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

# CSU3301 Computer Networks
# Assignment 1 Protocols

**Merlin Prasad Std# 19333557**

October 26,2021

# 1 Introduction

The assignment focused on designing a protocol that provides publish and subscribe mechanisms for processes based on UDP datagrams.

I decided to model my protocol of publishing and subscribing for a house with multiple rooms containing sensors. To demonstrate subscribing I have created a broker that can be subscribed to by a dashboard on a particular topic . Sensors publish information towards the broker and then the  broker forwards the correct packets to the dashboard. In addition to this I have created actuators that subscribe to the broker and a dashboard that can then publish instructions towards the actuators.

# 2.Theory of Topic

I researched socket programming and simple Client Server protocols to form a basis for my solution [1].  I also considered what information should be included in my packets to allow publishing and subscribing functionality.

## 2.1 Broker

The broker is the center of communication. It decides what packets to send onto subscribers based on the packet it receives from publishers. Broker sends acknowledgements (ACK) towards any device that published or subscribed to it.

## 2.2 Sensors

Sensors publish information towards the broker . There are two types of sensors in my protocol. Temperature sensors that measure the temperature in a room in degrees celsius and light sensor that measures the brightness of a room in lux.
There is a light and temperature sensor in each room of the house.

## 2.3 Actuators

Actuators subscribe to the broker. Similar to the sensor there are light and temperature actuators and one of each kind in the rooms. When they are published an instruction it means they have been told to make the room change its temperature or lux value.

## 2.4 Dashboard

Dashboards can subscribe and publish on a given topic. When a dashboard subscribes to a user defined topic or subtopic it will receive the data on the topic and tell the user whether the temperature or light reading in the room is adequate.
When a dashboard publishes it asks the user what topic and instruction to publish to the actuator.The dashboard handles most of the user interface.

## 2.5 Packet description

The following table explains the information enclosed in my header.

| Name | Description |
|---|---|
| Format | Shows whether the packet sent was a publish (pub) or subscribe (sub) format to let the broker know what type of connection is needed to be made. |
| Device | Describes the type of component sending the packet. For example sensors , actuators or dashboard . |
| Room | Tells what room the device sends the information from . There are currently 3 rooms in the building: pc room , bedroom and kitchen. |
| Topic | This is the subject the devices are measuring. My protocol deals with temperature (temp) and light topics. |
| Packet Info | The current value of the devices. Temperature is measured in degrees celsius and light is measured in lux. Current value for sensors is the number they measured in a room. Current value for actuators is their most recent instruction from the dashboard. |
| ID | The unique two digit code every device in my protocol has for easy identification |

