THE ANNALS OF "DUNĂREA DE JOS" UNIVERSITY OF GALATI
FASCICLE III, 2020, VOL. 43, SPECIAL ISSUE, ISSN 2344-4738, ISSN-L 1221-454X
ELECTROTECHNICS, ELECTRONICS, AUTOMATIC CONTROL, INFORMATICS
Article DOI: https://doi.org/10.35219/eeaci.2020.2.03

COMMUNICATION IN A SMART HOME PLATFORM

Iosef Georgian

Computers and Information Technology - Advanced Informatics Technologies, Faculty of Automation, Computers, Electrical and Electronic Engineering - "Dunărea de Jos" University of Galati, Romania

Abstract: Smart things and IoT have become a normal thing in our daily routine. Without even realizing, we use them, interact with them and benefit from generated comfort. Starting from this reality, the current article proposes a deeper dive into IoT world creating a smart-home platform. Implemented solution provides real-time interaction between the house owner and his home environment, offering possibility to control various elements of the smart-home ecosystem and get feedback from the platform.

Keywords: Internet of Things, real-time, smart-home, Message Queuing Telemetry Transport

1. INTRODUCTION

Smart things are everywhere. From the phones in our hands to the traffic management systems or various solutions to predict diseases, they provide comfort, economy of resources and a better living. The idea to create a smart-home platform comes as something naturally in a world full of technology, a world constrained by time but concerned about the environment. An intelligent system meant to manage a house in a manner of optimizing resources and offers comfort to the owners must provide a climate control system, lighting and other devices control system, plant watering and prevention of possible accumulations of harmful gases. All of this must be accomplished in a secure mode and in real-time.

2. REAL-TIME

In the last years the technology has evolved to a level that we have never met before and human expectations has evolved exponentially with it, giving birth to new terms to describe these expectations. "Real-time" describe a very low latency communication between an emitter and a receptor. In our case, real-time means a very short time between data collection and data delivery between various smart-home platform components or between smart-home platform and end-user.

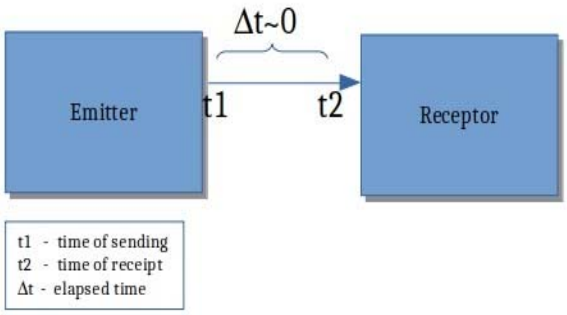

Fig. 1. Latency in communications

Communications protocols are meant to define a set of rules and norms that allows two or more entities to communicate, to transmit data, and may be implemented by hardware, software or a combination of both.

In this paper we will analyze and implement a solution based on two main protocols which helps to create complex communications systems in the world of Internet of Things: MQTT and Socket.IO/WebSockets.

2.1 MQTT

History of MQTT protocol begins in 1999, when two engineers, IBM's Andy Stanford-Clark and Cirrus

Article DOI: https://doi.org/10.35219/eeaci.2020.2.03

Link's Arlen Nipper, published first version. They had developed MQTT as a method to keep machine-to-machine communication on networks with limited bandwidth or unpredictable connectivity (www.u-blox.com).

Because of its lightweight and a very small code footprint, MQTT was considered as being ideal for low-power devices or those with battery, like smartphones or various sensors. MQTT isn't the only publish-subscribe (Pub/Sub) real-time messaging protocol from the market, but it has advantage of being widespread adopted in machine-to-machine communication.

