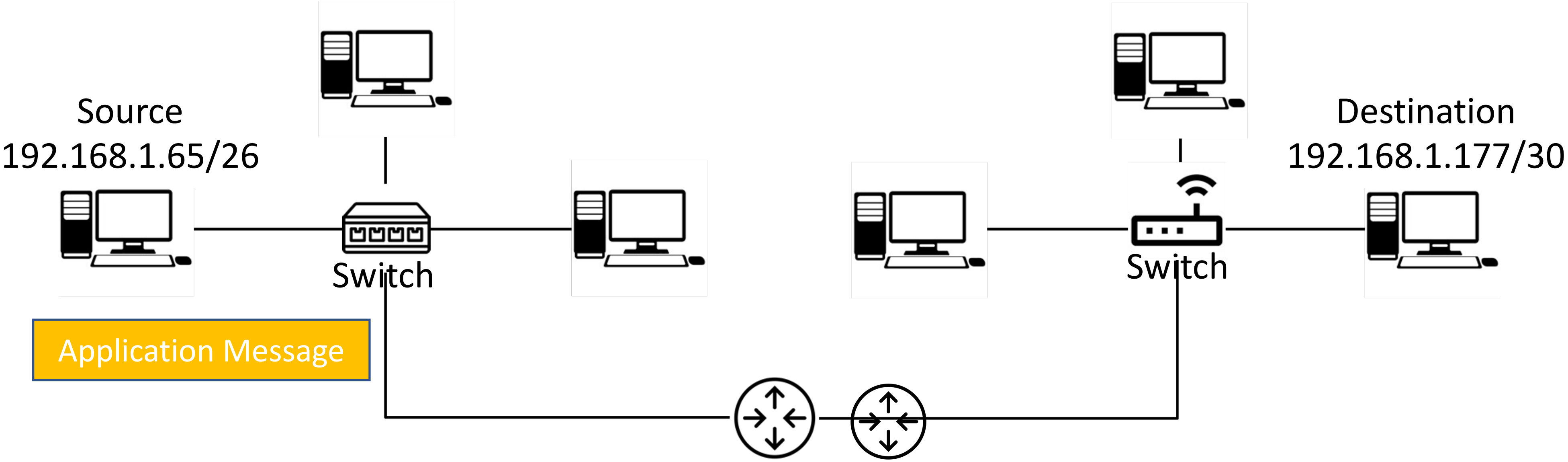
Q1. and Q2. UDP is more suitable for our application, because the it prioritises fluidity in gameplay and low latency. TCP It is designed to guarantee absolute delivery control (no loss of messages), connection and flow management, but with cost => more control, acknowledgment and notification messages, which reduce the usefull rate therefore reduce the latency, and a gives less fluidity when a message loss accure (during the retransmission of the lost message the process waits, which is not adapted to the real-time games: we prefer to lose a message on a movement for example and go to the next one (lag in common parlance) than get stuck until this movement is retransmitted, and resume from the last position.

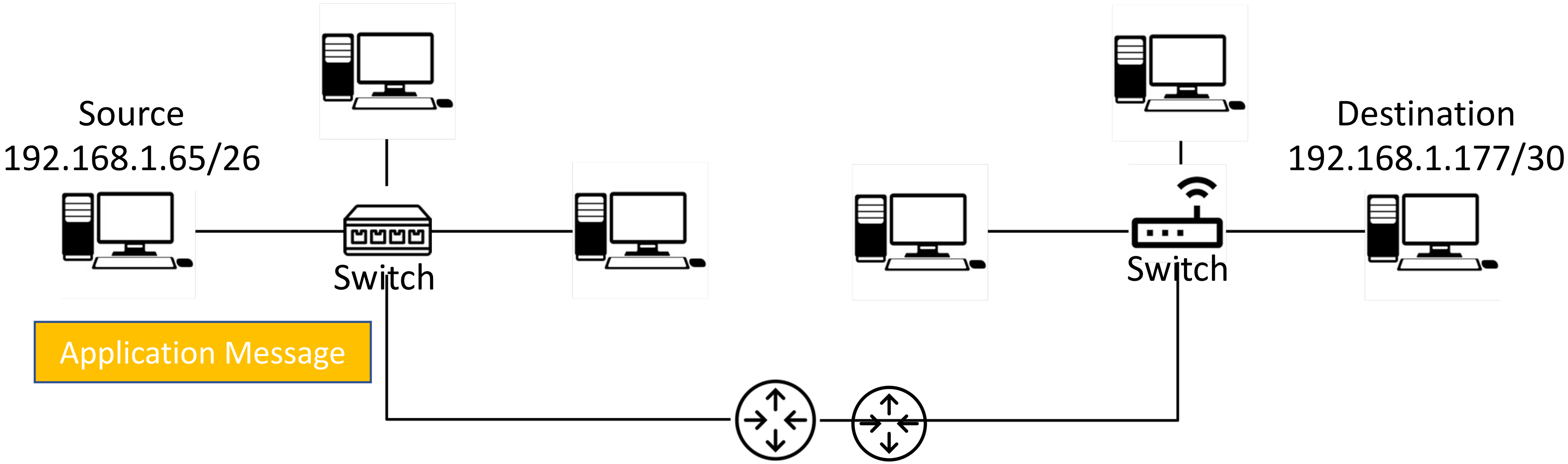
Q. 3


Source
192.168.1.65/26
Switch
Application Message
Switch
Destination
192.168.1.177/30

Q. 3

UDP

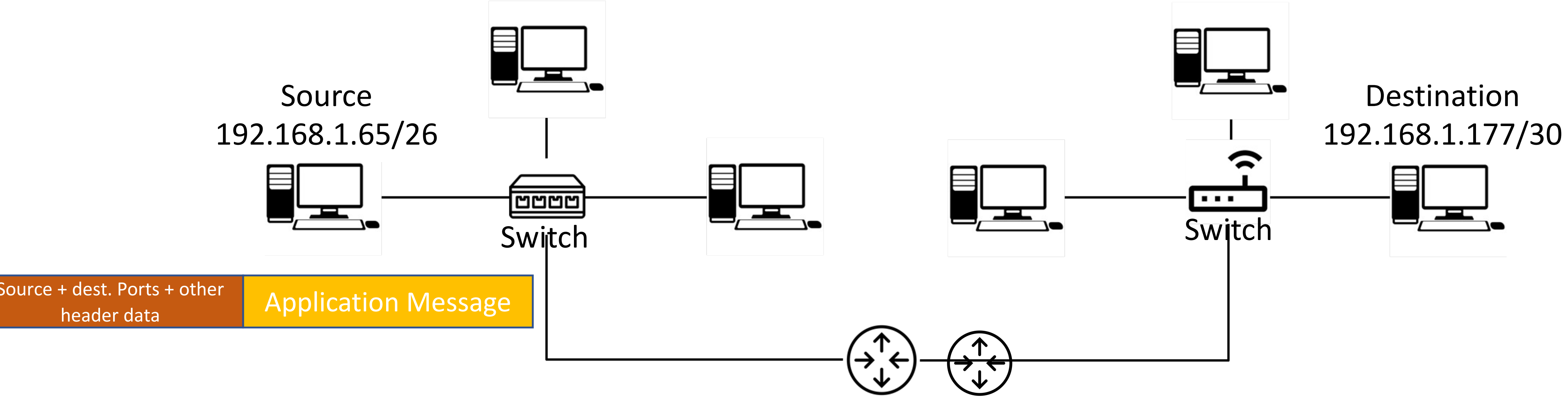
1. Receive application message from upper layer



Q. 3

UDP

1. Receive application message from upper layer
2. Add UDP header containing source port (given by the client application) and dest. Port (server application port)



Source
192.168.1.65/26

Switch

Switch

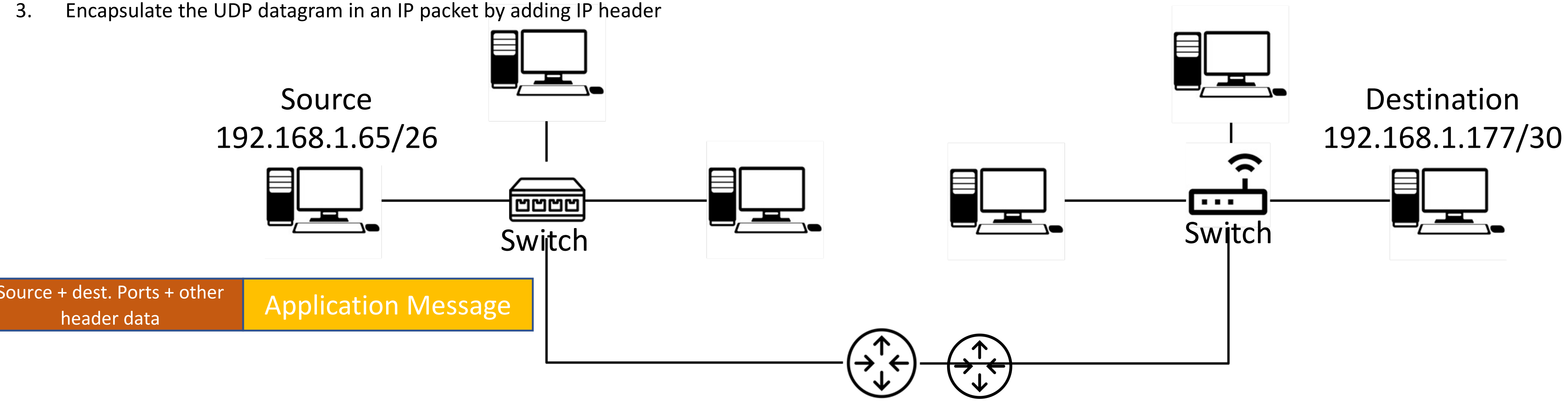
Destination
192.168.1.177/30

Source + dest. Ports + other header data	Application Message

Q. 3

UDP

1. Receive application message from upper layer
2. Add UDP header containing source port (given by the client application) and dest. Port (server application port)
3. Encapsulate the UDP datagram in an IP packet by adding IP header



Source
192.168.1.65/26

Destination
192.168.1.177/30

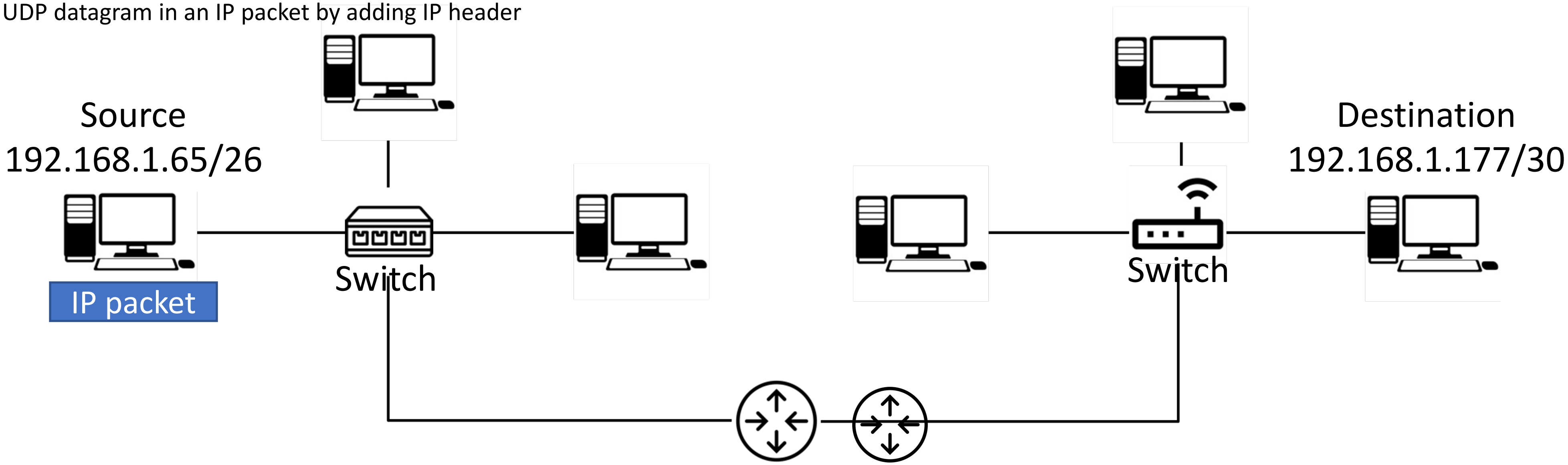
ource + dest. Ports + other header data

Application Message

Q. 3

UDP

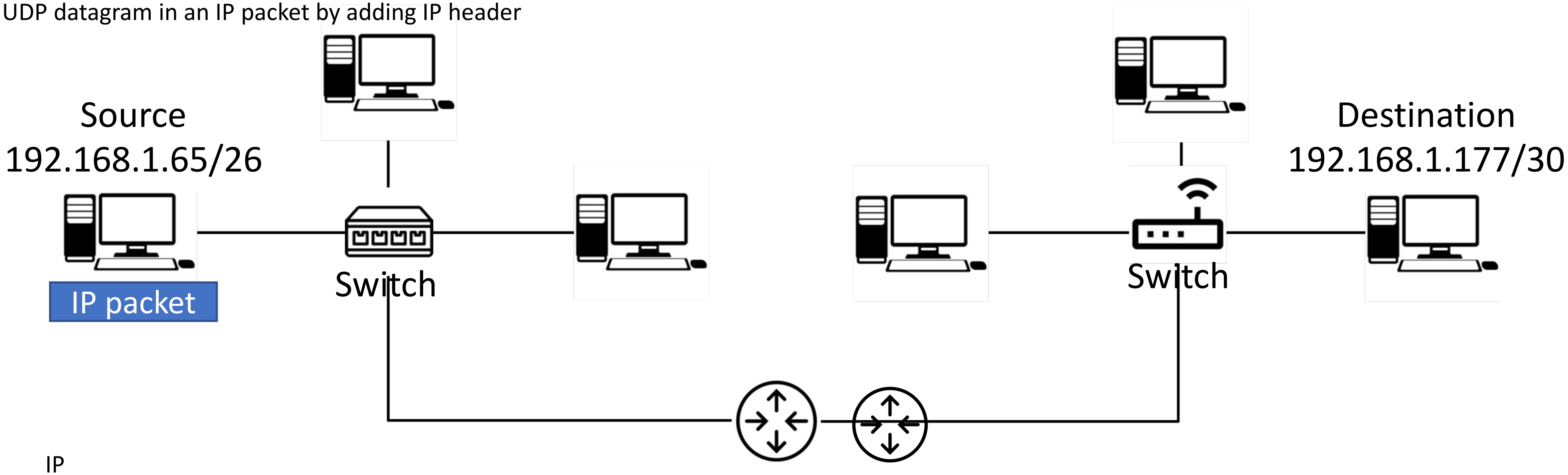
1. Receive application message from upper layer
2. Add UDP header containing source port (given by the client application) and dest. Port (server application port)
3. Encapsulate the UDP datagram in an IP packet by adding IP header



Q. 3

UDP

1. Receive application message from upper layer
2. Add UDP header containing source port (given by the client application) and dest. Port (server application port)
3. Encapsulate the UDP datagram in an IP packet by adding IP header



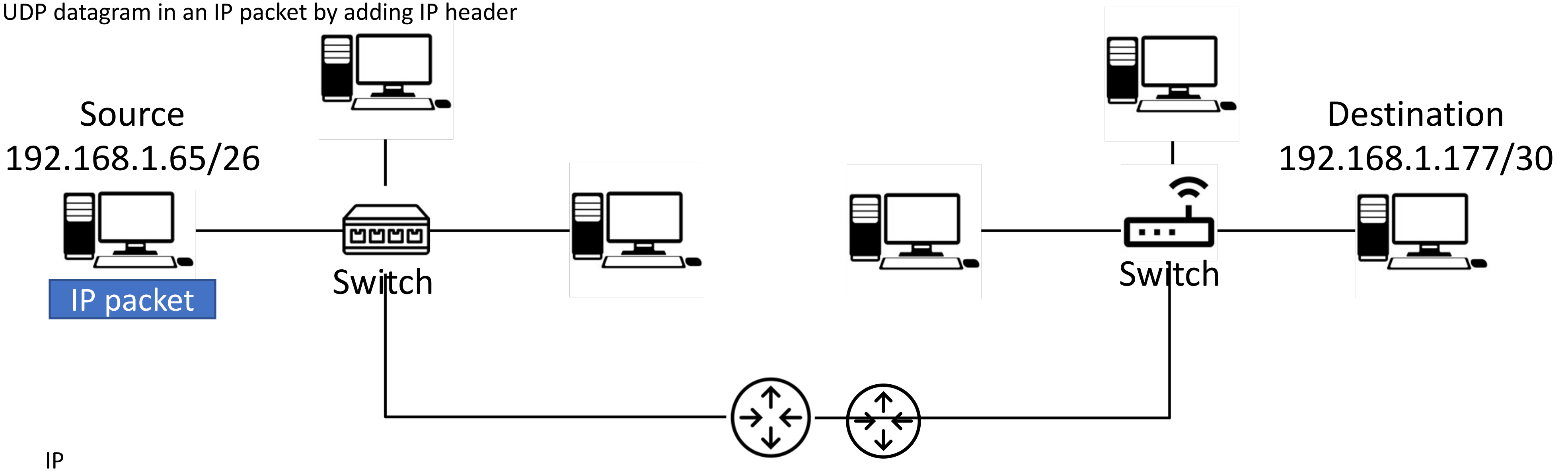
IP

1. Mask /26 on 192.168.1.177 = 192.168.1.128

Q. 3

UDP

1. Receive application message from upper layer
2. Add UDP header containing source port (given by the client application) and dest. Port (server application port)
3. Encapsulate the UDP datagram in an IP packet by adding IP header



