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

CSU33031 Computer Networks
Assignment #1: Publication/Subscription Protocols

Daniel Whelan, 19335045

November 1, 2021

Contents

1 Overall Design

In the first two sections I will discuss my understanding of Publishing, Subscribing and discussing the design of my network elements and packets sent across the network. I will then discuss the overall design of my implementation of these elements. Following this I will discuss the strengths and weaknesses of my design whilst giving a general summary and reflection on the overall assignment.

1.1 Publishing

Publishing in a Computer Network refers to the sending of instructions or information from one node in a network to other nodes that are present in this network. This is done by sending packets across the network that will be received by designated subscribers.

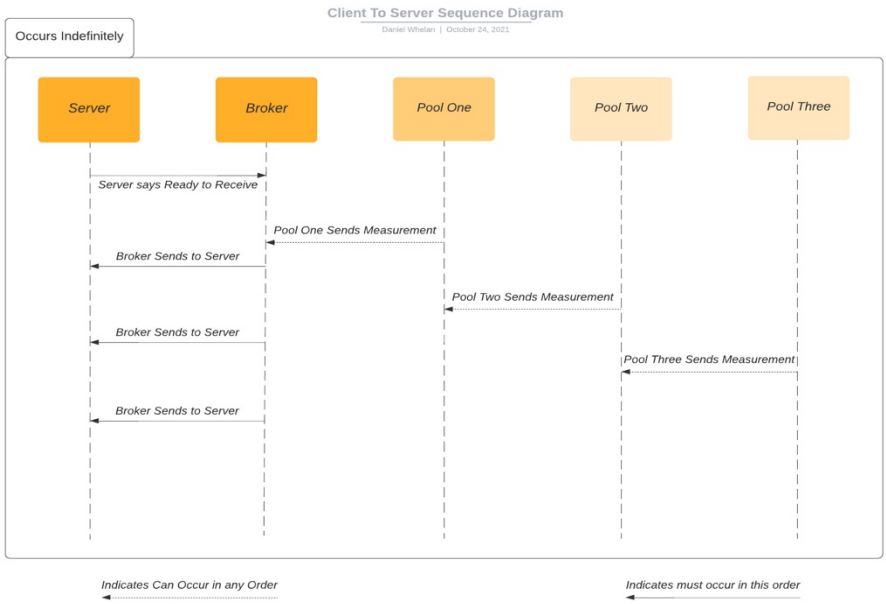

Figure 1: The figure above shows the general topology of the client publishing information to the server through the use of a broker in my designed protocol. The Client publishes information to the broker which in turn is sent to the Server.

1.2 Subscribing

Subscribing in a computer network involves the receiving of designated information that has been published into the network by another node(s). This is done by receiving packets that have been designated by a publisher for this exact subscriber.

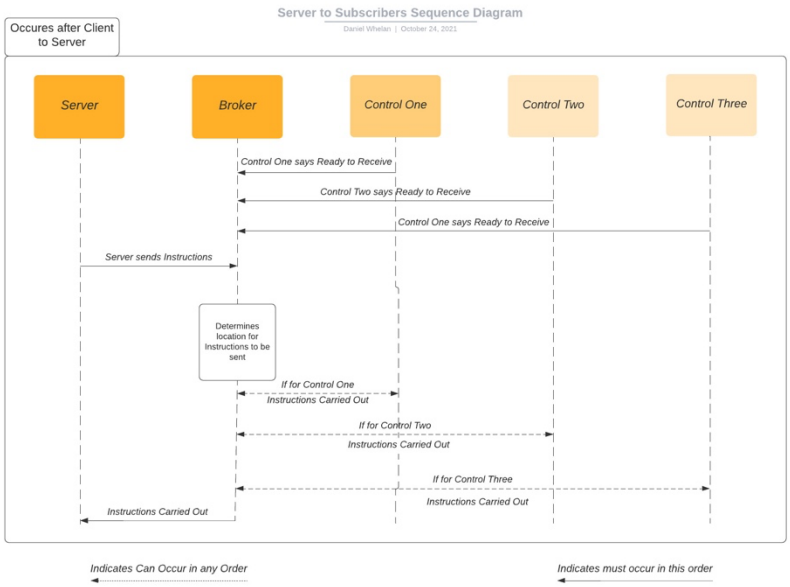

Figure 2: The figure above shows a general topology for a node subscribing to information published by the server through the use of a broker in my designed protocol. The Server publishes instructions to the broker which determines which node needs to subscribe to them, and in turn sends them to the designated subscribing node.