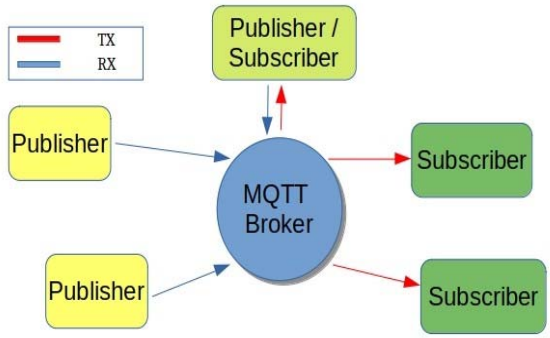

Fig. 2. MQTT working principle

A communication system built on MQTT consists of the broker and one or more clients. Every client can publish or subscribe to any topic, meaning clients can send data or receive data according to which channel has published or has subscribed.

A very useful characteristic of the MQTT broker is the fact they can be setup to accumulate messages related to channels that have no current subscribers. The accumulated messages will either be discarded or retained, depending on instructions in the control message. This is very helpful when some subscribers were disconnected in an unexpected mode. When they are reconnected, it requires the most recent recorded data, instead of waiting until the next message on desired topic is published.

Every exchanged message using MQTT protocol is defined by two components:

1. The first Byte stores data about message type (the client requests to connect, pings request etc.), a duplication flag, instructions for messages and information on the quality of service level (QoS).

2. The second Byte stores information on the remaining length of the message, the payload and any other optional header data. (www.u-blox.com, http://www.steves-internet-guide.com).

The main functionality which MQTT supports, quality of services, relies on a flag from the Byte 1 of the message, QoS. QoS flags may have the following values based on the intent and urgency of the message:

1. 0 means "at most once". Server sends and forgets and the messages may be lost or duplicated;

2. 1 means "at least once". The recipient confirms the delivery (ACK), the messages may be duplicated, but delivery is assured;

3. 2 equivalent for "exactly once". The server ensures that the delivery has been made and messages arrived precisely once without loss or duplication.

2.2 Socket.IO

Socket.IO was created in 2010 by Guillermo Rauch to use open connections to facilitate real-time communication.

Talking about Socket.IO means talking about WebSockets. WebSocket is the communication protocol which provides bidirectional communication between the client and the server over a TCP connection. It remains open all the time so it allows real-time data transfer.

When the client triggers the request to the server it does not close the connection after receiving the response, but it rather keeps the connection open. At the beginning, Socket.IO creates a long-polling connection using *xhr*-polling, then, after establishing this connection, it upgrades to the best connection method available which most frequently is WebSockets connection (https://socket.io/blog, https://www.npmjs.com). Then, when the sockets are used for communication, if the server receives a new message, it will send it to the clients and notify them, eliminating the need to send requests between clients and server, a channel for communications being already established.

One of the main advantages of using Socket.IO is the fact that has implemented automatic reconnect in case of unexpected disconnect and has support for broadcasting messages.

When are intending to send data over WebSocket (https://tools.ietf.org) protocol, an endpoint must perform following steps:

Article DOI: https://doi.org/10.35219/eeaci.2020.2.03

- The endpoint ensure the WebSocket connection is in the OPEN state and if in any moment the WebSocket connection changes his state, the endpoint will abort the following steps;
- Must encapsulate data in a WebSocket frame and if a big volume of data needs to be sent, may encapsulate the data in a series of frames;
- The opcode (frame-opcode) of the first frame containing the data must be set to the appropriate value for data that is to be interpreted by the recipient as text or binary data;
- The frame which has been formed will be transmitted over the underlying network connection.

3. IOT PROTOCOLS IN SMART-HOME ENVIRONMENT

When we speak about smart home, we speak first about a personal space where safety and comfort are needed. All of this can be accomplished by having real-time environmental data which help us to be in control of our private living zone.

To implement a system based on IoT (https://www.oracle.com/ro) protocols, easy to be maintained and upgraded, we chose Raspberry Pi platform organized as satellites and central station and Python programming language for all system levels, from the lowest to the highest.

