LAB ASSIGNMENT 5

IoT System using MQTT Protocol

MQTT Broker: We implemented our own mqtt broker. It hosts a TCP socket at the port 1883. Publishers and subscribers connects to that socket and communicates with the broker using MQTT protocol. Main loop of the broker listens for new connections. Before the main loop broker runs a thread for waiting messages. When a new message arrives, broker parses MQTT header and operates neccessary operations by looking at the header then sends a corresponding ACK message back to the client.

Project Description: We implemeted an IoT system using MQTT protocol. It consists of 3 parts; Broker, Publisher, Subscriber. Broker is a server that listens packets from publishers and subscribers. Publishers and subscribers connects to the broker using TCP/IP protocol. Publishers publish some messages under a topic and subscribers subscribe to these topics to recieve the messages when published. Publishers and subscribers doesn't communicate directly to each other. They communicate with the broker and broker acts like a middleman. We used windows terminals to imitate sensors and output devices.

To publish in publisher.py, type to the console:

PUBLISH {topic-name} {value}

To subscribe in subscribe.py, type to the console:

SUBSCRIBE {topic-name}

To unsubscribe:

UNSUBSCRIBE {topic-name}

To disconnect:

DISCONNECT

Problems Encountered: Creating header and parsing it was a bit tricky since it requires some bitwise operations but after trail and error we managed to overcome that problem.

Project Team:

Mert Ertürk 1700002485 (on the left)

Can İşcan 1800001775 (on the right)

Publisher:

```
import socket

import threading

import mqtt


# Global Variables

IP = socket.gethostbyname(socket.gethostname())

PORT = 1883

BUFFERSIZE = 1024
```

```
MSGENCODING = 'utf-8'


def sendTh(MQTTclient):
    while True:
        inpt = input("> ").split(' ', 2)

        if (len(inpt) == 3):
            if (inpt[0].upper() == 'PUBLISH'):
                MQTTclient.publish(inpt[1] + ' ' + inpt[2])

        elif (len(inpt) == 1 and inpt[0].upper() == 'DISCONNECT'):
            MQTTclient.disconnect()
            MQTTclient.sock.close()
            break

# Entry Point of the Program
def Main():
    sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    sock.connect((IP, PORT))

    MQTTclient = mqtt.MQTTclient(sock)

    th = threading.Thread(target=sendTh, args=(MQTTclient,))
    th.start()

    MQTTclient.connect('Publisher')

    while True:
        try:
            responseMsg = mqtt.MQTTPackage().parseMsg(sock.recv(BUFFERSIZE))
            print(responseMsg)
        except Exception as e:
            print(e)
```

```
            break


    sock.close()


if __name__ == '__main__':
    Main()
```

Subscriber:

```
import socket
import threading
import mqtt
# Non-Durable


# Global Variables
IP = socket.gethostbyname(socket.gethostname())
PORT = 1883
BUFFERSIZE = 1024
MSGENCODING = 'utf-8'


def sendTh(MQTTclient):
    while True:
        inpt = input("> ").split()


        if (len(inpt) == 2):
            if (inpt[0].upper() == 'SUBSCRIBE'):
                MQTTclient.subscribe(inpt[1])


            elif (inpt[0].upper() == 'UNSUBSCRIBE'):
                MQTTclient.unsubscribe(inpt[1])


        elif (len(inpt) == 1 and inpt[0].upper() == 'DISCONNECT'):
            MQTTclient.disconnect()
            MQTTclient.sock.close()
```

```
            break


# Entry Point of the Program
def Main():
    sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    sock.connect((IP, PORT))

    MQTTclient = mqtt.MQTTclient(sock)

    th = threading.Thread(target=sendTh, args=(MQTTclient,))
    th.start()

    MQTTclient.connect('Subscriber')

    while True:
        try:
            responseMsg = mqtt.MQTTPackage().parseMsg(sock.recv(BUFFERSIZE))
            print(responseMsg)
        except Exception as e:
            print(e)
            break

    sock.close()

if __name__ == '__main__':
    Main()
```