1.3 Design of Network Elements

My network consists of four coded elements which simulates 7 separate elements in a topology. There is a Client class which acts as the publisher of Chlorine Measurements in three separate swimming pools to a Server that subscribes to this information and determines whether these readings are too high, too low or a good measurement for the swimming pool. The Server then publishes the relevant instructions based on the recorded measurements to the Chlorine Controllers that are simulated in the Subscriber class which in turn carry out the instructions and reduce, increase or maintain chlorine levels in the designated swimming pool.

1.4 Packet Descriptions

My packets consist of two distinct elements; the header and the message. The message is self-explanatory and contains the message that is being sent across the network. The header is the more complicated piece of the packet consisting of: The Topic, which is the element that determines where in the network the packet is being sent and can be either Client, Broker, Server or Subscriber, The Sub-Topic, which is the specific area in that element of the network these instructions are to be sent, This is often, but not exclusive to, either Pool One, Pool Two or Pool Three, and The Message Length, which is how long the message being sent is.

2 Implementation

This section presents the implementation details of the individual network elements, the Client, the Broker, the Server and the Subscribers.

2.1 Client

The Client is where the program starts and its main purpose is to generate a measurement reading and a pool that this reading is from and send this information to the Server. It does this by generating random numbers between 0 and 30 for the measurement and 1-3 for the pool number

```
int measurement = generator.nextInt(UPPER_LIMIT); //UPPER_LIMIT = 30;
int poolNumber = generator.nextInt(POOL_THREE) + POOL_ONE;
//POOL_THREE = 3, POOL_ONE = 1
String content = String.valueOf(measurement);
```

Listing 1: At the Client, the program is generating a measurement that will be sent to the Server and the Pool Number that will determine the Sub-Topic that is associated with that packet.

The Constructor for Client sets out some important elements that relate to the class such as the destination address of the Broker alongside its own port number. Upon construction the start() method can be called inside which the generation of numbers occurs, the header is set up and the packet is sent to the destination address that is initialised in the constructor.

```
data[TYPE] = CLIENT; // TYPE = 0;
switch(poolNumber) {
case POOL_ONE:
        data[SUB_TOPIC] = POOL_ONE; //SUB_TOPIC = 1
        break;
case POOL_TWO:
        data[SUB_TOPIC] = POOL_TWO;
        break;
case POOL_THREE:
        data[SUB_TOPIC] = POOL_THREE;
        break;
default:
        System.err.println("ERROR: invalid subTopic");
}
data[MESSAGE_LENGTH] = (byte) content.length(); //MESSAGE_LENGTH = 2;
```

Listing 2: The initialising of the header of the packet. The Type is set to the relevant node, in this case Client. The Sub-Topic is set to the relevant pool that this packet is referring to. The Message Length is set to the length of the content that is being sent.

Once the header and packet are constructed they are sent to the destination address that has been initialised in the constructor, which in the case of this topology is the Broker. This start() method is then looped with a 15 second sleep after each iteration.

2.2 Broker

The broker is the main router of the network. It is here that all of the packets are sent and in turn the broker directs them to the correct node in the network. It is constructed with the port number that it must listen to in the network and is left running indefinitely until it begins receiving packets and rerouting them to their final destination.

```
switch(data[TYPE]) {
        case ACK: //ACK = 4
           System.out.println("Packet recieved");
           break;
        case CLIENT: // CLIENT = 2
           // make relevant response depending on sub-topic of packet.
              break;
        case SERVER: // SERVER = 3
           // make relevant response depending on sub-topic of packet.
              break;
        case SUBSCRIBER: // SUBSCRIBER = 4
           // make relevant response depending on sub-topic of packet.
              break;
        default:
          System.err.println("Error: Unexpected packet received");
          break;
      }
```

Listing 3: The Switch statement that is used to route the packets to their relevant locations based on the type that is contained in the packets header. The Type is stored as bytes in the initial element of the packet header.

Once the topic of the packet has been decided the sub-topic of the packet is then determined with a similar switch statement that is contained in each of the above cases of the encapsulating statement.

```
switch(data[SUB_TOPIC]) {
            case POOL_ONE:
               content = sendAck(packet, data);
               sendPacket(BROKER, POOL_ONE, content, serverAddress);
               break;
            case POOL_TWO:
               content = sendAck(packet, data);
               sendPacket(BROKER, POOL_TWO, content, serverAddress);
               break;
            case POOL_THREE:
               content = sendAck(packet, data);
               sendPacket(BROKER, POOL_THREE, content, serverAddress);
               break;
            default:
               System.err.println("ERROR: Unexpected Packet");
          }
          break;
```

Listing 4: The inner switch statement contained in each case of the routing switch statement. This particular one occurs when the packet is sent by the Client and the sub-topics are the separate pools that the measurements have come from.

