Proc. XXVII International Scientific Conference Electronics - ET2018, September 13 - 15, 2018, Sozopol, Bulgaria

Research of the Communication Protocols between the IoT Embedded System and the Cloud Structure

Neven Nikolov
Faculty of Computer Systems and Technologies
Technical University of Sofia
8 Kliment Ohridski blvd., 1000 Sofia, Bulgaria
n.nikolov@tu-sofia.bg

Abstract – **This article describes the study of communication protocols used between IoT Embedded Systems and Cloud Structure. Cloud structure architectures are described that use protocols such as MQTT and TCP sockets, and the experimental part includes their realization. It is proposed to implement a protocol through TCP Sockets using a JSON packet that compares with the MQTT protocol using the same JSON packet. Measurements are made with the size of the packet and the amount of data sent through the protocols.**

Keywords – **Protocol, MQTT, TCP socket, IoT, Embedded system, Cloud**

I. Introduction

Internet of Things - IoT are embedded systems that connect to a centralized or decentralized system by exchanging data with each other. The idea is to exchange data using protocols that are specialized for IoT embedded systems. The need for specialized protocols is due to the fact that embedded systems have limited capabilities and resources. Protocols must be reliable, consuming less power and a number of bytes across the network. The network is a resource that limits the amount of information transmitted. There are different types of physical transmission environments, such as ZigBee, LoraWan, NarrolBand, WiFi, Ethernet and others.

But there are exists different protocols with a particular idea, such as MQTT, TCP sockets, CoAP, XMPP, AMQP, RESTfull HTTP and others. Regarding IoT embedded systems, there are different implementation options for microcontrollers, radio modules and peripherals that are shaped by the physical network type and functionality. Embedded systems must bind to each other and exchange data directly or indirectly through Cloud Structure [1, 2] - servers that handle data entry and storage. The Cloud structure can be anywhere in the globe, which gives a great advantage over analog systems. The distances between cloud structures and IoT embedded systems can be large, with transmission without loss of data compared with damping in analogue transmission

This article describes the construction of the Cloud Structure [3,4,5] and the program implementation of the IoT embedded system. The main purpose is to study protocols such as MQTT and TCP sockets, through which a Java Script Object Notation (JSON) data packet is implemented. Describes how data is transmitted, embedded system architecture, and Cloud structure. The main idea is the study of data protocols and a solution for optimal data transmission is proposed.

II. Review protocols

The article describes the use, use and implementation of a JSON packet format for data transmission between cloud structure [6,7,8] and IoT embedded system, using the MQTT and TCP sockets [13,14].

MQTT is a message-based communication technology that is lightweight and relatively simple to integrate into many devices [11]. The technology works by using the so-called broker. Broker is an intermediate software module that implements the so-called Publish - Subscribe model, where Subscribers communicate with the broker and receive one or more messages sent by publishers. MQTT is a simple and easy protocol designed for limited devices and networks with low data transfer, high latency and unreliability. TCP / IP ports used by the MQTT protocol brokers are 1883 and 8883. The use of port 1883 is associated with an unencrypted connection that can listen and see and port 8883 uses SSL encryption encryption. A major drawback of using the SSL certificate is the introduction of extra workload, which results in productivity declining. Certificates consume more power in low-powered battery-powered devices.

The TCP sockets client opens with the server of a specific port number [12]. To make this happen, the server listens to a socket on the client, as it asks itself. The client needs to know "Hostname" - the name of the machine on which the server operates to perform the hanging and communication. The server accepts the connection request. Upon acceptance, the server receives a new socket connected to the same local port. On the client side, if the connection is accepted, a socket is created and the client can use the socket to communicate with the server. The client and the server can communicate by writing or reading the sockets. The endpoint is a combination of an IP address and a port number. Each TCP connection can be uniquely identified from its two endpoints. In this way there may be several connections between the host and the server. The connection between two computers is performed by a socket. It is a combination of IP address and port. IP addresses are executed on the network layer, which is the IP layer. Sockets are executed on the transport layer as part of the TCP header. TCP is designed for non-data-oriented applications. It has built-in error checking and will re-provide the missing packages.