IP

1. Mask /26 on 192.168.1.177 = 192.168.1.128

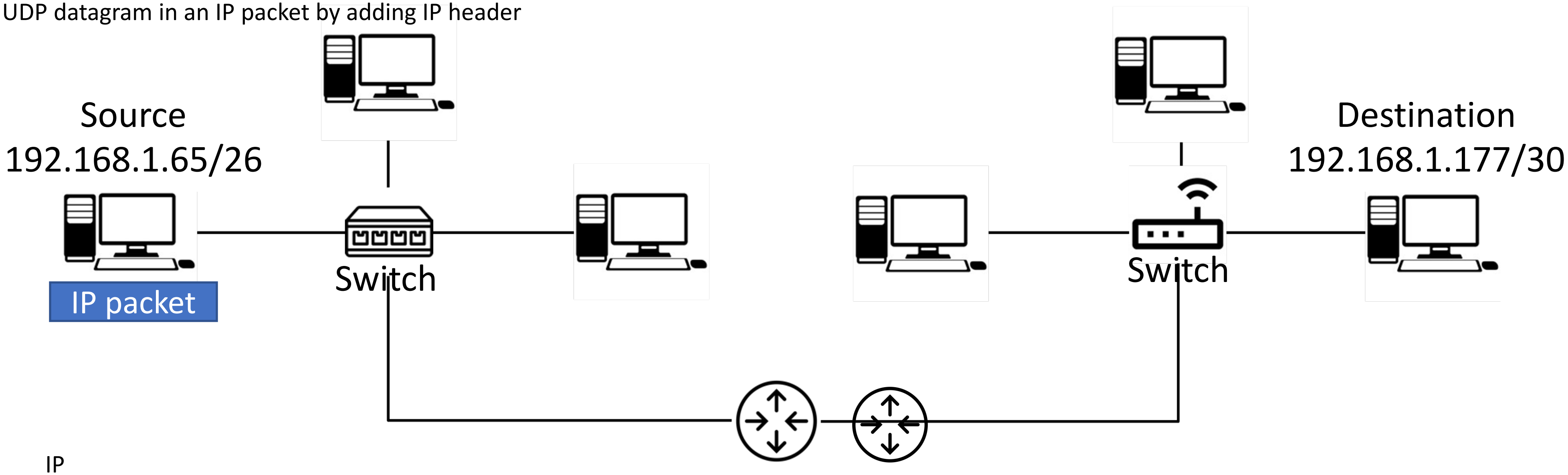
Source
192.168.1.65
↓
/26
↓
192.168.1.64 ≠

Destination
192.168.1.177
↓
/26
↓
192.168.1.128

Q. 3

UDP

1. Receive application message from upper layer
2. Add UDP header containing source port (given by the client application) and dest. Port (server application port)
3. Encapsulate the UDP datagram in an IP packet by adding IP header



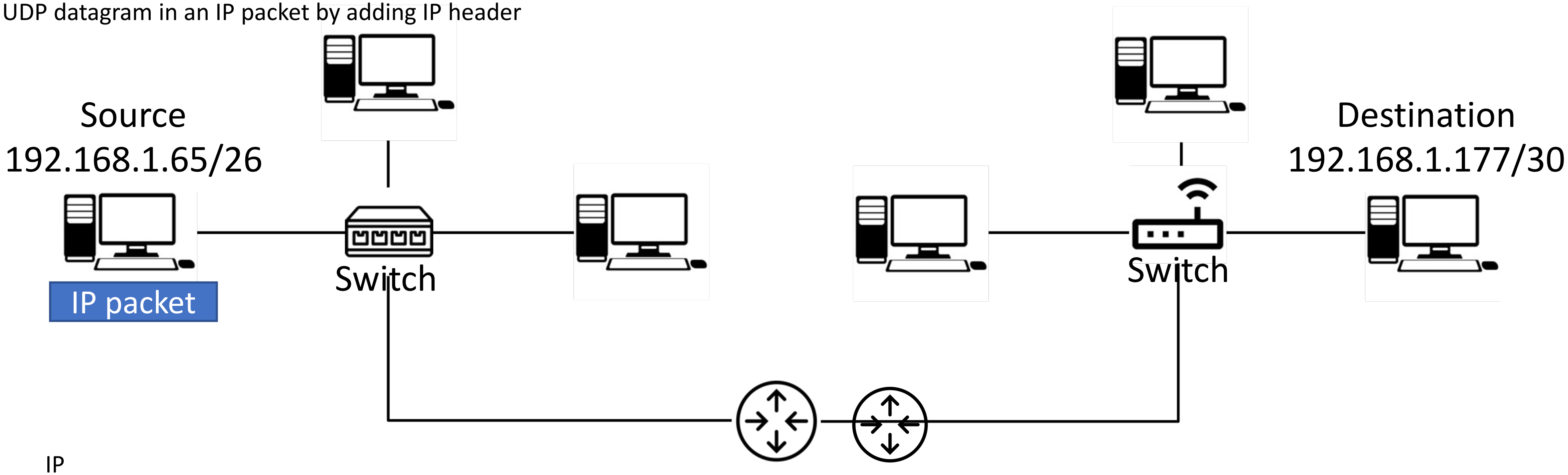
IP

1. Mask /26 on 192.168.1.177 = 192.168.1.128
2. 192.168.1.128 ≠ 192.168.1.64 (current) => destination on another network

Q. 3

UDP

1. Receive application message from upper layer
2. Add UDP header containing source port (given by the client application) and dest. Port (server application port)
3. Encapsulate the UDP datagram in an IP packet by adding IP header



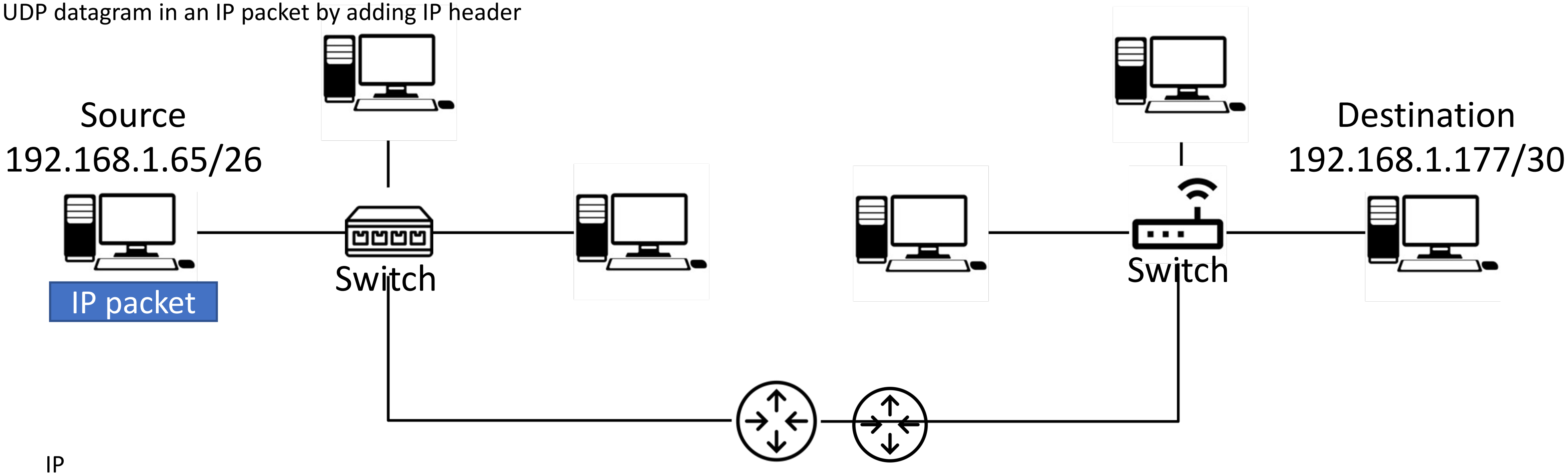
IP

1. Mask /26 on 192.168.1.177 = 192.168.1.128
2. 192.168.1.128 ≠ 192.168.1.64 (current) => destination on another network
3. Check the routing table to get the router IP address

Q. 3

UDP

1. Receive application message from upper layer
2. Add UDP header containing source port (given by the client application) and dest. Port (server application port)
3. Encapsulate the UDP datagram in an IP packet by adding IP header



IP

1. Mask /26 on 192.168.1.177 = 192.168.1.128
2. 192.168.1.128 ≠ 1192.168.1.64 (current) => destination on another network
3. Check the routing table to get the router IP address

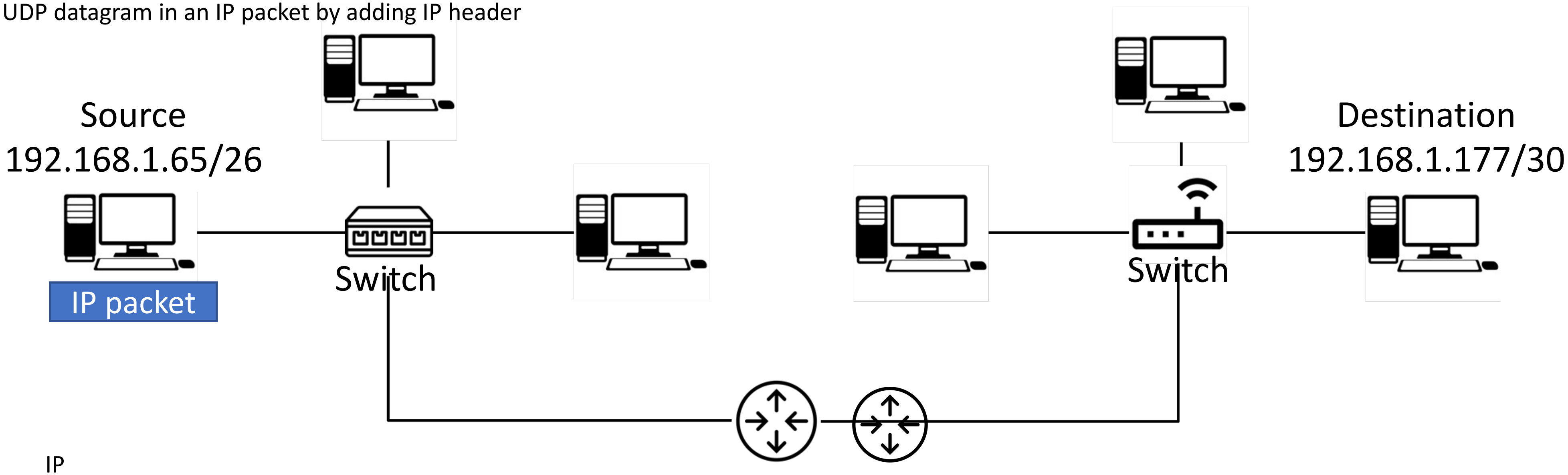
ARP

1. Check the router MAC address in the ARP table

Q. 3

UDP

1. Receive application message from upper layer
2. Add UDP header containing source port (given by the client application) and dest. Port (server application port)
3. Encapsulate the UDP datagram in an IP packet by adding IP header



IP

1. Mask /26 on 192.168.1.177 = 192.168.1.128
2. 192.168.1.128 ≠ 1192.168.1.64 (current) => destination on another network
3. Check the routing table to get the router IP address

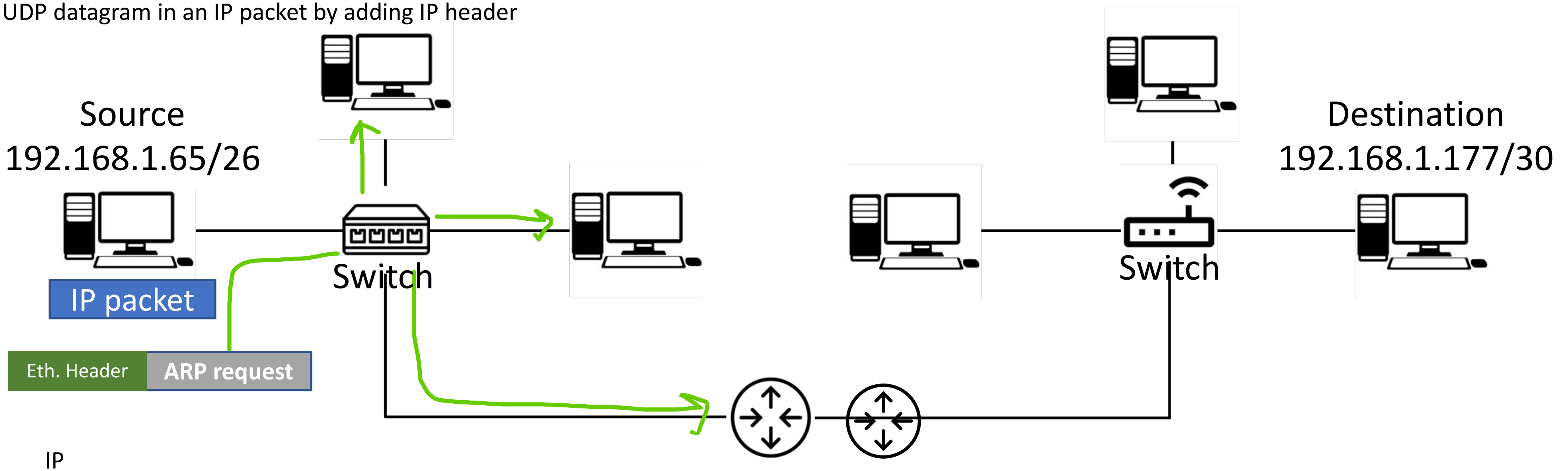
ARP

1. Check the router MAC address in the ARP table
2. If no MAC in ARP table : send ARP request

Q. 3

UDP

1. Receive application message from upper layer
2. Add UDP header containing source port (given by the client application) and dest. Port (server application port)
3. Encapsulate the UDP datagram in an IP packet by adding IP header



IP

1. Mask /26 on 192.168.1.177 = 192.168.1.128
2. 192.168.1.128 ≠ 1192.168.1.64 (current) => destination on another network
3. Check the routing table to get the router IP address

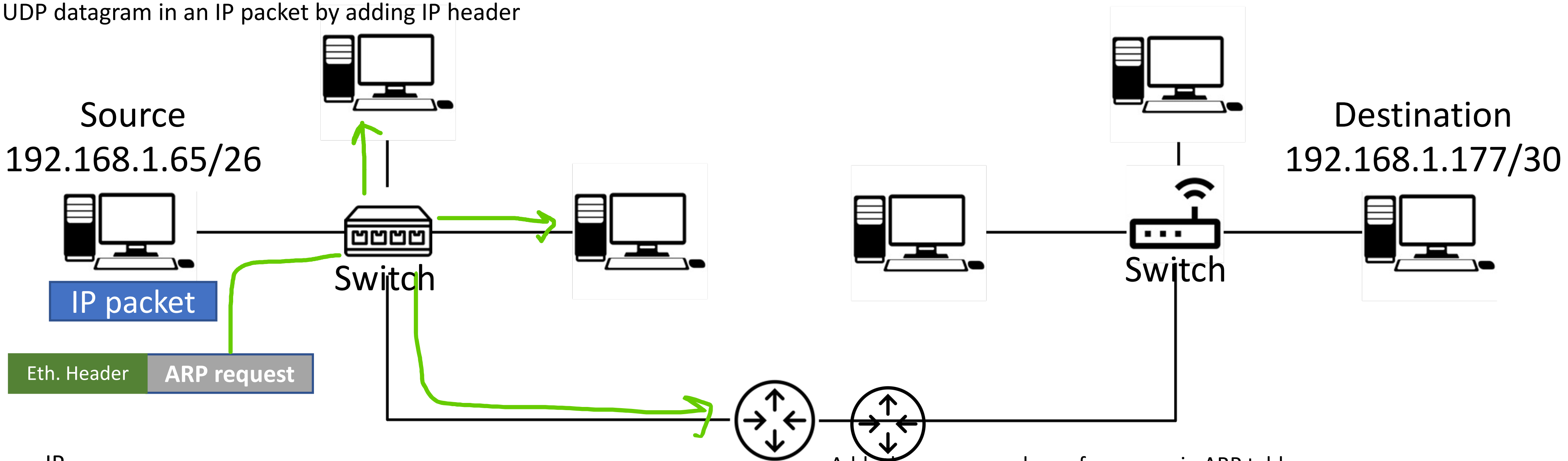
ARP

1. Check the router MAC address in the ARP table
2. If no MAC in ARP table : send ARP request

Q. 3

UDP

1. Receive application message from upper layer
2. Add UDP header containing source port (given by the client application) and dest. Port (server application port)
3. Encapsulate the UDP datagram in an IP packet by adding IP header



