Unit-2

Protocols for IoT

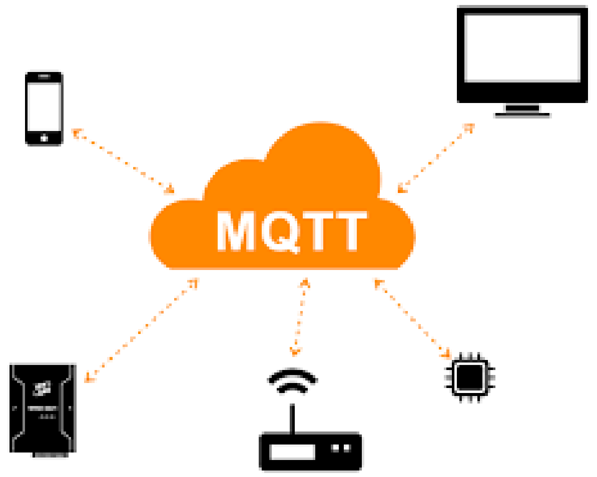
Messaging Protocols

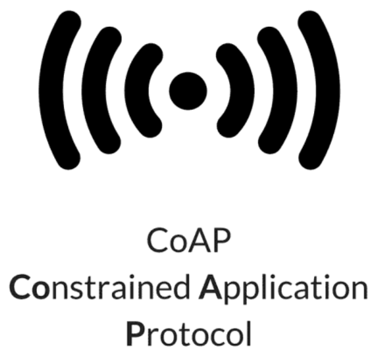
Section - 1

Messaging Protocols in IoT

- Messaging protocols are very important for the transfer of data in terms of messages.
- They are useful for send/receive of a message to/from the cloud in IoT applications.
- In the section, two messaging protocols are discussed.

1. Message Queuing Telemetry Transport (MQTT)
2. Constrained Application Protocol (CoAP)





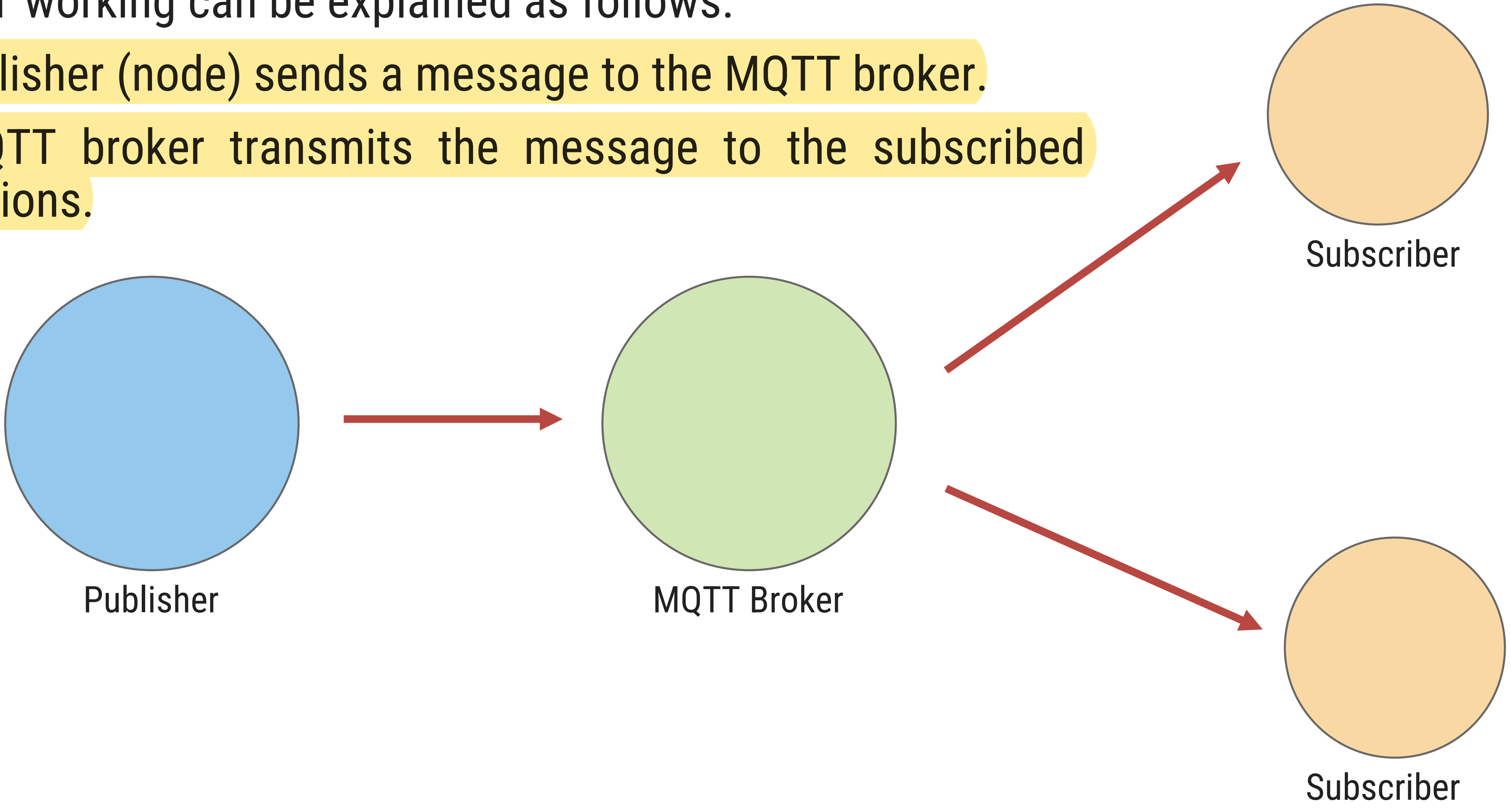
Message Queuing Telemetry Transport (MQTT)

- As we know, IoT has one of the biggest challenges of resource constraints, the lightweight protocol is more preferred.
- MQTT is widely used in IoT applications as it is a lightweight protocol.
- Here, lightweight means, it can work with minimal resources and does not require any specific hardware architecture or additional resources.

MQTT

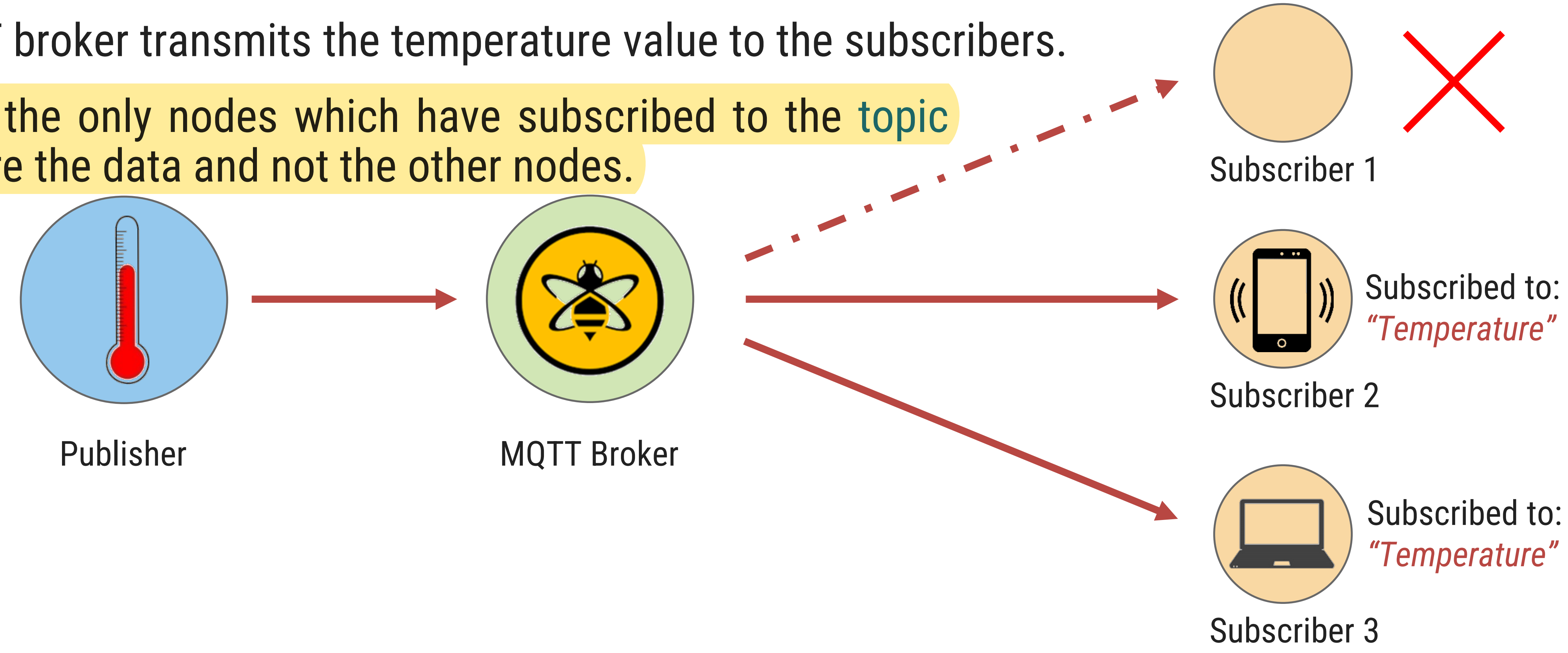
- MQTT works with “publish – subscribe” pattern.
- The MQTT working can be explained as follows.

1. The publisher (node) sends a message to the MQTT broker.
2. The MQTT broker transmits the message to the subscribed destinations.



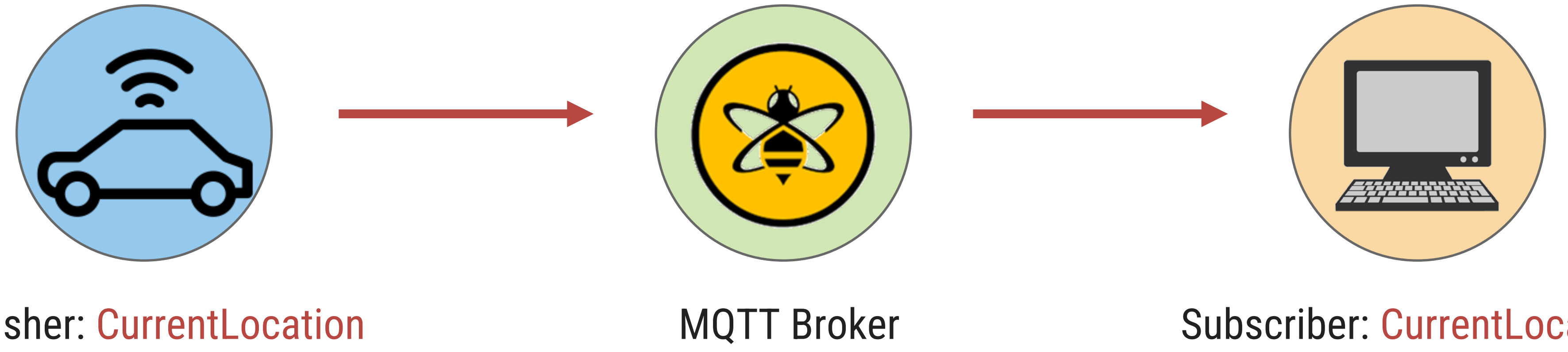
MQTT - Example

- Suppose a temperature sensor is connected to any system and we want to monitor that temperature.
- The controller connected to the temperature sensor publishes the temperature value to the MQTT broker with a topic name: *"Temperature"*. Hence, it is considered a publisher.
- The MQTT broker transmits the temperature value to the subscribers.
- Note that the only nodes which have subscribed to the topic will capture the data and not the other nodes.



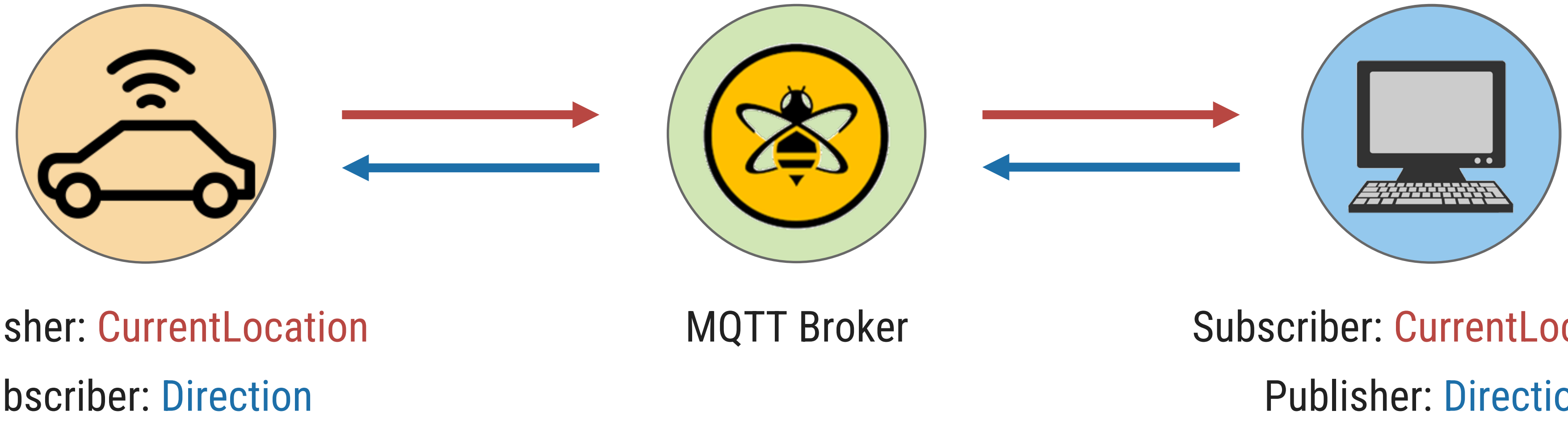
MQTT - Example

- Can publisher and subscriber interchange their role?
- Let us take an example of a Driverless car.
- A GPS sensor is connected to the car which gives the current location of the car.
- The system in car, publishes the location to MQTT broker with topic name – CurrentLocation.
- The server/computer subscribes to the topic CurrentLocation and receives the current location of the car.



MQTT - Example

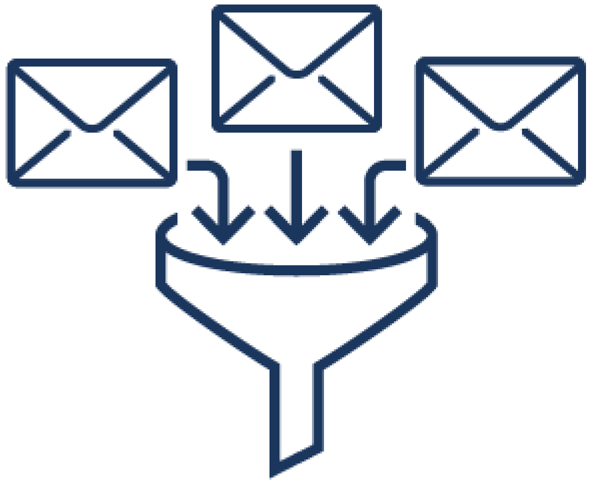
- Based on the currently selected destination, the server/computer computes the direction of the car.
- This data is published to an MQTT broker with a topic named – Direction.
- To get the command from the server, the system in the car has to subscribe to the topic named Direction.
- According to the command received from the server, the car will run in a particular direction.

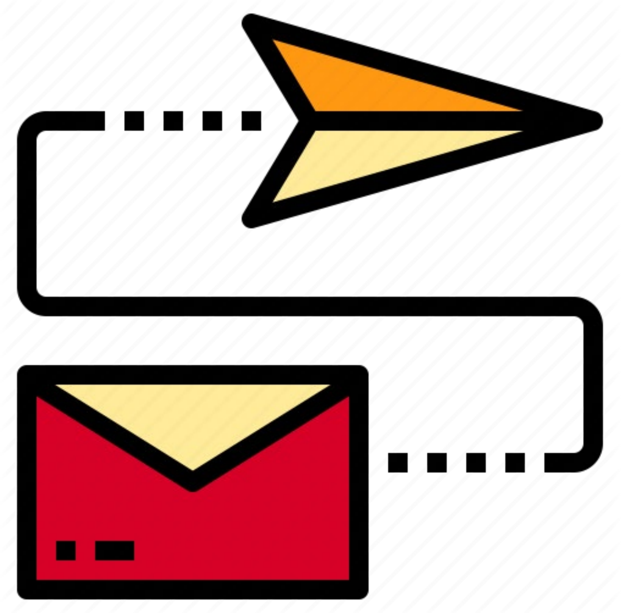


MQTT - Working

- In MQTT, clients do not have any address like a typical networking scheme.
- The broker filters the messages based on the subscribed topics.
- The messages will then be circulated to respective subscribers.
- The topic name can be anything in string format.
- The MQTT working can be explained with an example of the television broadcast.