III. ARCHITECTURE

There are exists different types of architectures and realizations [16]. Due to the fact that different protocols are used, it is the use of different technologies for the realization of architecture [10,15,17]. Fig. 1 shows the Cloud System architecture using TCP sockets, and fig. 3 are shown Cloud System architecture using the MQTT protocol.

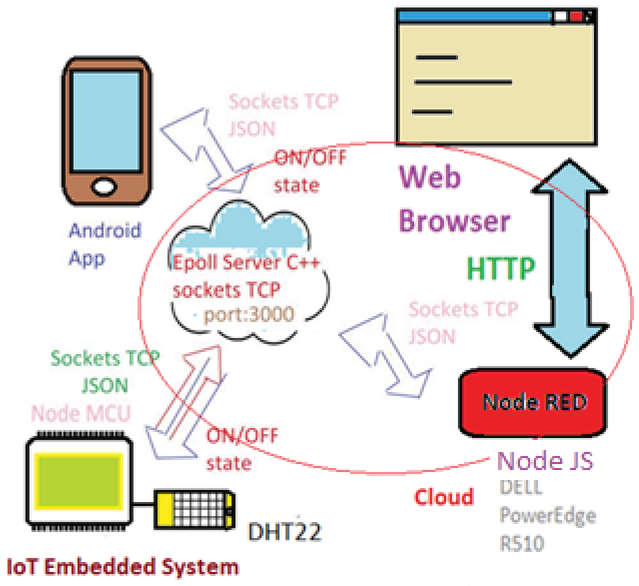

Fig. 1 Cloud structure architecture using TCP sockets

The architecture presented in fig. 1 uses TCP sockets, including technologies such as Epoll Server C ++, Node RED and Node JS. An Epoll server used to communicate with TCP sockets to the IoT embedded system and Node RED is an IBM technology that can realize an IoT server and graphical environment running on a WEB browser [9]. Node JS is a server technology using JavaScrip programming language on which Node RED works. In the web browser, the client can monitor parameters such as temperature and humidity and manage the IOT embedded system. A mobile application connects to the Epoll server via the JSON TCP sockets protocol. The protocol used is an HTTP between the Web browser and Node RED. The Epoll server communicates between Node REDs using the TCP socket protocol implementing the JSON data transmission format Fig.2.

```
{ "deviceID"  :  1,
  "type"      :  "IoTdevice",
  "humidity"  :  42,
  "name"      :  "ESPDH22",
  "state"     :  "on",
  "temp"      :  19 }
```

Fig.2 Data transmission between embedded system and Cloud structure in JSON format

In fig. 2 are shows the JSON format for data transmission between the Cloud and the IoT embedded system. The fields in the protocol are "deviceID", "type", "humidity", "name", "state", "temp", respectively the values sent by the built-in temperature and humidity system. Each device that connects to the Cloud sends a "deviceID" field in the JSON package, which is a unique IOT device number, and the "name" field displays the name of the device, which in this case is "ESPDH22".

In Fig. 3 are shows the architecture that using the MQTT protocol. At the core of Cloud Structure is Mosquitto MQTT Broker, Node RED IBM and Node JS Server. The IoT devices are connected to the broker by sending the messages to those customers subscribed to the topic. In the case of building a common environment, the same topic is used to which all customers are subscribed. The Node RED server has served as a client. The protocol between the Web Browser and the Cloud is HTTP, with the client managing the embedded system and monitoring the temperature and humidity readings. The mobile application uses the MQTT protocol by connecting to the Broker as a client.

