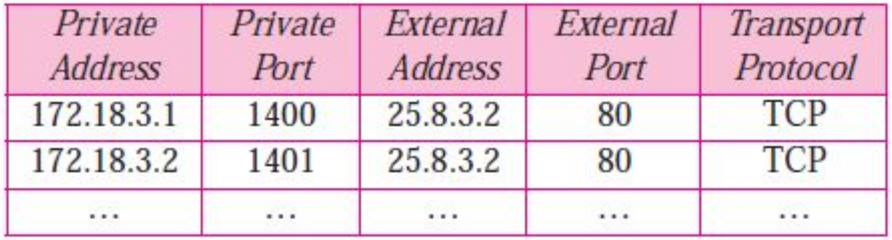
S: 138.76.29.7, 5001
D: 128.119.40.186, 80

S: 128.119.40.186, 80
D: 138.76.29.7, 5001

3: Reply arrives dest. address: 138.76.29.7, 5001

S: 128.119.40.186, 80
D: 10.0.0.1, 3345

4: NAT router changes datagram dest addr from 138.76.29.7, 5001 to 10.0.0.1, 3345



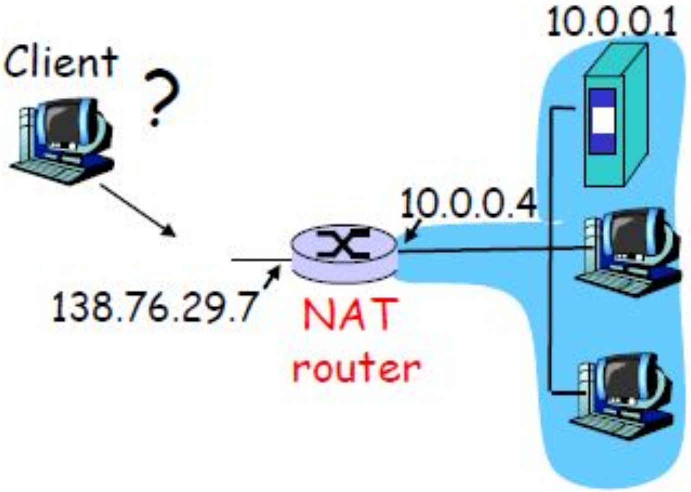
Private Address	*Private Port*	*External Address*	*External Port*	*Transport Protocol*
172.18.3.1	1400	25.8.3.2	80	TCP
172.18.3.2	1401	25.8.3.2	80	TCP
...	...	...	...	...

NAT: Network Address Translation

- 16-bit port-number field:
  - 60,000 simultaneous connections with a single LAN-side address!
- NAT is controversial:
  - routers should only process up to layer 3
  - violates end-to-end argument
    - NAT possibility must be taken into account by app designers, eg, P2P applications
  - address shortage should instead be solved by IPv6

NAT traversal problem

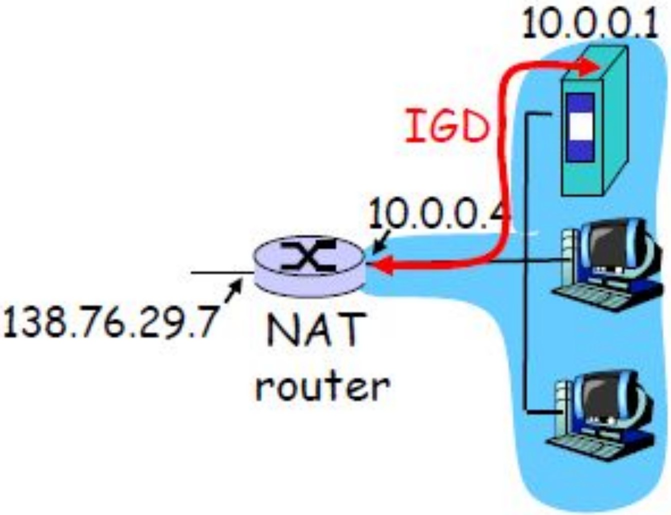
- client wants to connect to server with address 10.0.0.1
  - server address 10.0.0.1 local to LAN (client can't use it as destination addr)
  - only one externally visible NATted address: 138.76.29.7
- solution 1: statically configure NAT to forward incoming connection requests at given port to server
  - e.g., (123.76.29.7, port 2500) always forwarded to 10.0.0.1 port 25000