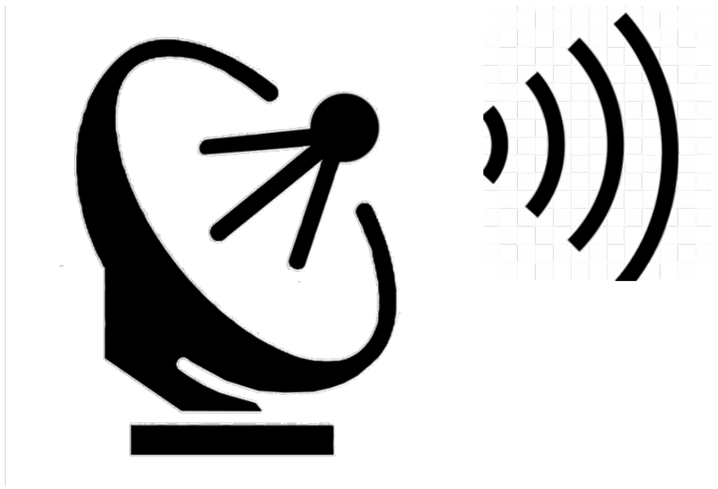
~~Client_Address~~

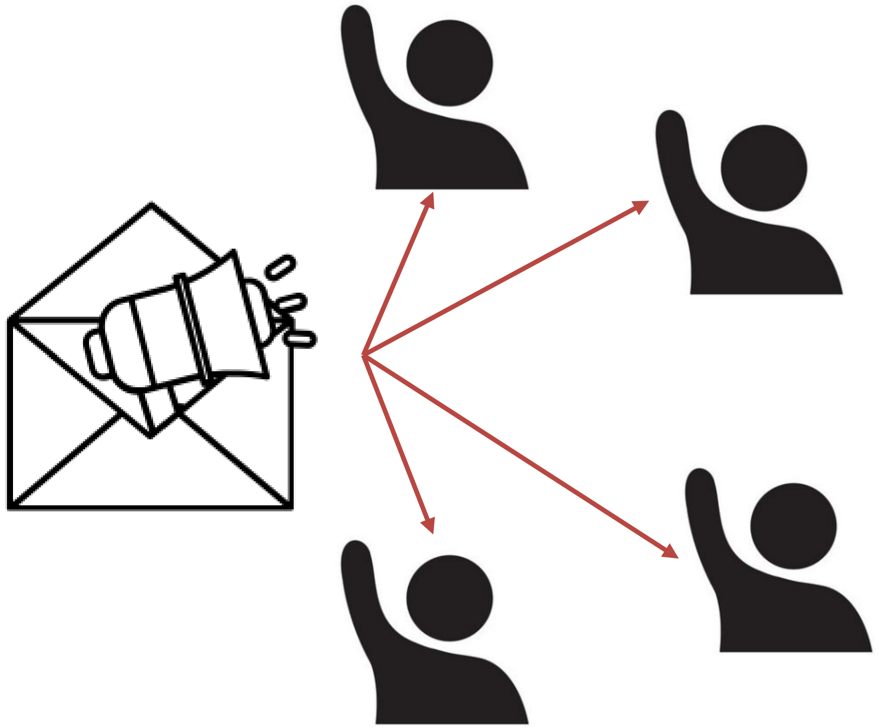




MQTT - Working

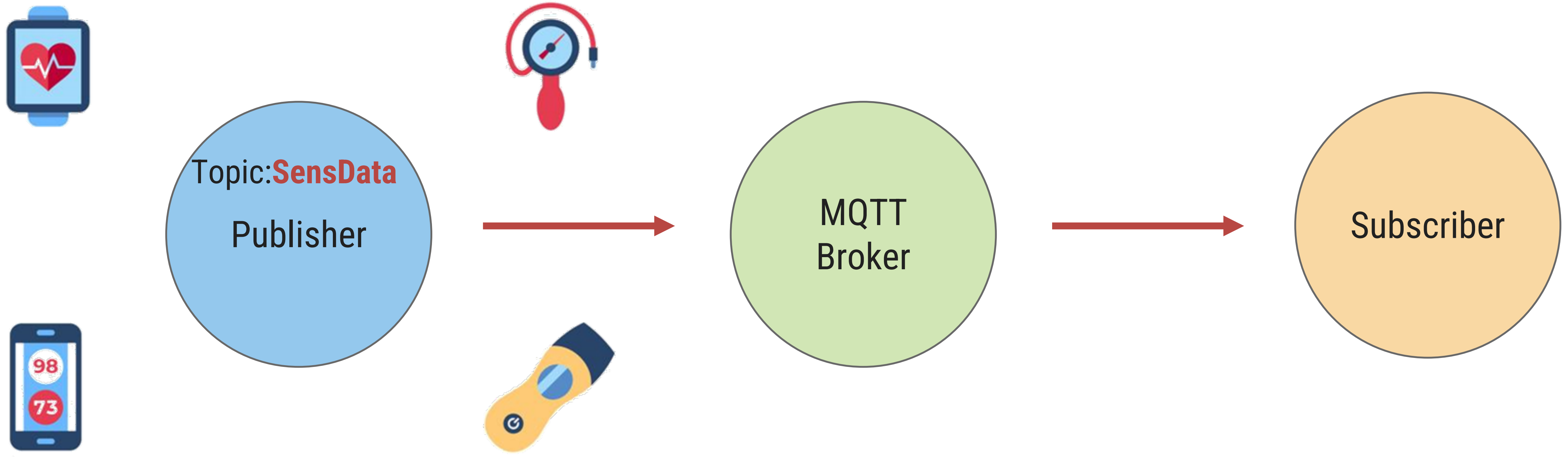
- Television broadcaster station broadcasts all the channels.
- The viewers subscribe to their favorite channels only.
- Similarly, in MQTT, the publisher node publishes data with the topic name and interested subscribers subscribe to the topic.
- Note that there can be multiple subscribers of a same topic as well as a single subscriber can subscribe to multiple topics.





MQTT – Node(s)

- Node(s) in MQTT collects the information from sensors or other i/p devices in case of the publisher.
- Connects it to the messaging server known as the MQTT broker.
- A specific string – Topic is used to publish the message and let other nodes understand the information. These nodes are considered subscribers.
- Any node can be publisher or subscriber and node is also referred to as client.



QoS in MQTT

- MQTT supports different QoS (Quality of Service) levels.
- There are 3 layers of QoS supported by MQTT.

1. Level 0 = At most once (Best effort, No Acknowledgement)
2. Level 1 = At least once (Acknowledged, retransmitted if *ack* not received)
3. Level 2 = Exactly once (Requested to send, clear to send)

MQTT Publisher – Code Implementation

- The code for MQTT publisher is as shown in below.

MQTT_publisher.ino

```
#include <ESP8266WiFi.h>
#include <PubSubClient.h>

const char* ssid = "OnePlusTJ";
const char* pwd = "Tjm12347";
const char* mqtt_server =
        "broker.mqtt-dashboard.com";

WiFiClient espClient;
PubSubClient client(espClient);
int value = 0;
unsigned long lastMsg = 0;
#define MSG_BUFFER_SIZE  (50)
char msg[MSG_BUFFER_SIZE];
```

Including the “ESP8266WiFi” library for enabling Wi-Fi connection with protocols.

Including the “PubSubClient” library for use of MQTT protocol and their functions.

Defining parameters for Wi-Fi connection.

Defining MQTT server/broker for publishing the message on particular topic.

MQTT Publisher – Code Implementation

MQTT_publisher.ino

```
void setup_wifi() {
  delay(10);
  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);
  WiFi.mode(WIFI_STA);
  WiFi.begin(ssid, pwd);

  while(WiFi.status()!=WL_CONNECTED)
  {
    delay(500);
    Serial.print(".");
  }
  Serial.println("WiFi connected");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
}
```

Initializing Wi-Fi module as station mode.

Connecting Wi-Fi module with given SSID and Password

Try until Wi-Fi is not connected to given SSID with delay of half second.

MQTT Publisher – Code Implementation

MQTT_publisher.ino

```
void setup() {
  Serial.begin(115200);
  setup_wifi();
 client.setServer(mqtt_server,1883);
  while (!client.connected()) {
    Serial.print("Attempting MQTT");
    String clientId="ESPClient1234";
    if (client.connect(
                   clientId.c_str())) {
      Serial.println("connected");
      client.publish("darshan/6thsem
        /ce", “You are in Darshan");
    }
  }
}
```

Connects the client to the MQTT Server stored in *mqtt_server* with port number 1883.

Execute the syntax in loop until ESP is not connected to MQTT broker.

Defining a *clientId* for MQTT server/broker.

Returns the pointer of string *clientId* to the array.

Publishing announcement message after connecting to the MQTT broker with Topic name : darshan/6thsem/ce

MQTT Publisher – Code Implementation

MQTT_publisher.ino

```
void loop() {

  client.loop();
  unsigned long now = millis();
  if (now - lastMsg > 2000)
  {
    lastMsg = now;
    ++value;
    snprintf (msg, MSG_BUFFER_SIZE,
         "hello world #%ld", value);
   Serial.print("Publish message:");
    Serial.println(msg);
    client.publish("darshan/6thsem/
                               ce", msg);
  }
}
```

This function allows client to process incoming messages, publish data and refresh the connection.

Creating a string in *msg* with given buffer size and assigning the value “hello world” with concatenation of integer named *value*.

Publishes the *msg* in the given topic name: *darshan/6thsem/ce*

MQTT Subscriber – Code Implementation

- The code for MQTT subscriber is as shown in below.

MQTT_subscriber.ino

```
#include <ESP8266WiFi.h>
#include <PubSubClient.h>

const char* ssid = "OnePlusTJ";
const char* pwd = "Tjm12347";
const char* mqtt_server =
        "broker.mqtt-dashboard.com";

WiFiClient espClient;
PubSubClient client(espClient);
```

Including the "ESP8266WiFi" library for enabling Wi-Fi connection with protocols.

Including the "PubSubClient" library for use of MQTT protocol and their functions.

Defining parameters for Wi-Fi connection.

Defining MQTT server/broker for publishing the message on particular topic.

MQTT Subscriber – Code Implementation

MQTT_subscriber.ino

```
void setup_wifi() {
  delay(10);
  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);
  WiFi.mode(WIFI_STA);
  WiFi.begin(ssid, pwd);

  while(WiFi.status()!=WL_CONNECTED)
  {
    delay(500);
    Serial.print(".");
  }
  Serial.println("WiFi connected");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
}
```

Initializing Wi-Fi module as station mode.

Connecting Wi-Fi module with given SSID and Password

Try until Wi-Fi is not connected to given SSID with delay of half second.

MQTT Subscriber – Code Implementation

MQTT_subscriber.ino

```
void callback(char* topic,
byte* payload, unsigned int length)
{
  Serial.print("Message arrived [");
  Serial.print(topic);
  Serial.print("] ");
  for (int i = 0; i < length; i++)
  {
    Serial.print((char)payload[i]);
  }
  Serial.println();
}
```

A *callback* function for receiving messages every time when *client.loop( )* executes.

This function accepts the arguments *char* of topic*, *char* of payload* and the *length of payload*.

Executing the for loop until all the characters of the payload are printed.

Typecasting of *payload[i]* into equivalent character and then printing on serial monitor.

MQTT Subscriber – Code Implementation

MQTT_subscriber.ino

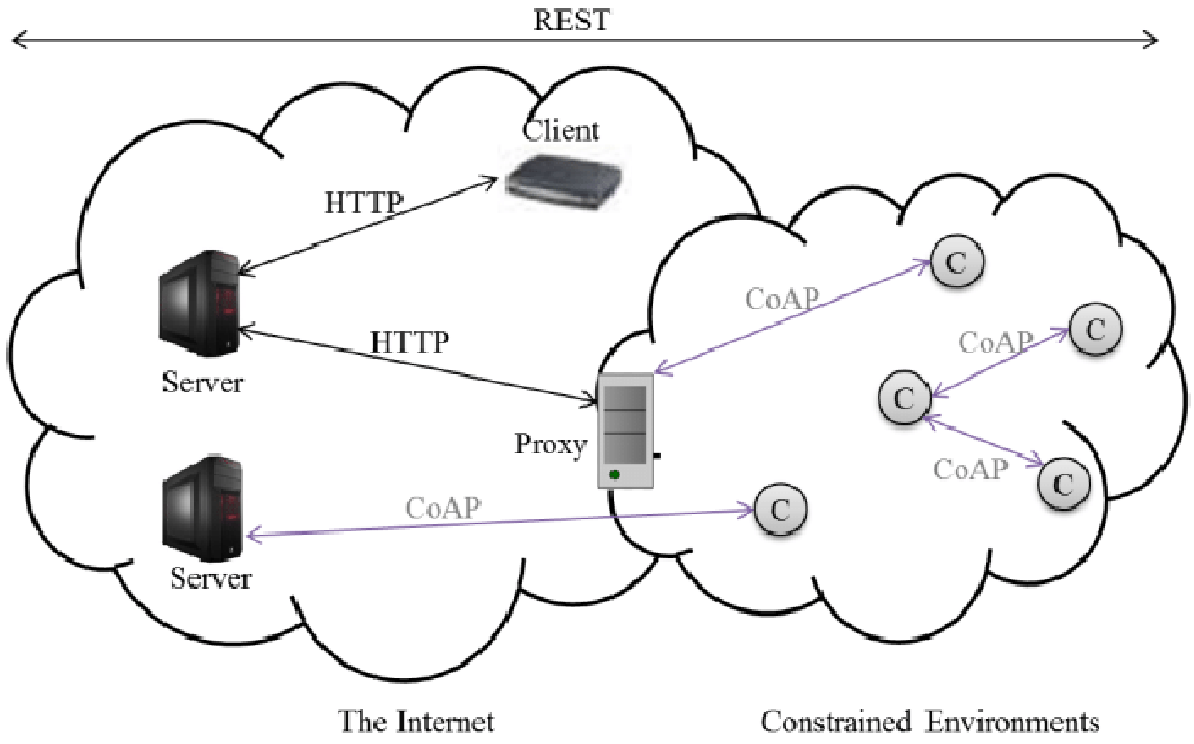
```
void setup() {
  Serial.begin(115200);
  setup_wifi();
 client.setServer(mqtt_server,1883);
  client.setCallback(callback);
  while (!client.connected()) {
   Serial.print("Attempting MQTT");
   String clientId="ESPClient1234";
   if (client.connect(
                 clientId.c_str())) {
    Serial.println("connected");
    client.subscribe("darshan/Msg");
    }
  }
}
void loop() {
  client.loop(); }
```

This is built-in function *client.setCallback( )* in *PubSubClient* library and it calls a function *callback( )* every time when *client.loop( )* function is executed.

Subscribing to the Topic named “darshan/Msg”

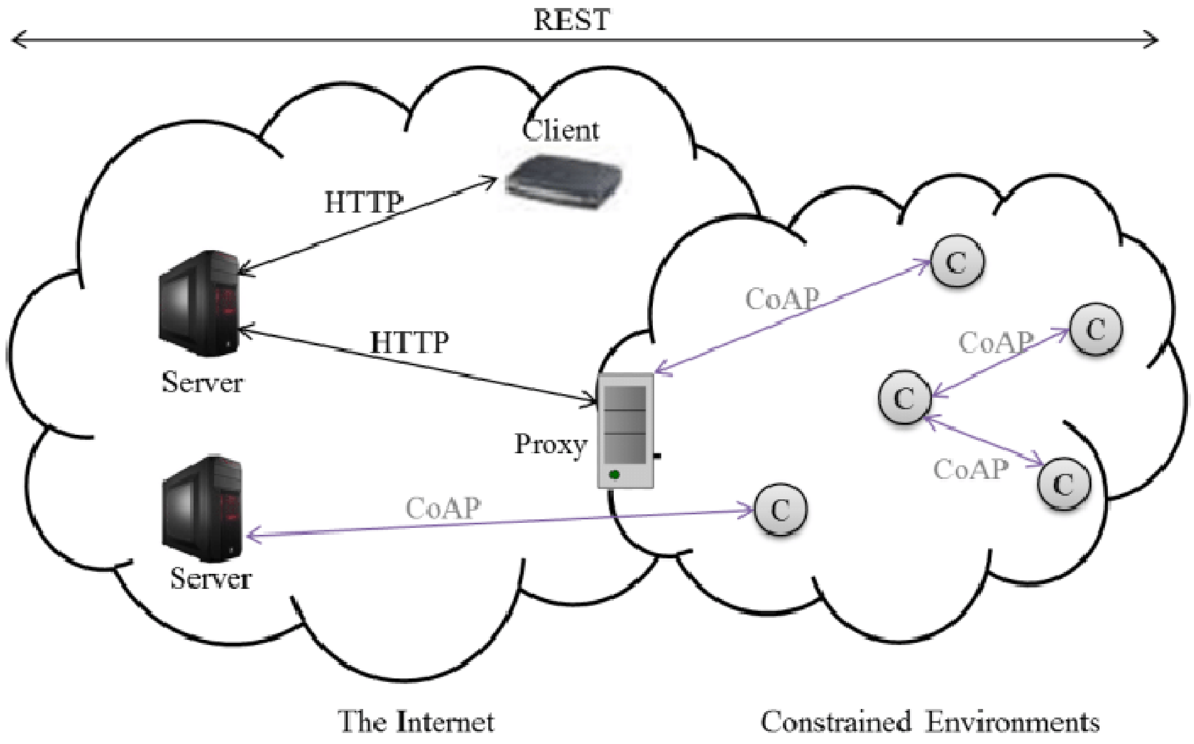
Constrained Application Protocol (CoAP)

- CoAP is designed by IETF (Internet Engineer Task Force) to work in constrained environment.
- It is a one to one communication protocol.
- CoAP is also light weight like MQTT protocol and it uses less resources than HTTP.
- HTTP runs over TCP and is connection oriented while CoAP runs over UDP and it is connection less.



CoAP Architecture

- CoAP is based on the RESTful architecture (Representational State Transfer).
- REST approach ensures the secure, fault tolerant and scalable system.
- The CoAP optimizes the length of the datagram.
- CoAP is connectionless protocol and it requires retransmission support.

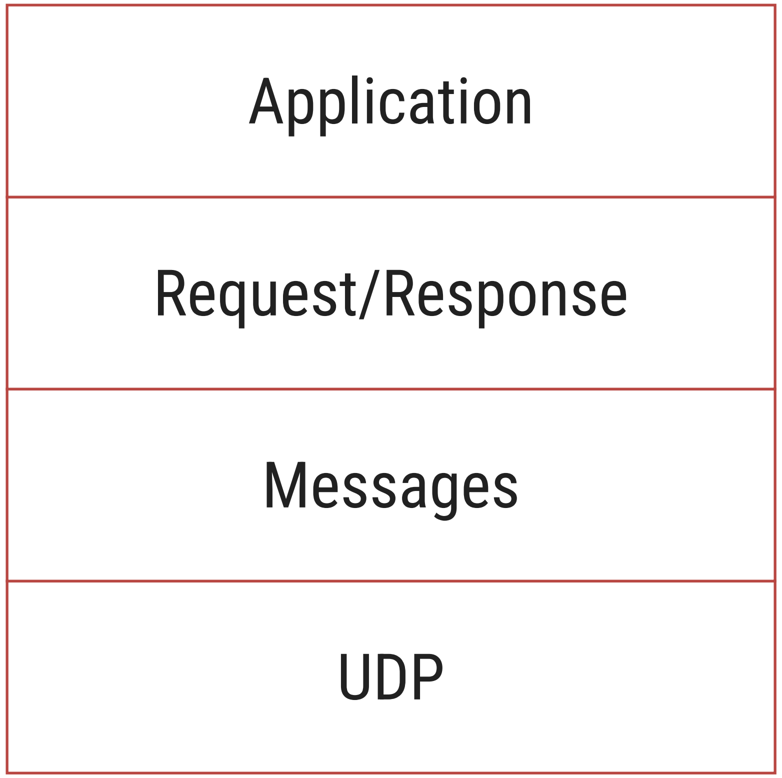


CoAP Layers

- CoAP has four layers:

1. UDP
2. Messages
3. Request-Response
4. Application

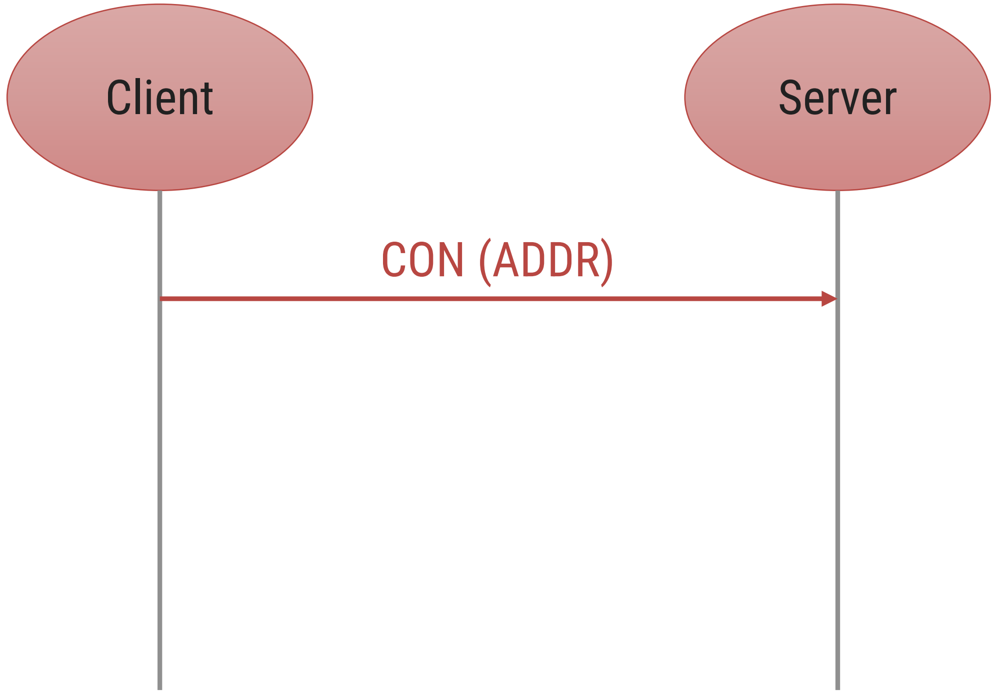
- The message layer is designed to deal with UDP and asynchronous switching.
- The request/response layer concerns communication method and deal with request/response messages.
- In the request/response layer, clients may use GET/PUT/DELET methods to transmit the message.