Source
192.168.1.65/26

IP packet

Eth. Header | ARP request

Switch

Destination
192.168.1.177/30

Switch

Adds the correspondance for source in ARP table

IP

1. Mask /26 on 192.168.1.177 = 192.168.1.128
2. 192.168.1.128 ≠ 1192.168.1.64 (current) => destination on another network
3. Check the routing table to get the router IP address

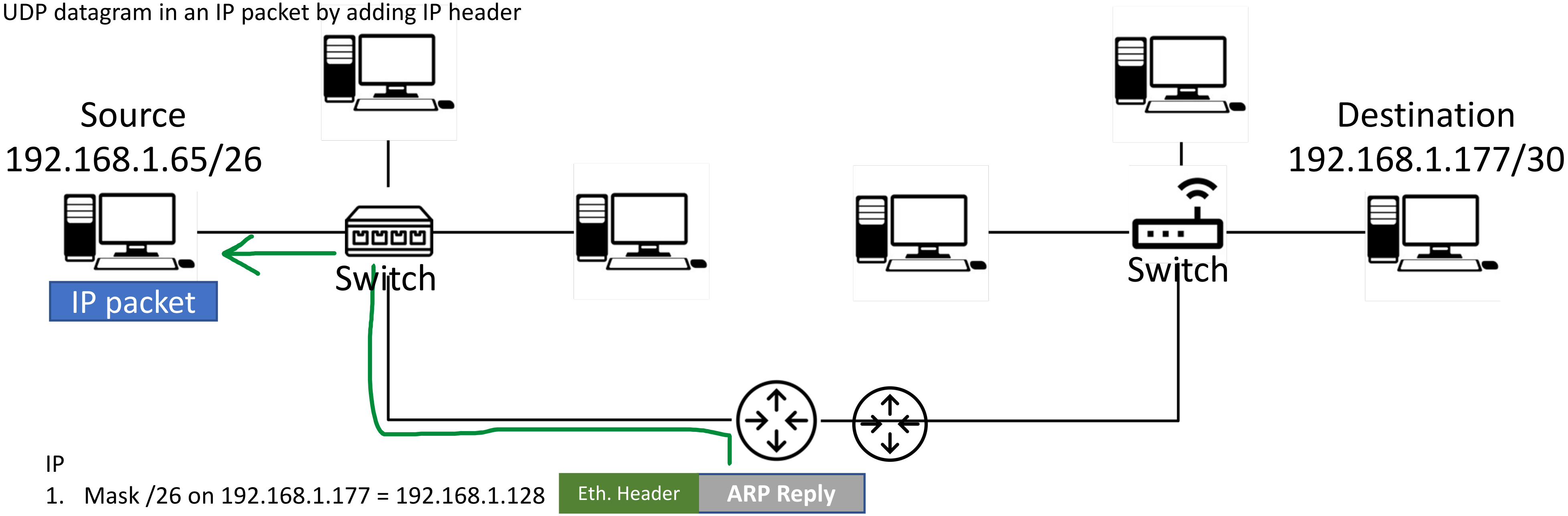
ARP

1. Check the router MAC address in the ARP table
2. If no MAC in ARP table : send ARP request

Q. 3

UDP

1. Receive application message from upper layer
2. Add UDP header containing source port (given by the client application) and dest. Port (server application port)
3. Encapsulate the UDP datagram in an IP packet by adding IP header



IP

1. Mask /26 on 192.168.1.177 = 192.168.1.128
2. 192.168.1.128 ≠ 1192.168.1.64 (current) => destination on another network
3. Check the routing table to get the router IP address

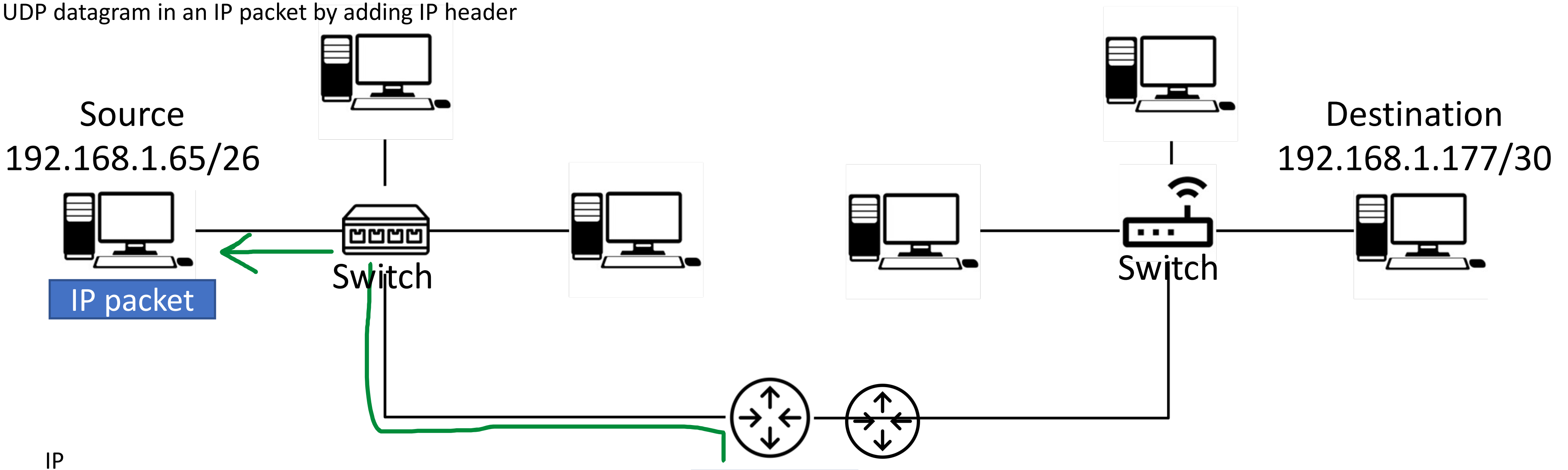
ARP

1. Check the router MAC address in the ARP table
2. If no MAC in ARP table : send ARP request : who has the router IP address

Q. 3

UDP

1. Receive application message from upper layer
2. Add UDP header containing source port (given by the client application) and dest. Port (server application port)
3. Encapsulate the UDP datagram in an IP packet by adding IP header



IP

1. Mask /26 on 192.168.1.177 = 192.168.1.128
2. 192.168.1.128 ≠ 1192.168.1.64 (current) => destination on another network
3. Check the routing table to get the router IP address

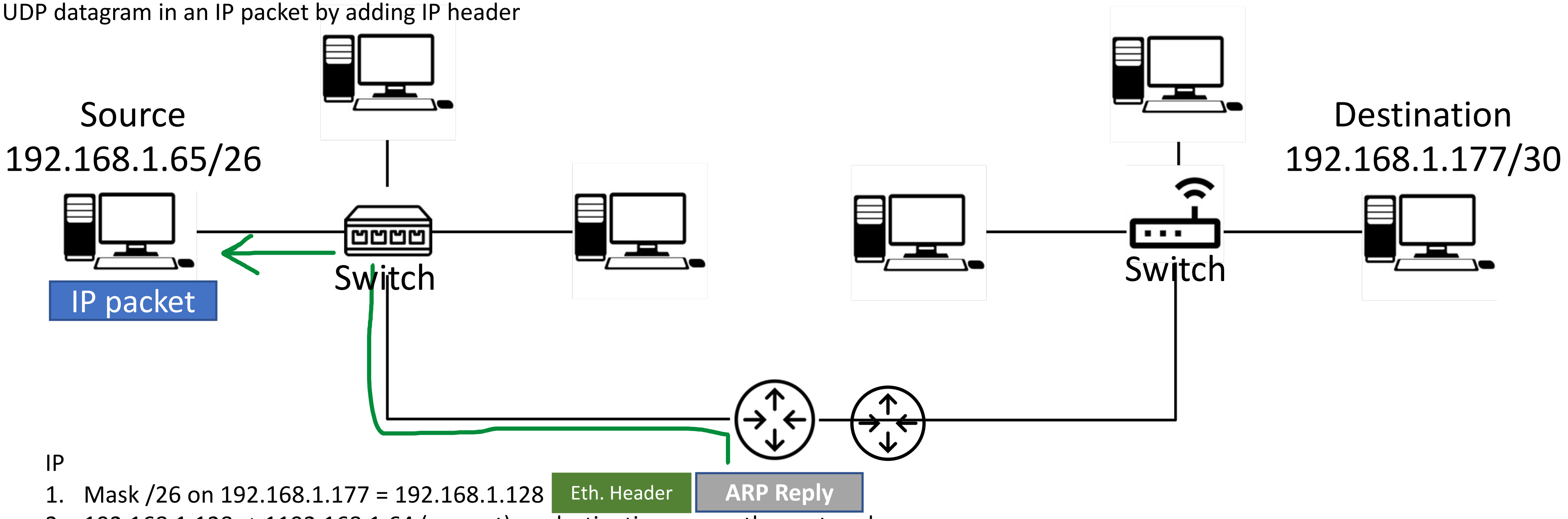
ARP

1. Check the router MAC address in the ARP table
2. If no MAC in ARP table : send ARP request
3. Adds the correspondance MAC-IP of the router in ARP table

Q. 3

UDP

1. Receive application message from upper layer
2. Add UDP header containing source port (given by the client application) and dest. Port (server application port)
3. Encapsulate the UDP datagram in an IP packet by adding IP header



IP

1. Mask /26 on 192.168.1.177 = 192.168.1.128
2. 192.168.1.128 ≠ 1192.168.1.64 (current) => destination on another network
3. Check the routing table to get the router IP address

ARP

1. Check the router MAC address in the ARP table
2. If no MAC in ARP table : send ARP request
3. Adds the correspondance MAC-IP of the router in ARP table

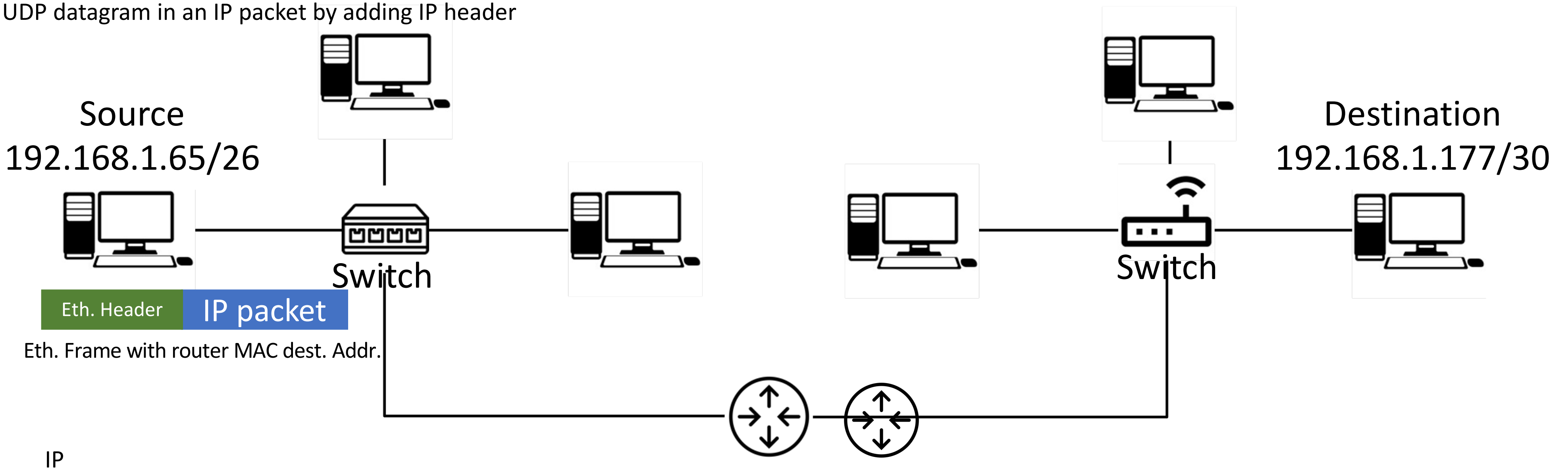
MAC

1. Create an Ethernet frame with router MAC address, put IP packet in it and send it

Q. 3

UDP

1. Receive application message from upper layer
2. Add UDP header containing source port (given by the client application) and dest. Port (server application port)
3. Encapsulate the UDP datagram in an IP packet by adding IP header



IP

1. Mask /26 on 192.168.1.177 = 192.168.1.128
2. 192.168.1.128 ≠ 1192.168.1.64 (current) => destination on another network
3. Check the routing table to get the router IP address

ARP

1. Check the router MAC address in the ARP table
2. If no MAC in ARP table : send ARP request
3. Adds the correspondance MAC-IP of the router in ARP table

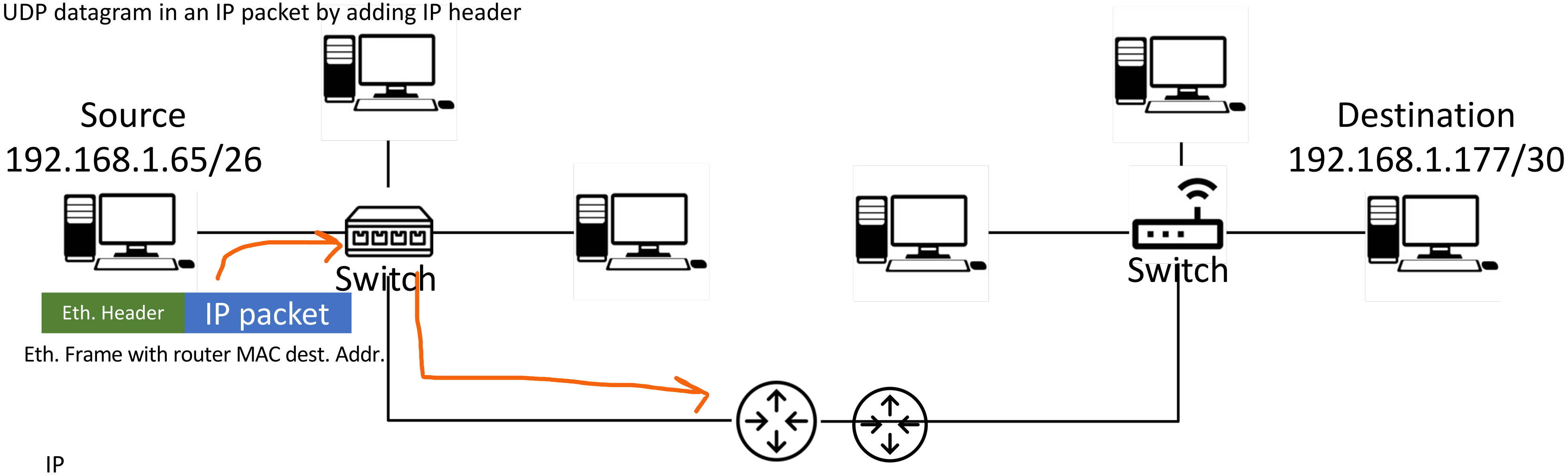
MAC

1. Create an Ethernet frame with router MAC address, put IP packet in it and send it

Q. 3

UDP

1. Receive application message from upper layer
2. Add UDP header containing source port (given by the client application) and dest. Port (server application port)
3. Encapsulate the UDP datagram in an IP packet by adding IP header



IP

1. Mask /26 on 192.168.1.177 = 192.168.1.128
2. 192.168.1.128 ≠ 1192.168.1.64 (current) => destination on another network
3. Check the routing table to get the router IP address

ARP

1. Check the router MAC address in the ARP table
2. If no MAC in ARP table : send ARP request
3. Adds the correspondance MAC-IP of the router in ARP table