NAT traversal problem

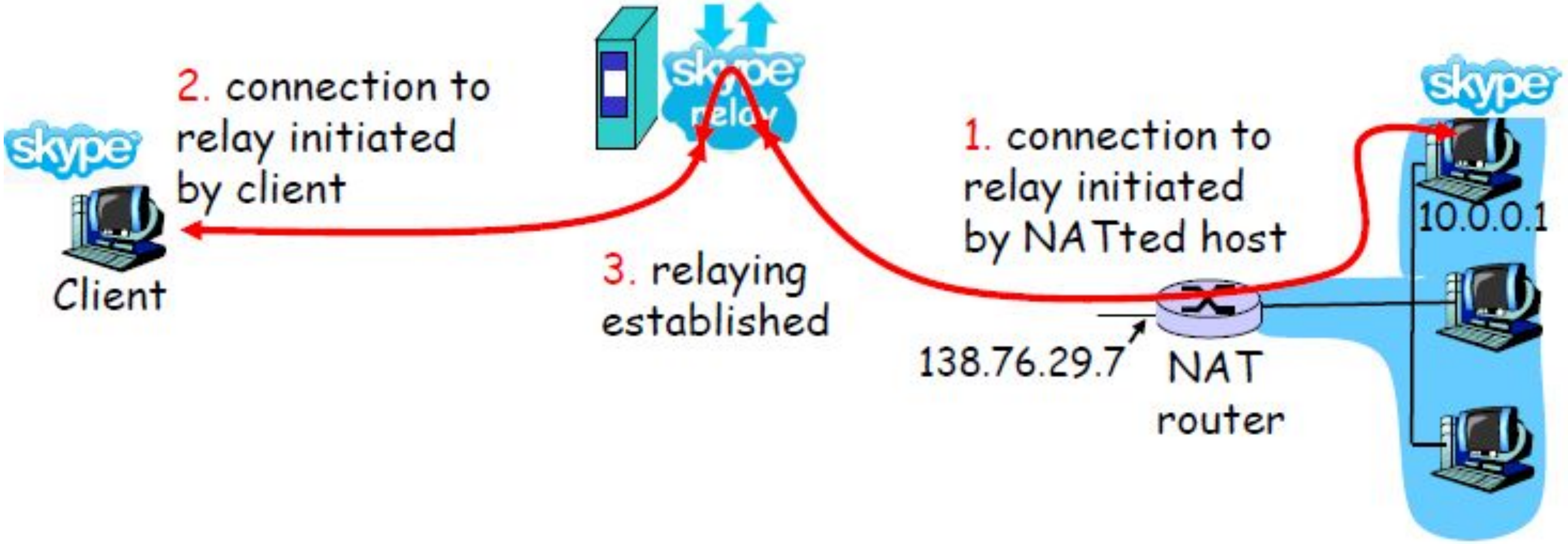
- solution 2: Universal Plug and Play (UPnP) Internet Gateway Device (IGD) Protocol. Allows NATted host to:
  - learn public IP address (138.76.29.7)
  - add/remove port mappings (with lease times)

  i.e., automate static NAT port map configuration



NAT traversal problem

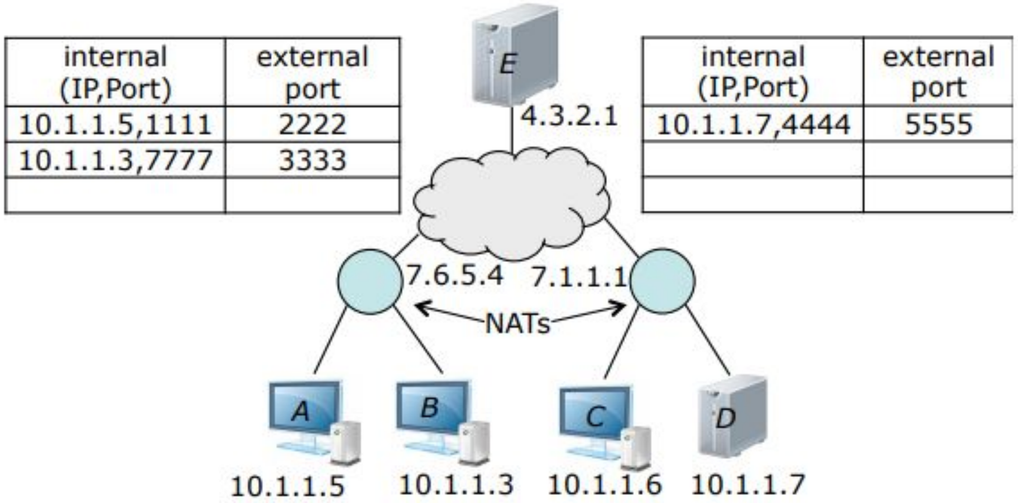
- solution 3: relaying (used in Skype)
  - NATed client establishes connection to relay
  - External client connects to relay
  - relay bridges packets between to connections