Message Layer in CoAP

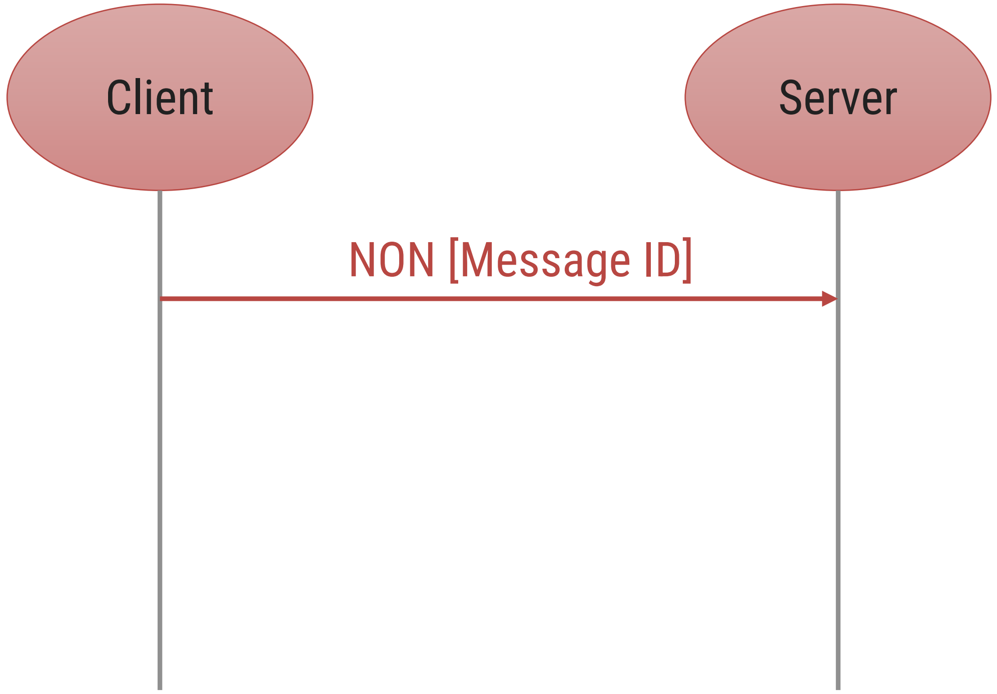
- CoAP message layer supports four types of messages:

1. Confirmable – Reliable Messaging (CON):
   - This is reliable approach because the retransmission of message occurs until the acknowledgement of the same message ID is received.
   - If there is a timeout or fail of acknowledgement, the RST message will be sent from the server as a response.
   - Hence, the client will retransmit the message, this resolves the retransmission and it is considered a reliable approach.



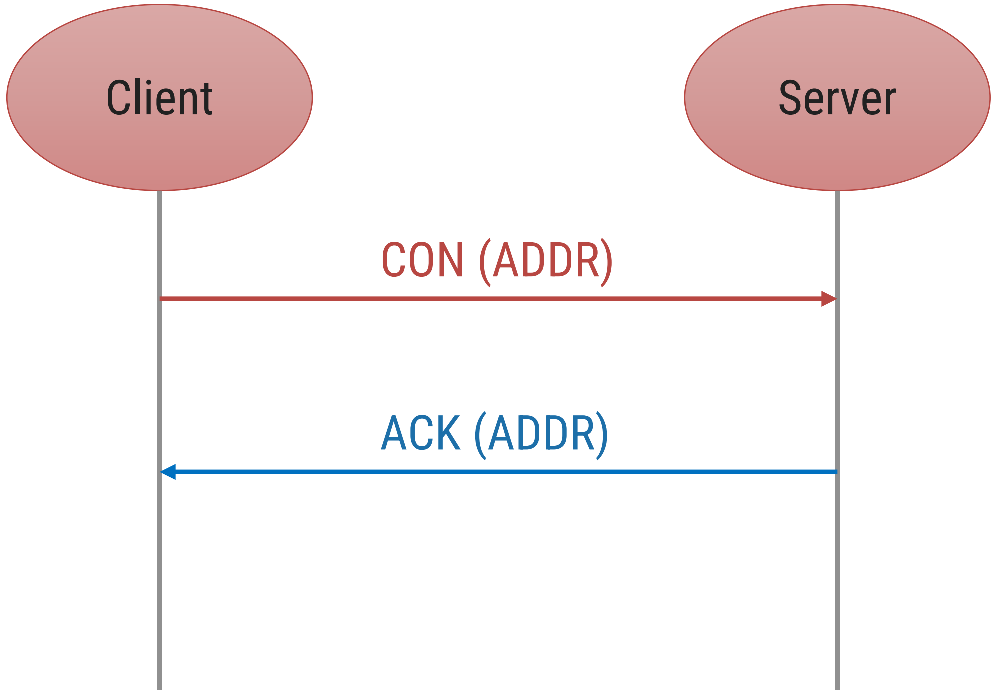
Message Layer in CoAP (Contd.)

2. Non-confirmable – Non-reliable Messaging (NON):
   - In this message transmission style, there is no reliability is ensured.
   - The acknowledgement is not issued by the server.
   - The message has ID for supervision purpose, if the message is not processed by the server it transmits the RST message.



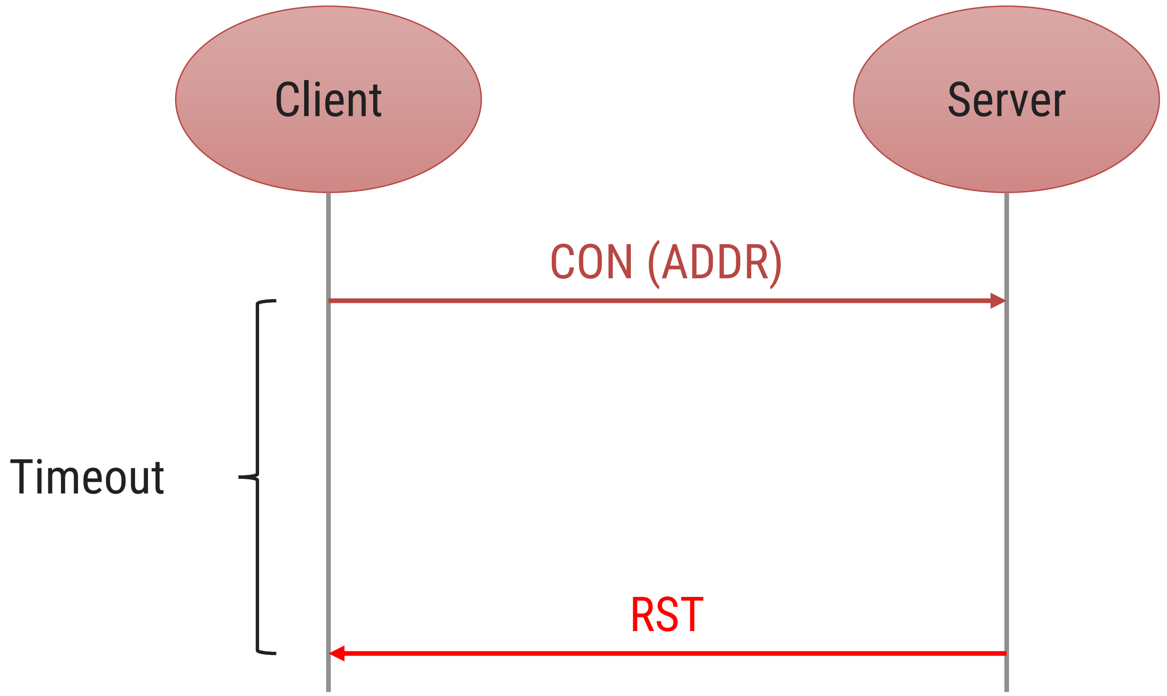
Message Layer in CoAP (Contd.)

3. **Acknowledgement (ACK):**
    - In this messaging, the traditional acknowledgement message sent as usual protocol.
    - We can compare this with regular handshaking scheme.
    - The handshake is automated process that establishes a link for communication before actual data transfer begins.



Message Layer in CoAP (Contd.)

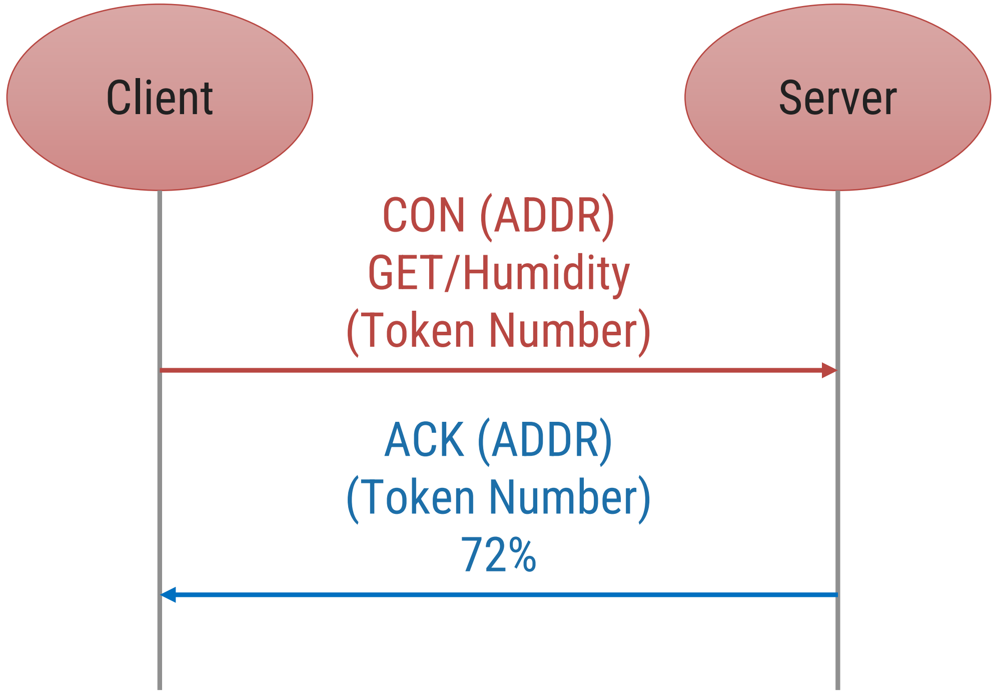
4. Reset (RST):
   - The receiver is expecting the message from sender of a particular message ID.
   - If no message is processed or received before specific amount of time (called timeout), the message called RESET is transmitted from receiver.
   - This message informs the sender that there is a trouble in transmission of messages.

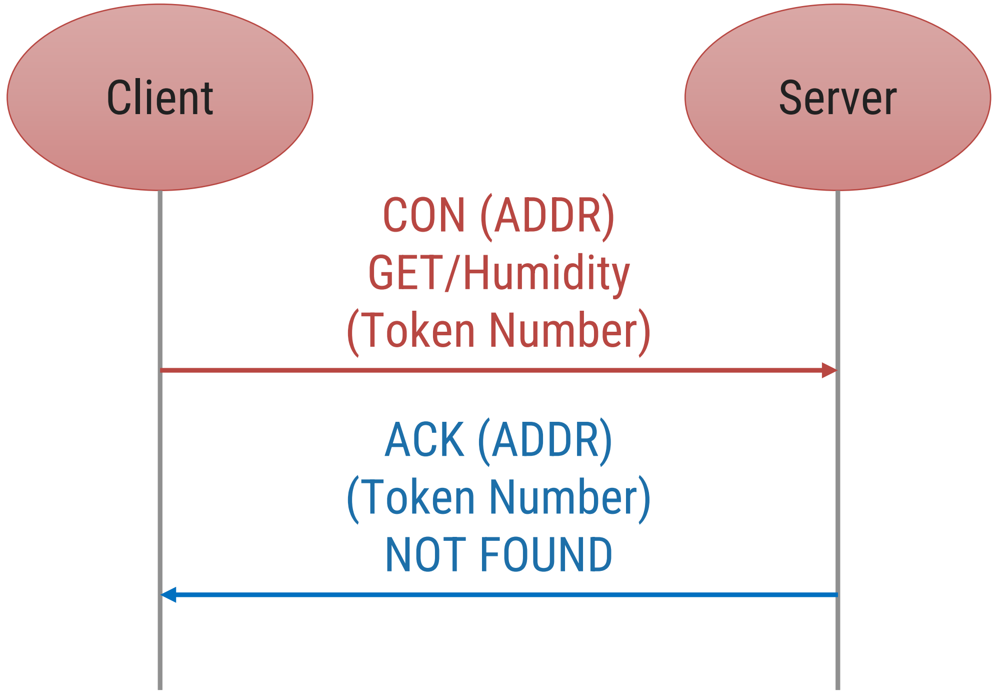


Request-Response Layer in CoAP

- There are three modes in CoAP request-response layer.

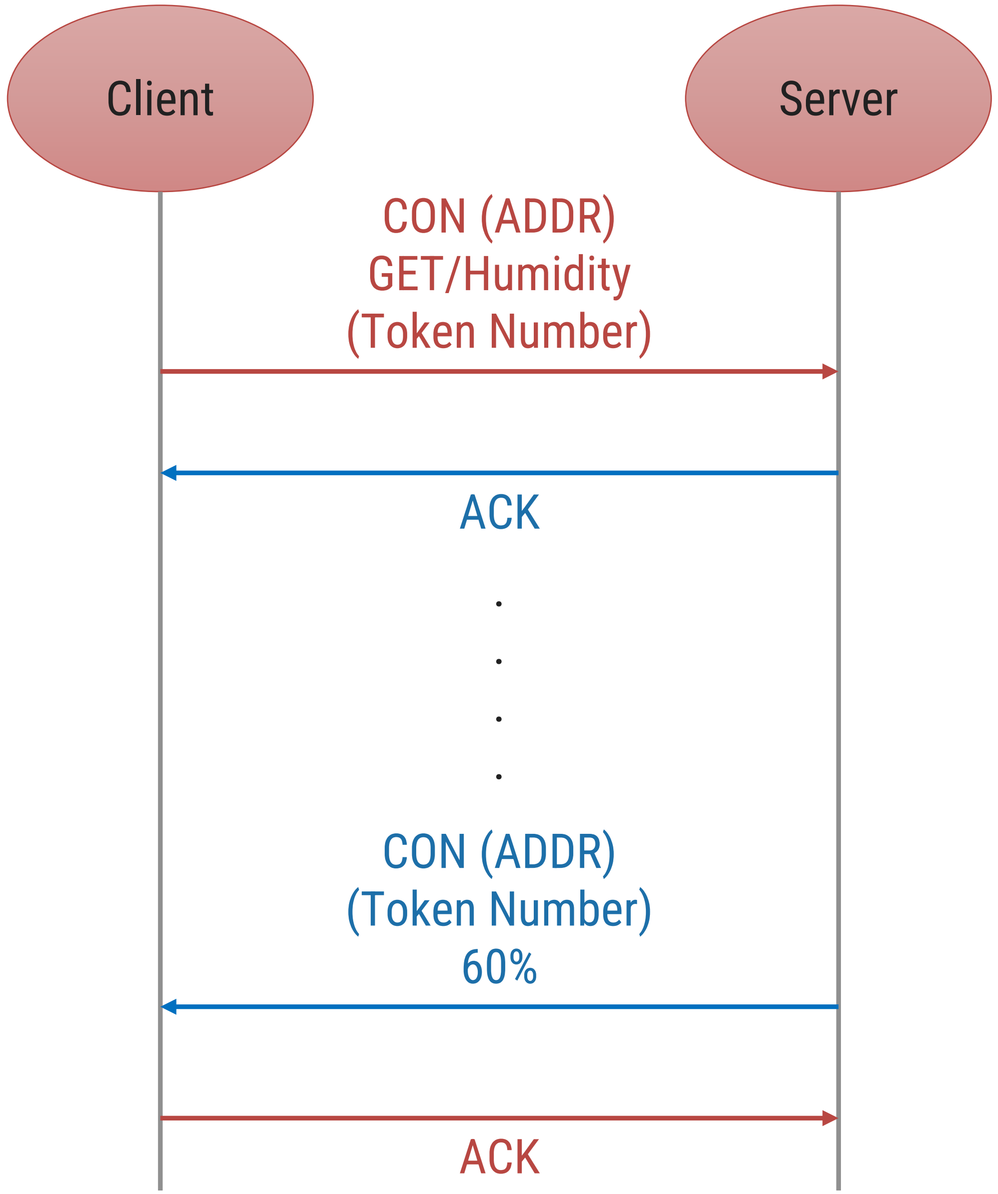
1. Piggy-Backed:
   - In this mode, the client sends the data with particular method (GET, PUT etc.) with token number and CON messaging method.
   - The ACK is transmitted by server immediately with corresponding token number and message.
   - If the message is not received by server or data is not available then the failure code is embedded in ACK.





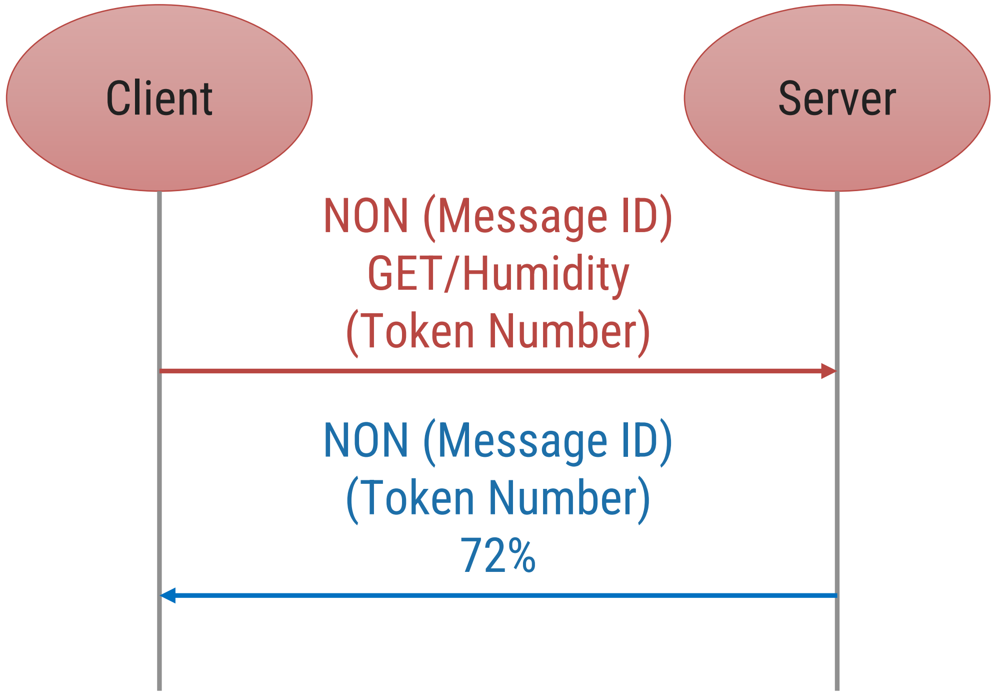
Request-Response Layer in CoAP (Contd.)