## 2.6 Communication when dashboard is subscribing

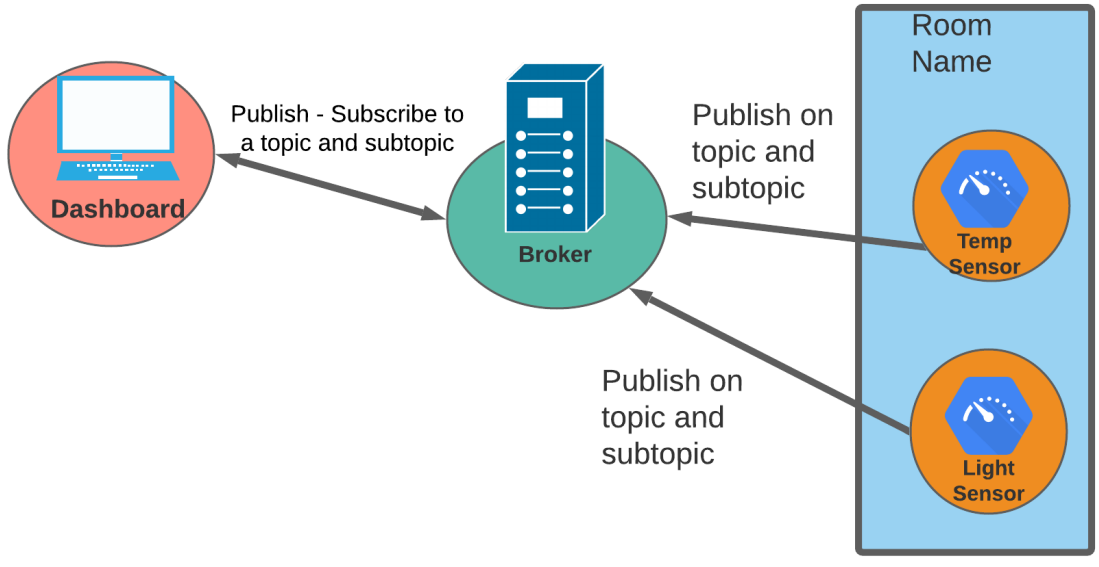


Figure 1 : A dashboard process can subscribe to a Broker on a given topic e.g “light”. This will mean any sensor that publishes to the broker on the light topic after this subscription will be forwarded to the dashboard by the broker.

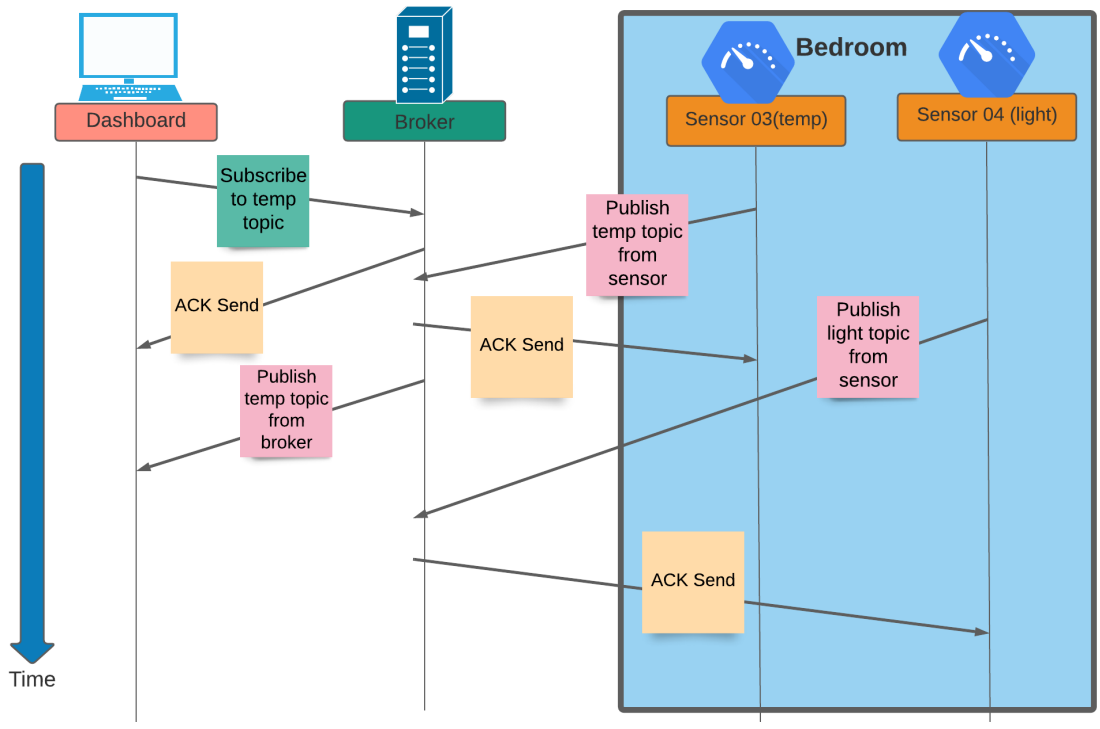


Figure 2: Shows a dashboard subscribed to temperature(temp) topic. All information on this topic from sensors is forwarded to the dashboard.