skype relay
2. connection to relay initiated by client
skype
Client
3. relaying established
1. connection to relay initiated by NATted host
skype
10.0.0.1
138.76.29.7
NAT router

(10 points). The figure at right shows two residential networks with routers that implement NAT. Suppose host *A* is connected to the web server at host *E*.

In the left-hand NAT table, add an entry that would allow *A* to communicate with *E*. You may choose any port numbers you like, but the internal port numbers should be different from the external port numbers.



internal (IP,Port)	external port
10.1.1.5,1111	2222
10.1.1.3,7777	3333

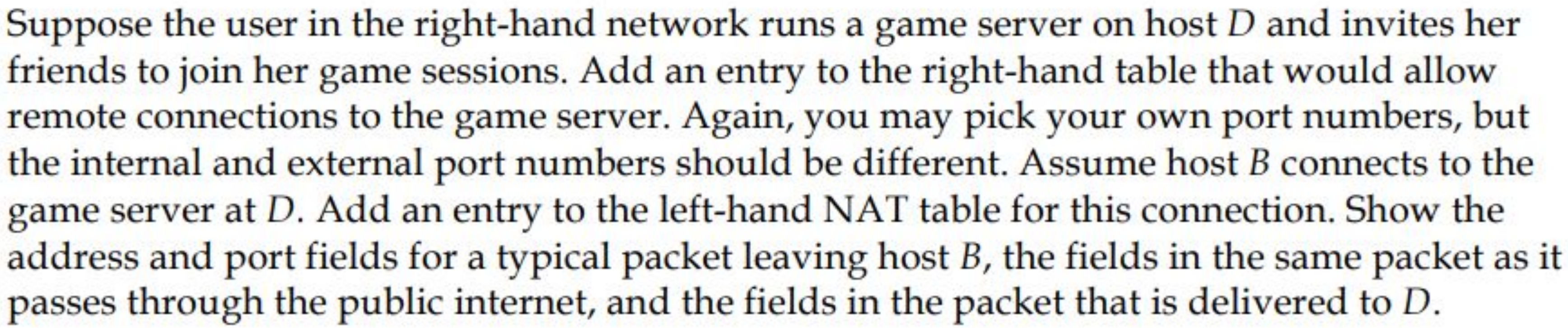
internal (IP,Port)	external port
10.1.1.7,4444	5555

Show the values of the address and port fields in the diagram below, for a typical packet sent by host *A*.

src adr	dest adr	src port	dest port
10.1.1.5	4.3.2.1	7777	80

Show the fields in the packet as it might appear when it reaches *E*.