2. Separate Response:
   - In this mode, the client sends the data with particular method (GET, PUT etc.) with token number and CON messaging method.
   - If the server is unable to respond immediately, an empty ACK will be reverted.
   - After some time, when the server is able to send the response, it sends a CON message with data.
   - In response to that, ACK is sent back from client to server.



Request-Response Layer in CoAP (Contd.)

3. Non-Confirmable Request and Response:
   - In this mode, the client sends the data with particular method (GET, PUT etc.) with token number and NON messaging method.
   - The server does not give ACK in NON messaging method, so it will send a NON type response in return with token number and its data.



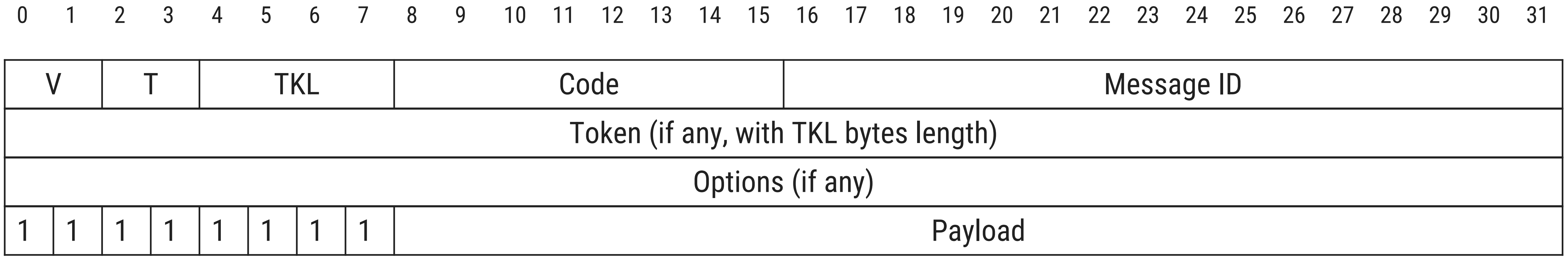
CoAP Message Format

- The CoAP message format is of 16 Bytes consisting various fields.
- V : It refers to version of CoAP. It is two bit unsigned integer.
- T : It refers to message type. It is also two bit unsigned integer.
  - Confirmable (0)
  - Non-Confirmable (1)
  - ACK (2)
  - RESET (3)
- TKL : It refers to the token length and is four bit unsigned integer.
- Code: It refers to the response code.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
V		T		TKL				Code								Message ID															
Token (if any, with TKL bytes length)																															
Options (if any)																															
1	1	1	1	1	1	1	1	Payload																							

CoAP Message Format (Contd.)

- Message ID: It is identifier of the message and to avoid duplication.
- Token : Optional field whose size is indicated by the Token Length field.
- Options : This field is used if any options available and having length of 4 Bytes.
- Payload : This field refers to the payload data where actual data is transmitted.



2. **Advanced Message Queuing Protocol (AMQP) :**
   AMQP is an open standard session layer protocol useful for sending transnational messages that's why mainly used in financial industry. It runs over TCP (Transmission Control Protocol) or UDP (User Datagram Protocol) which is nearly similar to that of MQTT. But only difference is that Broker(Dealer) contains Exchange and Queues as a result it is focused on not losing messages. The exchange receives messages from publishers and distributes them to respective queues. Different subscribers collect topic wise sensory data from respective topic queues.

Parameter	AMQP	HTTP
Full Form	Advanced Message Queuing Protocol.	Hyper Text Markup Protocol.
Developed by	It was developed by JPMorgan Chase.	It was developed by Tim Berners-Lee.
Communication Nature	It has asynchronous communication nature.	It has synchronous communication nature.
Usage	It is easy to setup and manage.	It is user centric and it can be used in every aspect.
Message Delivery	It has guaranteed message delivery.	It has no guarantee for message delivery.
Interface	It provides publish/subscribe interface.	It provides point to point interface.
Fault Tolerance	AMQP protocol can bear the server broke issue on its own.	HTTP protocol is not capable to react to the server breakdown issue.
Segmentation	It has the property of segmentation and can process messages into slots.	It does not has this capability to treat each message as segments.
Protocol Characteristics	It is specific protocol used for specific purposes.	It is general purpose protocol and is used for multiple purposes.
Advantages	It is fast, flexible and cost effective protocol.	It is well known, efficient and multi-purpose protocol.

Parameter	MQTT	HTTP
Abbreviation	Message Queuing Telemetry Transport	Hyper Text Transfer Protocol
Architecture	It works on publish/subscribe model.	It works on request/response model.
Complexity	It has less complexity.	It is more complex.
Runs over	It runs over Transmission Control Protocol.	It runs over Transmission Control Protocol (TCP) and can also adapted to User Datagram Protocol.
Protocol Design	This protocol's design is Data centric.	This protocol's design is Document centric.
Message Size	The message size generated is less as it uses binary format.	The message size generated is more as it uses ASCII format.
Header Size	It is of 2 bytes.	It is of 8 bytes.
Port Number	It works on 1883 port.	It works on 80 or 8080 port.
Data Security	It provides data security with SSL/TLS.	It does not provide security but Https is built for that.

XMPP Protocol

- The full form of XMPP is Extensible Messaging and Presence Protocol.
- It is designed for messaging, chat, video, voice calls and collaboration.
- The fundamental base of this protocol is XML and it is used for messaging services such as WhatsApp.
- The XMPP was originally named as Jabber and later known as XMPP.

XMPP denotation

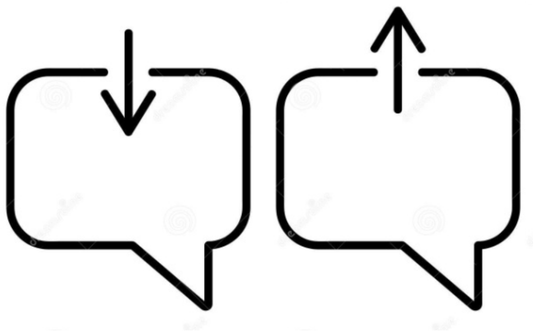
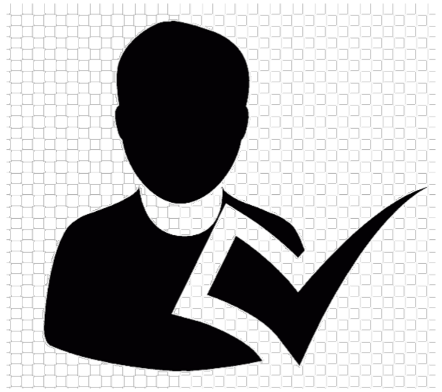
- The denotation for XMPP can be given as following:
- X – X denotes eXtensible.
  - It is considered extensible as it is designed to accept and accommodate any changes.
  - The protocol is defined in open standard and it is extensible because of open standard approach.
- M – M denotes Messaging.
  - XMPP supports sending message to the recipients in real time.
- P – P denotes Presence.
  - It is helpful to identify the status such as “Offline” or “Online” or “Busy”.
  - It also helps in understanding if the recipient is ready to receive the message or not.
- P – P denotes Protocol.
  - The set of standards together acting as a protocol.

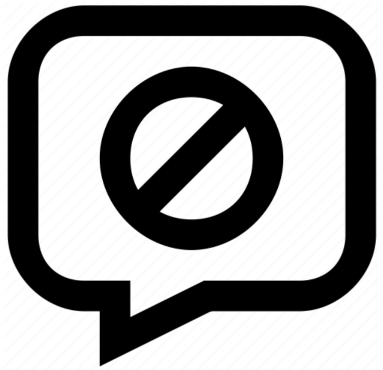
Messenger Requirements

- Any messenger requires the following features inbuilt :

1. Message sending and receiving features.
2. Understanding the presence/absence status.
3. Managing subscription details.
4. Maintaining the contact list.
5. Message blocking services.

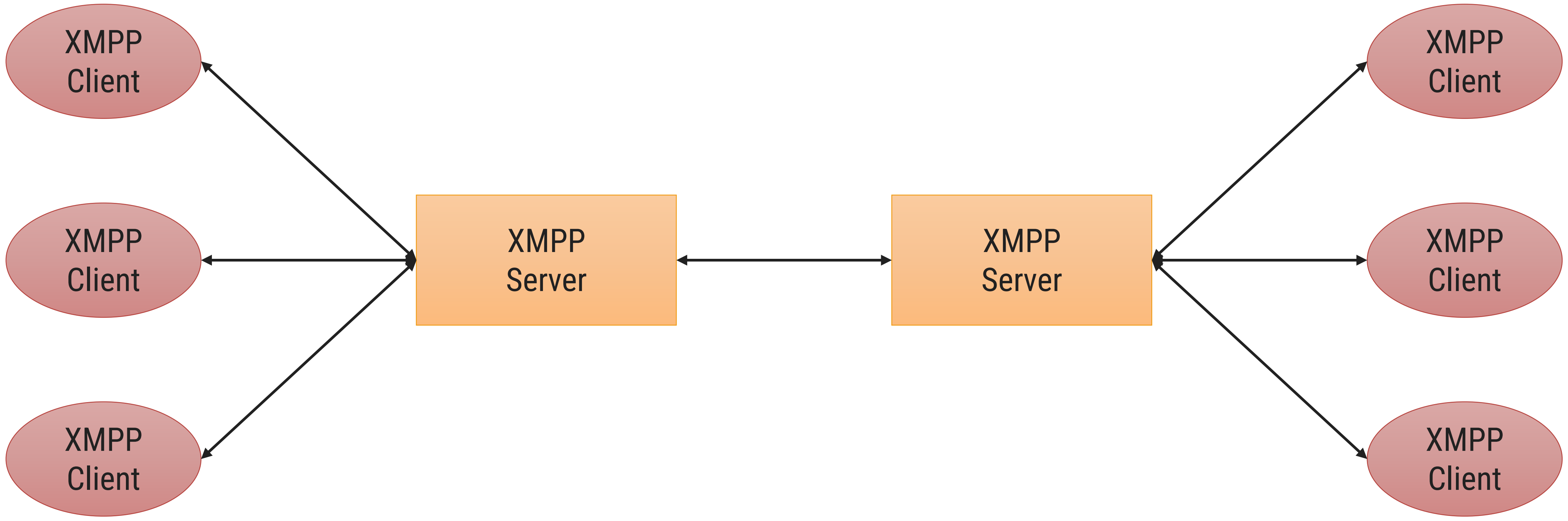
- Note that all the listed features are built in XMPP.



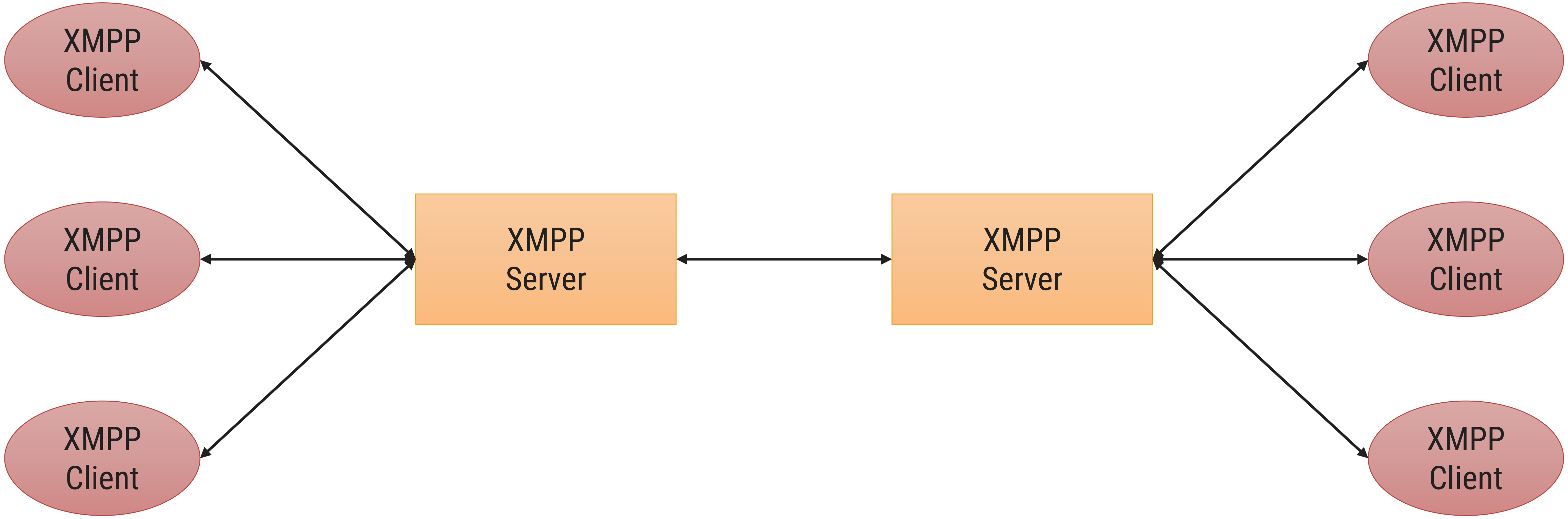
XMPP Architecture

- The architecture of XMPP protocol is shown in the figure :
- The XMPP clients communicate with XMPP server bidirectional.
- There can be any numbers of clients connected to XMPP server.
- The XMPP servers may be connected to the other clients or other XMPP servers.



XMPP Architecture (Contd.)

- The initial version of the XMPP was with TCP using open ended XML.
- After certain period of time, the XMPP was developed based on HTTP.
- The XMPP can work with HTTP is through two different methods Polling and Binding.
- What is Polling and Binding?



XMPP Architecture (Contd.)

- In polling method, the messages stored in the server are pulled or fetched.
- The fetching is done by XMPP client through HTTP GET and POST requests.
- In binding method, Bidirectional-streams Over Synchronous HTTP (BOSH) enables the server to push the messages to the clients when they are sent.

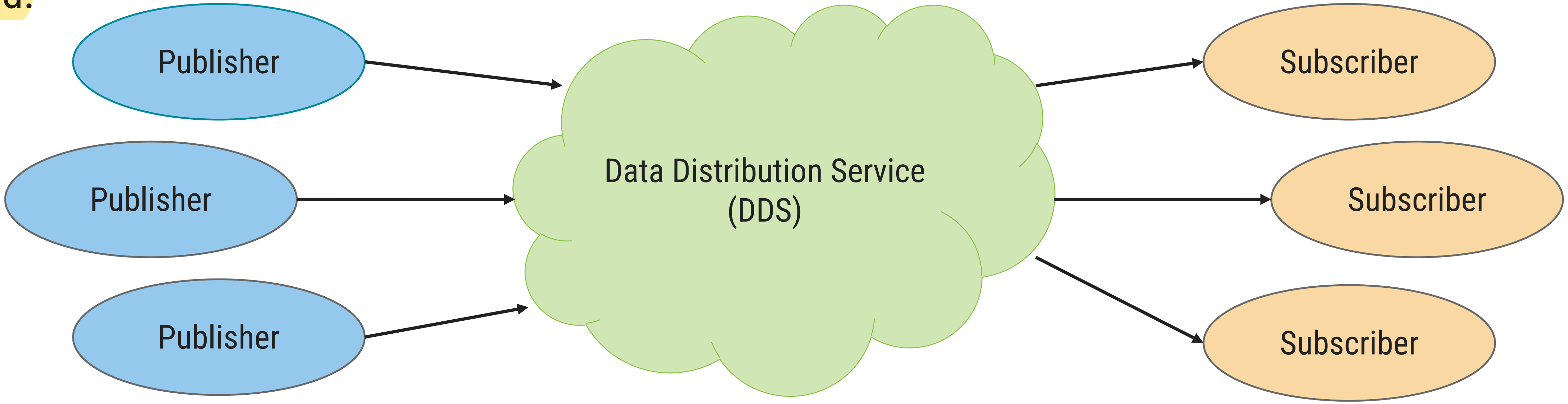
- The binding approach is more effective than polling.
- Also both method uses HTTP, hence the firewall allows the fetch and post activity without any difficulties.

Data Distribution Service (DDS)

- Data Distribution Service (DDS) is one of the protocol being used in the Internet of Things (IoT) applications.
- DDS is brokerless service and the protocol was developed by the Object Management Group (OMG).
- The purpose behind development of DDS is to get better machine to machine communication.
- Like MQTT, DDS is also using publish-subscribe approach.

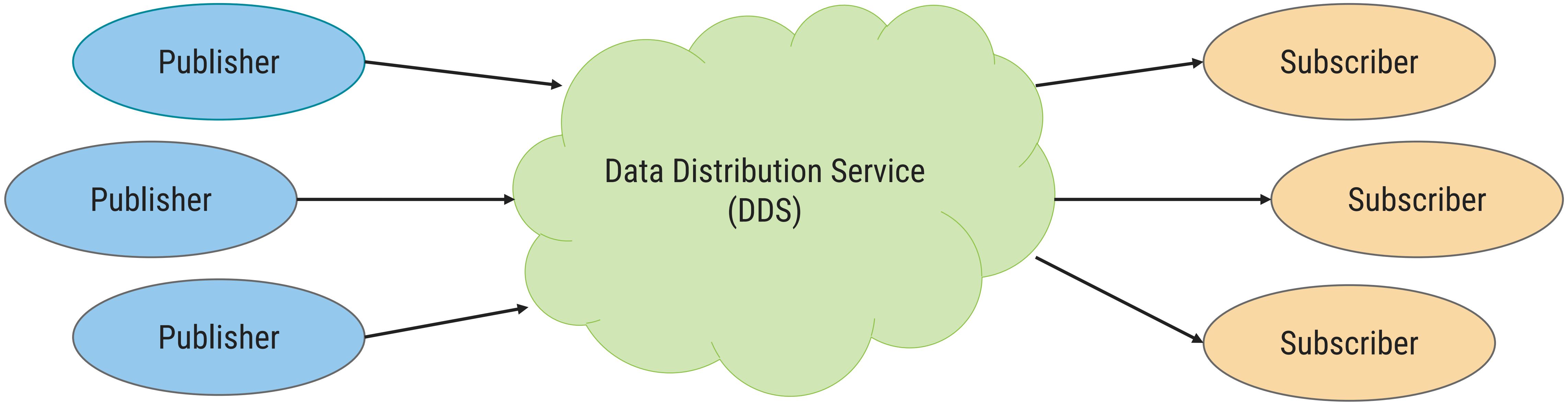
DDS Architecture

- DDS is basically a publish – subscribe model based architecture.
- Everything including sending and receiving data, event updates, command within and among the nodes are all accomplished through the publish – subscribe approach.
- The publisher sends the data to DDS cloud with particular Topic.
- The interested nodes (Subscribers) subscribes the particular Topic and receives data from DDS Cloud.



DDS Architecture (Contd.)