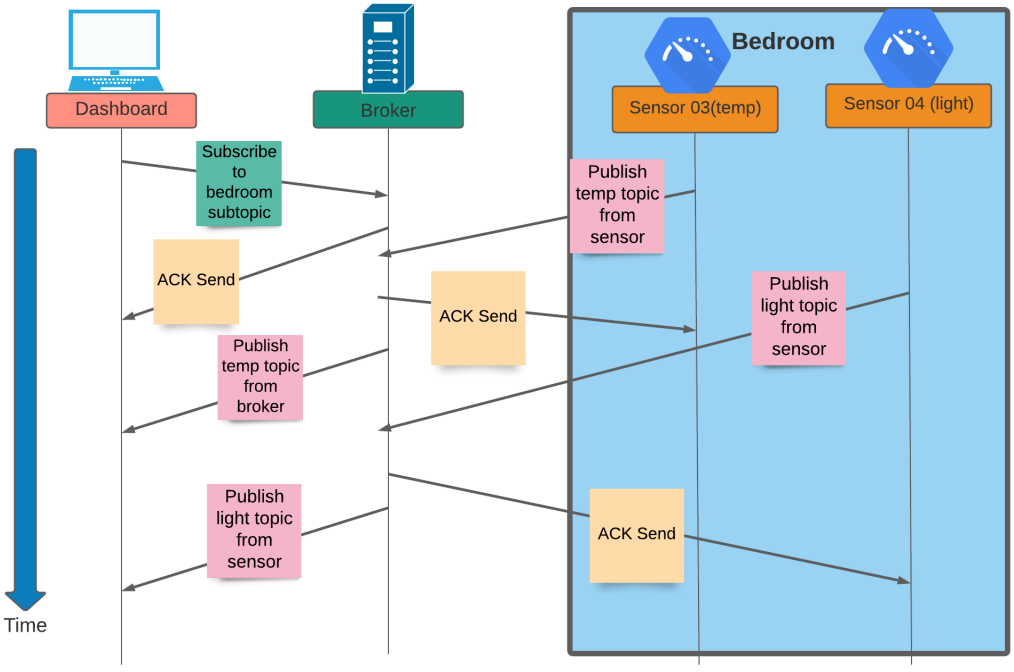


Figure 3: Shows a dashboard subscribed to the bedroom subtopic. All information on this topic from sensors is forwarded to the dashboard.