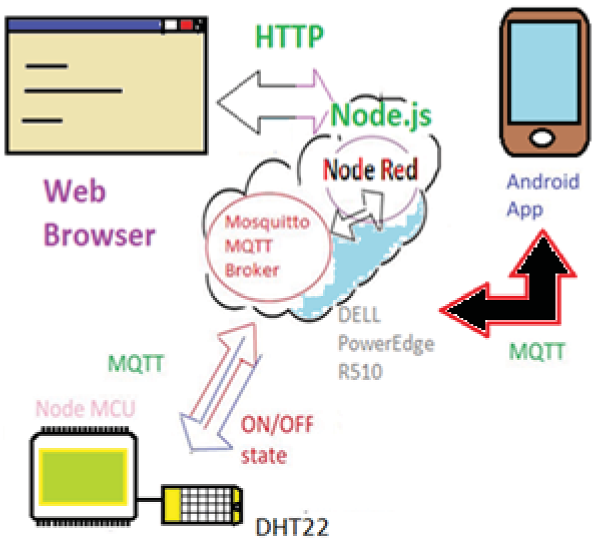

Fig. 3 Cloud structure architecture using MQTT sockets

IV. SOFTWARE REALIZATION

For building a Cloud structure, a Dell Power edge R510 server was used and the Centos 7 operating system was installed. Embedded systems are connected via WiFi to the internet network [18].

For the implementation of Epoll Server, a language C ++ is used, and the Cmake tool is used to compile the program code [19]. The Epoll server works on the following principle, devices need to connect, open a socket, and send a message with the current temperature, humidity, and condition of GPIO pins. The Epoll server should respond to the query from the embedded system within the shortest possible time. When the mobile application is connected to the cloud [22] and GPIO pin activation command is sent, the server will pass the corresponding parameter to the IoT embedded system. This will trigger GPIO pins on the IoT embedded system.

The MQTT Broker Mosquitto has been installed on the server, which acts as the publisher-subscriber model distributor. IBM's Node Red Technology, which uses Node JS server technology running Java Script, is used for a client and management unit.

The implementation of the IoT embedded system uses the C ++ programming language [20,21]. To define and initialize the sensor DHT22 for temperature and humidity are used following rows:

```
#define DHTTYPE DHT22

DHT dht(D2, DHTTYPE);
Serial.begin(115200);
pinMode(D5, OUTPUT);
digitalWrite(D5, LOW);
dht.begin();
```

where the function dht() are initialized OneWire communication to sensor DHT22. Function Serial.begin() create initialization of UART module on embedded system, who is used for debugging. Functions pinMode() and digitalWrite() are used for initialization GPIO pin. To define the JSON package, the data format is set from the following rows:

```
const int    deviceID   = 1;
const String type       = "IoTdevice";
const String Name       = "ESPDH22";
```

the fields describing the embedded system parameters that are read from the Cloud. The following lines of program code are used to initiate and connect to the Cloud Structure:

```
    WiFiClient          clientTCP,wifiClientS;
    PubSubClient clientMQTT(mqtt_serverS, 1883,
wifiClientS);

    if (clientMQTT.connect(clientIDS, mqtt_usernameS,
mqtt_passwordS )) {

        Serial.println("Connected to MQTT Broker!");
    }
    clientMQTT.subscribe("test");

    if (!clientTCP.connect("78.83.114.192", 3000)) {
        Serial.println("Connection to echo server failed");
        while (true) {}
    }
```

where the function “clientMQTT.connect”is used to connect to the MQTT broker, and the clientTCP.connect method connects the embedded system to the Epoll server. The function clientMQTT() are used to establish connection with MQTT broker on port 1883. To send the JSON package to the Epoll server and the MQTT broker, there are used the following program code rows:

```
clientTCP.print(jsonS);
clientMQTT.publish(mqtt_topicS, string2char(jsonS));
```

through which these DHT22 temperature and humidity sensor data are sent serialized to the Cloud structure [23]. Accepting data from the Cloud Structure to IoT embedded system is done through the following program code rows:

```
    while (true) {
        while (clientTCP.available() == 0) {}
        tempVar[i] = clientTCP.read();

    void callback(String topicS, byte* messageS, unsigned int
length) {
```

where the function “clientTCP.read ()” reads the returned data from the Epoll server, and the callback method is used to read a received message from the MQTT broker. The data needs to be retrieved from the JSON package - Parse in order to read the contents [23].