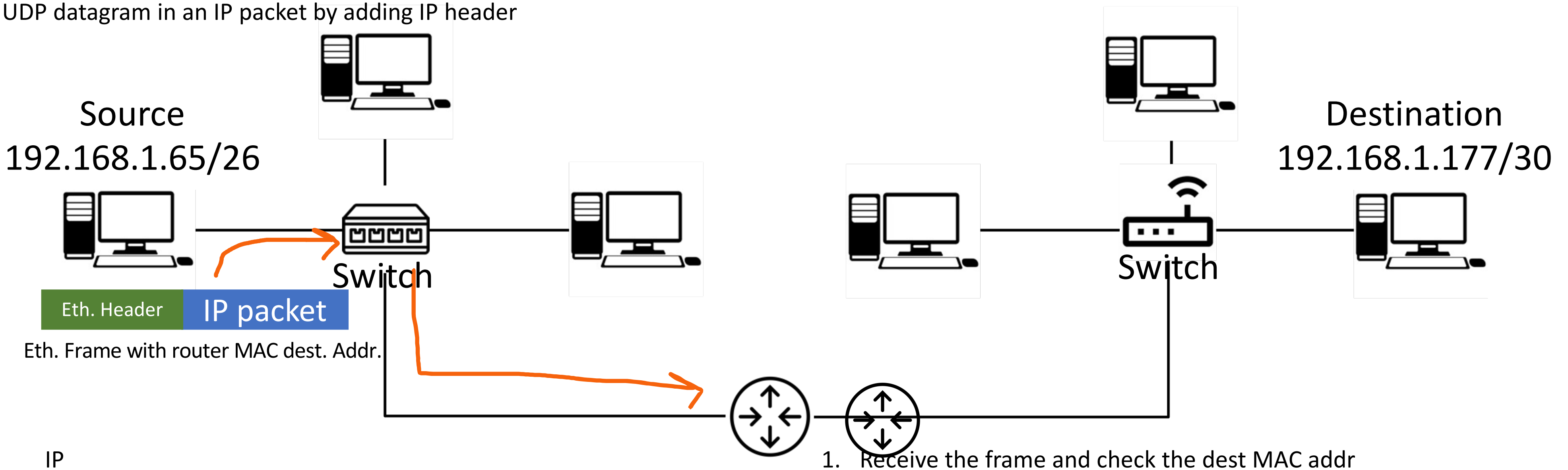
MAC

1. Create an Ethernet frame with router MAC address, put IP packet in it and send it

Q. 3

UDP

1. Receive application message from upper layer
2. Add UDP header containing source port (given by the client application) and dest. Port (server application port)
3. Encapsulate the UDP datagram in an IP packet by adding IP header



1. Receive the frame and check the dest MAC addr

IP

1. Mask /26 on 192.168.1.177 = 192.168.1.128
2. 192.168.1.128 ≠ 1192.168.1.64 (current) => destination on another network
3. Check the routing table to get the router IP address

ARP

1. Check the router MAC address in the ARP table
2. If no MAC in ARP table : send ARP request
3. Adds the correspondance MAC-IP of the router in ARP table

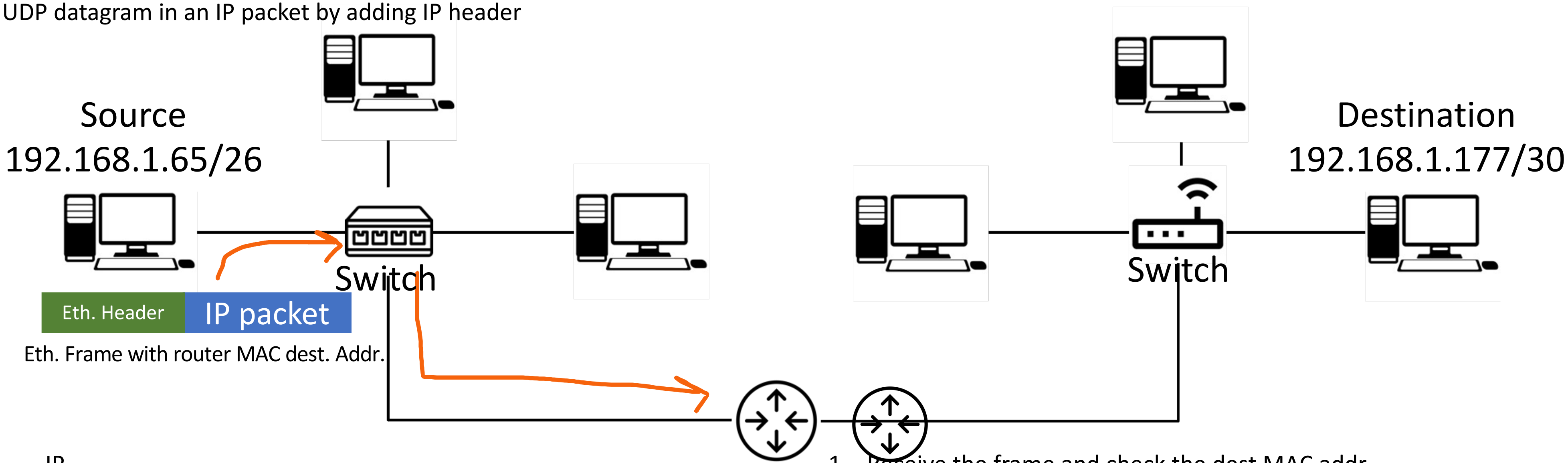
MAC

1. Create an Ethernet frame with router MAC address, put IP packet in it and send it

Q. 3

UDP

1. Receive application message from upper layer
2. Add UDP header containing source port (given by the client application) and dest. Port (server application port)
3. Encapsulate the UDP datagram in an IP packet by adding IP header



Source
192.168.1.65/26

Destination
192.168.1.177/30

Switch

Switch

Eth. Header | IP packet

Eth. Frame with router MAC dest. Addr.

1. Receive the frame and check the dest MAC addr
2. Extract the IP packet

IP

1. Mask /26 on 192.168.1.177 = 192.168.1.128
2. 192.168.1.128 ≠ 1192.168.1.64 (current) => destination on another network
3. Check the routing table to get the router IP address

ARP

1. Check the router MAC address in the ARP table
2. If no MAC in ARP table : send ARP request
3. Adds the correspondance MAC-IP of the router in ARP table

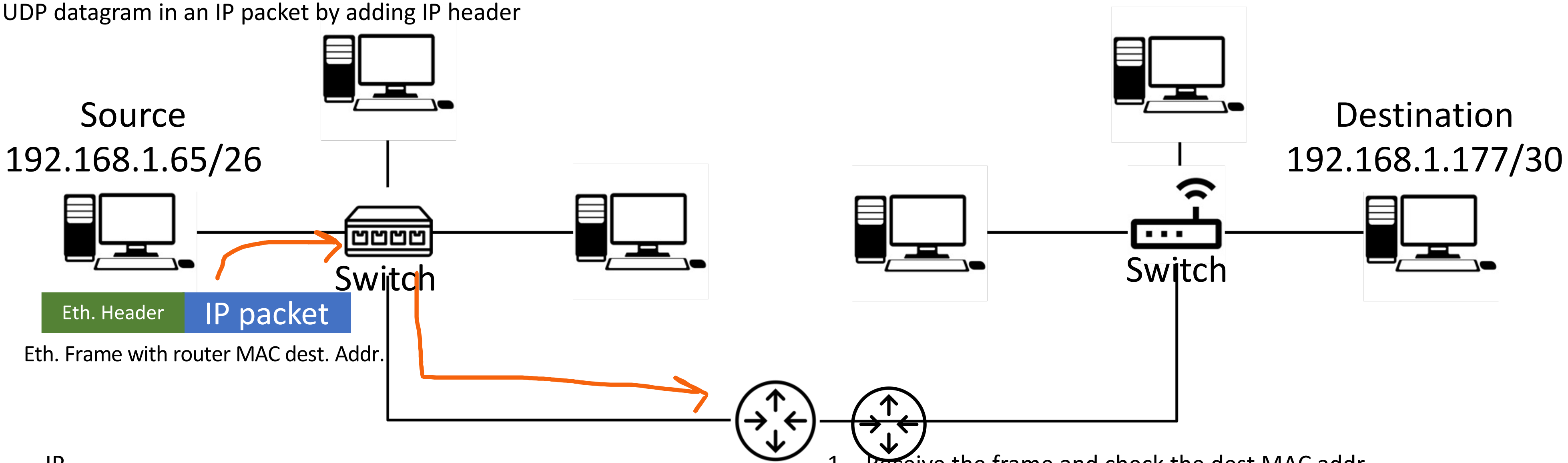
MAC

1. Create an Ethernet frame with router MAC address, put IP packet in it and send it

Q. 3

UDP

1. Receive application message from upper layer
2. Add UDP header containing source port (given by the client application) and dest. Port (server application port)
3. Encapsulate the UDP datagram in an IP packet by adding IP header



Eth. Frame with router MAC dest. Addr.

IP

1. Mask /26 on 192.168.1.177 = 192.168.1.128
2. 192.168.1.128 ≠ 1192.168.1.64 (current) => destination on another network
3. Check the routing table to get the router IP address

1. Receive the frame and check the dest MAC addr
2. Extract the IP packet
3. Check the routing table to find the next dest. (destination IP addr.)

ARP

1. Check the router MAC address in the ARP table
2. If no MAC in ARP table : send ARP request
3. Adds the correspondance MAC-IP of the router in ARP table

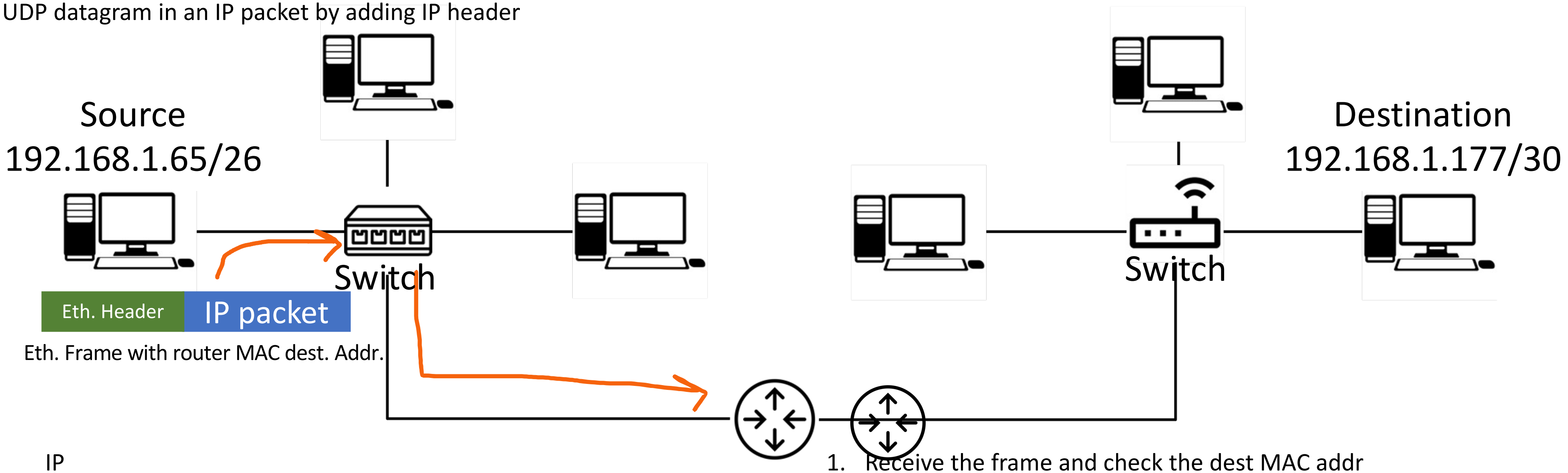
MAC

1. Create an Ethernet frame with router MAC address, put IP packet in it and send it

Q. 3

UDP

1. Receive application message from upper layer
2. Add UDP header containing source port (given by the client application) and dest. Port (server application port)
3. Encapsulate the UDP datagram in an IP packet by adding IP header



IP

1. Mask /26 on 192.168.1.177 = 192.168.1.128
2. 192.168.1.128 ≠ 1192.168.1.64 (current) => destination on another network
3. Check the routing table to get the router IP address

ARP

1. Check the router MAC address in the ARP table
2. If no MAC in ARP table : send ARP request
3. Adds the correspondance MAC-IP of the router in ARP table

MAC

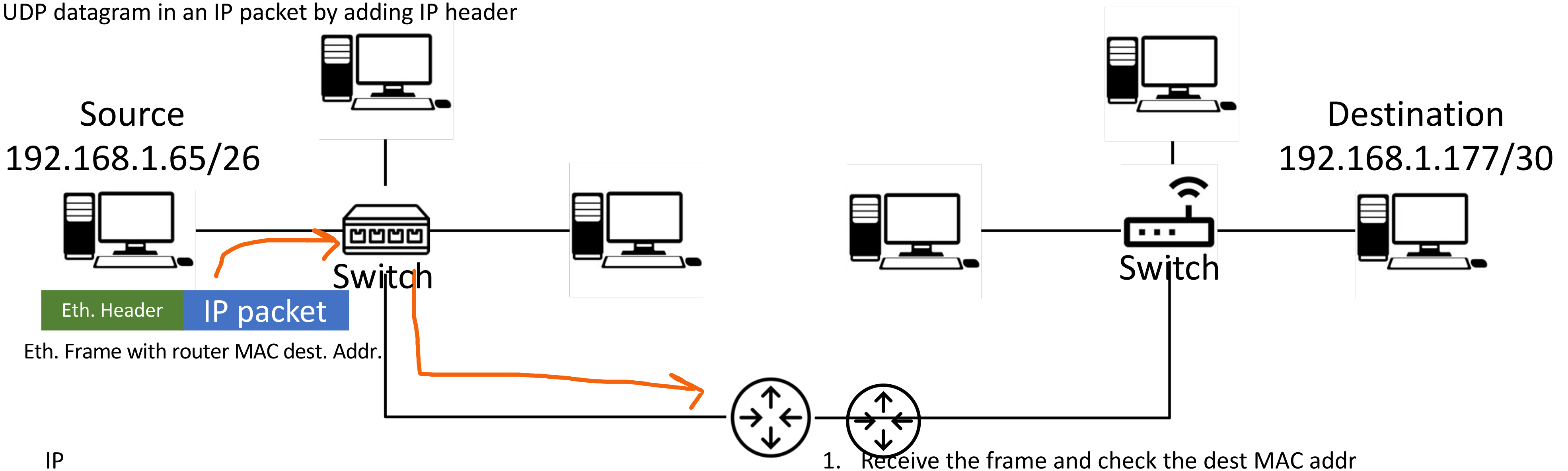
1. Create an Ethernet frame with router MAC address, put IP packet in it and send it

1. Receive the frame and check the dest MAC addr
2. Extract the IP packet
3. Check the routing table to find the next dest. (destination IP addr.)
4. Check the MTU -> fragmentation

Q. 3

UDP

1. Receive application message from upper layer
2. Add UDP header containing source port (given by the client application) and dest. Port (server application port)
3. Encapsulate the UDP datagram in an IP packet by adding IP header



IP

1. Mask /26 on 192.168.1.177 = 192.168.1.128
2. 192.168.1.128 ≠ 1192.168.1.64 (current) => destination on another network
3. Check the routing table to get the router IP address

ARP

1. Check the router MAC address in the ARP table
2. If no MAC in ARP table : send ARP request
3. Adds the correspondance MAC-IP of the router in ARP table

MAC

1. Create an Ethernet frame with router MAC address, put IP packet in it and send it

1. Receive the frame and check the dest MAC addr
2. Extract the IP packet
3. Check the routing table to find the next dest. (destination IP addr.)
4. Check the MTU -> fragmentation

ARP

Same process

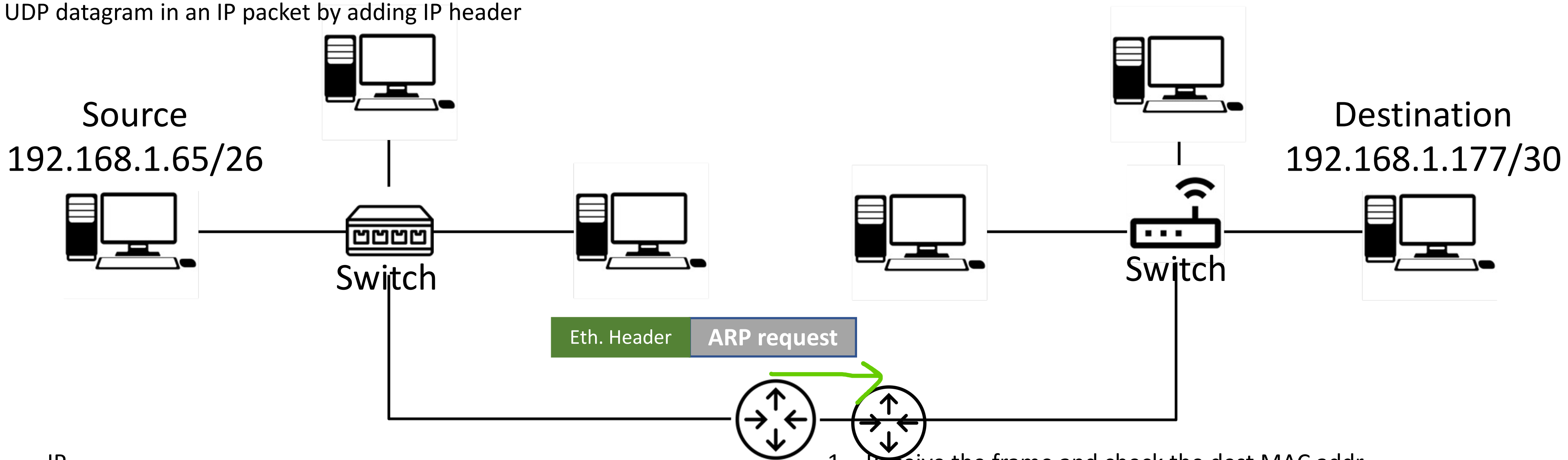
MAC

Same process

Q. 3

UDP

1. Receive application message from upper layer
2. Add UDP header containing source port (given by the client application) and dest. Port (server application port)
3. Encapsulate the UDP datagram in an IP packet by adding IP header