- Here, DDS Cloud has to handle all the attributes of data transfer like addressing, marshalling data, control of delivery, control of flow, etc.
- The marshalling of data means to convert any data or objects into a byte-stream, while unmarshalling is exactly the reverse process.
- This enables the communication between publisher and subscriber working on multiple different platforms.



Transport Protocols

Section - 2

Bluetooth 4.0 (BLE)

- BLE – Bluetooth Low Energy
- BLE is open low energy short range radio communication technology.
- The BLE is also known as Bluetooth Smart.
- This protocol is designed by Bluetooth Special Interest Group (SIG).
- BLE also works same al Bluetooth in wireless Personal Area Network (PAN).
- It can be considered the advanced version of Bluetooth.

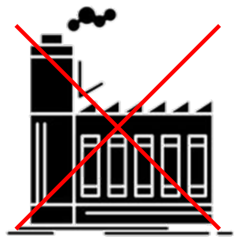
Bluetooth 4.0 (BLE) (Cont.)

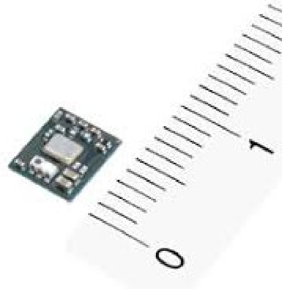
- BLE is superior to classic Bluetooth because of following reasons:
  - It is power saving.
  - It is supported by Android, IOS, Blackberry, etc.
  - It is also supported by computer OS like Windows 8/10, MAC-OS, and Linux variants.
- Because of certain advantages of BLE over classic Bluetooth, it is widely used in the areas like:
  - Healthcare
  - Entertainment
  - Fitness (in form of wearable devices)
  - Proximity related applications
  - Tracking devices

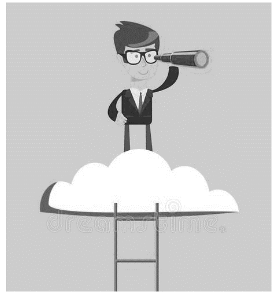
Features of BLE

- The key features of BLE are as following:

1. It is licence free and does not add any overhead costing.
2. There is no restrictions to Manufacturer.
3. BLE modules are inexpensive and affordable.
4. The size of BLE modules are small. So it can be accommodated in any system easily.
5. The power consumption for BLE modules is very less and hence suitable for IoT applications.
6. The range of BLE has very high range compared to classic Bluetooth.

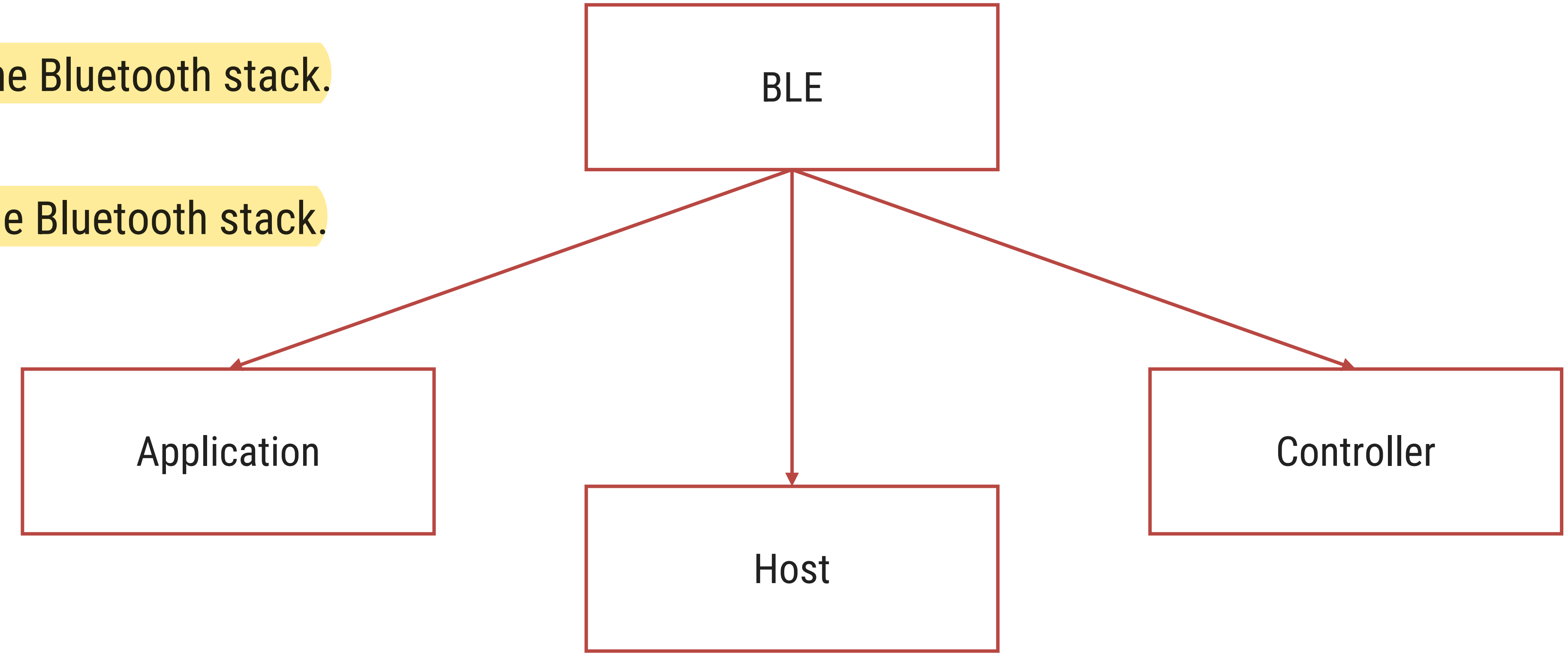


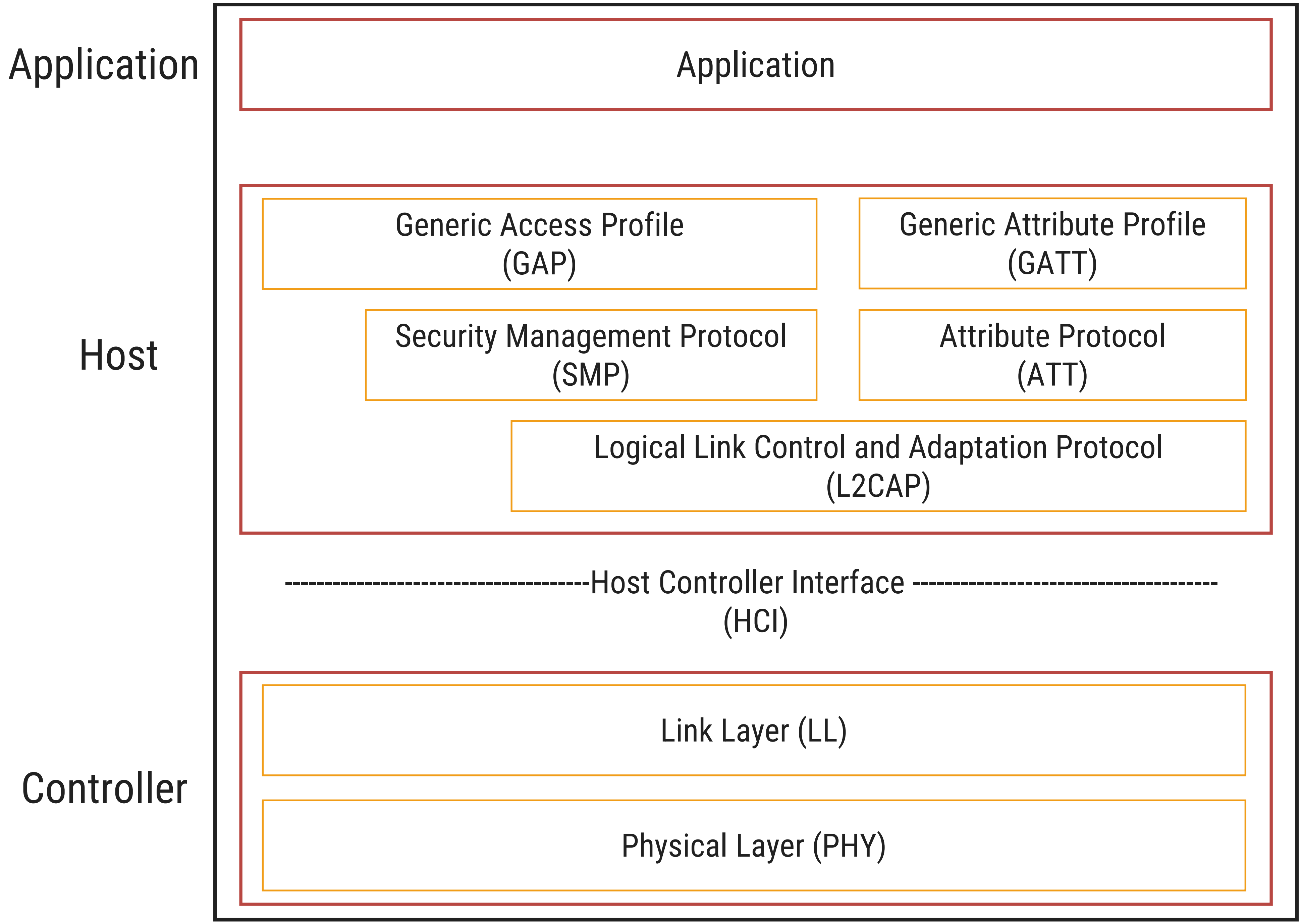




BLE Architecture

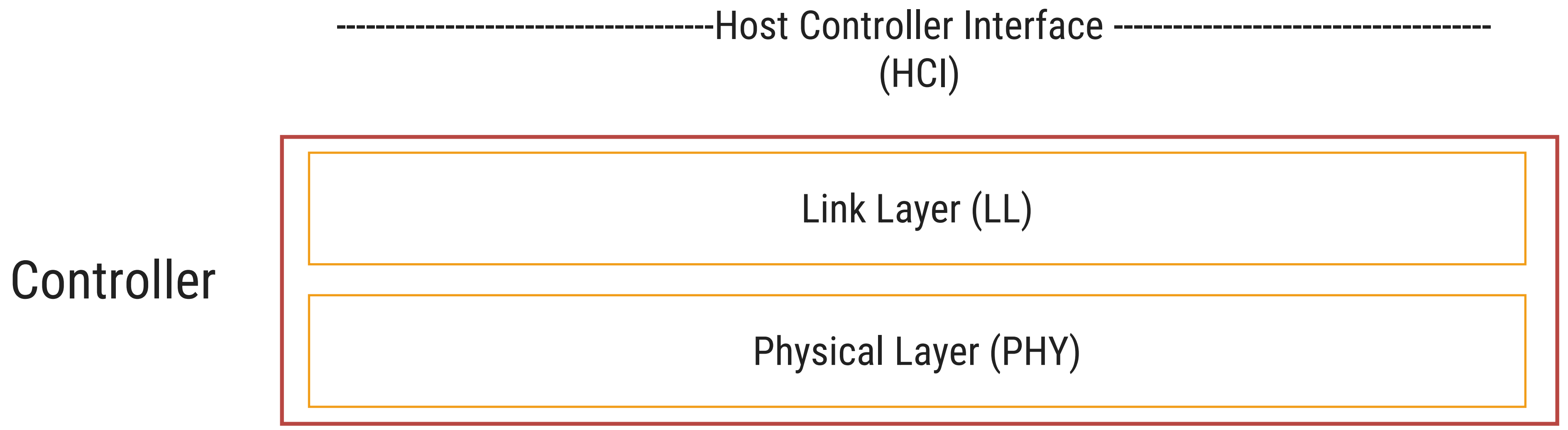
- The BLE architecture has three components.
- Application Block
  - The user application is accommodated in this block.
  - It interacts directly with Bluetooth stack.
- Host Block
  - It is upper layer of the Bluetooth stack.
- Controller Block
  - It is lower layer of the Bluetooth stack.




Application
Application
Host
Generic Access Profile (GAP)
Generic Attribute Profile (GATT)
Security Management Protocol (SMP)
Attribute Protocol (ATT)
Logical Link Control and Adaptation Protocol (L2CAP)
------------------------------------Host Controller Interface ------------------------------------ (HCI)
Controller
Link Layer (LL)
Physical Layer (PHY)

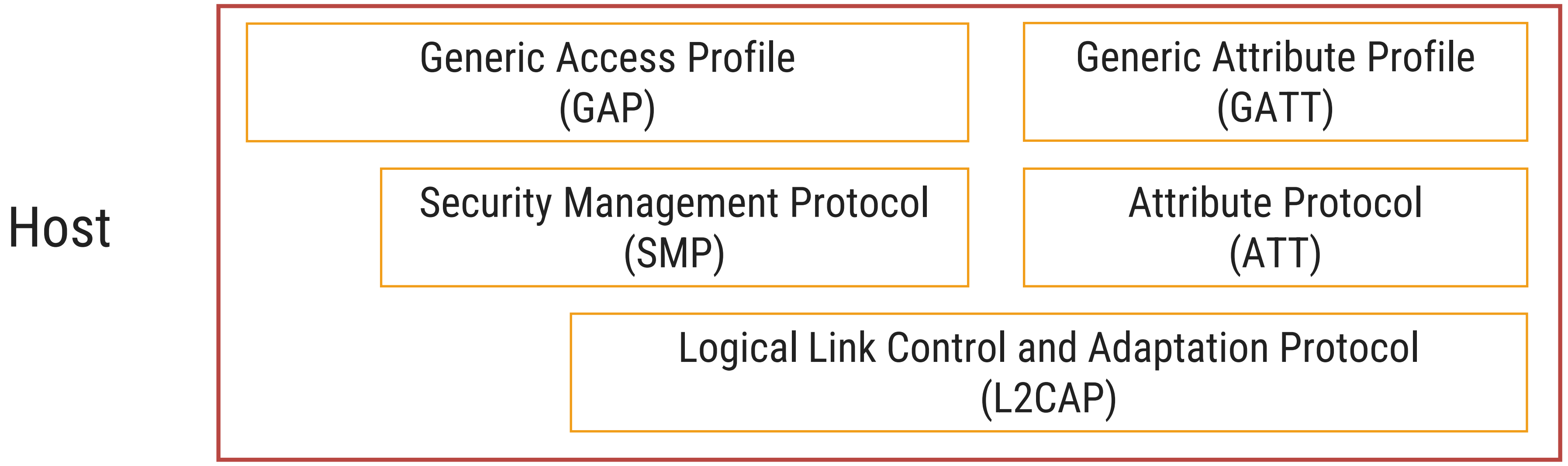
BLE Architecture

1. Controller : It has three components.
   a) Host Controller Interface (HCI) : is used to enable interoperability between hosts and controllers assembled by different manufacturers
   b) Link Layer (LL) : It defines the packet structure
   c) Physical Layer (PHY) : It takes care of the transmission/reception, modulation/demodulation and analog-to-digital conversion.


--------------------------------------Host Controller Interface --------------------------------------
(HCI)
Controller
Link Layer (LL)
Physical Layer (PHY)

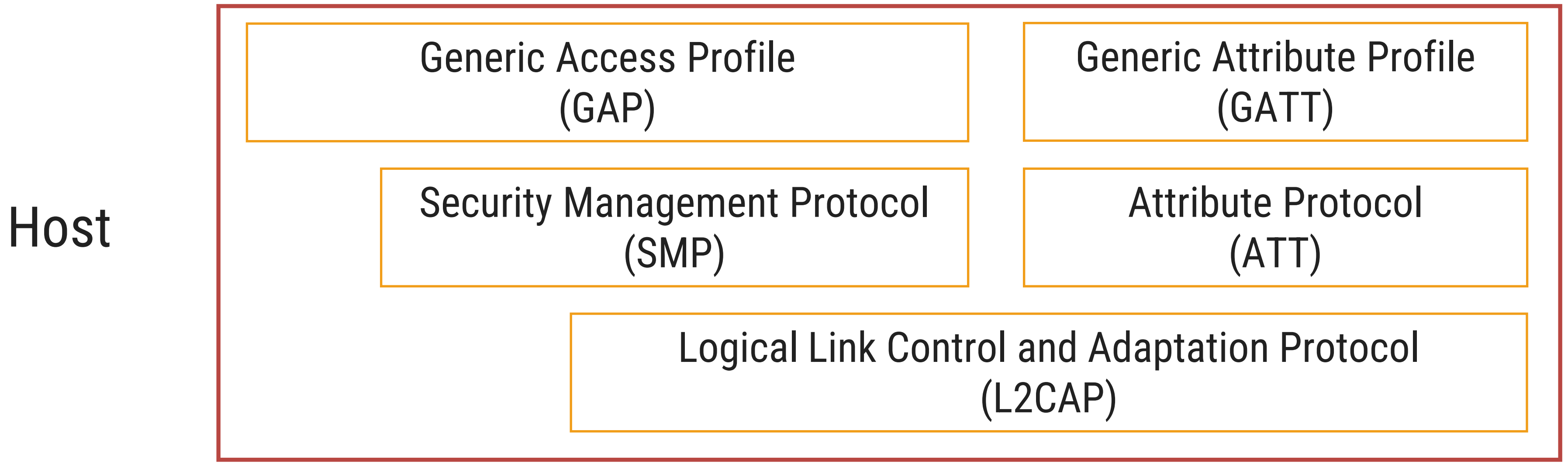
BLE Architecture

2. Host : It has following components with functionalities
   a) Generic Access Profile (GAP) : It takes care of the device discovery, connection establishment, connection management and security.
   b) Generic Attribute Profile (GATT) : It is responsible for the data exchange process. Whenever there is a need to push data and to read or write data, the generic guidelines with respect to the process are governed by GATT.
   c) Attribute Protocol (ATT) : It is the protocol for accessing data.


Host
Generic Access Profile (GAP)
Generic Attribute Profile (GATT)
Security Management Protocol (SMP)
Attribute Protocol (ATT)
Logical Link Control and Adaptation Protocol (L2CAP)

BLE Architecture

2. Host : It has following components with functionalities
   d) Logical Link Control and Adaptation Protocol (L2CAP) : It is responsible for fragmentation and de-fragmentation of the application data. Also, it is responsible for the multiplexing and de-multiplexing of channels over the shared logical link.
   e) Security Manager (SM) : It takes care of pairing, authentication, and encryption. Everything related to security is taken care here.
   f) Host Controller Interface (HCI) : The functionality remains the same as that in the controller. It is used to enable interoperability between hosts and controllers assembled by different manufacturers.

