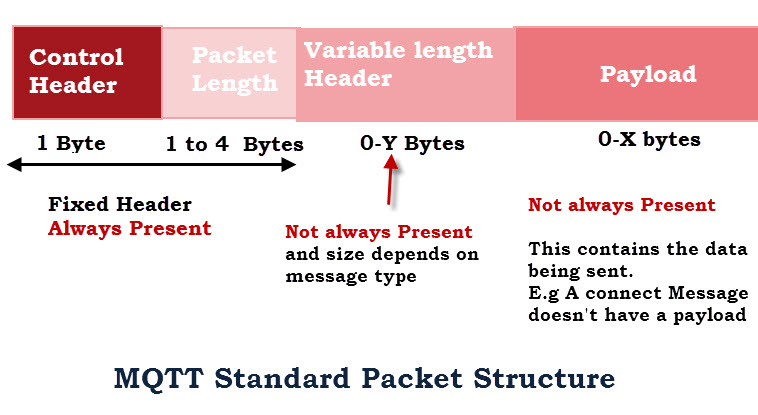
**ANSWERS FOR ASSINGMENT 4**

ANSWER 1

MQTT is a binary based protocol were the control elements are binary bytes and not text strings. MQTT uses a command and command acknowledgement format. That means each command has an associated acknowledgement.   
Topic names, Client ID, User names and Passwords are encoded as UTF-8 strings.  
The Payload excluding MQTT protocol information like Client ID etc is binary data and the content and format is application specific. The MQTT packet or message format consists of a 2 byte fixed header (always present) + Variable-header (not always present) + payload (not always present).



Possible Packet formats are:

Fixed Header (Control field + Length) – Example CONNACK

Fixed Header (Control field + Length) + Variable Header -Example PUBACK

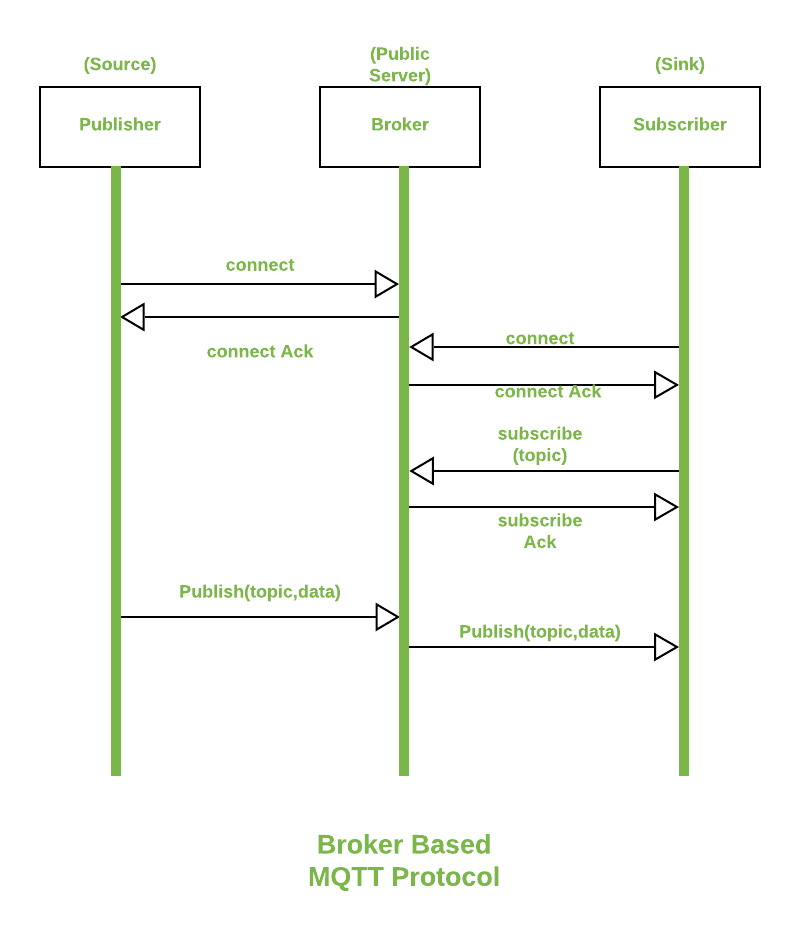
Fixed Header (Control field + Length) + Variable Header + payload -Example CONNECT

The fixed header field consists of the control field and the variable length packet length field.