IP

1. Mask /26 on 192.168.1.177 = 192.168.1.128
2. 192.168.1.128 ≠ 1192.168.1.64 (current) => destination on another network
3. Check the routing table to get the router IP address

ARP

1. Check the router MAC address in the ARP table
2. If no MAC in ARP table : send ARP request
3. Adds the correspondance MAC-IP of the router in ARP table

MAC

1. Create an Ethernet frame with router MAC address, put IP packet in it and send it

1. Receive the frame and check the dest MAC addr
2. Extract the IP packet
3. Check the routing table to find the next dest. (destination IP addr.)
4. Check the MTU -> fragmentation

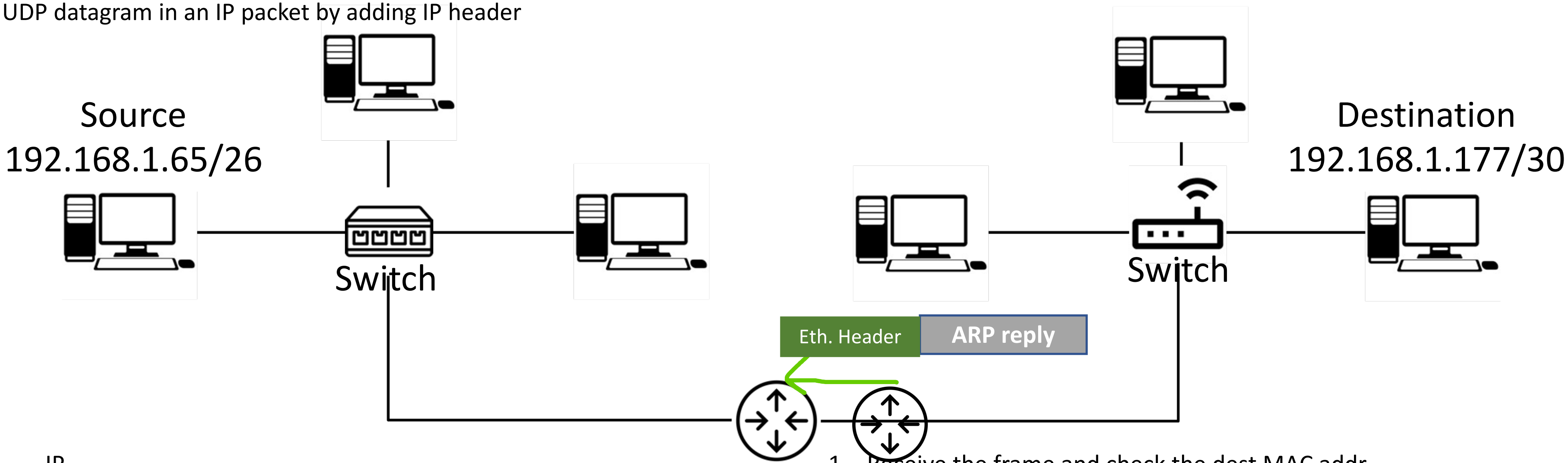
ARP
Same process

MAC
Same process

Q. 3

UDP

1. Receive application message from upper layer
2. Add UDP header containing source port (given by the client application) and dest. Port (server application port)
3. Encapsulate the UDP datagram in an IP packet by adding IP header



IP

1. Mask /26 on 192.168.1.177 = 192.168.1.128
2. 192.168.1.128 ≠ 1192.168.1.64 (current) => destination on another network
3. Check the routing table to get the router IP address

ARP

1. Check the router MAC address in the ARP table
2. If no MAC in ARP table : send ARP request
3. Adds the correspondance MAC-IP of the router in ARP table

MAC

1. Create an Ethernet frame with router MAC address, put IP packet in it and send it

1. Receive the frame and check the dest MAC addr
2. Extract the IP packet
3. Check the routing table to find the next dest. (destination IP addr.)
4. Check the MTU -> fragmentation

ARP

Same process

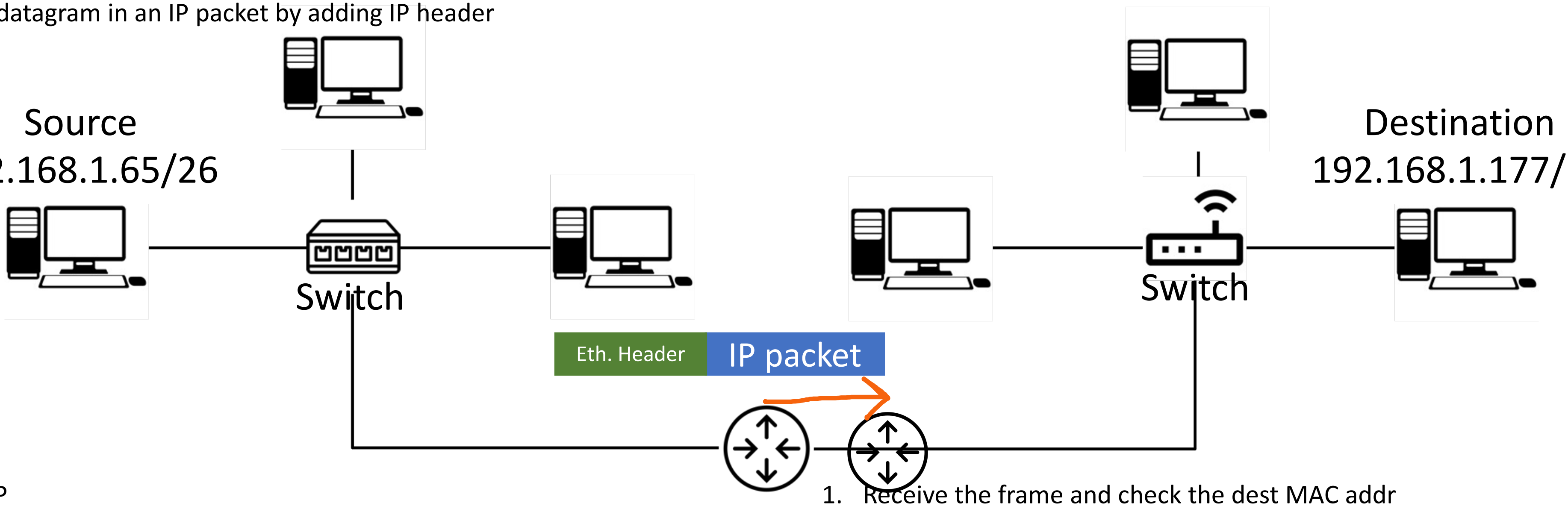
MAC

Same process

Q. 3

UDP

1. Receive application message from upper layer
2. Add UDP header containing source port (given by the client application) and dest. Port (server application port)
3. Encapsulate the UDP datagram in an IP packet by adding IP header



IP

1. Mask /26 on 192.168.1.177 = 192.168.1.128
2. 192.168.1.128 ≠ 1192.168.1.64 (current) => destination on another network
3. Check the routing table to get the router IP address

ARP

1. Check the router MAC address in the ARP table
2. If no MAC in ARP table : send ARP request
3. Adds the correspondance MAC-IP of the router in ARP table

MAC

1. Create an Ethernet frame with router MAC address, put IP packet in it and send it

1. Receive the frame and check the dest MAC addr
2. Extract the IP packet
3. Check the routing table to find the next dest. (destination IP addr.)
4. Check the MTU -> fragmentation

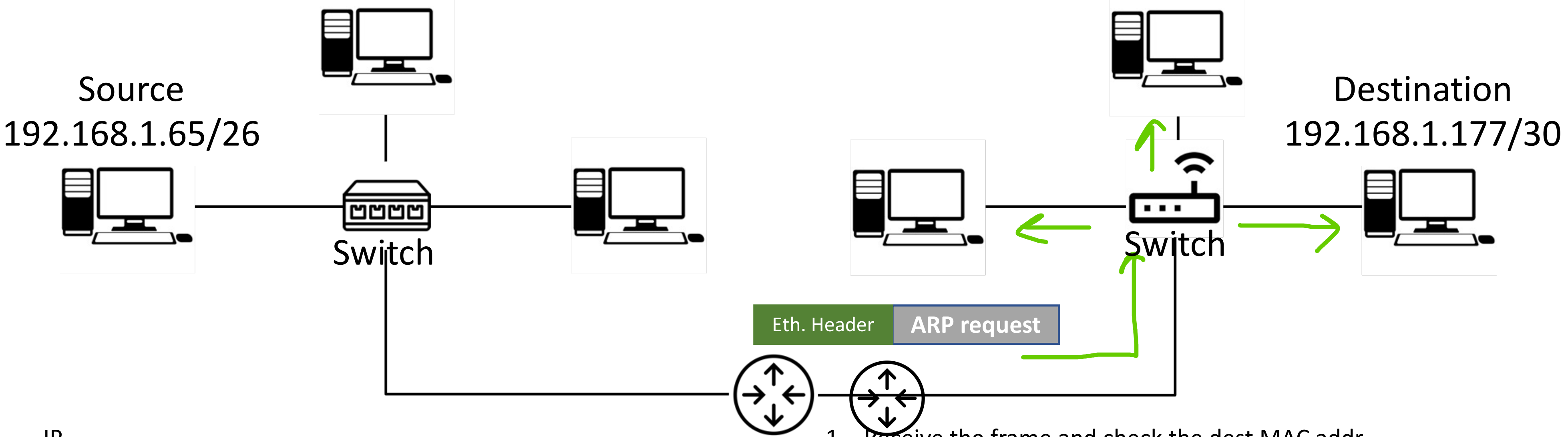
ARP
Same process

MAC
Same process

Q. 3

UDP

1. Receive application message from upper layer
2. Add UDP header containing source port (given by the client application) and dest. Port (server application port)
3. Encapsulate the UDP datagram in an IP packet by adding IP header



IP

1. Mask /26 on 192.168.1.177 = 192.168.1.128
2. 192.168.1.128 ≠ 1192.168.1.64 (current) => destination on another network
3. Check the routing table to get the router IP address

ARP

1. Check the router MAC address in the ARP table
2. If no MAC in ARP table : send ARP request
3. Adds the correspondance MAC-IP of the router in ARP table

MAC

1. Create an Ethernet frame with router MAC address, put IP packet in it and send it

1. Receive the frame and check the dest MAC addr
2. Extract the IP packet
3. Check the routing table to find the next dest. (destination IP addr.)
4. Check the MTU -> fragmentation

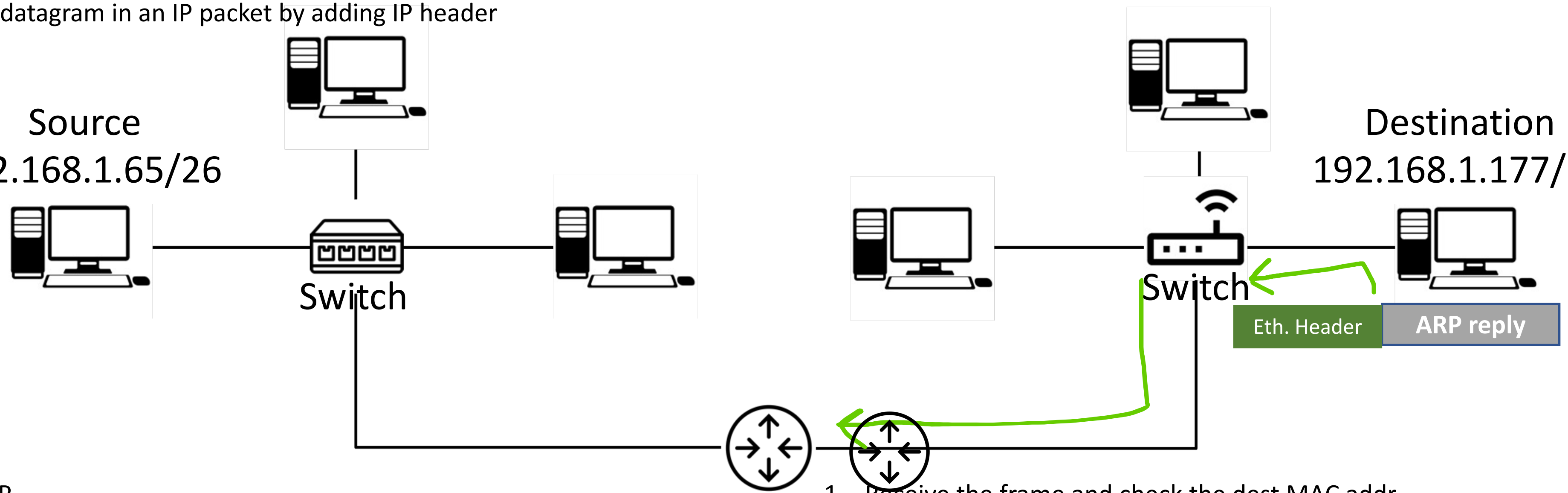
ARP
Same process

MAC
Same process

Q. 3

UDP

1. Receive application message from upper layer
2. Add UDP header containing source port (given by the client application) and dest. Port (server application port)
3. Encapsulate the UDP datagram in an IP packet by adding IP header



IP

1. Mask /26 on 192.168.1.177 = 192.168.1.128
2. 192.168.1.128 ≠ 1192.168.1.64 (current) => destination on another network
3. Check the routing table to get the router IP address

ARP

1. Check the router MAC address in the ARP table
2. If no MAC in ARP table : send ARP request
3. Adds the correspondance MAC-IP of the router in ARP table

MAC

1. Create an Ethernet frame with router MAC address, put IP packet in it and send it

1. Receive the frame and check the dest MAC addr
2. Extract the IP packet
3. Check the routing table to find the next dest. (destination IP addr.)
4. Check the MTU -> fragmentation

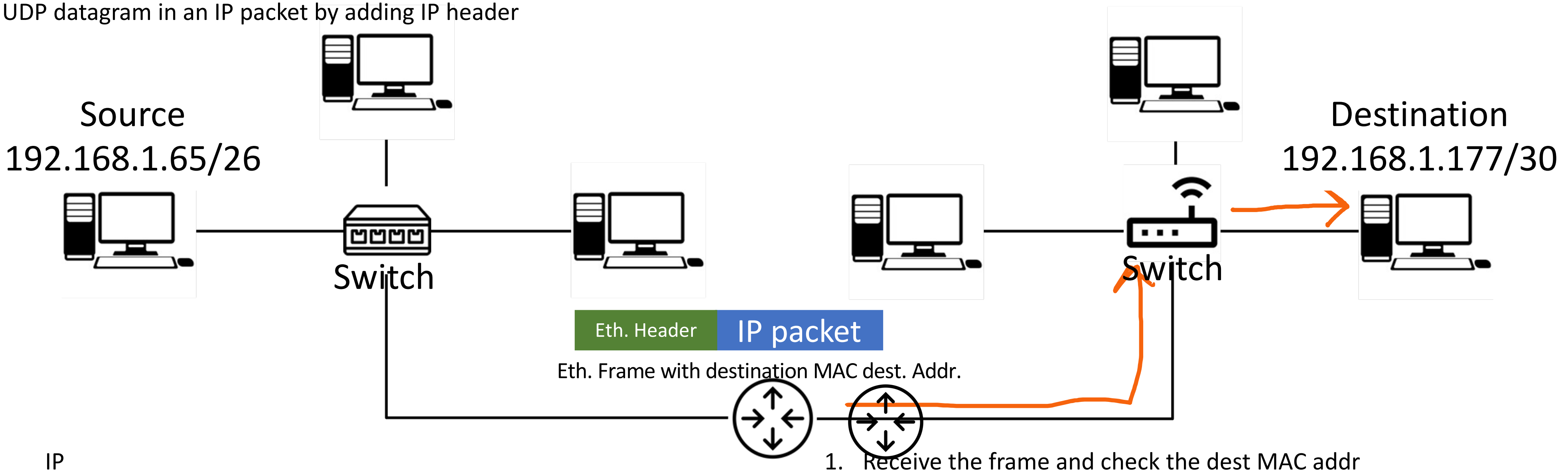
ARP
Same process

MAC
Same process

Q. 3

UDP

1. Receive application message from upper layer
2. Add UDP header containing source port (given by the client application) and dest. Port (server application port)
3. Encapsulate the UDP datagram in an IP packet by adding IP header



IP

1. Mask /26 on 192.168.1.177 = 192.168.1.128
2. 192.168.1.128 ≠ 1192.168.1.64 (current) => destination on another network
3. Check the routing table to get the router IP address

ARP

1. Check the router MAC address in the ARP table
2. If no MAC in ARP table : send ARP request
3. Adds the correspondance MAC-IP of the router in ARP table

MAC

1. Create an Ethernet frame with router MAC address, put IP packet in it and send it

1. Receive the frame and check the dest MAC addr
2. Extract the IP packet
3. Check the routing table to find the next dest. (destination IP addr.)
4. Check the MTU -> fragmentation