V. Experimental Results

In this sections, we can summarize and compare on the basis of the results obtained by measuring with the Wireshark software tool the size of the package and its contents. The two protocols MQTT and TCP sockets are compared in Table 1, as the basis for comparison is the same message sent by the embedded system through the MQTT and TCP sockets protocols. The message is composed of a JSON package, which is presented in fig. 2 The message size of the JSON package is 88 bytes. Table 1 shows the comparison of the two protocols, and in Fig. 4 is a graphical representation of the results of the table

TABLE 1

COMPARISON OF THE MQTT AND TCP SOCKET PROTOCOLS ON A SINGLE JSON PACKAGE SENT FROM THE IOT DEVICE TO THE CLOUD

Protocol	Message bytes	Packet Frame Bytes	Data Payload Bytes	Header Bytes	Topic
TCP socket	88	142	88	54	-
MQTT	88	150	96	54	„test“

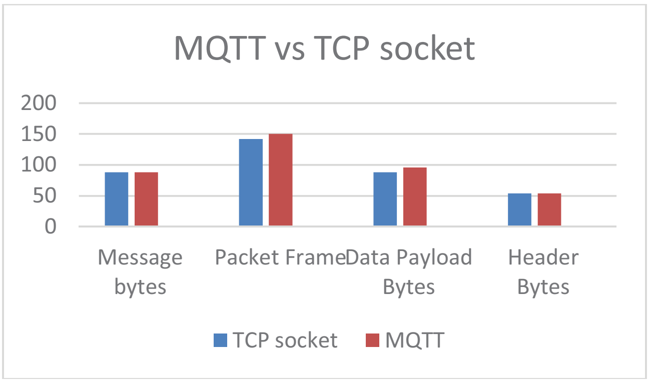

Fig. 4. Compare MQTT and TCP socket protocols with the same JSON
packet between the Cloud and the IoT device

From the fig. 4, it can be concluded that when sending the same JSON packet between the Cloud and the IoT device, the Packet Frame parameter has a greater length of the MQTT protocol than the TCP socket protocol. From the experiment, it appears that the TCP socket protocol uses fewer bytes to transmit the JSON format to the Cloud, so it will be more appropriate in situations where the amount of sent bytes is critical. For example, it would be more appropriate for low - capacity networks such as mobile, IoT networks and others, and the amount of bytes passed will cost the service. The IoT embedded system sends in every time S = 2000ms for sensor status data, and Table 2 shows the amount of bytes sent per 1 hour and 24 hours respectively, as well as the amount of transmitted bytes for that period. The Q for 1 and 24 hours can be calculated using the following formula Eq.1:

$$Q = PF \times (T\,(\text{hours}) \times 3600) / S \qquad (1)$$

Q - Quantity of Q data for 1 and 24 hours;

PF - Packet Frame (bytes 2 sec);

T - Time for which the IoT embedded system sent data;

S - The time the embedded system sends data to the Cloud;

Table 2 shows the amount of data used Q from the internet network calculated, respectively, Eq 1.

TABLE 2

COMPARISON OF THE MQTT AND TCP SOCKET PROTOCOLS ON A SINGLE JSON PACKAGE SENT FROM THE IOT DEVICE TO THE NUMBER OF QUANTITIES USED DATA FROM THE NETWORK

Protocol	Message bytes	Packet Frame Bytes for 2sec	Q for 1hour	Q for 24 hour
TCP socket	88	142	249.609375 kbytes	5 990.625 kbytes
MQTT	88	150	263.671875 kbytes	6328.125 kbytes

VI. Conclusion

From the measurements made, tables, graphs and comparisons in sections IV, it can be concluded that the use of TCP sockets uses fewer data bytes than the MQTT protocol. But this advantage is appropriate when we use the Client-Server architecture for IoT purposes. The MQTT protocol has advantages over TCP sockets in situations where an embedded system needs to send data to many others subscribed to the topic - Topic. The implementation of the protocol through JSON and TCP sockets is faster due to the fact that an Epoll server, characterized by its performance, is used. The protocol offers fewer transmitted bytes across the network, so IoT devices with battery power are longer lifetime than the power consumed.