## 2.7 Communication when dashboard is publishing

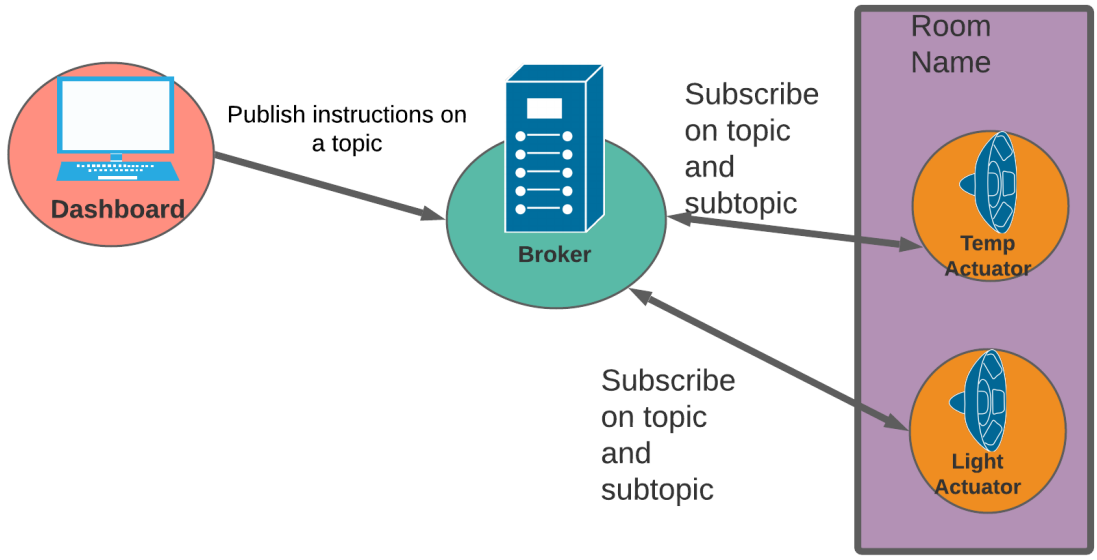


Figure 4 : Actuators subscribe at a Broker for instructions on a topic e.g “Light”. Dashboards that publish on this topic will be forwarded to the actuator.

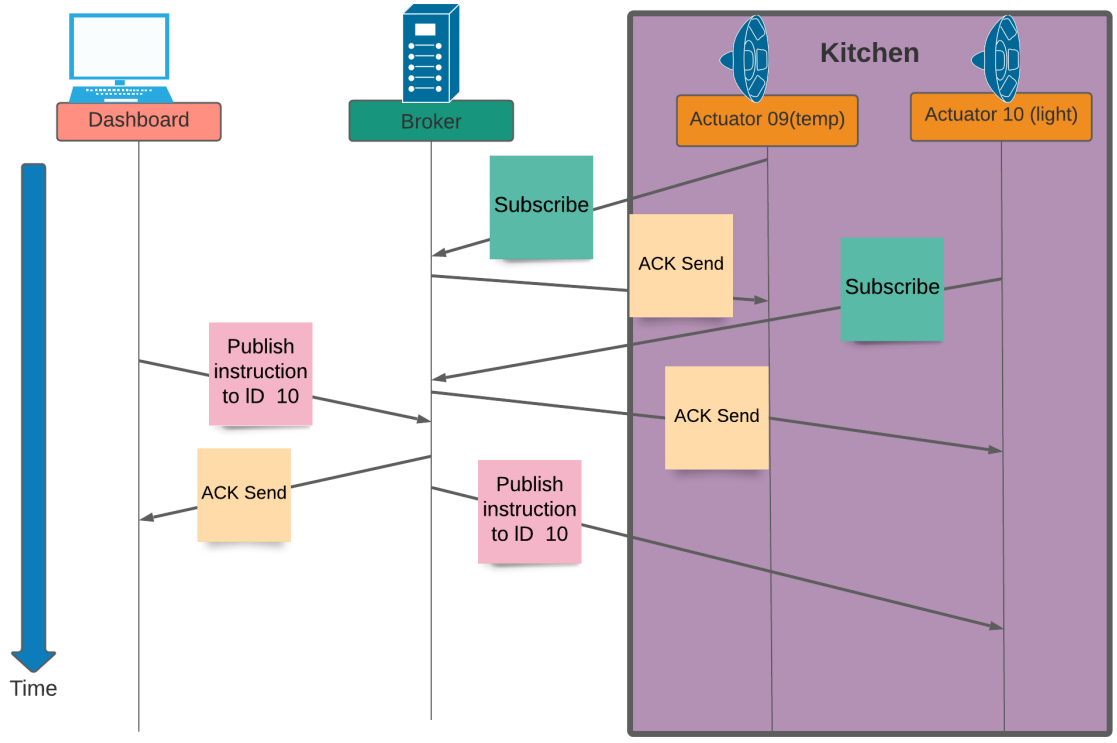


Figure 5: Shows actuators from the kitchen subscribed to the broker. Dashboard publishes to subtopic 10 so instructions are forwarded to this actuator specifically.

# 3.Implementation

I used the docker hello world program as the starting block of my implementation [2]. The code is run in 5 separate containers in docker called broker , sensor, dashboard, dashboardB and actuator. By enabling Xlaunch tcdlib.io terminals are able to launch on windows and the packets are also available to view on Wireshark by searching UDP.

Each of the separate components run on their own port number and using Docker I resolve the ip addresses of each device using container name and destination port. With Docker I am able to demonstrate how my protocol can work on multiple hosts .