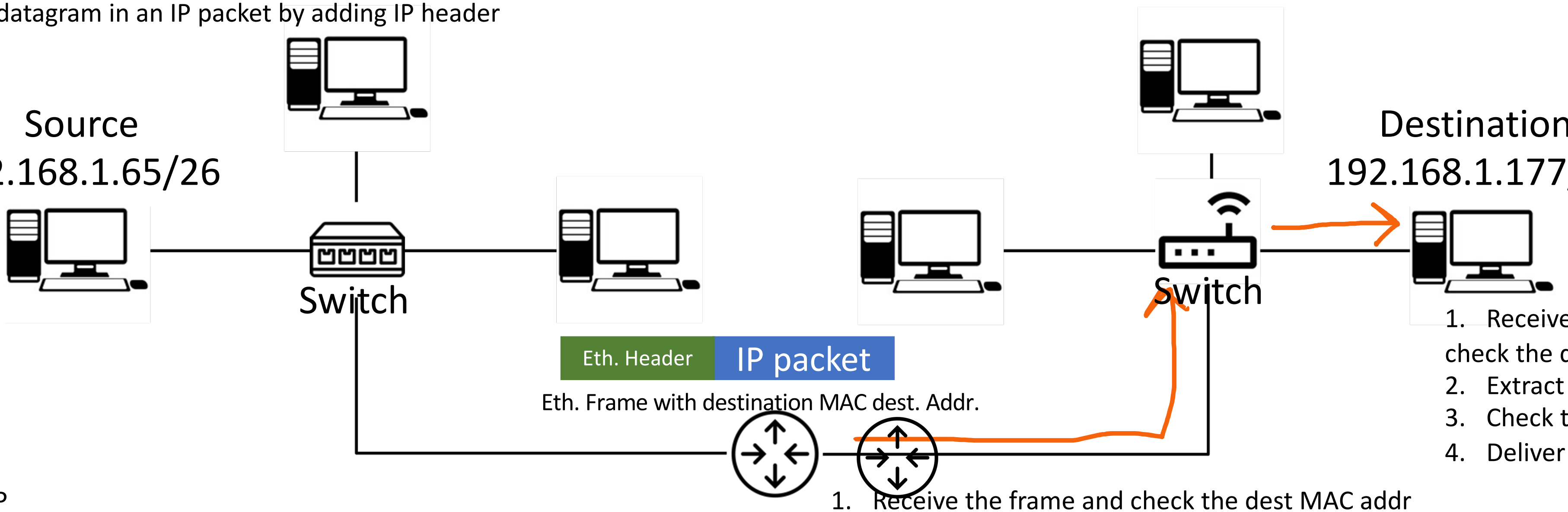
ARP
Same process

MAC
Same process

Q. 3

UDP

1. Receive application message from upper layer
2. Add UDP header containing source port (given by the client application) and dest. Port (server application port)
3. Encapsulate the UDP datagram in an IP packet by adding IP header



1. Receive the frame and check the dest MAC addr
2. Extract the IP packet
3. Check the IP address
4. Deliver it to the Transport layer (UD

IP

1. Mask /26 on 192.168.1.177 = 192.168.1.128
2. 192.168.1.128 ≠ 1192.168.1.64 (current) => destination on another network
3. Check the routing table to get the router IP address

ARP

1. Check the router MAC address in the ARP table
2. If no MAC in ARP table : send ARP request
3. Adds the correspondance MAC-IP of the router in ARP table

MAC

1. Create an Ethernet frame with router MAC address, put IP packet in it and send it

1. Receive the frame and check the dest MAC addr
2. Extract the IP packet
3. Check the routing table to find the next dest. (destination IP addr.)
4. Check the MTU -> fragmentation

ARP
Same process

MAC
Same process

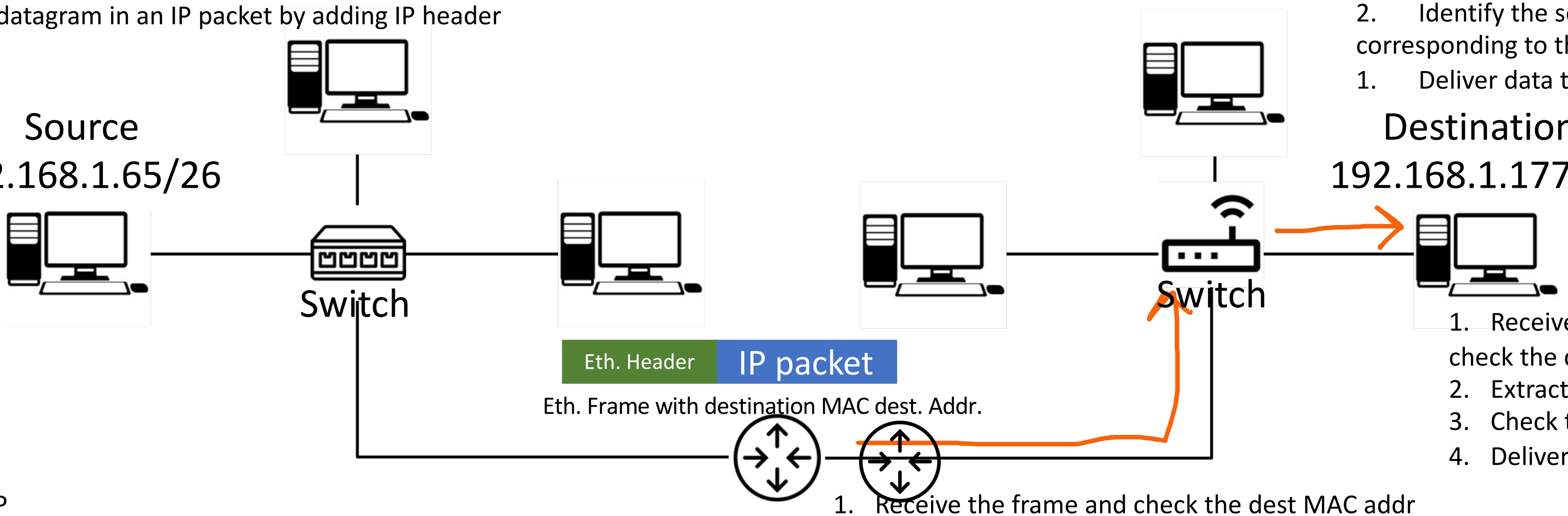
Q. 3

UDP

1. Receive application message from upper layer
2. Add UDP header containing source port (given by the client application) and dest. Port (server application port)
3. Encapsulate the UDP datagram in an IP packet by adding IP header

UDP

1. Analyse te UDP header
2. Identify the service (server in our case) corresponding to the dest. port
1. Deliver data to the server application



Eth. Frame with destination MAC dest. Addr.

1. Receive the frame and check the dest MAC addr
2. Extract the IP packet
3. Check the IP address
4. Deliver it to the Transport layer (UD

IP

1. Mask /26 on 192.168.1.177 = 192.168.1.128
2. 192.168.1.128 ≠ 1192.168.1.64 (current) => destination on another network
3. Check the routing table to get the router IP address

ARP

1. Check the router MAC address in the ARP table
2. If no MAC in ARP table : send ARP request
3. Adds the correspondance MAC-IP of the router in ARP table

MAC

1. Create an Ethernet frame with next hop MAC address, put IP packet in it and send it

1. Receive the frame and check the dest MAC addr
2. Extract the IP packet
3. Check the routing table to find the next dest. (destination IP addr.)
4. Check the MTU -> fragmentation

ARP

Same process

MAC

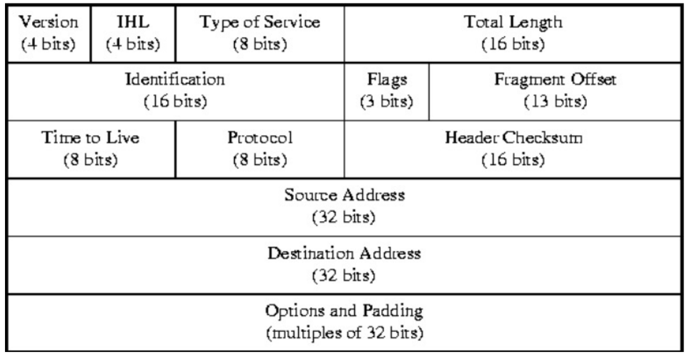
Same process

Q. 4

Frame 1

FF FF FF FF FF FF 08 00 20 02 45 9E 08 06 00 01 08 00 06 04 00 01

08 00 20 02 45 9E C0 A8 01 41 00 00 00 00 00 00 C0 A8 01 64



Version (4 bits)	IHL (4 bits)	Type of Service (8 bits)	Total Length (16 bits)

Identification (16 bits) | Flags (3 bits) | Fragment Offset (13 bits)

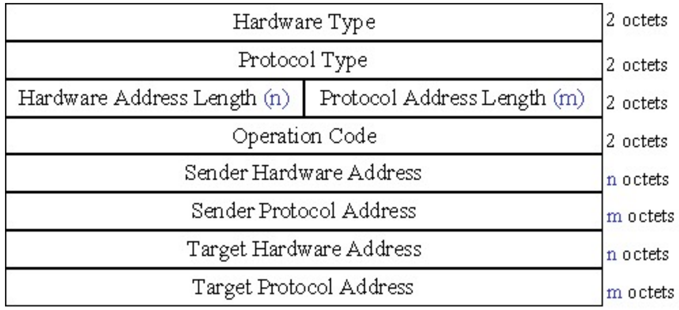
Time to Live (8 bits) | Protocol (8 bits) | Header Checksum (16 bits)

Source Address (32 bits)

Destination Address (32 bits)

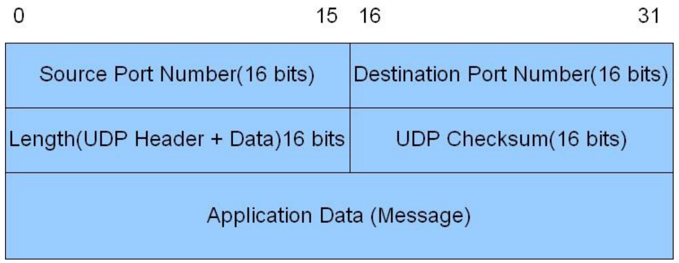
Options and Padding (multiples of 32 bits)

IP



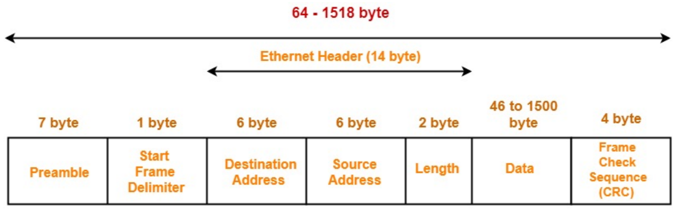
Hardware Type	2 octets
Protocol Type	2 octets
Hardware Address Length (n) | Protocol Address Length (m)	2 octets
Operation Code	2 octets
Sender Hardware Address	n octets
Sender Protocol Address	m octets
Target Hardware Address	n octets
Target Protocol Address	m octets

ARP



0 15	16 31
Source Port Number(16 bits)	Destination Port Number(16 bits)
Length(UDP Header + Data)16 bits	UDP Checksum(16 bits)
Application Data (Message)	

UDP



64 - 1518 byte

Ethernet Header (14 byte)

7 byte	1 byte	6 byte	6 byte	2 byte	46 to 1500 byte	4 byte
Preamble	Start Frame Delimiter	Destination Address	Source Address	Length	Data	Frame Check Sequence (CRC)

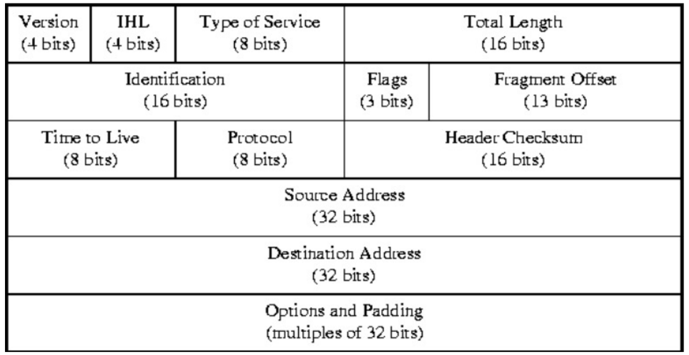
IEEE 802.3 Ethernet Frame Format

Q. 4

Frame 1

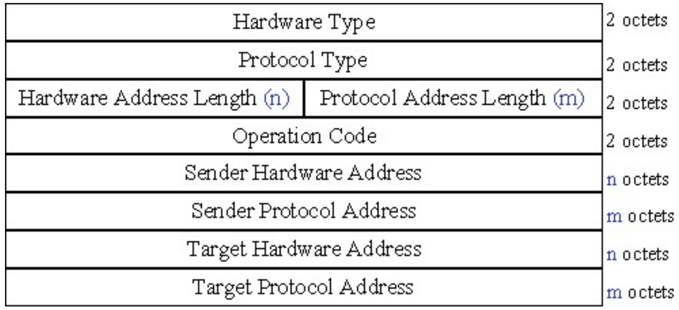
Dest mac addr : broadcast	Source mac addr :	ARP inside	
FF FF FF FF FF FF	08 00 20 02 45 9E	08 06	00 01 08 00 06 04 00 01

08 00 20 02 45 9E C0 A8 01 41 00 00 00 00 00 00 C0 A8 01 64



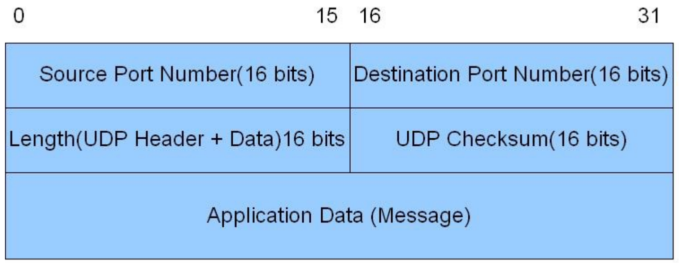
Version (4 bits)	IHL (4 bits)	Type of Service (8 bits)	Total Length (16 bits)	
Identification (16 bits)			Flags (3 bits)	Fragment Offset (13 bits)
Time to Live (8 bits)		Protocol (8 bits)	Header Checksum (16 bits)	
Source Address (32 bits)				
Destination Address (32 bits)				
Options and Padding (multiples of 32 bits)				

IP



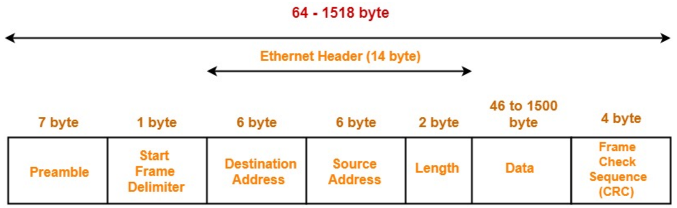
Hardware Type		2 octets
Protocol Type		2 octets
Hardware Address Length (n)	Protocol Address Length (m)	2 octets
Operation Code		2 octets
Sender Hardware Address		n octets
Sender Protocol Address		m octets
Target Hardware Address		n octets
Target Protocol Address		m octets

ARP



0 15	16 31
Source Port Number(16 bits)	Destination Port Number(16 bits)
Length(UDP Header + Data)16 bits	UDP Checksum(16 bits)
Application Data (Message)	

UDP



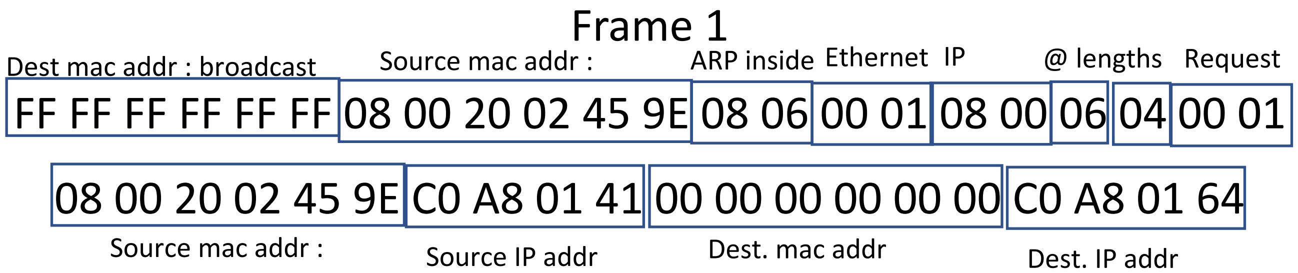
64 - 1518 byte

Ethernet Header (14 byte)

7 byte	1 byte	6 byte	6 byte	2 byte	46 to 1500 byte	4 byte
Preamble	Start Frame Delimiter	Destination Address	Source Address	Length	Data	Frame Check Sequence (CRC)