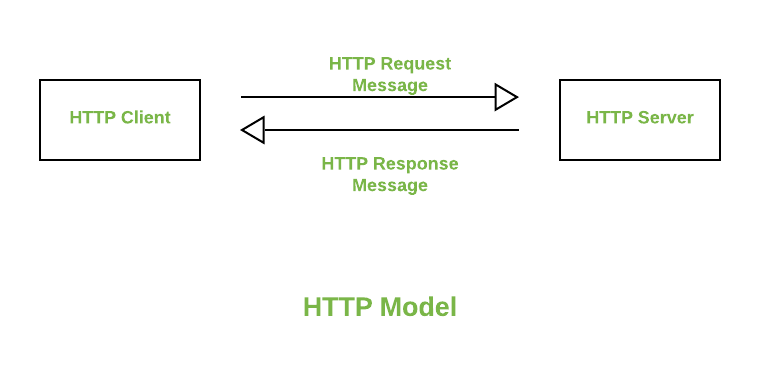
The minimum size of the packet length field is 1 byte which is for messages with a total length less than 127 bytes. (not including control and length fields).The maximum packet size is 256MB. Small packets less than 127 bytes have a 1 byte packet length field. Packets larger than 127 and less than 16383 will use 2 bytes. etc. Note: 7 bits are used with the 8th bit being a continuation bit. The minimum packet size is just 2 bytes with a single byte control field and a single byte packet length field. E.g the disconnect message is only 2 bytes.

ANSWER 2

**Message Queuing Telemetry Transport (MQTT) :**  
It was created by Andy Sandford-Clark and Arlen Nipper. It is an IoT interaction protocol based on the Publish/Subscribe model. This model is a simple model that provides support for QoS (Quality of Service). Due to its abilities, it can be found in every second IoT based device. This protocol has many features as it is over TCP and uses SSL/TLS for security. For messaging between server it uses CONNECT, PUBLISH, SUBSCRIBE, DISCONNECT, etc.



[**Hyper Text Transfer Protocol (HTTP)**](https://www.geeksforgeeks.org/http-non-persistent-persistent-connection/) :  
It is used by [World Wide Web (WWW)](https://www.geeksforgeeks.org/world-wide-web-www/) for defining how its messages are going to be transmitted and formatted. This protocol is responsible for the action that a server has to take while sending information over the network. When a URL is being entered into the browser, this protocol sends an HTTP request to the server and then an HTTP response is sent back to the browser. This protocol is also responsible for the controlling of webpages on the World Wide Web for their formatting and representation.



HTTP is (or was) primarily a request-response protocol built on top of TCP. i.e. a client (web browser) sends a request (open a TCP connection and send a specifically formatted text message) to a server (a web server). The server then sends back an appropriate response and the connection is closed.

HTTP 1.1/2 introduce persistent connections, binary framing mechanisms, web sockets etc. More information on the request / response mechanism - HTTP Messages

MQTT is a publish-subscribe protocol based on TCP. A client connects to a server. The client can then subscribe to various topics. The client and the server can then publish messages to various topics and the server is responsible for delivering it to various interested clients.

ANSWER 3

JSON stands for JavaScript Object Notation  
JSON is a lightweight format for storing and transporting data  
JSON is often used when data is sent from a server to a web page  
JSON is "self-describing" and easy to understand

Example: -

{ "students in hexnbit":[

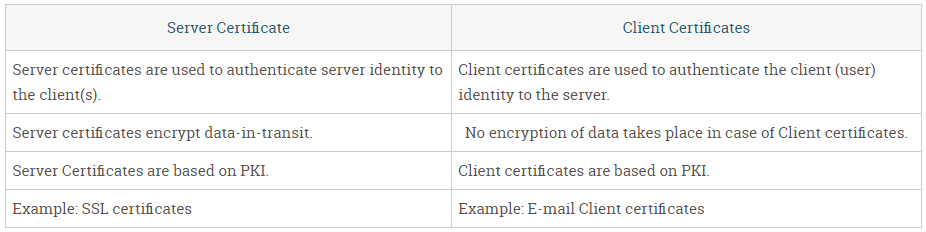
{"firstName":"somesh", "lastName":"rao"},  
 {"firstName":"likhith", "lastName":"readdy"},  
 {"firstName":"sumanth", "lastName":"Jones"},  
 {"firstName":"satvik", "lastName":"cooper"}

]  
}

ANSWER 4

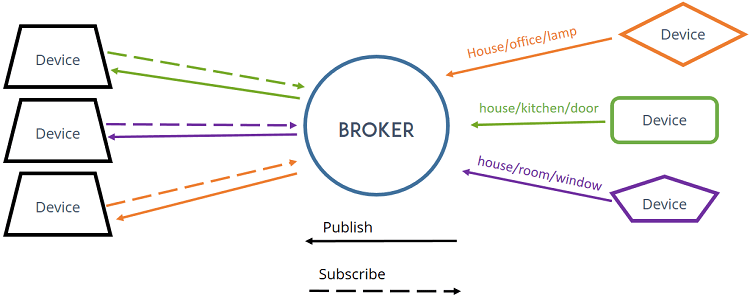
Server certificates (SSL certificates) are used to authenticate the identity of a server. When installed on a website, an SSL certificate turns the protocol on the website from HTTP to HTTPS [Difference b/w HTTP and https] and installs indicators that vouch for the authenticity of the website. Thus, users can know the website belongs to the said entity. Apart from authentication, SSL certificates also facilitate Encryption. Meaning, any information a user sends to the server is protected from the reaches of any ill-intended 3rd party.

