Trinity College Dublin
Coláiste na Tríonóide, Baile Átha Cliath
The University of Dublin

School of Computer Science and Statistics

I have read and I understand the plagiarism provisions in the General Regulations of the University

Student Name	Tianze Zhang
Student ID Number	19334401
Course Title	Computer Science
Module Title	CSU33031 Computer Networks
Lecturer(s)	Stefan Weber
Title	IoT Publish-/Subscribe Protocol – Assignment 1 Report
Date Submitted	02/11/2021
Word Count	2000

Contents

1 Introduction

The aim of the report is to design protocol which supporting broker-mediated publish-/subscribe model for end processes based on UDP datagrams. The protocol supports topic-based publishing and subscription with broker as a coordinator to decouple them. This report provides the detailed description on the problem analysis, design, implementation and evaluation.

The layered views are applied when analysing and designing the solution to resolve the problem: From the high-level network application processes and low-level network communication protocol. From the application layer point of view, the three key components named publisher, subscriber, and broker are defined, while from network point of view, the customised UDP protocol is specified to support the communication among these components.

The docker containerized based simulated network environment is built for the experimental implementation and testing. Wireshark as a packet sniffer and analysis tool, is used to capture network traffic and analyse the packet. The description on the proposed use cases and their implementations are also provided in the report.

Further evaluation and discussion on the protocol are also proposed.

2 Design Consideration

As the problem is described as topic-based publish and subscribe interaction style with broker as a middle man to coordinate and decouple the publisher and subscriber. In this sense, the application can break down three key components(processes) based on the roles they play, publisher, subscriber and broker. When considering these components (end processes) communicate with each other through the network, a specific protocol which supporting the topic-based packets is required and agreed among them, from this point of view, a network protocol design is the key for this solution, specifically the customised UDP protocol for the IoT scenario.

In this section, the design consideration on high level key applications components and network protocol (Transport Layer - UDP) are explained. The experimental environment setup is also described.

2.1 Problem statement and analysis

It is topic-organized publish and subscribe style solution, decoupled with broker.

2.2 Key Challenges

- Protocol supporting publish and subscribe model
- Protocol appropriate for IoT device communication
- Network environment to simulated the proposed application environment.

2.3 Theory of Topic

2.3.1 Application point of view

- Broker: It is a manager, which decouple the publisher and subscriber, also manages the topics. It also plays the central role in the architecture.
- Components: publisher, subscriber.
- Application processes: using Socket programming with UDP, together with port numbers for different application processes.

2.3.2 Network Communication and Protocol Design

- Data Link/IP Layer: handled by docker network with docker bridge.
- UDP protocol: UDP packet design with customised headers to support topic related info.

2.4 Key Components

- Publisher: response to publish message to the topic. Only interact with broker.
- Broker: Middle man for publishing and subscription, decouple the subscriber and publisher. Centralized manages the topics. Play as coordinator and manager roles.
- Subscriber: Interest and subscribe to the topics, messages get pushed by the broker.