Above it can be seen that two methods, sendAck and sendPacket, are called, these are called in every node within this topology and server as the two base methods for sending and receiving information in the topology.

```
private String sendAck(DatagramPacket packet, byte[] data) {
    try{
        String content;
        DatagramPacket response;
        byte[] buffer = new byte[data[MESSAGE_LENGTH]];
        System.arraycopy(data, HEADER_LENGTH, buffer, 0, data[MESSAGE_LENGTH]);
        content = new String(buffer);
        data = new byte[HEADER_LENGTH]; // HEADER_LENGTH = 3
        data[TYPE] = ACK;
        data[SUB_TOPIC] = 0;
        data[ACKCODE] = ACKPACKET; //ACKCODE = 2;
        response = new DatagramPacket(data, data.length);
        response.setSocketAddress(packet.getSocketAddress());
        socket.send(response);
        return content;
    } catch(Exception e) {
        e.printStackTrace();
        return "";
    }
}
```

Listing 5: The sendAck method. It takes in the packet that was sent and the data contained within the packet. Within this method the content of the received packet is extracted and stored in a String variable until returned at the end of the method. An Ack is sent from the method to the sender of the packet by creating a new packet containing the ACK topic.

```
private void sendPacket(byte type, byte subTopic, String content, InetSocketAddress dstAddress){
    try {
        byte[] data;
        DatagramPacket packet;
        lastPacketSent = type;
        data = new byte[HEADER_LENGTH + content.length()];
        data[TYPE] = BROKER;
        data[SUB_TOPIC] = subTopic;
        data[MESSAGE_LENGTH] = (byte) content.length();
        System.arraycopy(content.getBytes(), 0, data, HEADER_LENGTH, content.length());
        packet = new DatagramPacket(data, data.length);
        packet.setSocketAddress(dstAddress);
        socket.send(packet);
    } catch(Exception e) {
        e.printStackTrace();
    }
}
```

Listing 6: The sendPacket method. It takes in the topic of the packet, the sub-topic of the packet, the content to be sent in the packet and the destination address of the packet. It constructs the packet in a standard way, formulating the header followed by content in a byte array and in turn storing this in a packet to be sent to the relevant address.

The Broker continues this method of operation indefinitely until the network is shut down.

2.3 Server

The server acts as the measurement control system for this network of swimming pools. Packets containing measurements of chlorine levels are sent here by the Client via the Broker and in turn appropriate instructions are relayed by the Server to the Chlorine Control Subscribers once again via the Broker. The instructions that are sent depend on the reading of the measurement. If less than 5 parts per million (ppm), which means milligrams per litre, then the Server sends instructions to increase levels over 5ppm. If over 15ppm the Server sends instructions to lower readings to less than 15ppm but above 5ppm. If the readings are between 5 and 15ppm then the Server orders the controllers to maintain the current levels.

```
int measurement = Integer.parseInt(content);
String response = "";
if(measurement <= UPPER_CHLORINE_LIMIT && measurement >= LOWER_CHLORINE_LIMIT) {
        response = "Continue as normal";
}
else if(measurement > UPPER_CHLORINE_LIMIT){
        response = "Reduce Chlorine Levels to lower than " + UPPER_CHLORINE_LIMIT;
}
else {
        response = "Increase Chlorine levels to higher than " + LOWER_CHLORINE_LIMIT;
}

//UPPER_CHLORINE_LIMIT = 30
//LOWER_CHLORINE_LIMIT = 0
```

Listing 7: The portion of the sendPacket implementation for the Server in which the instructions to be sent to the Subscriber are determined based off of the measurement that was sent by the Client of the Chlorine measurements in a specified Pool.

The instructions that are sent by the Server are sent to the Broker and in tr=urn are routed to the Subscriber based off the Topic that is sent in the header of the packet

2.4 Subscriber

The Subscriber for this topology is a series of chlorine controllers that maintain the correct chlorine levels of the swimming pools that measurements are read from in the Client. The subscriber reads the instructions from the Server and in turn the adjustments that are required to be made are made by the relevant chlorine controller to the pool that measurement came from.

```
private void doInstruction(String content, byte[] data) {
    switch(data[SUB_TOPIC]) {
        case POOL_ONE:
            //Carry out instructions given
            break;
        case POOL_TWO:
            //Carry out instructions given
            break;
        case POOL_THREE:
            //Carry out instructions given
            break;
        default:
            System.err.println("Error: Pool does not exist in this system");
            break;
    }
```

Listing 8: The Switch statement that reads he Sub-Topic, i.e. the pool that the reading came from, and in turn orders the controller relating to that swimming pool to carry out the instructions that have been given to the controller by the Server.