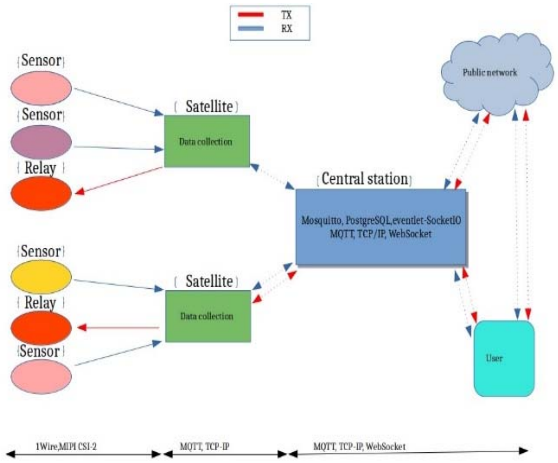

Fig. 3. Smart-home platform workflow

At the lowest level we collect data from the various sensors, as can be seen in Figure 4, where we use a function for read temperature sensors data, we process that information and using MQTT protocol we send all the information to the next layer of the smart-home platform.

```
def get_temperature():
    datastring = {}
    for sensor in glob.glob("/sys/bus/w1/devices/28*/w1_slave"):
        id = sensor.split("/")[5]
        try:
            f = open(sensor, "r")
            data = f.read()
            f.close()
            if "YES" in data:
                (discard, sep, reading) = data.partition(' t=')
                t = float(reading) / 1000.0 #reports temperature
                #datastring = datastring + id + ':' + str(t)
                datastring[id] = t
        except:
            pass
    return json.dumps(datastring)
```

Fig. 4. Python code for temperature reading

The most complex part of the application is at the level of central station where we collect all information from the satellites and user inputs and make them all to work together. At the central station level the MQTT broker is running which manage all the communications with the satellites. A RDBMS which is represented by PostgreSQL stores information in a persistent mode for future analyze, and Python Flask framework which act as an interface between data collecting and processing level and the user.

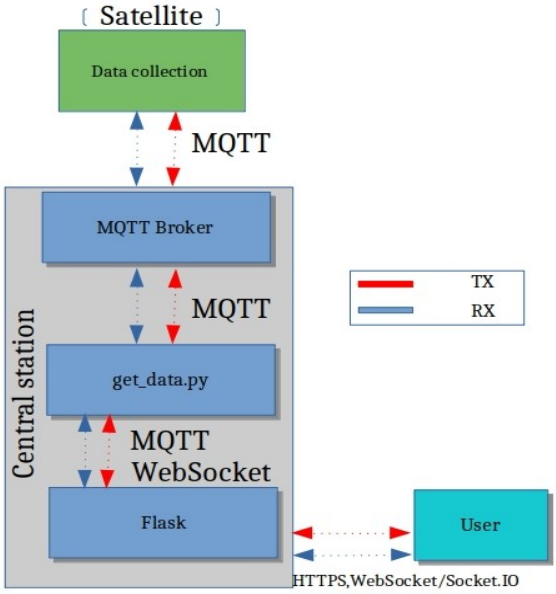

Fig. 5. Message transport - central station level

Because internet browsers don't have MQTT support built-in we need an adapter which, in our case, is represented by flask-socketio package who add Websockets support in Flask framework (https://blog.miguelgrinberg.com).

Article DOI: https://doi.org/10.35219/eeaci.2020.2.03

For a better understanding, at the level of central station, as it can be seen on Figure 5, a Python script receive data from the MQTT broker and routes that information through a channel to the frontend zone, as is shown in the code from Figure 6. In the frontend, a JavaScript Socket.IO library receives data and presents it to the user through graphical user interfaces.

```python
def on_message(client, userdata, message):
    ourMessage = message.payload.decode("utf-8")
    ourTopic = message.topic
    if "temperatura" in ourTopic:
        config_pairings = configparser.ConfigParser()
        config_pairings.read('/smart-home-reborn/app/pairing.conf')
        #sensors defined/paired in our database
        known_sensors =  dict(config_pairings.items('sensorspairing'))
        decodedMessage = json.loads(ourMessage)
        #if it's a topic for temperature readings then
        #proceed with themperature processing and emiting
        get_rooms_sensors_readings(decodedMessage,known_sensors)
    elif "relaystate" in ourTopic:
        socketio.emit('relay_state',ourMessage)
    elif "camerastatus" in ourTopic:
        print(ourTopic,ourMessage)
        socketio.emit('camera_state',ourMessage)
```

