A Project Report

On

MQTT-S – A Publish/Subscribe Protocol For Wireless Sensor Networks

Submitted in requirement for the course

ADVANCED COMPUTER NETWORKS (CSN-503)

of Bachelor of Technology in Computer Science and Engineering

by

ANKUSH PATEL (Enroll No. 15114012)

KANISHK GOYAL (Enroll No. 15114035)

UJJAWAL (Enroll No. 15114074)

to

Dr. P. Sateesh Kumar

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY, ROORKEE

ROORKEE- 247667 (INDIA)

TABLE OF CONTENTS

Introduction

Wireless Sensor Networks (WSNs) have been gaining increasing attention, both from commercial and technical point of views, because of their potential of enabling of novel and attractive solutions in areas such as industrial automation, asset management, environmental monitoring, transportation business, etc. Many of these applications require the transfer of data collected by the sensors to applications residing on a traditional network infrastructure. The WSNs need to be integrated with the traditional networks.

But within the WSNs, a large number of battery-operated Sensor/Actuator (SA) devices, usually equipped with a limited amount of storage and processing capabilities have to be installed with low power consuming protocols for data transfer. On the other hand, there are many problems associated with WSNs. SA devices may change their network addresses at any time. Wireless links are quite likely to fail. Applications can be hosted and run on any machines anywhere in the Integrated Wireless Sensor Networks traditional network.

The problem described above is solved by using a well known data-centric communication approach- Publish/Subscribe messaging system, in which information is delivered to the consumers not based on their network addresses, but rather as a function of their contents and interests. . These features are achieved by decoupling the various communicating components from each other such that it is easy to add new data sources/consumers or to replace existing modules

Publish/Subscribe Systems

The main purpose of the publish/subscribe (pub/sub) communication model in MQTT-S is for the decoupling of the various communicating components from each other. The components which are interested in consuming certain information register their interest. This process of registering an interest is called subscription, the interested party is therefore called a subscriber. Components which want to produce certain information do so by publishing their information and are called publishers. The entity which ensures that the data gets from the publishers to the

subscribers is the broker. The broker coordinates subscriptions, and subscribers usually have to contact the broker explicitly to subscribe.

The set of publishers and subscribers that are coupled together by matching topics can dynamically change over time without the subscribers or publishers being aware of this.

MQTT - Message Queuing Telemetry Transport

The MQTT protocol provides a lightweight method of carrying out messaging using a publish/subscribe model. This makes it suitable for Internet of Things messaging such as with low power sensors or mobile devices such as phones, embedded computers or microcontrollers. MQTT is designed in such a way that its implementation on the client's side is very simple. All of the system complexities reside on the broker's side. MQTT does not specify any routing or networking techniques; it assumes that the underlying network provides a point-to-point, session-oriented, auto-segmenting data transport service with in-order delivery (such as TCP/IP) and employs this service for the exchange of messages.

MQTT-S

❖ Architecture

There are two types of components: MQTT-S clients and MQTT-S gateways (GWs). MQTT-S clients are on the WSN side and enable the SA devices to access the pub/sub services of a MQTT broker located on the traditional network. They connect to the gateway using the MQTT-S protocol, and the gateway connects to the broker using MQTT. There can be two types of gateways - Aggregating or Transparent. Transparent gateway is simple to implement but Aggregating is more feasible in terms of scalability.

The below diagram shows how both of them work differently.

- ### Support for Multiple Gateways

  Because of high risk of link failures in WSNs, it is desirable to have at least two gateways for uninterrupted connection to the broker. Another benefit of having multiple gateways is that it helps in balancing the load between multiple gateways. If any one gateway is overwhelmed, the sensors can connect to the other gateways.

- ### Support of short message payload

  Since the control overhead allows only around 60 bytes of data to be transmitted in one packet, the CONNECT messages are divided into several small message fragments and the PUBLISH messages are allowed to have larger payload by minimizing the control information, such as giving topic IDs to all the topics.

Work Description

This project works on multiple instances of clients running on a single device only, but we tried our best to understand the various problems associated with WSNs and how MQTT-S helps in solving them. We have used Node.js to implement the gateway and client which communicate through different ports on the same device. Following is the list of files/modules involved in this project.

- Index.js File

  This acts as MQTT-S gateway which connects the clients to the broker(Mosquitto Server). We have used MongoDb as well to store the previously published information.

- Client.js File

  This implements a simple client where a client can either publish some information or gather previously published information on the basis of interests.
- Eclipse Mosquitto

  We have used this as our broker. It is a open source, lightweight message broker that implements MQTT.
- Index.html File

  This is the UI for client.

Conclusion

This project has been a great source of learning for us in the recent days. The implementation helped us in learning how MQTT and other publish/subscribe models work, about the major problems in WSNs and how to deal with them with the help of MQTT-S.
Although, we cannot simulate the exact behaviour of WSNs on our laptops, we have tried our best to understand the different underlying concepts by implementing Gateway and Client in such a way that both are on the same device but communicate through different ports.