src adr	dest adr	src port	dest port
7.6.5.4	4.3.2.1	2222	80



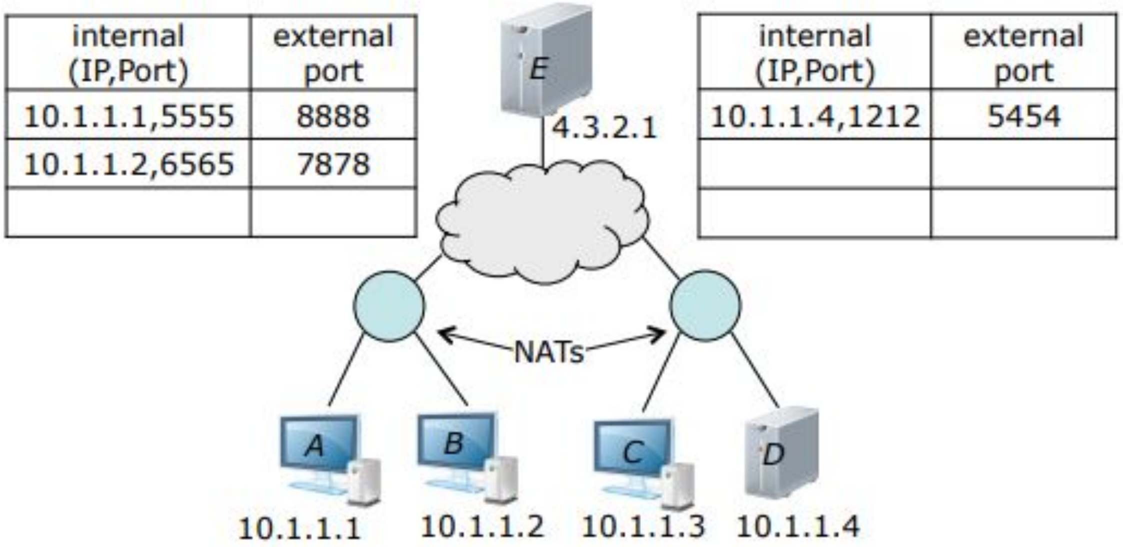
Suppose the user in the right-hand network runs a game server on host *D* and invites her friends to join her game sessions. Add an entry to the right-hand table that would allow remote connections to the game server. Again, you may pick your own port numbers, but the internal and external port numbers should be different. Assume host *B* connects to the game server at *D*. Add an entry to the left-hand NAT table for this connection. Show the address and port fields for a typical packet leaving host *B*, the fields in the same packet as it passes through the public internet, and the fields in the packet that is delivered to *D*.

src adr	dest adr	src port	dest port
10.1.1.3	7.1.1.1	7777	5555

src adr	dest adr	src port	dest port
7.6.5.4	7.1.1.1	3333	5555

src adr	dest adr	src port	dest port
7.6.5.4	10.1.1.7	3333	4444

(15 points). The diagram below shows two residential networks with routers that implement NAT and a remote server with a public internet address



internal (IP,Port)	external port
10.1.1.1,5555	8888
10.1.1.2,6565	7878

internal (IP,Port)	external port
10.1.1.4,1212	5454

The packet header diagrams at right are for a packet from a host in the left-hand network, going to the server. The first shows the header when the packet arrives at the router, the second shows it when the packet leaves the router. Add an entry to the left-hand NAT table that is consistent with these two packet headers. What is the public IP address of the left-hand router?

src adr	dest adr	src port	dest port
10.1.1.1	4.3.2.1	5555	3333
3.7.5.7	4.3.2.1	8888	3333

3.7.5.7

The three header diagrams at right are for a packet from a host in the right-hand network, going to a host in the left-hand network. Fill in the blank fields. Add entries to the two NAT tables that are consistent with this sequence of packet headers. What is the public IP address of the right-hand router?

src adr	dest adr	src port	dest port
10.1.1.4	3.7.5.7	1212	7878
5.3.5.2	3.7.5.7	5454	7878
5.3.5.2	10.1.1.2	5454	6565

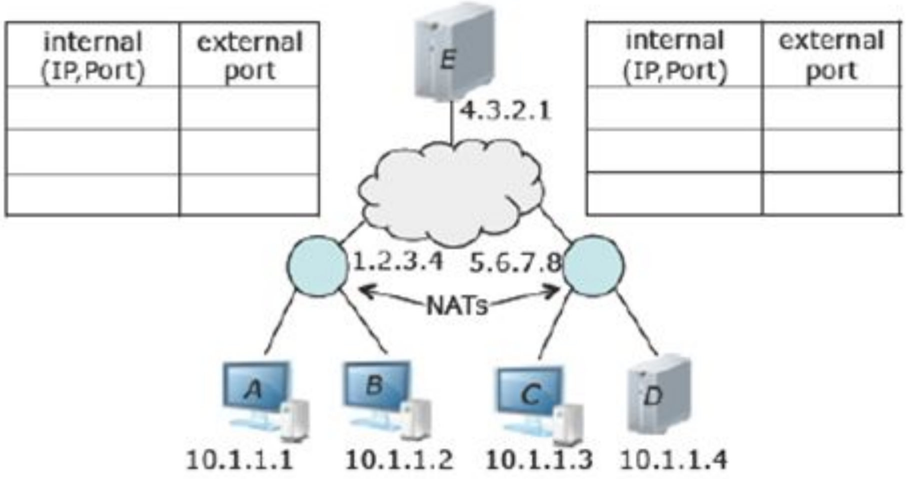
5.3.5.2

In the diagrams at right, fill in the header fields that would be used by a response to the last packet, (the response goes from the right hand network to the left).

src adr	dest adr	src port	dest port
10.1.1.2	5.3.5.2	6565	5454
3.7.5.7	5.3.5.2	7878	5454
3.7.5.7	10.1.1.4	7878	1212

2.