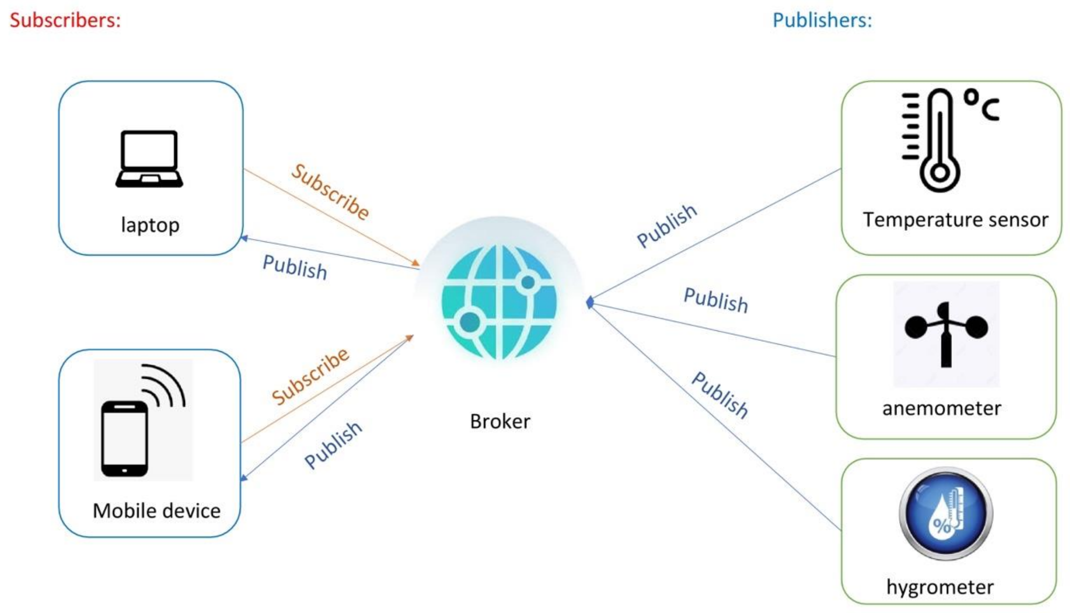
2.5 Environment Setup

- Docker: utilize docker bridge to create simulated environment on the local machine. The physical, data link and IP layer (routing, ip tables) are automatically provided by docker network and bridge. The running containers are the simulation of the network nodes. (p52, Rajdeep)
- Wireshark for packet sniff, and protocol analysis.

3 Implementation

My implementation contains 6 components: 1 broker and 2 subscribers (Subscriber.java and SubscriberA.java) which corresponds to a laptop and computer device, and 3 publishers (Publisher.java PublisherA.java and PublisherB.java) which corresponds to a temperature sensor, anemometer and hygrometer. A laptop/mobile device subscribes for data with a given topic “temperature”, “wind speed”, “humidity”. Data that is published by any of the publishers after subscription will be forwarded back to the laptop/mobile device.

Each component runs in a separate container in docker and is connected to the same docker network(cs2031).



3.1 Publisher

The publishers only have the functionality to send packets but they never receive. Therefore, the publishers don’t have its own port number but they need to know the broker’s port number. Each publisher works the same but sends a different topic. They pack the data with the header, and send the packets to the broker using sockets.

3.2 Subscriber

Unlike Publishers, the subscribers need to not only to be able to send packets but also to receive packets. Therefore, each subscriber is assigned to a different port number: the mobile device has a port number of “12349” while the laptop has a port number of “12346”. The subscribers work as follows:

- The subscriber asks the user what topic they want to subscribe to, then the packet is made differently by the subscriber depending on which topic the user enters.
- The packet is then sent to the broker and the subscriber will wait for packets to be sent back from the broker.

- When it receives packet from the broker, it unpacks the packet and outputs the data in console.
- It gives the user an option to subscribe to a new topic or continue waiting for packets based on the currently subscribed topics.

3.3 Broker (manager)

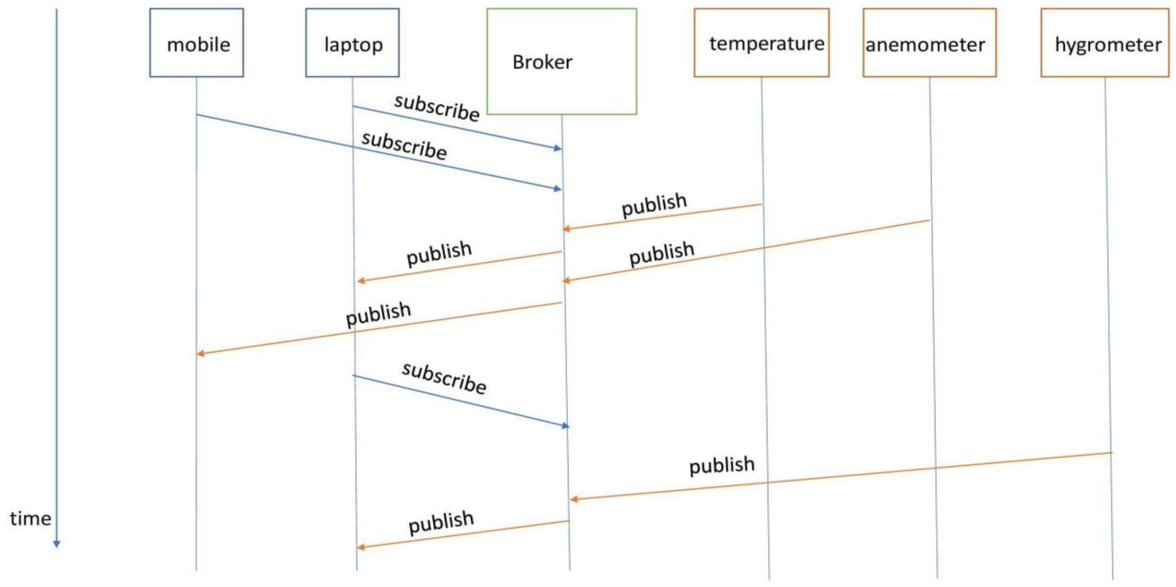
The broker acts like a manager because it needs to manage all packets sent from the subscriber and publisher. The Broker consists of both the functionalities of sending and receiving. Its port number is 12345. The Broker repeatedly listens for a packet, so that the subscriber and publishers are able to send to the Broker at any time. The Broker manages the packets as follow:

- It unpacks the packet and check whether the packet comes from a publisher or a subscriber.
- If the packet comes from a subscriber, the broker looks at the topic the subscriber wants to subscribe to, and then adds the subscriber to the list of subscribers subscribing to the same topic.
- If the packet comes from the publisher, the broker looks at the topic it is publishing, then forwards the packet to all of the subscribers subscribing to this topic, using their port numbers from the header of each subscriber in the list.

3.4 Process and Interaction

The system could interact in different ways. The process I'm demonstrating is as follows:

1. The laptop user subscribes to temperature and sends it to broker.
2. The mobile user subscribes to the wind speed and sends it to broker.
3. The temperature sensor publishes to the broker.
4. The broker forwards temperature to the laptop.
5. The anemometer publishes to the broker.
6. The broker forwards wind speed to the mobile user.
7. The laptop user wants to make another subscription to the humidity and sends to the broker.
8. The hygrometer publishes to the broker.
9. The broker forwards the humidity to the laptop user.



3.5 Protocol and packet encoding

The packet is treated as a byte array and contains two different parts: the first 8 bytes which is the header, and the rest is the payload/data.

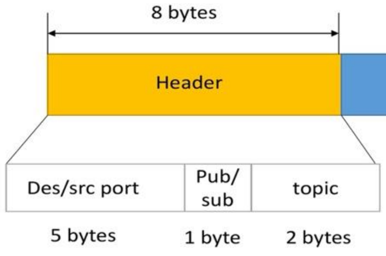
3.51 Customized header:

The header is customized, it is designed overall similar to a UDP header but much simpler. I designed the header as port number (5 bytes) + byte indicating publisher and subscriber + topic (2 bytes).

The port number is the first 5 bytes of the header. For a subscriber, its own source port is written in the header. However, for publishers it is the destination port(brokers port). The port number is chosen to be included in the header because it helps the broker know where the packet should be sent to.

The byte indicating publisher and subscriber follows directly after the port number (the 6th byte). This byte helps the broker to know whether the packets is sent from a publisher and a subscriber. In my implementation, this byte is “0” when it comes from a subscriber and “1” when it comes from a publisher.

The last two bytes are the topic. In my implementation, “01” represents temperature, “02” represents wind speed and “03” represents humidity.



3.52 Encoding

The way encoding works is as follows: The data/payload in string format is first converted into a byte array. The header is then constructed as 8-byte array. The header is then inserted in front of the payload’s array to form the packet.

3.53 Decoding

The way decoding works is as follows: the packet gets split into different parts: for example, header and payload. They are split and works on the part they are interested in.

4 Evaluation and Discussion

- Broker based architecture for IoT solution, which decouple subscriber and publisher and centralized topic management.
- UDP protocol in nature is speedy protocol (connectionless, no ACK, no ordering, and etc), which is needed for streaming and IoT applications. (Kurose, p229)
- Experimental Environment Limitation: Docker bridged based to simulate the nodes in a local machine, not the real internet environment.
- Disadvantage: UDP vs TCP on the transport layer. Trade-off analysis for different use cases.

- Reflection and Improvement: Security and Cache, such as supported by MQTT.

5 Summary

In this report, it covers the detailed design and implementation for the problem and IoT use cases. The design approach and considerations are also explained in detail and in-depth. Specifically, the detailed descriptions on the implementation are provided to facility to understand the solutions which have been done. The further evaluation and discussion are also suggested.

Reference

Kurose-Ross, 2017 Computer Networking. A Top-Down Approach, Seventh Edition, Pearson Publishing

Rajdeep Due, 2016 Learning Docker Networking, Packt publishing.