IEEE 802.3 Ethernet Frame Format

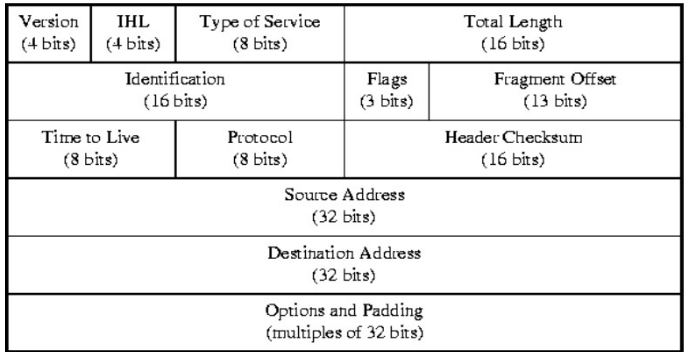
Q. 4



Frame 1

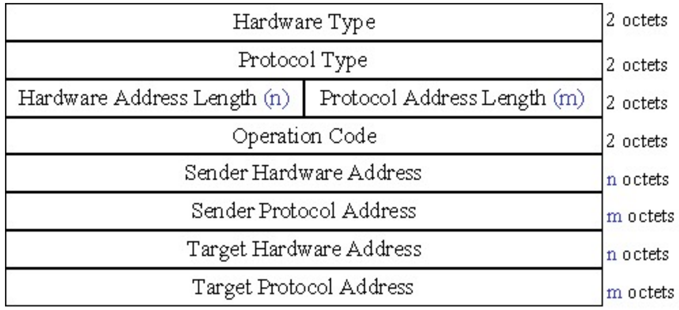
Dest mac addr : broadcast	Source mac addr :	ARP inside	Ethernet	IP	@ lengths		Request
FF FF FF FF FF FF	08 00 20 02 45 9E	08 06	00 01	08 00	06	04	00 01

08 00 20 02 45 9E	C0 A8 01 41	00 00 00 00 00 00	C0 A8 01 64
Source mac addr :	Source IP addr	Dest. mac addr	Dest. IP addr



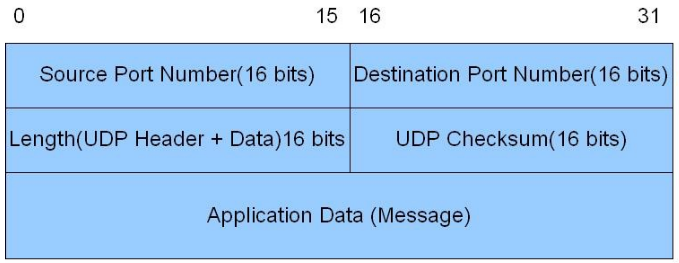
Version (4 bits)	IHL (4 bits)	Type of Service (8 bits)	Total Length (16 bits)
Identification (16 bits)		Flags (3 bits)	Fragment Offset (13 bits)
Time to Live (8 bits)		Protocol (8 bits)	Header Checksum (16 bits)
Source Address (32 bits)			
Destination Address (32 bits)			
Options and Padding (multiples of 32 bits)			

IP



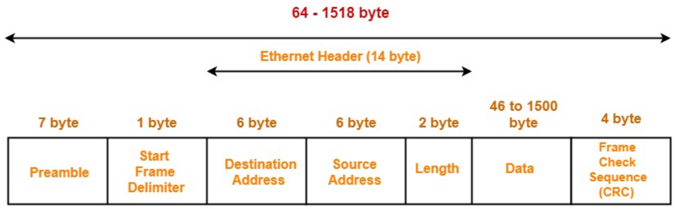
Hardware Type		2 octets
Protocol Type		2 octets
Hardware Address Length (n)	Protocol Address Length (m)	2 octets
Operation Code		2 octets
Sender Hardware Address		n octets
Sender Protocol Address		m octets
Target Hardware Address		n octets
Target Protocol Address		m octets

ARP



0 15	16 31
Source Port Number(16 bits)	Destination Port Number(16 bits)
Length(UDP Header + Data)16 bits	UDP Checksum(16 bits)
Application Data (Message)	

UDP



64 - 1518 byte

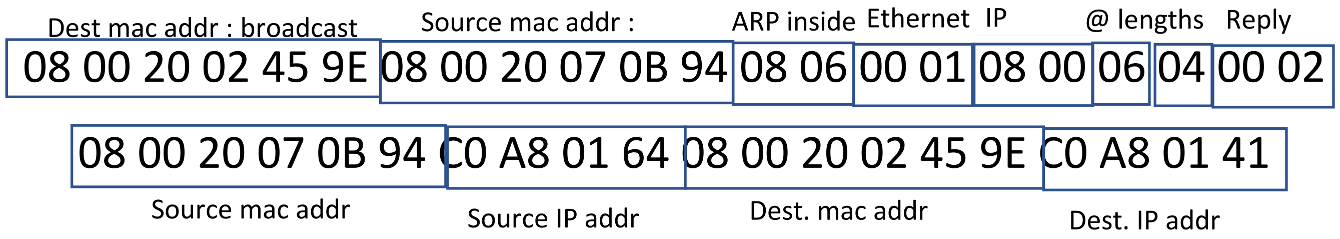
Ethernet Header (14 byte)

7 byte	1 byte	6 byte	6 byte	2 byte	46 to 1500 byte	4 byte
Preamble	Start Frame Delimiter	Destination Address	Source Address	Length	Data	Frame Check Sequence (CRC)

IEEE 802.3 Ethernet Frame Format

Q. 4

Frame 2



Dest mac addr : broadcast

Source mac addr :

ARP inside

Ethernet

IP

@ lengths

Reply

08 00 20 02 45 9E | 08 00 20 07 0B 94 | 08 06 | 00 01 | 08 00 | 06 | 04 | 00 02

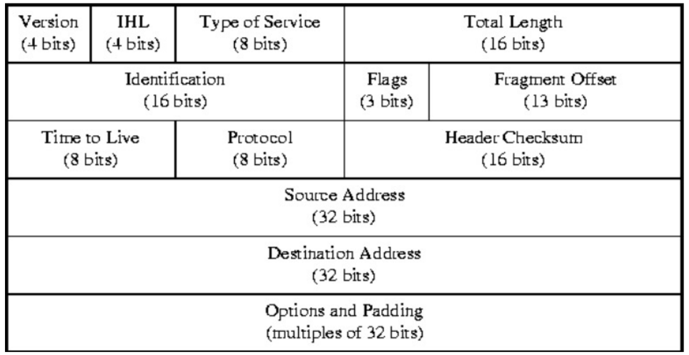
08 00 20 07 0B 94 | C0 A8 01 64 | 08 00 20 02 45 9E | C0 A8 01 41

Source mac addr

Source IP addr

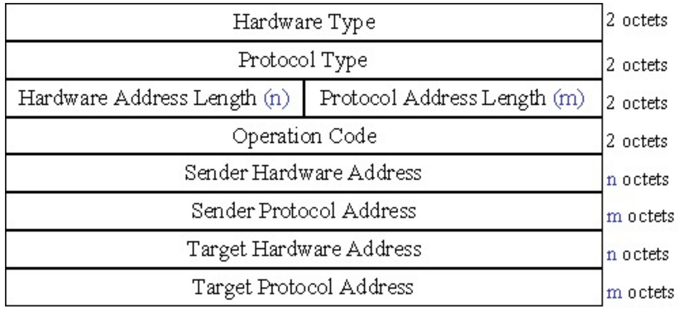
Dest. mac addr

Dest. IP addr



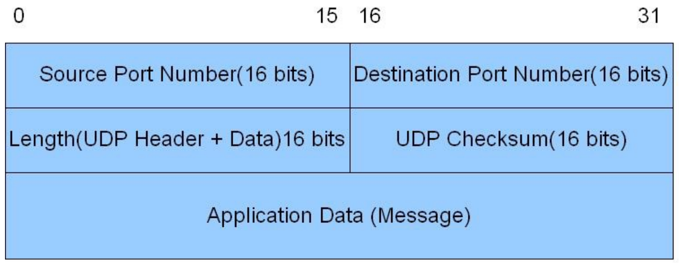
Version (4 bits)	IHL (4 bits)	Type of Service (8 bits)	Total Length (16 bits)
Identification (16 bits)		Flags (3 bits)	Fragment Offset (13 bits)
Time to Live (8 bits)		Protocol (8 bits)	Header Checksum (16 bits)
Source Address (32 bits)			
Destination Address (32 bits)			
Options and Padding (multiples of 32 bits)			

IP



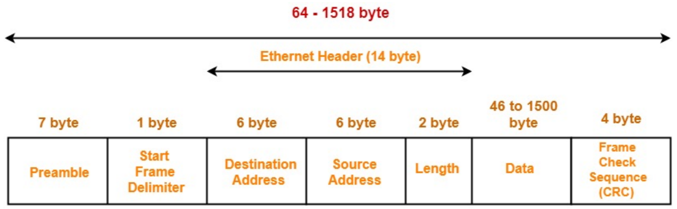
Hardware Type		2 octets
Protocol Type		2 octets
Hardware Address Length (n)	Protocol Address Length (m)	2 octets
Operation Code		2 octets
Sender Hardware Address		n octets
Sender Protocol Address		m octets
Target Hardware Address		n octets
Target Protocol Address		m octets

ARP



0 15	16 31
Source Port Number(16 bits)	Destination Port Number(16 bits)
Length(UDP Header + Data)16 bits	UDP Checksum(16 bits)
Application Data (Message)	

UDP



64 - 1518 byte

Ethernet Header (14 byte)

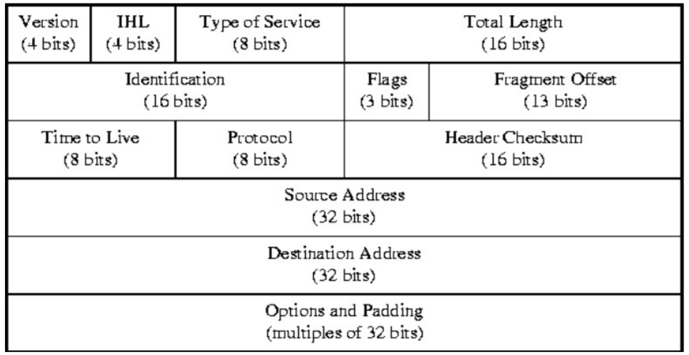
7 byte	1 byte	6 byte	6 byte	2 byte	46 to 1500 byte	4 byte
Preamble	Start Frame Delimiter	Destination Address	Source Address	Length	Data	Frame Check Sequence (CRC)

IEEE 802.3 Ethernet Frame Format

Q. 4

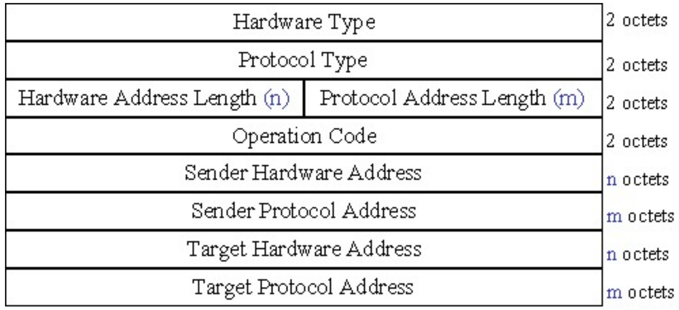
Frame 3

```
08 00 20 07 0B 94 08 00 20 02 45 9E 08 00 45 00 00 1D 7B BD 00 00
80 11 3A E5 C0 A8 01 41 C0 A8 01 B1 23 82 23 83 00 09 33 A9 01 01
```



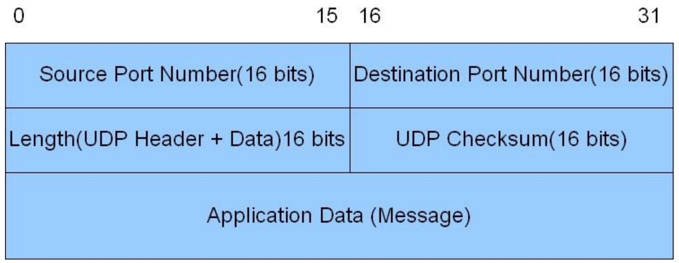
Version (4 bits)	IHL (4 bits)	Type of Service (8 bits)	Total Length (16 bits)
Identification (16 bits)		Flags (3 bits)	Fragment Offset (13 bits)
Time to Live (8 bits)	Protocol (8 bits)	Header Checksum (16 bits)	
Source Address (32 bits)			
Destination Address (32 bits)			
Options and Padding (multiples of 32 bits)			

IP



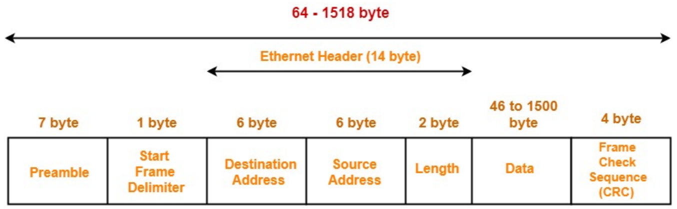
Field	Size
Hardware Type	2 octets
Protocol Type	2 octets
Hardware Address Length (n) | Protocol Address Length (m)	2 octets
Operation Code	2 octets
Sender Hardware Address	n octets
Sender Protocol Address	m octets
Target Hardware Address	n octets
Target Protocol Address	m octets

ARP



0 – 15	16 – 31
Source Port Number(16 bits)	Destination Port Number(16 bits)
Length(UDP Header + Data)16 bits	UDP Checksum(16 bits)
Application Data (Message)	

UDP



7 byte	1 byte	6 byte	6 byte	2 byte	46 to 1500 byte	4 byte
Preamble	Start Frame Delimiter	Destination Address	Source Address	Length	Data	Frame Check Sequence (CRC)

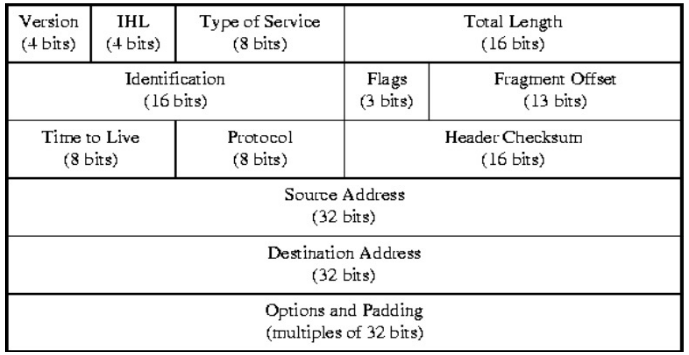
IEEE 802.3 Ethernet Frame Format

Q. 4

Frame 3

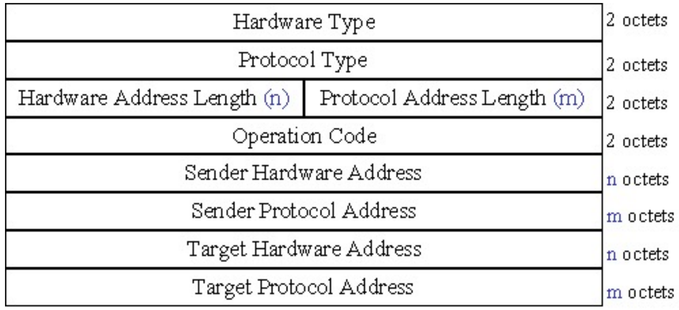
Dest mac addr: 08 00 20 07 0B 94
Source mac addr : 08 00 20 02 45 9E
IP inside: 08 00

```
08 00 20 07 0B 94 08 00 20 02 45 9E 08 00 45 00 00 1D 7B BD 00 00
80 11 3A E5 C0 A8 01 41 C0 A8 01 B1 23 82 23 83 00 09 33 A9 01 01
```



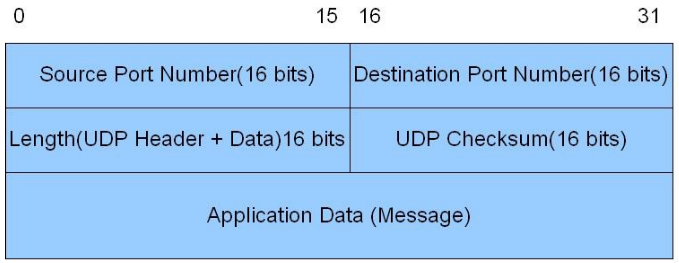
Version (4 bits)	IHL (4 bits)	Type of Service (8 bits)	Total Length (16 bits)	
Identification (16 bits)			Flags (3 bits)	Fragment Offset (13 bits)
Time to Live (8 bits)		Protocol (8 bits)	Header Checksum (16 bits)	
Source Address (32 bits)				
Destination Address (32 bits)				
Options and Padding (multiples of 32 bits)				

IP



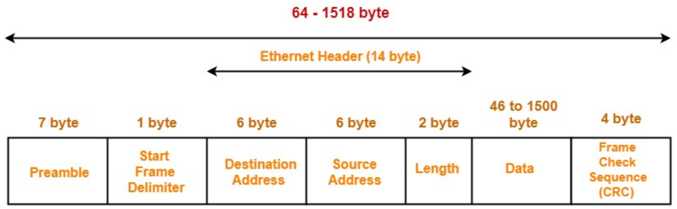
Hardware Type		2 octets
Protocol Type		2 octets
Hardware Address Length (n)	Protocol Address Length (m)	2 octets
Operation Code		2 octets
Sender Hardware Address		n octets
Sender Protocol Address		m octets
Target Hardware Address		n octets
Target Protocol Address		m octets