Client certificates are used to validate the identity of a client (user). The user, in this case, might be a website user or an email user. Simply put, it works as a password, but without any intervention/input from the user. This way, the server makes sure that it’s connecting to the permitted user and that party is safe to communicate with



ANSWER 5

An MQTT broker is a server that receives all messages from the clients and then routes the messages to the appropriate destination clients. An MQTT client is any device (from a micro controller up to a full-fledged server) that runs an MQTT library and connects to an MQTT broker over a network.



ANSWER 6

#include "mgos.h"

#include "mgos\_mqtt.h"

#include "mgos\_dht.h"

static struct mgos\_dht \*s\_dht = NULL;

static void led\_timer\_cb(void \*arg) {

  bool val = mgos\_gpio\_toggle(2);

  LOG(LL\_INFO, ("%s uptime: %.2lf, RAM: %lu, %lu free", val ? "Tick" : "Tock",

                mgos\_uptime(), (unsigned long) mgos\_get\_heap\_size(),

                (unsigned long) mgos\_get\_free\_heap\_size()));

  (void) arg;

}

static void net\_cb(int ev, void \*evd, void \*arg) {

  switch (ev) {

    case MGOS\_NET\_EV\_DISCONNECTED:

      LOG(LL\_INFO, ("%s", "Net disconnected"));

      break;

    case MGOS\_NET\_EV\_CONNECTING:

      LOG(LL\_INFO, ("%s", "Net connecting..."));

      break;

    case MGOS\_NET\_EV\_CONNECTED:

      LOG(LL\_INFO, ("%s", "Net connected"));

      break;

    case MGOS\_NET\_EV\_IP\_ACQUIRED:

      LOG(LL\_INFO, ("%s", "Net got IP address"));

      break;

  }

  (void) evd;

  (void) arg;

}

static void report\_temperature(void \*arg) {

  char topic[100], message[160];

  struct json\_out out = JSON\_OUT\_BUF(message, sizeof(message));

  time\_t now=time(0);

  struct tm \*timeinfo = localtime(&now);

  snprintf(topic, sizeof(topic), "event/temperature");

  json\_printf(&out, "{total\_ram: %lu, free\_ram: %lu, temperature: %f, humidity: %f, device: \"%s\", timestamp: \"%02d:%02d:%02d\"}",

              (unsigned long) mgos\_get\_heap\_size(),

              (unsigned long) mgos\_get\_free\_heap\_size(),

              (float) mgos\_dht\_get\_temp(s\_dht),

              (float) mgos\_dht\_get\_humidity(s\_dht),

              (char \*) mgos\_sys\_config\_get\_device\_id(),

              (int) timeinfo->tm\_hour,

              (int) timeinfo->tm\_min,

              (int) timeinfo->tm\_sec);

  bool res = mgos\_mqtt\_pub(topic, message, strlen(message), 1, false);

  LOG(LL\_INFO, ("Published to MQTT: %s", res ? "yes" : "no"));

  (void) arg;

}

static void button\_cb(int pin, void \*arg) {

  float t = mgos\_dht\_get\_temp(s\_dht);

  float h = mgos\_dht\_get\_humidity(s\_dht);

  LOG(LL\_INFO, ("Button presses on pin: %d", pin));

  LOG(LL\_INFO, ("Temperature: %f \*C Humidity: %f %%\n", t, h));

  report\_temperature(NULL);

  (void) arg;

}

enum mgos\_app\_init\_result mgos\_app\_init(void) {

  /\* Blink built-in LED every  seconds \*/

  mgos\_gpio\_set\_mode(2, MGOS\_GPIO\_MODE\_OUTPUT);

  mgos\_set\_timer(1000, MGOS\_TIMER\_REPEAT, led\_timer\_cb, NULL);

  /\* Report temperature to AWS IoT Core every  30 seconds \*/

  mgos\_set\_timer(30000, MGOS\_TIMER\_REPEAT, report\_temperature, NULL);