After the instructions have been given the Subscriber sends a confirmation message to the Server to confirm that the instructions given have been followed by the relevant controller. The process of the network terminates after this message has been sent and waits for a new measurement to be sent by the Client into the network from another or the same swimming pool.

3 Discussion

The strengths of this program are its efficiency and scalability. Upon executing the program the packets are sent near instantly with no noticeable wait time and hence provide a good platform for scalability. This protocol is scalable enough to be implemented with a multitude of additional swimming pool nodes and could be easily implemented into an actual control network of a swimming pool by attaching the actual sensors and controllers to the Client and Subscriber classes respectfully. It, however, is a very simplistic system and the use of a simulation of multiple clients and subscribers rather than actual multiples does hinder the appeared viability of the protocol. Despite this the protocol is quite viable in this regard as through the use of Topics and Sub-Topics the destinations of all packets, regardless of source, can be easily discerned. The idea of simulating multiple Clients and Subscribers was done in order to prevent the repetition of code and also highlight the realism of the proposed topology as sensors and controllers would, in reality, be linked to a central node that sends information to the sensor.

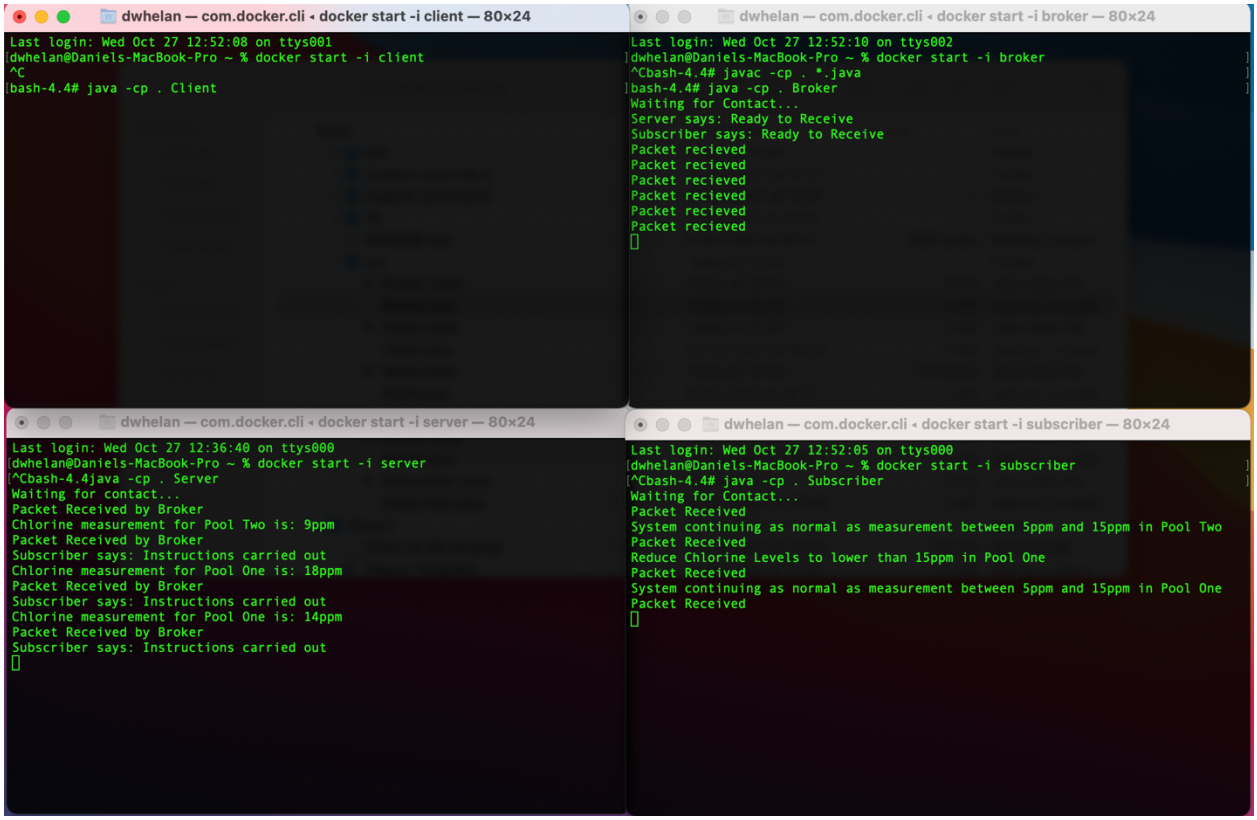

Figure 4 is a demonstration of the measurements from the Client being sent through the various network elements.