ARP



Source Port Number(16 bits)	Destination Port Number(16 bits)
Length(UDP Header + Data)16 bits	UDP Checksum(16 bits)
Application Data (Message)	

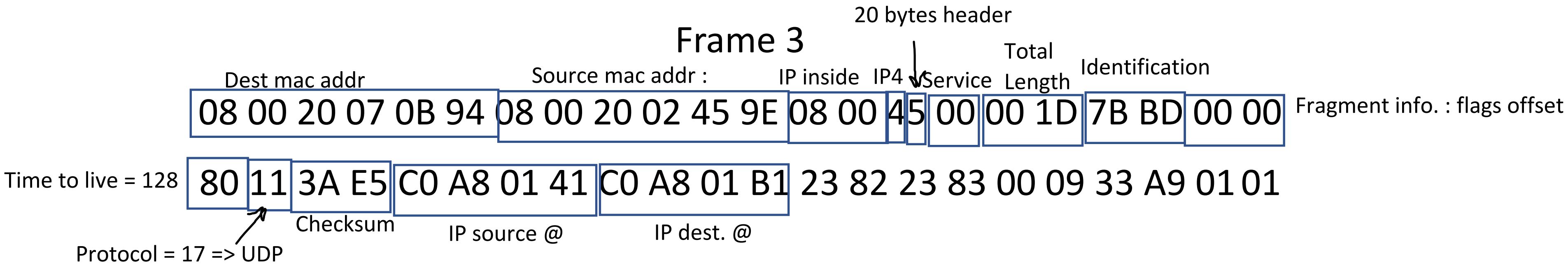
UDP



7 byte	1 byte	6 byte	6 byte	2 byte	46 to 1500 byte	4 byte
Preamble	Start Frame Delimiter	Destination Address	Source Address	Length	Data	Frame Check Sequence (CRC)

IEEE 802.3 Ethernet Frame Format

Q. 4



Frame 3

20 bytes header

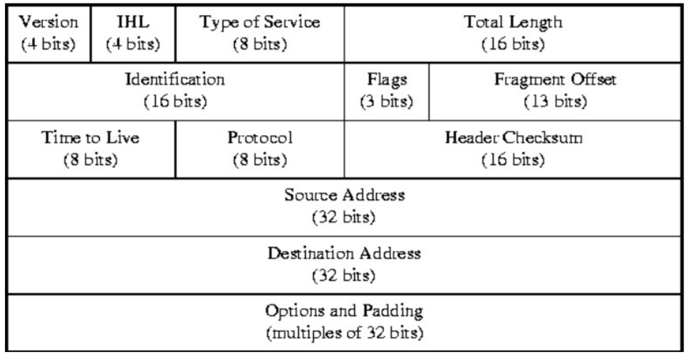
Dest mac addr | Source mac addr : | IP inside | IP4 | Service | Total Length | Identification

08 00 20 07 0B 94 | 08 00 20 02 45 9E | 08 00 | 4 | 5 | 00 | 00 1D | 7B BD | 00 00 Fragment info. : flags offset

Time to live = 128 | 80 | 11 | 3A E5 | C0 A8 01 41 | C0 A8 01 B1 | 23 82 23 83 00 09 33 A9 01 01

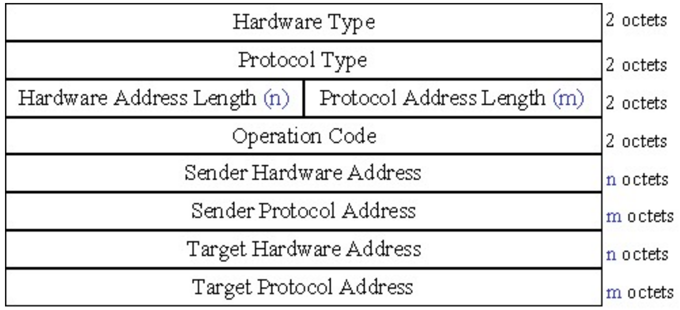
Checksum | IP source @ | IP dest. @

Protocol = 17 => UDP



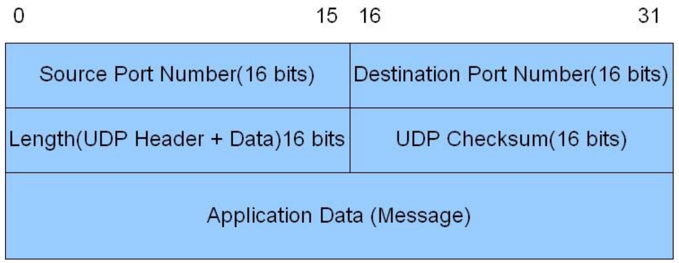
<table>
<tr><td>Version
(4 bits)</td><td>IHL
(4 bits)</td><td>Type of Service
(8 bits)</td><td colspan="2">Total Length
(16 bits)</td></tr>
<tr><td colspan="3">Identification
(16 bits)</td><td>Flags
(3 bits)</td><td>Fragment Offset
(13 bits)</td></tr>
<tr><td colspan="2">Time to Live
(8 bits)</td><td>Protocol
(8 bits)</td><td colspan="2">Header Checksum
(16 bits)</td></tr>
<tr><td colspan="5">Source Address
(32 bits)</td></tr>
<tr><td colspan="5">Destination Address
(32 bits)</td></tr>
<tr><td colspan="5">Options and Padding
(multiples of 32 bits)</td></tr>
</table>

IP



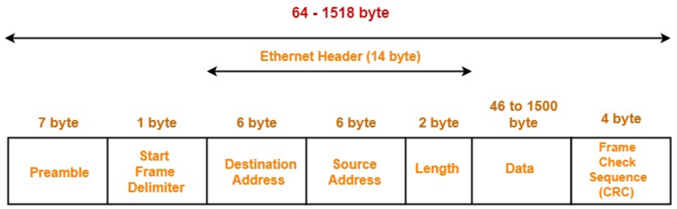
<table>
<tr><td colspan="2">Hardware Type</td><td>2 octets</td></tr>
<tr><td colspan="2">Protocol Type</td><td>2 octets</td></tr>
<tr><td>Hardware Address Length (n)</td><td>Protocol Address Length (m)</td><td>2 octets</td></tr>
<tr><td colspan="2">Operation Code</td><td>2 octets</td></tr>
<tr><td colspan="2">Sender Hardware Address</td><td>n octets</td></tr>
<tr><td colspan="2">Sender Protocol Address</td><td>m octets</td></tr>
<tr><td colspan="2">Target Hardware Address</td><td>n octets</td></tr>
<tr><td colspan="2">Target Protocol Address</td><td>m octets</td></tr>
</table>

ARP



<table>
<tr><td>0 15</td><td>16 31</td></tr>
<tr><td>Source Port Number(16 bits)</td><td>Destination Port Number(16 bits)</td></tr>
<tr><td>Length(UDP Header + Data)16 bits</td><td>UDP Checksum(16 bits)</td></tr>
<tr><td colspan="2">Application Data (Message)</td></tr>
</table>

UDP



64 - 1518 byte

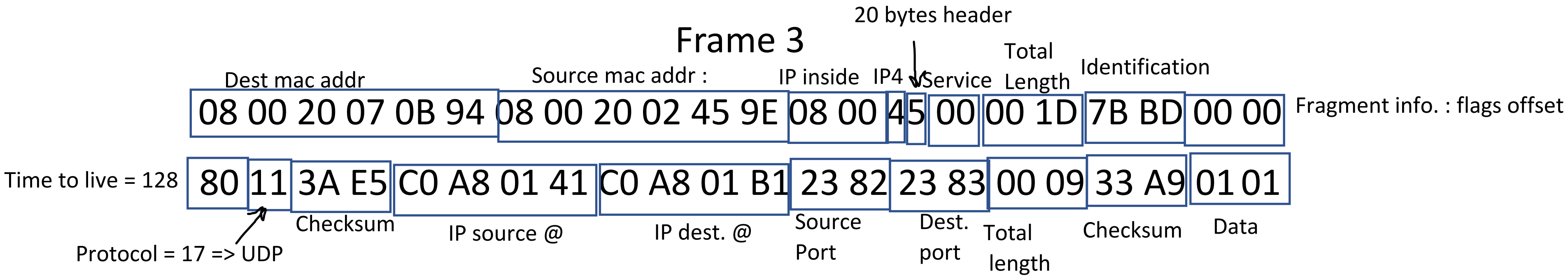
Ethernet Header (14 byte)

7 byte	1 byte	6 byte	6 byte	2 byte	46 to 1500 byte	4 byte
Preamble	Start Frame Delimiter	Destination Address	Source Address	Length	Data	Frame Check Sequence (CRC)

IEEE 802.3 Ethernet Frame Format

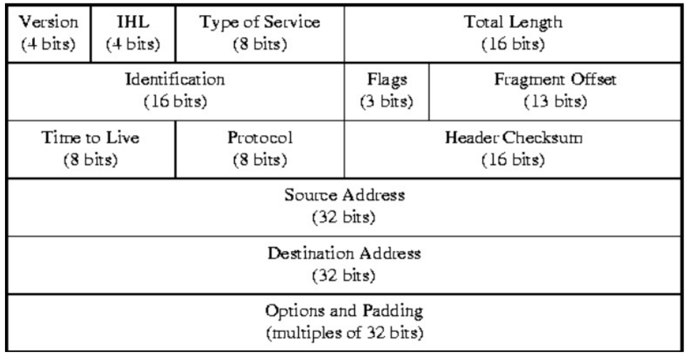
Q. 4

Frame 3



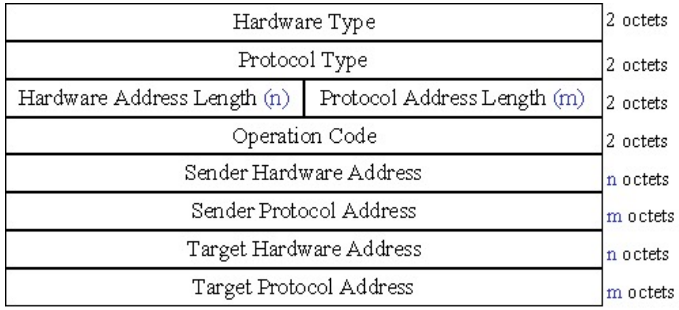
Dest mac addr	Source mac addr :	IP inside	IP4	20 bytes header	Service	Total Length	Identification	Fragment info. : flags offset
08 00 20 07 0B 94	08 00 20 02 45 9E	08 00	4	5	00	00 1D	7B BD	00 00

Time to live = 128	Protocol = 17 => UDP	Checksum	IP source @	IP dest. @	Source Port	Dest. port	Total length	Checksum	Data
80	11	3A E5	C0 A8 01 41	C0 A8 01 B1	23 82	23 83	00 09	33 A9	01 01



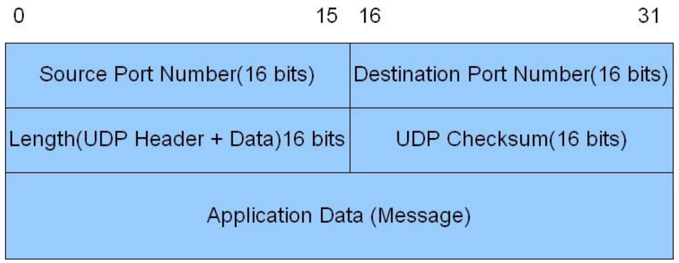
Version (4 bits)	IHL (4 bits)	Type of Service (8 bits)	Total Length (16 bits)	
Identification (16 bits)			Flags (3 bits)	Fragment Offset (13 bits)
Time to Live (8 bits)		Protocol (8 bits)	Header Checksum (16 bits)	
Source Address (32 bits)				
Destination Address (32 bits)				
Options and Padding (multiples of 32 bits)				

IP



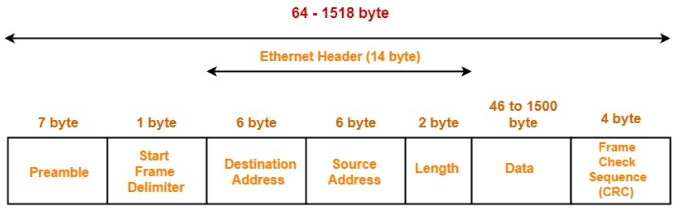
Hardware Type		2 octets
Protocol Type		2 octets
Hardware Address Length (n)	Protocol Address Length (m)	2 octets
Operation Code		2 octets
Sender Hardware Address		n octets
Sender Protocol Address		m octets
Target Hardware Address		n octets
Target Protocol Address		m octets

ARP



0 15	16 31
Source Port Number(16 bits)	Destination Port Number(16 bits)
Length(UDP Header + Data)16 bits	UDP Checksum(16 bits)
Application Data (Message)	

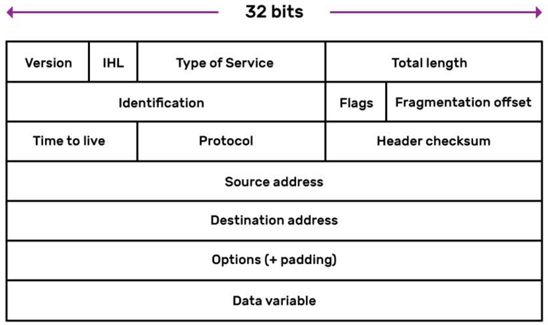
UDP



7 byte	1 byte	6 byte	6 byte	2 byte	46 to 1500 byte	4 byte
Preamble	Start Frame Delimiter	Destination Address	Source Address	Length	Data	Frame Check Sequence (CRC)

IEEE 802.3 Ethernet Frame Format

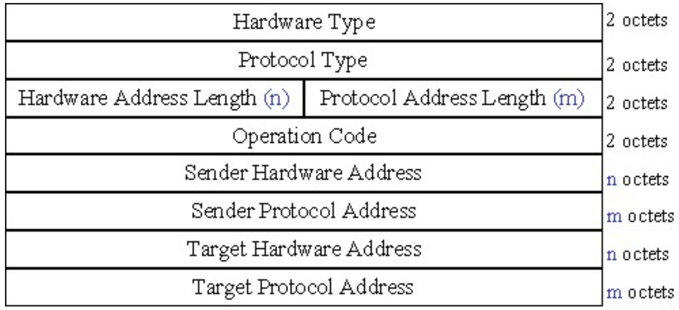
Q5. frame 1 and 2 : ARP request from sender to router 1 and reply from router 1 to sender
Frame 3 : UDP datagram going from the sender client application to the receiver
server application, captured between the sender and router 1 Chapter 4 : Network layer



32 bits

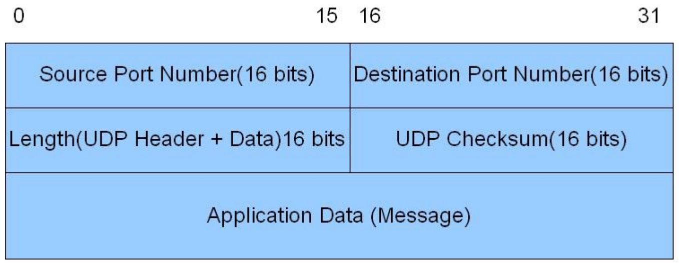
Version	IHL	Type of Service	Total length
Identification		Flags	Fragmentation offset
Time to live	Protocol		Header checksum
Source address			
Destination address			
Options (+ padding)			
Data variable			

IP



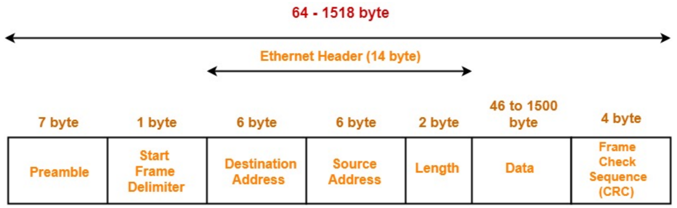
Hardware Type		2 octets
Protocol Type		2 octets
Hardware Address Length (n)	Protocol Address Length (m)	2 octets
Operation Code		2 octets
Sender Hardware Address		n octets
Sender Protocol Address		m octets
Target Hardware Address		n octets
Target Protocol Address		m octets

ARP



0 15	16 31
Source Port Number(16 bits)	Destination Port Number(16 bits)
Length(UDP Header + Data)16 bits	UDP Checksum(16 bits)
Application Data (Message)	

UDP



64 - 1518 byte

Ethernet Header (14 byte)

7 byte	1 byte	6 byte	6 byte	2 byte	46 to 1500 byte	4 byte
Preamble	Start Frame Delimiter	Destination Address	Source Address	Length	Data	Frame Check Sequence (CRC)

IEEE 802.3 Ethernet Frame Format