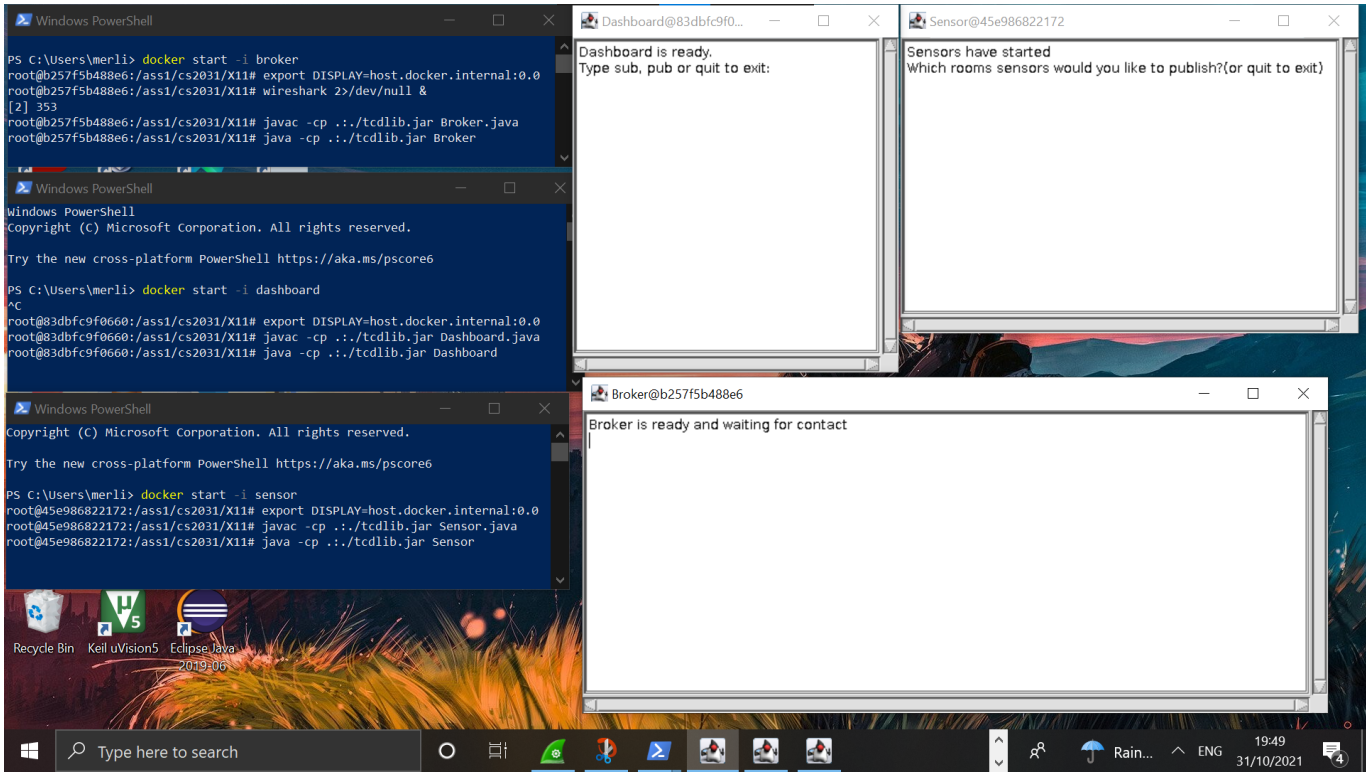


Figure 6 : Shows the sensor dashboard and broker containers started up and their corresponding terminal windows open and ready for input.

## 3.1Broker

Broker was implemented by extending the node class. It sends acknowledgment to each device that subscribes or publishes to it successfully Using threads it waits for the other components to publish and subscribe. It then sends the packets to specific components based on the ip address. A broker displays messages on the screen for every packet sent which can help for analysing wireshark.