The figure at right shows two residential networks with routers that implement NAT. Suppose host *A* is connected to the web server at host *E*. In the left-hand NAT table, add an entry that would allow *A* to communicate with *E*. You may choose any port numbers you like. Show the values of the address and port fields in the diagram below, for a typical packet sent by host *A*.



internal (IP,Port)	external port

internal (IP,Port)	external port

src adr	dest adr	src port	dest port

Show the fields in the packet as it might appear when it reaches *E*.

src adr	dest adr	src port	dest port

Suppose the user in the right-hand network wants to run a web server on host *D*. Add an entry to the right-hand table that would allow remote connections to the web server. Assume host *B* connects to the web server at *D*. Add an entry to the left-hand NAT table for this connection. Show the address and port field for a typical packet leaving host *B*, the fields in the same packet as it passes through the public internet, and the packet that is delivered to *D*.

src adr	dest adr	src port	dest port

src adr	dest adr	src port	dest port

src adr	dest adr	src port	dest port

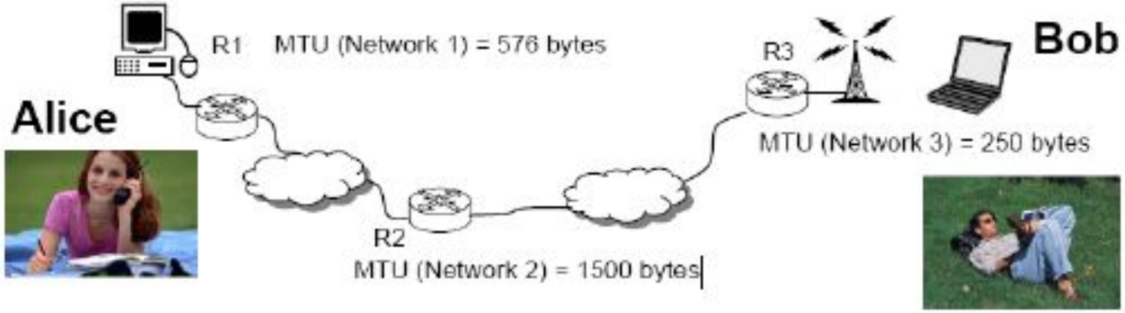
Questions

1. Alice is a musician and she wants to send one of her new song to Bob using a MP3 file. The MP3 file consists 1 million (1×106) bytes. Assume that TCP is used and the connection crosses through 3 networks as shown in the figure below. Assume link layer header is 26 bytes, IP header is 20 bytes, and TCP header is 20 bytes, UDP header is 8 bytes.

(a) How many packets/datagrams are generated in Alice's computer on the IP level? Explain clearly your result.

(b) How many fragments Bob receives on the IP level? Explain clearly your result.

(c) Show the first 4 and the last 5 IP fragments Bob receives and specify the values of all relevant parameters (number of bytes, ID, offset, Flag) in each fragment header. Fragment number need to be stated. Assume initial ID = 543


R1
MTU (Network 1) = 576 bytes
R3
Bob
Alice
MTU (Network 3) = 250 bytes
R2
MTU (Network 2) = 1500 bytes

- James Kurose, Keith Ross,” Computer Networking: A Top-Down Approach Featuring the Internet “, Addison Wesley
- Andrew S. Tanenbaum ,”Computer Networks “, Prentice-Hall Publishers
- Larry Peterson , Bruce Davie ,”Computer Networks a Systems Approach “, Morgan Kaufmann
- William Stallings ,”Data and Computer Communications”, Prentice Hall
- David Hanes, Gonzalo Salgueiro, Patrick Grossetete, Robert Barton, Jerome Henry, “IoT Fundamentals: Networking Technologies, Protocols, and Use Cases for the Internet of Things”, CISCO Press, 2017
- Rajkumar Buyya, and Amir Vahid Dastjerdi, eds. Internet of Things: Principles and paradigms. Elsevier, 2016.