In this article proposes a way of communicating an embedded system to a cloud structure by proposing a protocol using a JSON packet using TCP sockets.

References

[1] J. Weinman, “The Strategic Value of the Cloud”, IEEE Cloud Computing, vol.2, pp 66-70, 2015

[2] W., Peng Xu, L. T. Yang, Secure Data Collection, Storage and Access in Cloud-Assisted IoT, IEEE Cloud Computing, 2018

[3] A.Alshehri, R. Sandhu, “Access Control Models for Virtual Object Communication in Cloud-Enabled IoT”, IEEE Computer society, pp:16-25, 2017

[4] H.Truong, S.Dustdar, “Principles for Engineering IoT Cloud Systems”, vol.2, pp.68-76, 2015.

[5] S.Nastic, H. Truong, S. Dustdar, “A programming model for resource-constrained iot cloud edge devices”, Banff, IEEE international Conference on systems, 2017

[6] E. Lingg, G. Leone, K. Spaulding, R. BFar, “Cloud based employee health and wellness integrated wellness application with a wearable device and the HCM data store”, Internet of Things (WF-IoT), Seoul, 2014

[7] N. Fujii, N. Koike, A FPGA-based Remote Laboratory in the Hybrid Cloud, Chester, Cyberworlds , 2017

[8] G.Merlino, D.Bruneo, F.Longo, A,Puliafito, S.Distedano, “A Novel Paradigm for Smart Cities through IoT Clouds”, IEEE press, 2015

[9] M. Lekic, G. Garadasevic, “IoT sensor integration to Node-RED platform”, East Sarajevo, Infoteh-Jahorina, 2018

[10] Y. Obuchi, T. Yamasaki, K. Aizawa, S.Toriumi, M.Hayashi, “Measurement and evapuation of comfort levels of apartments using IoT sensors”, Las Vegas, Consumer Electronics (ICCE), 2018

[11] A. Stanford-Clark, H.Linh Truong, “MQTT For Sensor Networks (MQTT-SN)”, Protocol Specification, November 14, 2013, IBM.

[12] R. Preethika , G. Prabhu, A. Prabhu, “AN ANALYSIS OF COMMUNICATION WITH IOT DEVICES USING WEBSOCKETS”, 2017

[13] Frédéric Camps” , Connected objects, IoT, M2M, architecture, protocols”, 2017

[14] Postscapes, “IoT Standards and Protocols”, 2018

[15] A. Botta, W. de Donato, V. Persico, A. Pescap´e, “Integration of Cloud Computing and Internet of Things: a Survey”, September 18, 2015

[16] Cloud Standards Customer Concil, “Cloud Customer Architecture for IoT”, 2016

[17] Yu Liu1 , B. Dong , B. Guo , J. Yang and W. Peng, “Combination of Cloud Computing and Internet of Things (IOT) in Medical Monitoring Systems”, 2015

[18] STMicroelectronics, “Wireless connectivity for IoT applications”, 2015

[19] Robert Love, “Linux System Programming”, 2007 O'Reilly Media

[20] Eclipse Paho, “MQTT C++ Client for Posix and Windows”, 2018

[21] debihiga.wordpress.com, “Using MQTT in Python, C and C++ in embedded and non-embedded systems”, 2018

[22] developer.android.com, “Developer Guides Android”, 2018

[23] techtutorialsx.com, “ESP32 Arduino Websocket server: Receiving and parsing JSON content”, 2017

[24] hackster.io, “Temperature Dashboard Using Arduino UNO”, ESP8266 And MQTT, 2016