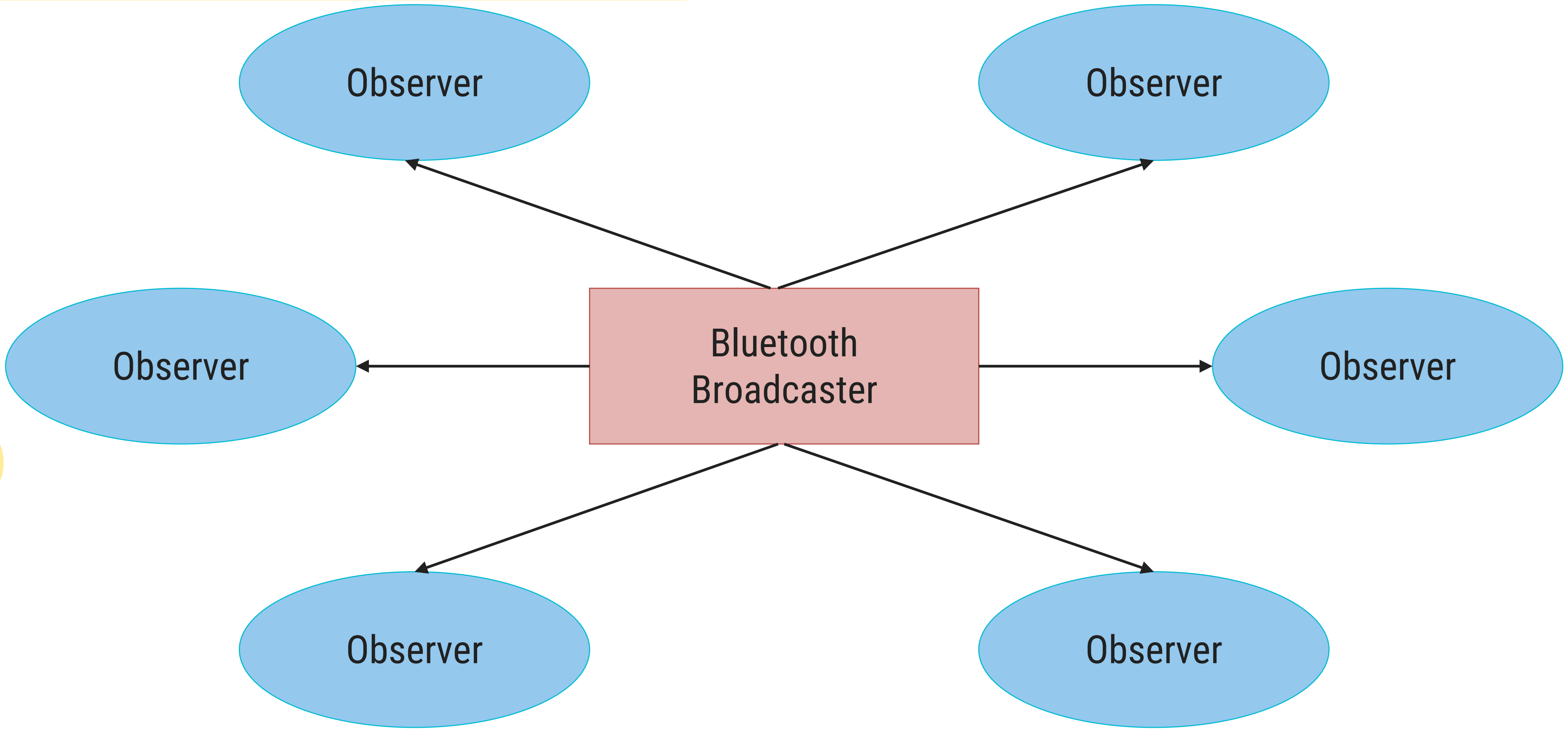

Host
Generic Access Profile (GAP)
Generic Attribute Profile (GATT)
Security Management Protocol (SMP)
Attribute Protocol (ATT)
Logical Link Control and Adaptation Protocol (L2CAP)

BLE Architecture

3. Application Layer :
   - ➥ The working of application layer in BLE is same as in OSI model. It is the top most layer in BLE and this layer is responsible for user interface (UI), logic and data handling.

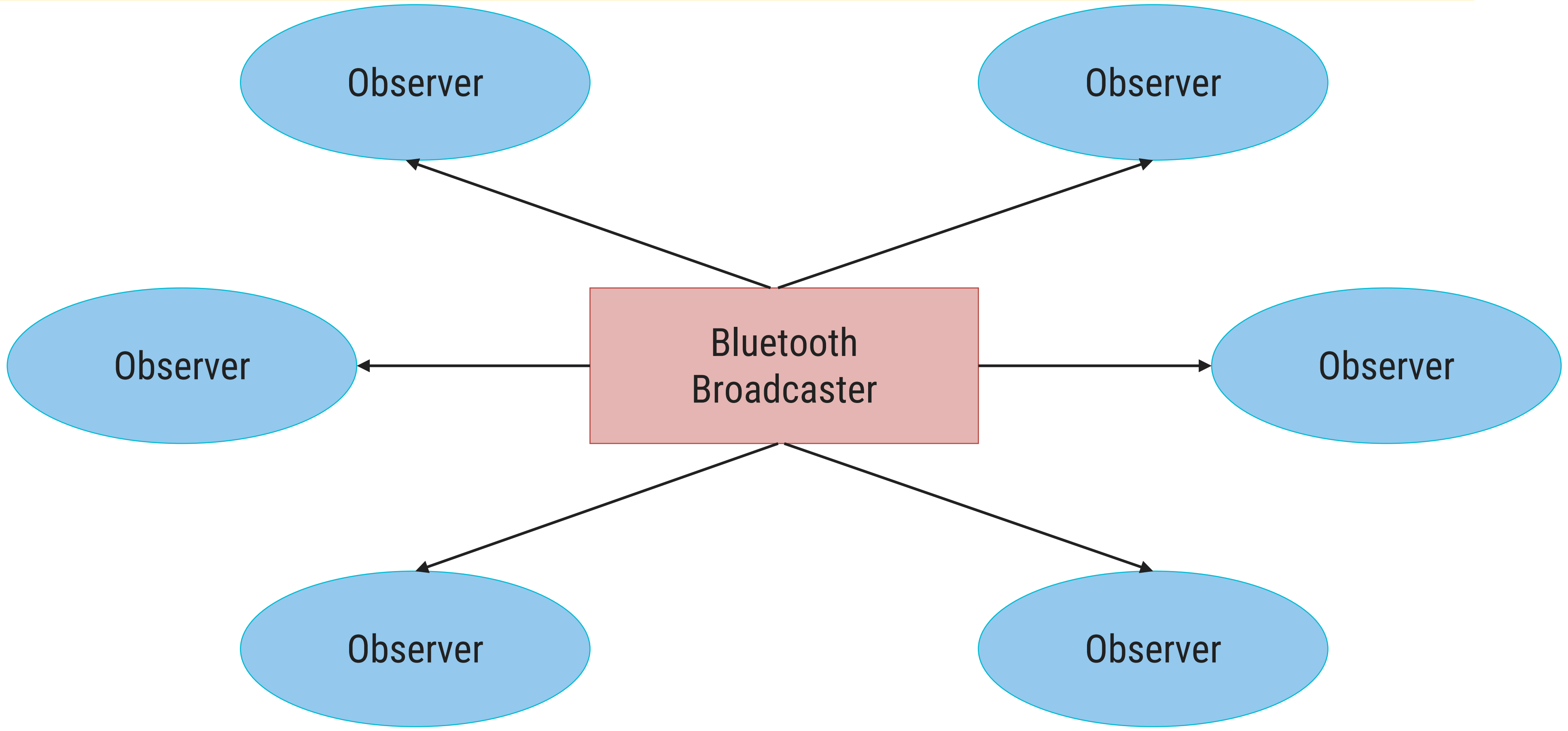
BLE Topology

- Broadcasting
  - The meaning of broadcasting is to send a message to more than one recipient. The same concept is followed in BLE.
  - The broadcast method will send the data in “one way” only.
  - The receiver in the vicinity of the broadcaster will receive the message or data.
  - There are two parts in broadcasting topology. Broadcaster and Observer.



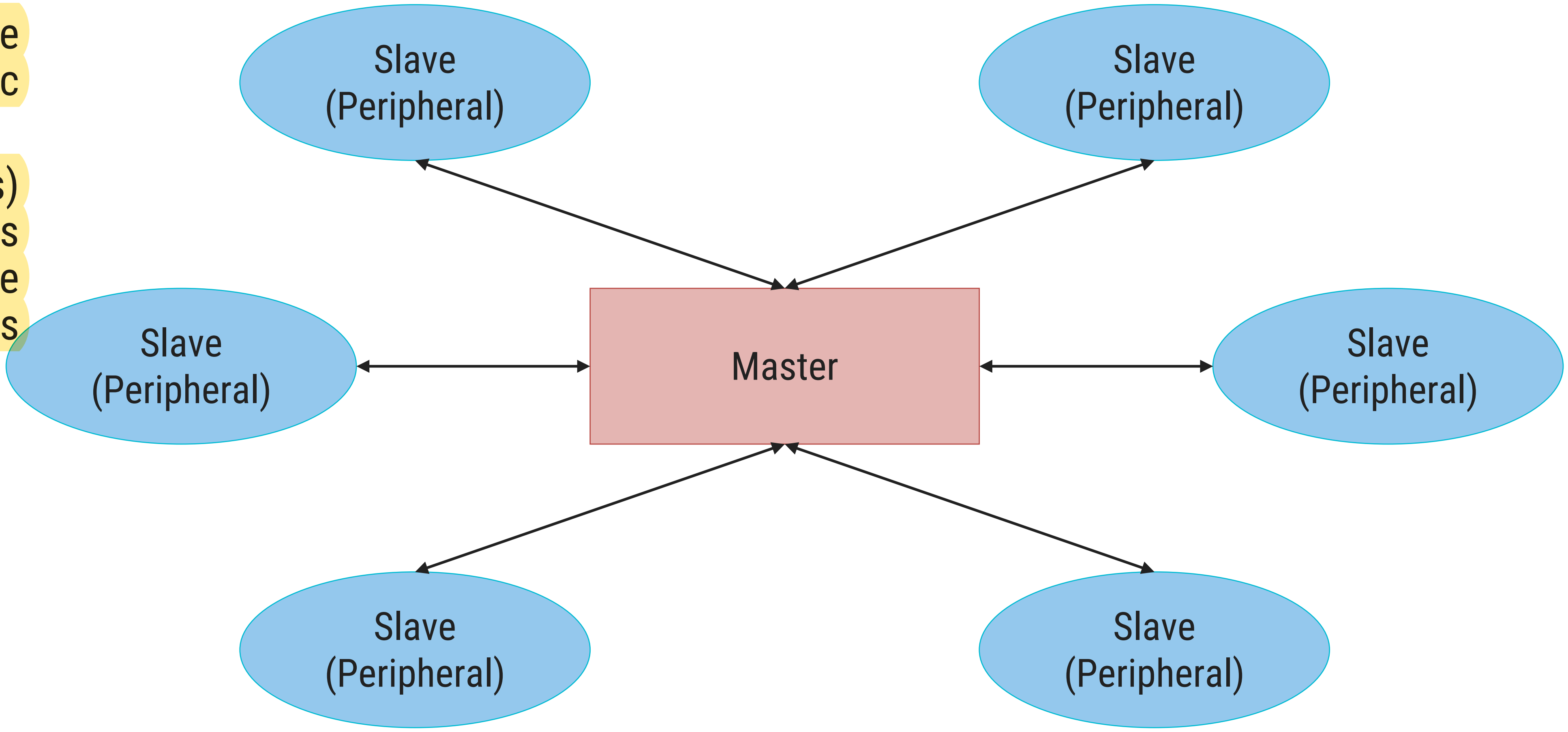
BLE Topology

- Broadcasting
  - Broadcaster : The broadcaster sends a "non-connectable" advertising packets in frequent intervals to all those who are willing to collect which is similar to a radio broadcast.
  - Observer : The observer will keep scanning predefined frequency to receive any non-connectable packets..



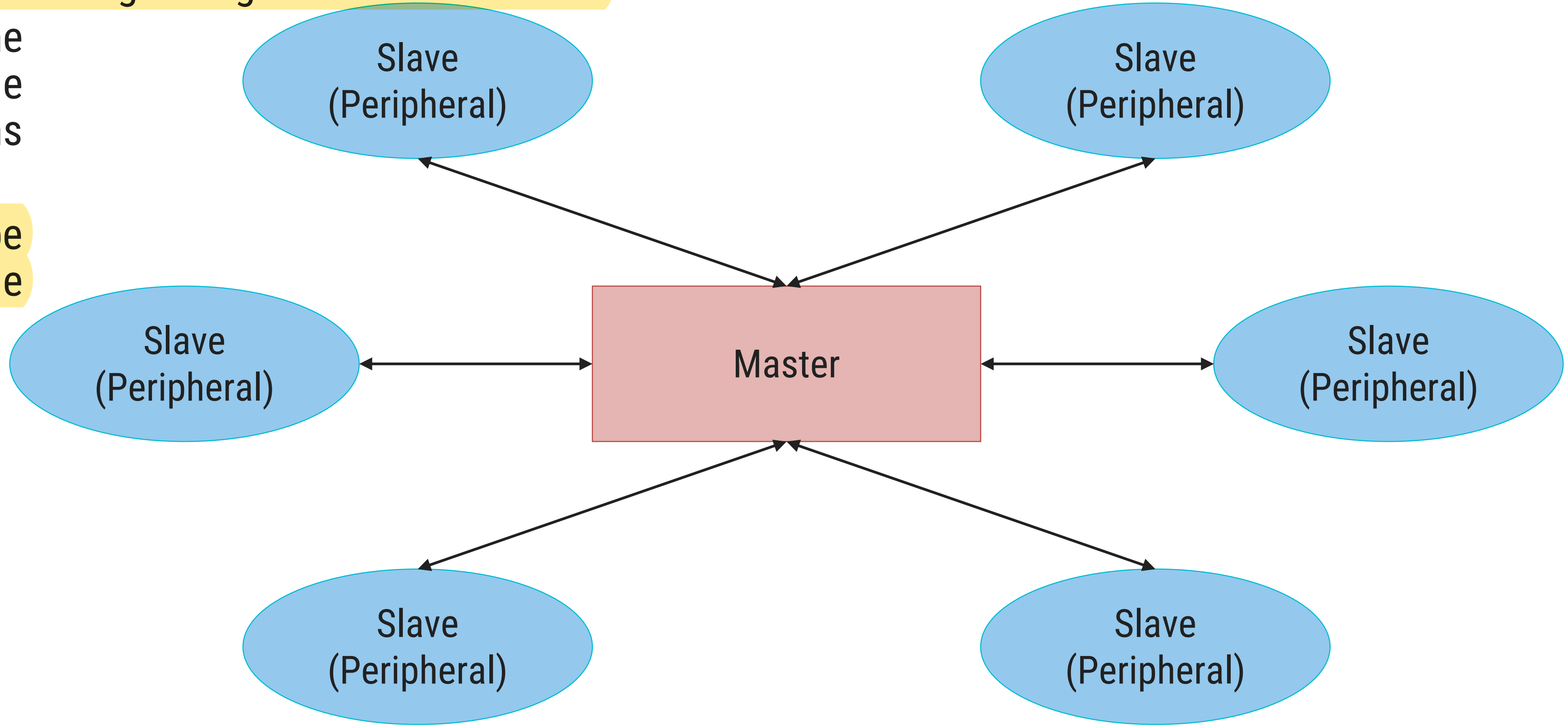
BLE Topology

- Connections
  - The connections topology works on the concept of master-slave configuration.
  - The master will try to find out the connectable advertising packets. Once the advertisement is found, it will initiate the connection to the slave.
  - The master will take care of periodic data exchange.
  - The slave(s) continuously sends the connectable advertising packets at regular interval.



BLE Topology

- Connections
  - Once the slave is connected, the event is called connection event.
  - In this topology, the device will work whenever required. For the remaining time the device is in sleep mode.
  - This provides power saving during the communication.
  - However, during the sleep mode also the connection remains present.
  - But the data will be transferred after the device invokes.

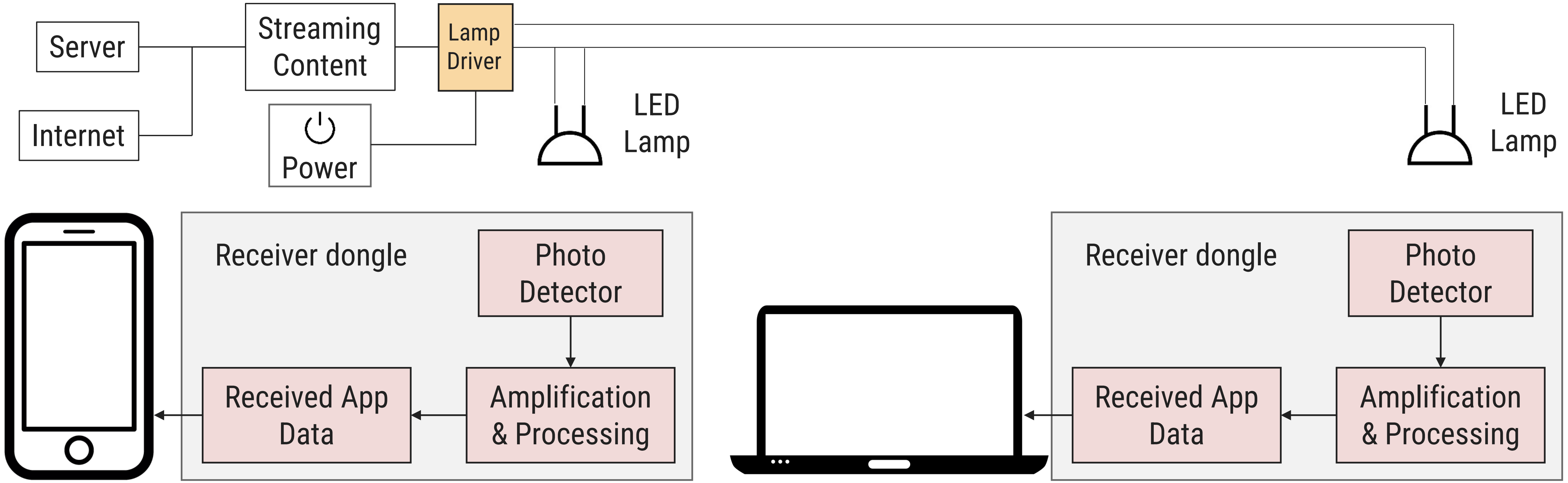


Light Fidelity (Li-Fi)

- The Li-Fi is fastest communication protocol right now among all existing protocols.
- The symbol of Li-Fi is as shown in the figure.
- It gives the following solutions to the problems which one can face during the use of the Wi-Fi.
  - More security than Wi-Fi.
  - More available bandwidth to the connected devices or equipments.
  - Less/No congestion when more number of devices connected.
- Note that the Li-Fi has phenomenal speed of 224GBps which is fastest among all.