Fig. 6. Python - parse data from MQTT to front-end

On the client side, a few lines of code are needed to intercept information and deliver it. We have to establish a connection with the server and then, on a defined event, execute a function to extract data we need from the received message, like in the code from Figure 7.

```
{% block scripts %}
   <script type="text/javascript" charset="utf-8">
  $(document).ready(function(){
// start up the SocketIO connection to the server
      var socket = io.connect(
                  'https://' + document.domain +
                  ':' + location.port);
      socket.on('room_temperature', function(msg) {
        var nDate = new Date();
          if(typeof msg.exterior !== "undefined") {
            exteriorRealTemperature = msg.exterior;
          }
        $('#temperature').text(msg.exterior).html();
      });
  });
</script>
{% endblock scripts %}
```

Fig. 7. Socket.IO temperature receive and display

Once the environmental data is exposed to the final user through the graphical user interface, the user can decide what to do next. He can manage, in manual mode, some aspects of the home automation platform or can instruct the platform to manage everything in real-time such that there is almost no difference if the user is physically in the house or he is remote.

We can see in Figure 8 how the state of the broker is checked and, if this is online, how is setup a client instance which will send user command on the right channel to the backend, at the satellite level to be executed (eg. lights on or off).

```python
def send_command(topic,command):
    print("Creating new instance")
    def on_connect(client, userdata, flags, rc):
        if rc==0:
            print("Connected to broker: ",broker," : ",
                  port," Topic: ",topic, "Command: ", command)
        else:
            print("Connect error : ", rc)
    if checkIfBrokerRunning('mosquitto'):
        # Setup SocketIO
        client = paho.Client("SendDataDevel",clean_session=True,
                             protocol=paho.MQTTv31,
                             transport='websockets')
        client.username_pw_set(username="publish",
                               password="publishpassword")
        client.tls_set(ca_certs=location+"ca.pem",
                       certfile=location+"client.pem",
                       keyfile=location+"client.key",
                       cert_reqs=ssl.CERT_REQUIRED,
                       tls_version=ssl.PROTOCOL_TLSv1, ciphers=None)
        client.tls_insecure_set(False)
        client.max_queued_messages_set(3)
        print("Connecting to broker to send relay command!")
        client.on_connect=on_connect
        client.connect(broker,port) #connect to broker
        client.loop_start()
        client.publish(topic,command,0,retain=False)
        time.sleep(1)
        client.disconnect()
    else:
```

Fig. 8. Python - Get user input and send to satellite

4. CONCLUSIONS

Real-time communication is very useful in a world in constant motion and full of low-powered devices. IoT protocols can be extended and used in other fields than personal home comfort or security. IoT begins to be present in hospitals, train stations or irrigation of various crops in an automatic and optimized way.

5. ACKNOWLEDGEMENT

This paper was developed as part of the Master thesis.

6. REFERENCES

https://www.u-blox.com/en/blogs/insights/mqtt-beginners-guide-2020

https://socket.io/blog/introducing-socket-io-1-0/

https://blog.miguelgrinberg.com/post/easy-websockets-with-flask-and-gevent

http://www.steves-internet-guide.com/mqtt-protocol-messages-overview/

https://www.oracle.com/ro/internet-of-things/what-is-iot.html

https://www.npmjs.com/package/socket.io

https://tools.ietf.org/html/rfc6455