  /\* Publish to MQTT on button press \*/

  mgos\_gpio\_set\_button\_handler(0, MGOS\_GPIO\_PULL\_UP, MGOS\_GPIO\_INT\_EDGE\_NEG, 200, button\_cb, NULL);

  if ((s\_dht = mgos\_dht\_create(5, DHT11)) == NULL) {

    LOG(LL\_INFO, ("Unable to initialize DHT11"));

  }

  /\* Network connectivity events \*/

  mgos\_event\_add\_group\_handler(MGOS\_EVENT\_GRP\_NET, net\_cb, NULL);

  return MGOS\_APP\_INIT\_SUCCESS;

}

ANSWER 9

#define CAYENNE\_PRINT Serial // Comment this out to disable prints and save space

#include <CayenneMQTTESP8266.h>

// WiFi network info.

char ssid[] = "tiger2.4GHz"; //Enter Your Router SSID

char wifiPassword[] = "hack\_it\_if\_u\_can"; //Enter your Router Password

// Cayenne authentication info. This should be obtained from the Cayenne Dashboard.

char username[] = "khadwiyegfiyef"; // Enter Cayenne Username

char password[] = "kjsxqsydxuqbdqhb"; // Enter Cayenne Password

char clientID[] = "ksjbciweugfIqdhiquwgdqu"; // Enter Cayenne ClientID

#define LED\_PIN D1 //Pin connected to LED

#define VIRTUAL\_CHANNEL 2 // Receive from Channel

void setup()

{

Serial.begin(9600); // Serial Communication

Cayenne.begin(username, password, clientID, ssid, wifiPassword);

pinMode(LED\_PIN,OUTPUT); // Make pin as OUTPUT

}

void loop()

{

Cayenne.loop();

}

CAYENNE\_IN(VIRTUAL\_CHANNEL)

{

int value = getValue.asInt();

// Write the value received to the digital pin.

digitalWrite(LED\_PIN, value);

}

ANSWER 10

#define CAYENNE\_PRINT Serial // Comment this out to disable prints and save space

#include <CayenneMQTTESP8266.h> // Library for ESP and Cayenne

// WiFi network info.

char ssid[] = "tiger"; // Enter you Router SSID

char wifiPassword[] = "hack\_me\_u\_can"; // Enter your Router Password

// Cayenne authentication info. This should be obtained from the Cayenne Dashboard.

char username[] = "xxxxxxxxxxxxxxxxxxxxx"; //Enter Cayenne Username

char password[] = "xxxxxxxxxxxxxxxxxxxxx"; // Enter Cayenne Password

char clientID[] = "xxxxxxxxxxxxxxxxxxxxx"; // Enter Cayenne Client ID

#define SENSOR\_PIN A0 //Pin connected to LDR

#define VIRTUAL\_CHANNEL 3 // Channel to Upload Data

void setup()

{

Serial.begin(9600); // Serial Communication

Cayenne.begin(username, password, clientID, ssid, wifiPassword);

}

void loop()

{

Cayenne.loop();

}

// This function is called at intervals to send sensor data to Cayenne.

CAYENNE\_OUT(VIRTUAL\_CHANNEL)

{

float voltage = value \* (3300.0/1023);

Cayenne.virtualWrite (VIRTUAL\_CHANNEL, LDR);

ANSWER 8

* **Trade capital expense for variable expense** – Instead of having to invest heavily in data centre and servers before you know how you’re going to use them, you can pay only when you consume computing resources, and pay only for how much you consume.
* **Benefit from massive economies of scale** – By using cloud computing, you can achieve a lower variable cost than you can get on your own. Because usage from hundreds of thousands of customers is aggregated in the cloud, providers such as AWS can achieve higher economies of scale, which translates into lower pay as-you-go prices.
* **Stop guessing capacity** – Eliminate guessing on your infrastructure capacity needs. When you make a capacity decision prior to deploying an application, you often end up either sitting on expensive idle resources or dealing with limited capacity. With cloud computing, these problems go away. You can access as much or as little capacity as you need, and scale up and down as required with only a few minutes’ notice.
* **Increase speed and agility** – In a cloud computing environment, new IT resources are only a click away, which means that you reduce the time to make those resources available to your developers from weeks to just minutes. This results in a dramatic increase in agility for the organization, since the cost and time it takes to experiment and develop is significantly lower.
* **Stop spending money running and maintaining data centers** – Focus on projects that differentiate your business, not the infrastructure. Cloud computing lets you focus on your own customers, rather than on the heavy lifting of racking, stacking, and powering servers.
* **Go global in minutes** – Easily deploy your application in multiple regions around the world with just a few clicks. This means you can provide lower latency and a better experience for your customers at minimal cost.