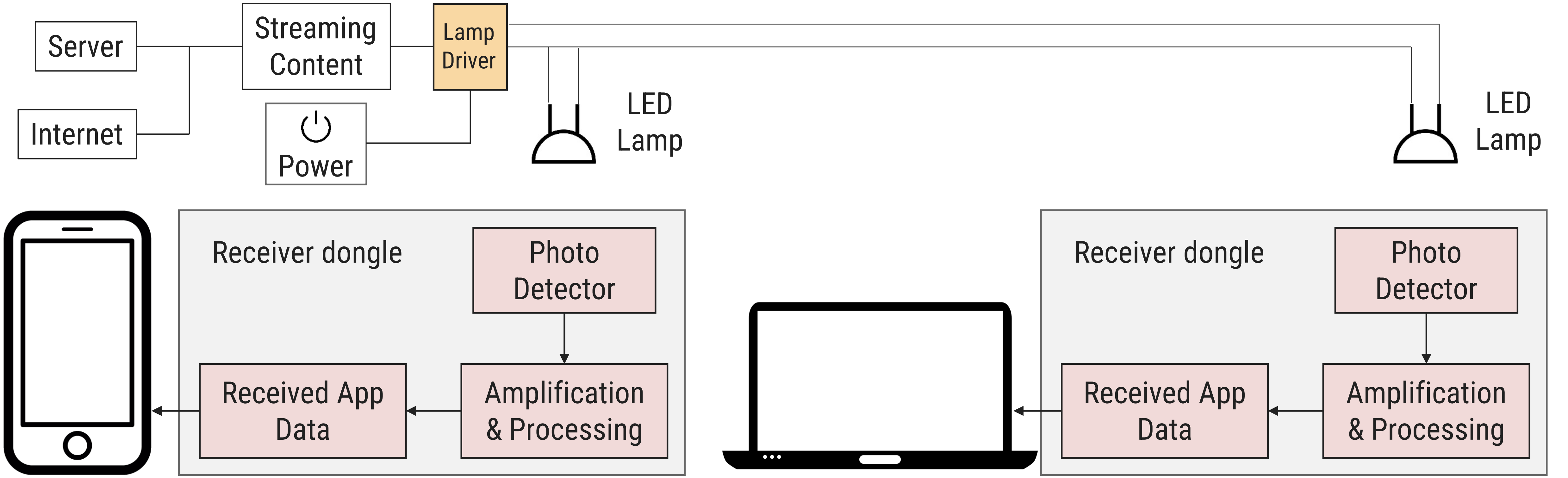
Li-Fi Working

- The server wants to transmit the data to the receiver with very high speed. The server is either connected to internet or intranet.
- The server connected to the internet generates a data which is converted into streaming content.
- The streaming content is given to Lamp driver which is powered by external source.



Li-Fi Working

- The LED lamp is turned on and off at very high speed that human eyes can not capture it.
- The light is received by the photo detector and it is passed for demodulation process and amplification to generate the data stream sent by transmitter.
- The received data stream is further given to receiver (e.g. mobile, PC etc.) via receiver app. All components working for receiving the data combined known as receiver dongle.



Li-Fi – Advantages & Disadvantages

- Advantages
  - Li-Fi is extremely secure as the light can not penetrate through walls and no data hijacking is possible.
  - Li-Fi is fastest and effective.
  - It is an effective alternate to RF communication.
  - It is inexpensive as the light is generated through LED lamps.

- Disadvantages
  - It can be implemented in short range as the presence of walls or any other object will become the interruption in communication.
  - The required time for set up is higher.
  - There is no clarity of bidirectional communication means how to send data from mobile or PC back to server.

Comparison	Li-Fi	Wi-Fi
Full form	It stands for Light Fidelity.	It stands for Wireless Fidelity.
Invented/Coined	Coined by Prof. Harald Haas in 2011.	By NCR corporation in 1991.
Operation	It transmits data using light by the help of LED bulbs.	It transmits data using radio waves using wifi router.
Technology	Present IrDA compliant devices	WLAN 802.11/b/g/n/ac/d standard compliant devices
Data Transfer Speed	About 1 Gbps	Ranges from IS0Mbps to maximum of 2Gbps
Standard	IEEE 802.15.7	IEEE 802.11
Privacy	Light is blocked by the walls hence provide more secure data transfer.	Walls cannot block radio waves so we need to employ more techniques to achieve secure data transfer.
Bandwidth	Availability of unlimited bandwidth.	Availability of limited bandwidth.
Frequency of operation	10. 000 times frequency spectrum of the radio	2.4Ghz. 4.9Ghz and SGhz
Coverage Distance	About 10 meters	About 32 meters(vary based on transmit power and antenna type)
Power Consumption	Power Consumption is Low.	Power Consumption is High.
Data density	Work with the high dense environment	Work in a less dense environment due to interference-related issues.
Cost	Low	High
Bare minimum Components used	LED bulb. LED driver and photo detector	Routers. Modems and access points

Merits	• Low cost because of VLC technology used which in turn uses highly efficient LED bulbs. • Faster transmission speed as light travels at fast speed. • Less interference. ability to travel across salt water. and ability to operate in dense areas. • Prevents unauthorized access because of the no penetration feature of visible light from the opaque walls.	• WiFi eliminated the need for direct physical connections to access local networks and the internet without requiring direct physical connections to servers. Users are granted access from anywhere within range because WiF, connections are wireless. • Easy to implement as you need to contact only an Internet Service Provider (ISP) to provide an internet connection and a WiFi Router to give an access point.
Demerits	• The infrastructure required to implement LiFi technology on a large enough scale is still essentially nonexistent because it is still an emerging concept. • To enable network access. the light source must always be on. • It becomes extremely vulnerable to outside interference because it uses light to communicate data. • Users are made stationary when using LiFi devices because of the short signal range emitted by light bulbs.	• WiFi networks are notoriously insecure. Its lack of security is primarily due to its large signal range. • More interference, inability to travel over salt water, and preference for less populated areas. • Signals are highly unreliable. This Is because of the fact that radio frequencies are still susceptible to numerous external interferences. Due to this much range of connection issues, including weak signals. poor reception, and even connection loss can occur.
Applications	Used in airlines. undersea exploration etc.	Used for internet browsing with the help of Wifi hotspot.
Broadcast	Li-Fi uses visible light with special chips and sensors to broadcast its signal	WIFI uses radio *waves* to broadcast its signal.

Addressing & Identification Protocols

Section - 3

Internet Protocol Version 4 (IPv4)

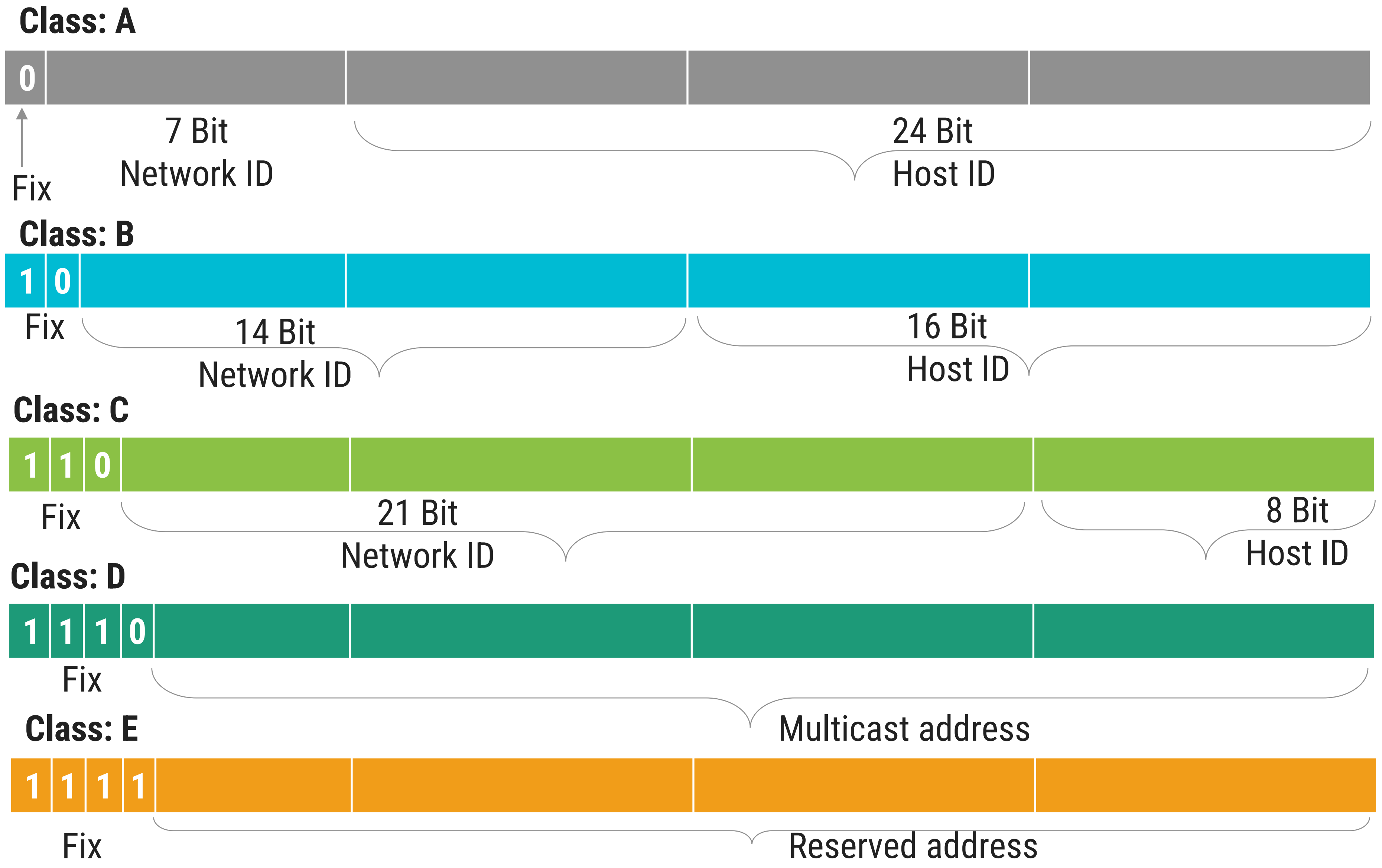
- IP addresses are useful in identifying a specific host in a network.
- IP addresses are 32 bit numbers which are divided into 4 octets. Each octet represents 8 bit binary number.
- Below is an example of an IP address:

10101100	00010000	11111110	00000001
172	**16**	**254**	**1**

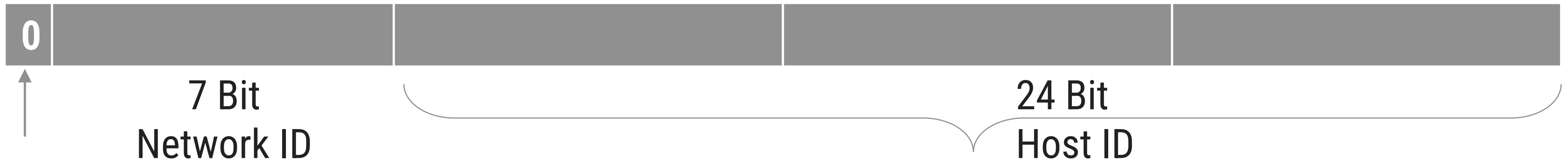
IP addresses are divided into 2 parts:
Network ID & Host ID

<NID> <HID> = IP Address

Classification of IP Address



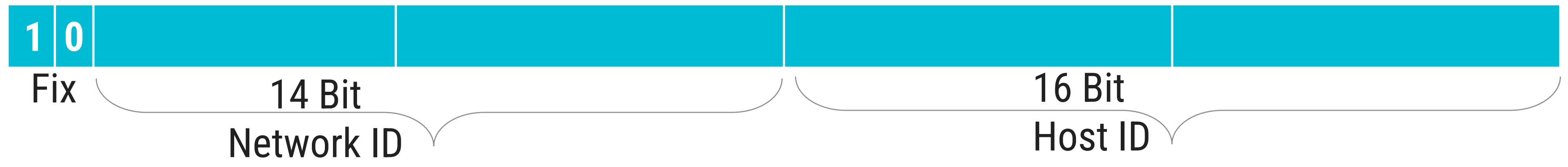
Class A



- Only 126 addresses are used for network address.
- All 0’s and 1’s in Network-ID are dedicated for special IP address. So, total number of IP address in class A can be represented as following.
- The range of Class A is 0.0.0.0 to 127.255.255.255

0.0.0.0	Special IP Address
00000001.0.0.1 1.0.0.2 1.0.0.3 . . . 126.255.255.254	$2^{24} - 2$ are Host IP
127.255.255.255	Special IP Address – Loopback

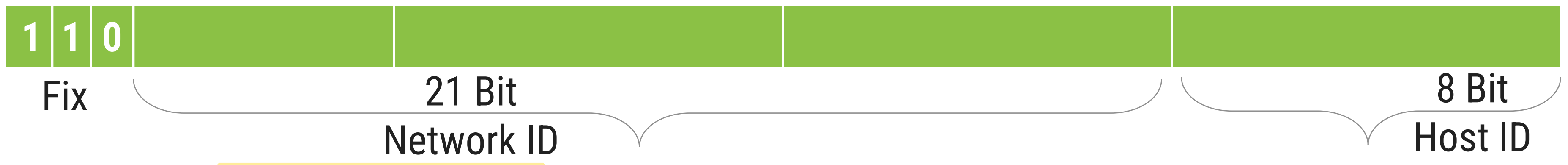
Class B



- No special network address here. All are usable.
- The total number of hosts connected in network can be given as following:
- It is used in medium size network.
- The range of Class B is 128.0.0.0 to 191.255.255.255

128.0.0.0	Special IP Address
10000001.0.0.1 130.0.0.2 130.0.0.3 . . . 190.255.255.254	$2^{16} – 2$ are Host IP
10111111.255.255.255	Special IP Address – Loopback

Class C



- This class is used in small networks.
- The calculation for number of hosts can be given as following:
- Normally, the LANs are configured with Class C.
- The range for Class C is given as 192.0.0.0 to 223.255.255.255

192.0.0.0	Special IP Address
11000001.0.0.1 194.0.0.2 194.0.0.3 . . . 222.255.255.254	$2^8 - 2$ are Host IP
11011111.255.255.255	Special IP Address – Loopback

Class D

- Very first four bits of the first octet in Class D IP addresses are set to 1110, giving a range of:

11100000 – 11101111
224 – 239

- Class D has IP address rage from 224.0.0.0 to 239.255.255.255.
- Class D is reserved for Multicasting.
- In multicasting data is not destined for a particular host, that is why there is no need to extract host address from the IP address, and Class D does not have any subnet mask.

Class E

- This IP Class is reserved for experimental purposes only for R&D or Study.
- IP addresses in this class ranges from 240.0.0.0 to 255.255.255.254.
- Like Class D, this class too is not equipped with any subnet mask.

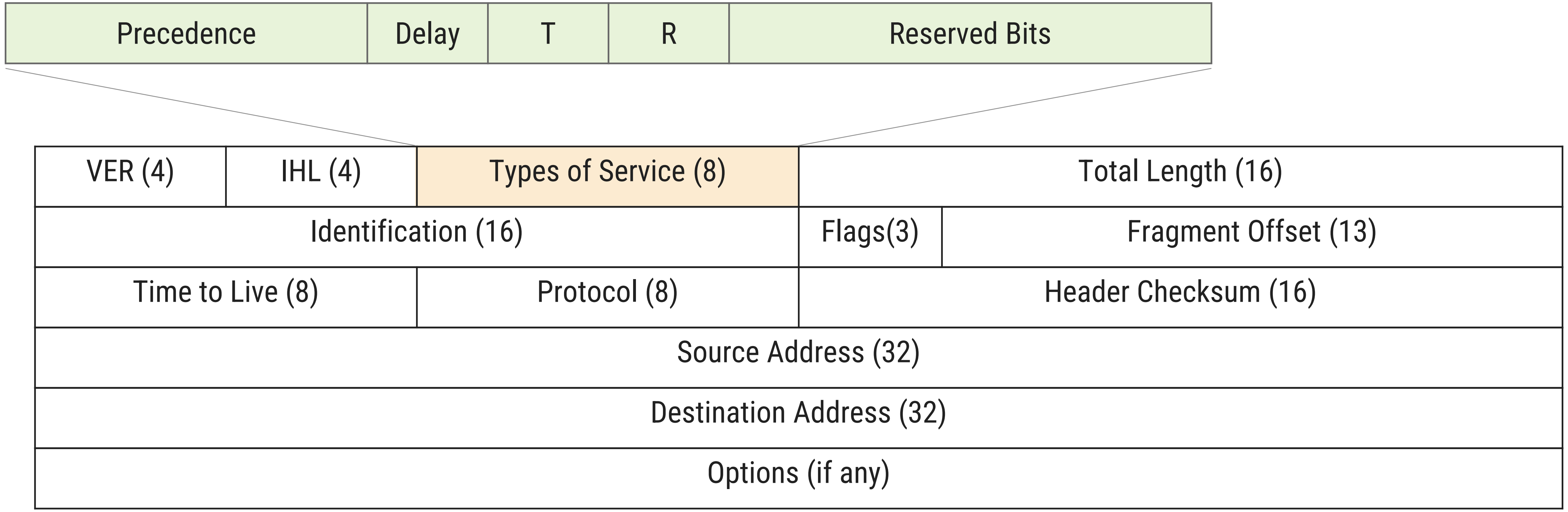
IPv4 Format

- Figure shows the format for IPv4.
- VER (Version):
  - It is a 4 bits field which indicates the version of IP.
- IHL (Internet Header Length):
  - It specifies the Internet Header Length in 32 bits word length and points the beginning of data.
- Types of Service:
  - It is a 8 bit field which specifies the Quality of Service (QoS). The format for these 8 bits are given here.

VER (4)	IHL (4)	Types of Service (8)	Total Length (16)	
Identification (16)			Flags(3)	Fragment Offset (13)
Time to Live (8)		Protocol (8)	Header Checksum (16)	
Source Address (32)				
Destination Address (32)				
Options (if any)				

IPv4 Format

- Types of Service:
  - Precedence (3) : They are used for future options.
  - Delay (1): It indicates the delay of either normal or lower.
  - Throughput (1): Indicates the speed of throughput of either normal or higher rate.
  - Reliability (1): It indicates normal reliability or high reliability.



IPv4 Format

- Total Length:
  - This is a 16 bit field defining the length of IPv4 datagram. The minimum length of datagram is 20 bytes and maximum is 65535.
- Identification:
  - This field is of 16 bits which helps in assembling the fragments and added by the senders.
- Flags:
  - There are3 bits in flags. First is always ‘1’. Second is DF (Don’t Fragment) and third is MF (More Fragment).

VER (4)	IHL (4)	Types of Service (8)	Total Length (16)	
Identification (16)			Flags(3)	Fragment Offset (13)
Time to Live (8)		Protocol (8)	Header Checksum (16)	
Source Address (32)				
Destination Address (32)				
Options (if any)				

IPv4 Format

- Fragment Offset:
  - ↳ When fragmentation is done, this specifies the offset or position of overall message.
- Time to Live:
  - ↳ This is 8 bit field that indicates the time period of datagram to survive.
- Protocol:
  - ↳ This field identifies the protocol for higher layer in the IP datagram.

VER (4)	IHL (4)	Types of Service (8)	Total Length (16)	
Identification (16)			Flags(3)	Fragment Offset (13)
Time to Live (8)		Protocol (8)	Header Checksum (16)	
Source Address (32)				
Destination Address (32)				
Options (if any)				

IPv4 Format

- Header Checksum:
  - It is 16 bit field to provide basic protection in transmission of header via checksum.
- Source Address:
  - This is 32 bit IP address of the source of the datagram.
- Destination Address:
  - This is 32 bit IP address of the destination of the datagram.
- Options:
  - There are many optional header available for debug and test purpose.