Above is an image of three iterations of the protocol. One can see the ready messages being sent to the Broker from both Server and Subscriber nodes. It can also be seen that the packet containing measurements is sent from the Client to the Broker and routed to the Server, then a packet containing instructions sent from Server to Subscriber via the broker and a confirmation packet being sent back across the same route. Below see images of some of the packet captures made by Wireshark during this transfer.

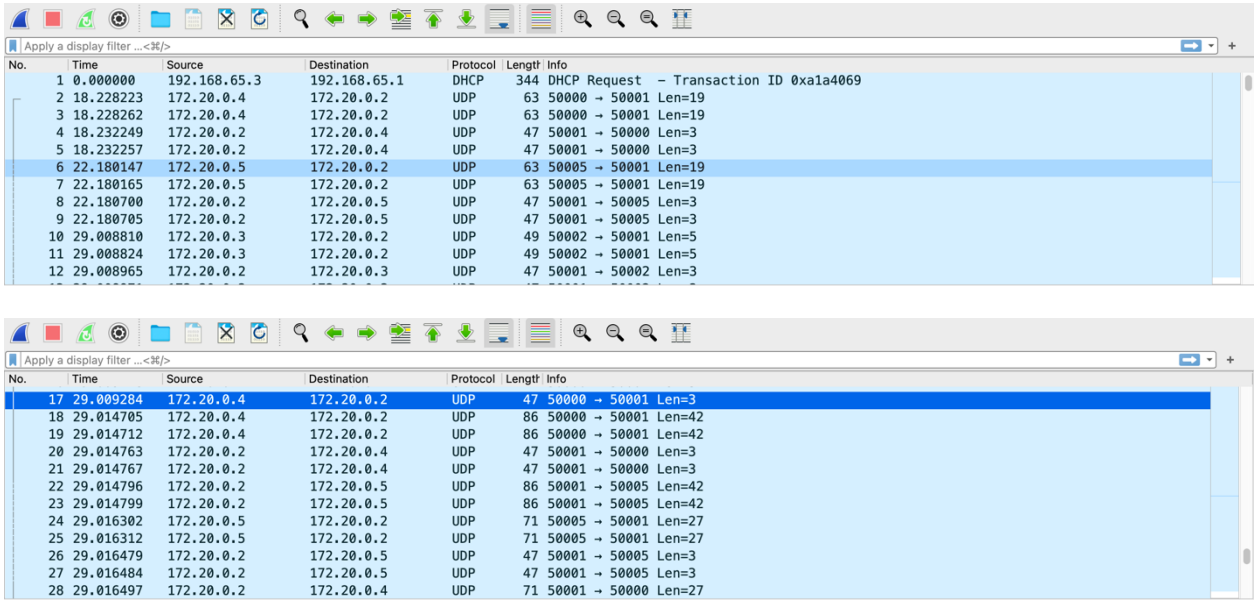

Figure 5: This figure shows some of the packets that make up the message exchange in Wireshark.

4 Summary

This report has described my attempt at implementing a Publish/Subscribe protocol for a system of Chlorine control in a series of swimming pools. It has highlighted the overall design of the topology that was created in order to complete the assignment and the methods that I used in order to emulate the network that was outlined in this document. The description of how the packets were designed was outlined and how the header was laid out, using Topic/Sub-Topic/Message Length as the basis of the organisation of the header. Furthermore, the classes that were used in each individual node were detailed and discussed and highlighted the programming choices that were made throughout the development of the topology.

5 Reflection

I found that the structure of these assignments was very helpful. The bi-weekly submission of a video demonstration and pcap files allowed for me to manage and maintain my time for the assignment and ensured that I was not only reaching the set deadlines but my own personal deadlines for the assignment. Overall I feel that I have learned a vast amount throughout this assignment. I have learned how to utilise docker to emulate the creation of a network and the communication that occurs between nodes within a topology. Furthermore I have learned the basics required for the creation of the header and content of a packet and how to send it using a program that I have written. As a final word, this assignment has definitely aided me in growing as a Software Developer.