VER (4)	IHL (4)	Types of Service (8)	Total Length (16)	
Identification (16)			Flags(3)	Fragment Offset (13)
Time to Live (8)		Protocol (8)	Header Checksum (16)	
Source Address (32)				
Destination Address (32)				
Options (if any)				

IPv6 Addressing

- As the number of devices grow connected to the internet, it is obvious that we need more number of IP addresses to assign all the devices.
- IPv4 IP addressing uses 32 bit IP addresses which can generate
- $2^{32} = 4294967296$
- The above value is just above 4 billion. In fact the actual available IP addresses in IPv4 is even less than theoretical value.
- The actual IP addresses that can be assigned in IPv4 are 3,720,249,092.
- The IPv6 IP addressing uses 128 bit IP addresses which can generate
- $2^{128} = 3.4028236692093846346337460743177\ x\ 10^{38}$
- The above value is very large that if we assign a unique IP to every item in the world then also it will have reserved IP addresses.,

Features of IPv6 Addressing

- It has better and efficient routing than IPv4.
- The additional flow label is added in header of IPv6 for improvement in QoS.
- IPv6 has new application like IP telephony, video/audio, interactive games etc. with ensured QoS.
- The plug and play abilities have been improved in IPv6.
- IPv6 has eliminated the need of Network Address Translation due to the huge availability of IP addresses.

IPv6 Header Format

- IPv6 header format has less fields than IPv4 header format.
- IPv6 header is 40 bytes which is twice than IPv4 header.
- The figure shows IPv6 header format.
- The IPv6 has simple packet handling and improved forwarding efficiency.
- The fields of IPv6 header can be explained as following:

IP Version Number (4)	Traffic Class (8)	Flow Label (20)
Payload Length (16)	Next Header (8)	Hop Limit (8)
Source Address (128)		
Destination Address (128)		

IPv6 Header Format

- IP version :
  - It is 4 bit field to represent the IP version being used.
- Traffic Class :
  - It is 8 bit field to identify the different classes or priorities of IPv6 packets.
  - It replaces the type of services in IPv4.
  - Upper 6 bits are used for type of services lower 2 bits are used for Explicit Congestion Notification.
- Flow Label :
  - It is 20 bit field used to identify the sequence of packets.
  - It is also useful for prioritizing the delivery of packet and providing real time service.
  - The higher priority packets can be delivered ahead of lower priority packets.

IP Version Number (4)	Traffic Class (8)	Flow Label (20)
Payload Length (16)	Next Header (8)	Hop Limit (8)
Source Address (128)		
Destination Address (128)		

IPv6 Header Format

- Payload Length :
  - It is a 16 bit field which identifies the length of payload of IPv6.
- Next Header :
  - It is a 8 bit field which is similar to protocol field in IPv4 header.
  - It represents the type of extension header that follows the primary IPv6 header.
- Hop Limit :
  - It is 8 bit field equivalent to Time to Live in IPv4 Header.
  - The value is decremented by 1 every time when it is forwarded by the host.
  - When this value reaches 0, the packet is discarded.

IP Version Number (4)	Traffic Class (8)	Flow Label (20)
Payload Length (16)	Next Header (8)	Hop Limit (8)
Source Address (128)		
Destination Address (128)		

IPv6 Header Format

- Source Address :
  - It represents the 128 bit IP address of source or sender.
- Destination Address :
  - It represents the 128 bit IP address of destination or receiver.

IP Version Number (4)	Traffic Class (8)	Flow Label (20)
Payload Length (16)	Next Header (8)	Hop Limit (8)
Source Address (128)		
Destination Address (128)		

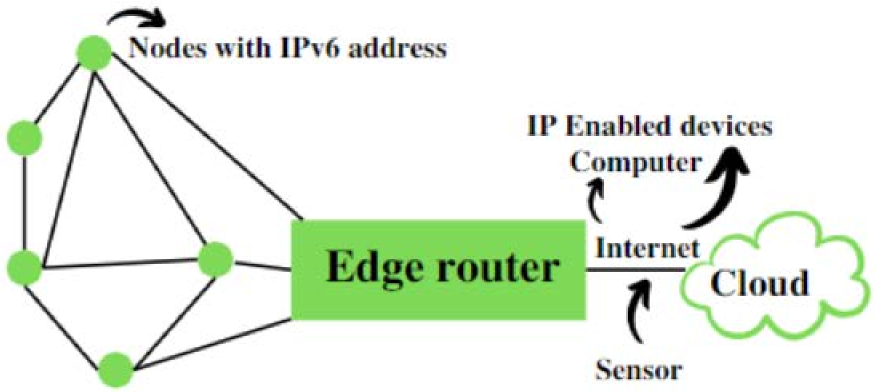
IPv4	IPv6
IPv4 has a 32-bit address length	IPv6 has a 128-bit address length
It Supports Manual and DHCP address configuration	It supports Auto and renumbering address configuration
In IPv4 end to end, connection integrity is Unachievable	In IPv6 end-to-end, connection integrity is Achievable
It can generate 4.29×10^9 address space	The address space of IPv6 is quite large it can produce 3.4×10^{38} address space
The Security feature is dependent on the application	IPSEC is an inbuilt security feature in the IPv6 protocol
Address representation of IPv4 is in decimal	Address Representation of IPv6 is in hexadecimal
Fragmentation performed by Sender and forwarding routers	In IPv6 fragmentation is performed only by the sender
In IPv4 Packet flow identification is not available	In IPv6 packet flow identification are Available and uses the flow label field in the header
In IPv4 checksum field is available	In IPv6 checksum field is not available
It has a broadcast Message Transmission Scheme	In IPv6 multicast and anycast message transmission scheme is available
In IPv4 Encryption and Authentication facility not provided	In IPv6 Encryption and Authentication are provided
IPv4 has a header of 20-60 bytes.	IPv6 has a header of 40 bytes fixed
IPv4 can be converted to IPv6	Not all IPv6 can be converted to IPv4
IPv4 consists of 4 fields which are separated by addresses dot (.)	IPv6 consists of 8 fields, which are separated by a colon (:)
IPv4's IP addresses are divided into five different classes. Class A , Class B, Class C, Class Da , Class E.	IPv6 does not have any classes of the IP address.
IPv4 supports VLSM(Variable Length subnet mask).	IPv6 does not support VLSM.
Example of IPv4: 66.94.29.13	Example of IPv6: 2001:0000:3238:DFE1:0063:0000:0000:FEFB

6LoWPAN is an IPv6 protocol, and It's extended from is IPv6 over Low Power Personal Area Network. As the name itself explains the meaning of this protocol is that this protocol works on Wireless Personal Area Network i.e., WPAN.

WPAN is a Personal Area Network (PAN) where the interconnected devices are centered around a person's workspace and connected through a wireless medium. You can read more about WPAN at WPAN. 6LoWPAN allows communication using the IPv6 protocol. IPv6 is Internet Protocol Version 6 is a network layer protocol that allows communication to take place over the network. It is faster and more reliable and provides a large number of addresses.

6LoWPAN initially came into existence to overcome the conventional methodologies that were adapted to transmit information. But still, it is not so efficient as it only allows for the smaller devices with very limited processing ability to establish communication using one of the Internet Protocols, i.e., IPv6. It has very low cost, short-range, low memory usage, and low bit rate.

It comprises an Edge Router and Sensor Nodes. Even the smallest of the IoT devices can now be part of the network, and the information can be transmitted to the outside world as well. For example, LED Streetlights.



- It is a technology that makes the individual nodes IP enabled.
- 6LoWPAN can interact with 802.15.4 devices and also other types of devices on an IP Network. For example, Wi-Fi.
- It uses AES 128 link layer security, which AES is a block cipher having key size of 128/192/256 bits and encrypts data in blocks of 128 bits each. This is defined in IEEE 802.15.4 and provides link authentication and encryption.

Basic Requirements of 6LoWPAN:

1. The device should be having sleep mode in order to support the battery saving.
2. Minimal memory requirement.
3. Routing overhead should be lowered.

Features of 6LoWPAN:

1. It is used with IEEE 802.15,.4 in the 2.4 GHz band.
2. Outdoor range: ~200 m (maximum)
3. Data rate: 200kbps (maximum)
4. Maximum number of nodes: ~100

Advantages of 6LoWPAN:

1. 6LoWPAN is a mesh network that is robust, scalable, and can heal on its own.
2. It delivers low-cost and secure communication in IoT devices.
3. It uses IPv6 protocol and so it can be directly routed to cloud platforms.
4. It offers one-to-many and many-to-one routing.
5. In the network, leaf nodes can be in sleep mode for a longer duration of time.

Disadvantages of 6LoWPAN:

1. It is comparatively less secure than Zigbee.
2. It has lesser immunity to interference than that Wi-Fi and Bluetooth.
3. Without the mesh topology, it supports a short range.

Applications of 6LoWPAN:

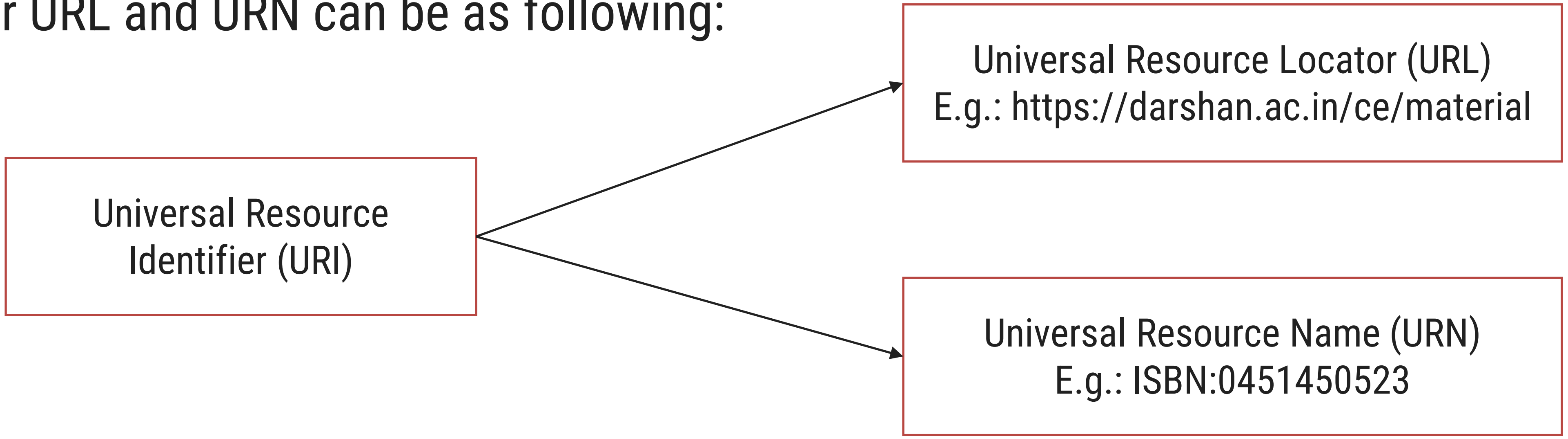
1. It is a wireless sensor network.
2. It is used in home-automation,
3. It is used in smart agricultural techniques, and industrial monitoring.
4. It is utilised to make IPv6 packet transmission on networks with constrained power and reliability resources possible.

Security and Interoperability with 6LoWPAN:

- **Security**: 6LoWPAN security is ensured by the AES algorithm, which is a link layer security, and the transport layer security mechanisms are included as well.
- **Interoperability:** 6LoWPAN is able to operate with other wireless devices as well which makes it interoperable in a network.

Uniform Resource Identifier (URI)

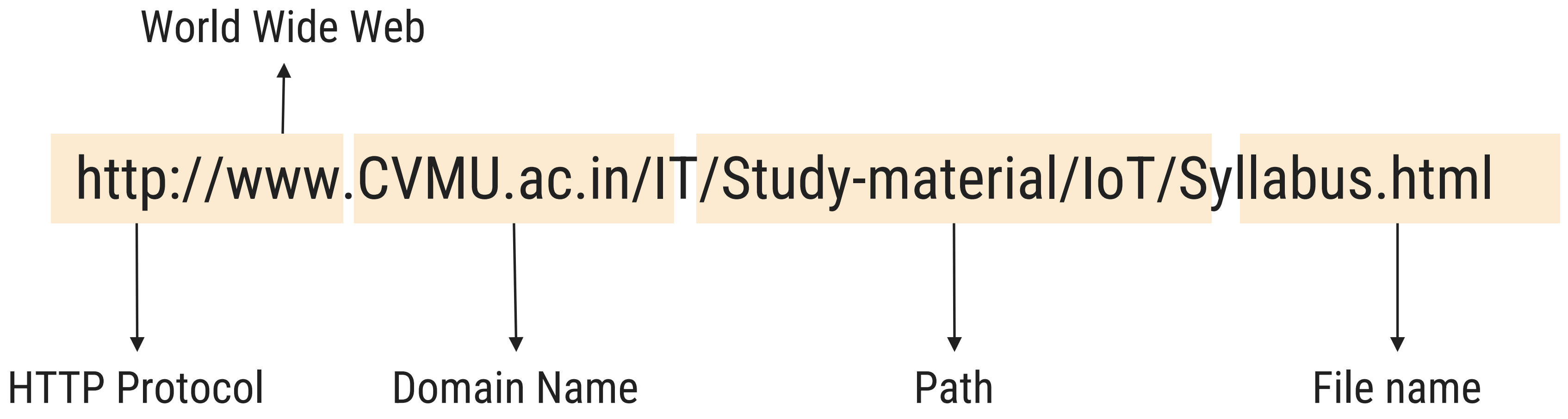
- Uniform Resource Identifier (URI) is used to identify the resources with the help of sequence of characters.
- The URI protocol was developed by IETF.
- The URI can be URL or URN.
- The sample for URL and URN can be as following:


Universal Resource Identifier (URI)
Universal Resource Locator (URL)
E.g.: https://darshan.ac.in/ce/material
Universal Resource Name (URN)
E.g.: ISBN:0451450523

- URN mostly provides information of unique name without locating or retrieving the resource information.
- URL provides locating and retrieving information on a network.

Uniform Resource Locator (URL)

- The URL contains the information of how to fetch a resource information from its location.
- URL always begin with a protocol like HTTP/HTTPS
- A URL is used when client request the server for the service.
- The sample URL and its fields are explained below:



What is Software-Defined Networking (SDN)?

- **Software-Defined Networking (SDN)** is an approach to networking that uses software-based controllers or application programming interfaces (APIs) to communicate with underlying hardware infrastructure and direct traffic on a network.
- This model differs from that of traditional networks, which use dedicated hardware devices (i.e., routers and switches) to control network traffic. SDN can create and control a virtual network – or control a traditional hardware – via software.
- software-defined networking enables a new way of controlling the routing of data packets through a centralized server.

Features of SDN.

- **Increased control with greater speed and flexibility:** Instead of manually programming multiple vendor-specific hardware devices, developers can control the flow of traffic over a network simply by programming an open standard software-based controller. Networking administrators also have more flexibility in choosing networking equipment, since they can choose a single protocol to communicate with any number of hardware devices through a central controller.
- **Customizable network infrastructure:** With a software-defined network, administrators can configure network services and allocate virtual resources to change the network infrastructure in real time through one centralized location. This allows network administrators to optimize the flow of data through the network and prioritize applications that require more availability.
- **Robust security:** A software-defined network delivers visibility into the entire network, providing a more holistic view of security threats. With the proliferation of smart devices that connect to the internet, SDN offers clear advantages over traditional networking. Operators can create separate zones for devices that require different levels of security, or immediately quarantine compromised devices so that they cannot infect the rest of the network.

Why SON is Important?

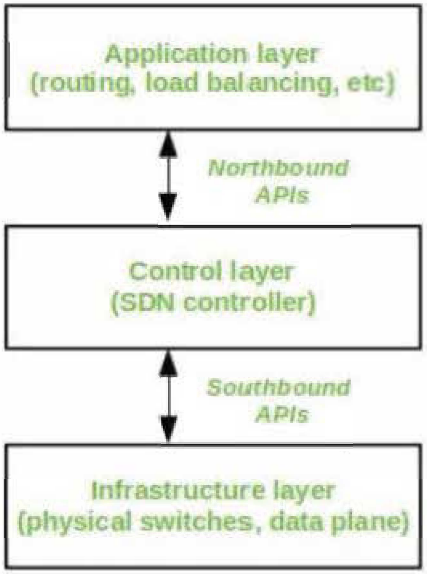
- **Better Network Connectivity:** SON provides very better network connectivity for sales, services, and internal communications. SON also helps in faster data sharing.
- **Better Deployment of Applications:** Deployment of new applications, services, and many business models can be speed up using Software Defined Networking.
- **Better Security:** Software-defined network provides better visibility throughout the network. Operators can create separate zones for devices that require different levels of security. SON networks give more freedom to operators.
- **Better Control with High Speed:** Software-defined networking provides better speed than other networking types by applying an open standard software-based controller.

Software Defined Networking	Traditional Networking
Software Defined Network is a virtual networking approach.	A traditional network is the old conventional networking approach.
Software Defined Network is centralized control.	Traditional Network is distributed control.
This network is programmable.	This network is nonprogrammable.
Software Defined Network is the open interface.	A traditional network is a closed interface.
In Software Defined Network data plane and control, the plane is decoupled by software.	In a traditional network data plane and control plane are mounted on the same plane.

A typical SON architecture consists of three Layers.

- Application Layer: It contains the typical network applications like intrusion detection, firewall, and load balancing
- Control layer: It consists of the SON controller which acts as the brain of the network. It also allows hardware abstraction to the applications written on top of it.
- Infrastructure layer: This consists of physical switches which form the data plane and carries out the actual movement of data packets.

The Layers communicate via a set of interfaces called the north-bound APIsjbetween the application and control layer! and southbound APIs(between the control and infrastructure layer).



SON Architecture

Advantages of SON

- The network is programmable and hence can easily be modified via the controller rather than individual switches.
- Switch hardware becomes cheaper since each switch only needs a data plane.
- Hardware is abstracted, hence applications can be written on top of the controller independent of the switch vendor.
- Provides better security since the controller can monitor traffic and deploy security policies. For example, if the controller detects suspicious activity in network traffic, it can reroute or drop the packets.

Disadvantages of SON

- The central dependency of the network means a single point of failure, i.e. if the controller gets corrupted, the entire network will be affected.
- The use of SON on large scale is not properly defined and explored.

Software-Defined Networking (SDN) can significantly enhance Internet of Things (IoT) network management and security in several ways:

1. **Centralized Management**: SDN allows for centralized management of network devices, which can greatly simplify the task of managing large-scale and heterogeneous IoT networks. This centralized controller and view of the network is one of the key features of SDN.

2. **Dynamic Configuration**: The dynamic configuration of the network becomes a necessity with the massive use of connected devices. SDN provides programmability of the network and separation of data plane and control plane, which can facilitate and improve the management of the network devices in terms of configuration, control, and security.

3. **Improved Security**: SDN introduces many opportunities to enhance security. It enables quick reactions to security threats, granular traffic filtering, and dynamic security policies deployment. Moreover, it provides the potential to distribute security rules across multiple domains without compromising the security of any single domain.

4. **Efficient Use of Network Resources**: SDN provides a global view of the network, which can help use network resources more efficiently. This can reduce the overhead of network management and improve the flexibility of networks.

5. **Customization**: By embracing SDN in IoT, organizations can unlock the potential for greater control, customization, and security within their networks. This paves the way for optimized performance and improved management of IoT deployments.

In conclusion, SDN presents a great potential to solve or mitigate the emerging problems of IoT, especially in terms of network management and security.