## 3.2 Sensor

On startup the sensor requests which rooms sensors should publish. It then publishes to the broker container by using the container names and its destination Port.

## 3.3 Actuator

Actuator takes in user input on which rooms actuators should subscribe to. Subscribe packets are then sent to the broker which contain actuators topics and subtopics and their last instruction as well. Brokers will forward instructions to the subscribed actuators based on topics.

## 3.4 Dashboard

Dashboard allows for the user to both subscribe and publish to a topic .When a dashboard publishes to an actuator I ensure the instruction length is not too long to reduce byte overhead . Dashboard tells the user if a room is hospitable when it receives packets on a subscribed topic ..After researching I decided to choose temperature values between 18 to 22 degrees [3] celsius as optimum and light values between 100 to 500 lux as optimum [4]. I compare the current value reading from the sensors with these numbers to determine if the room is well heated and lit.

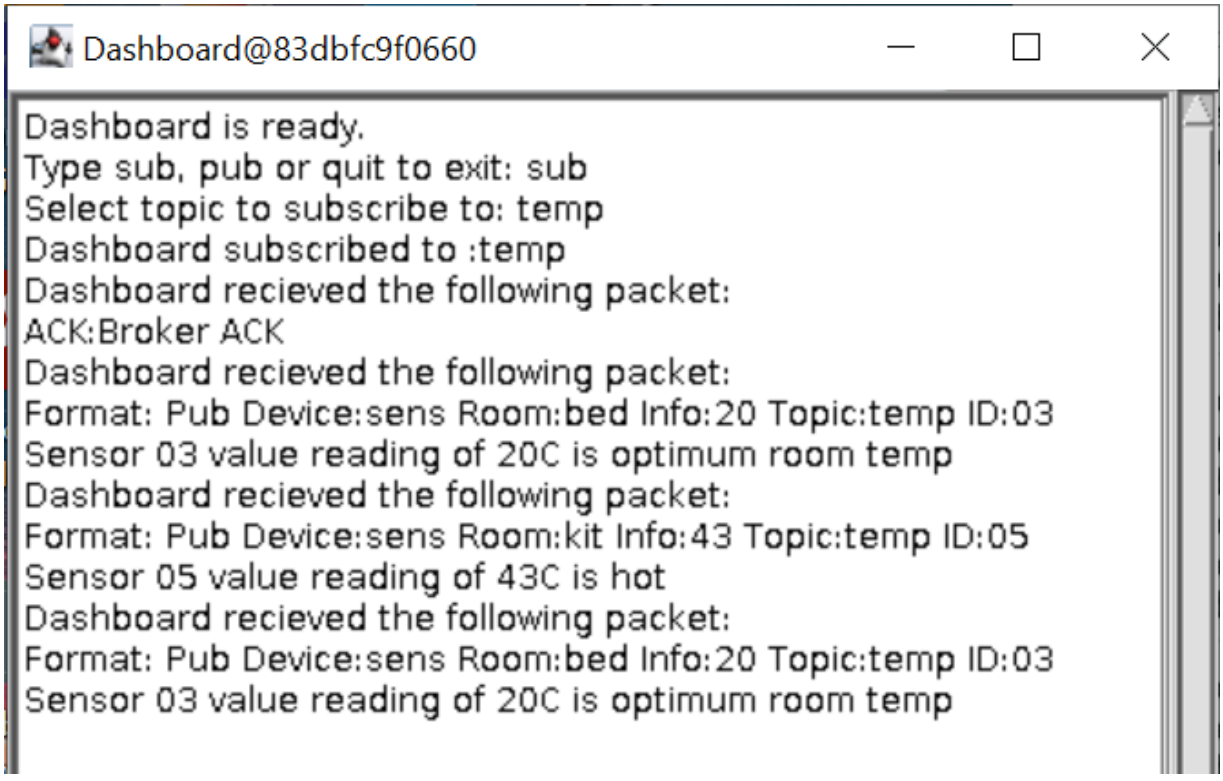


Figure 7 : Shows a dashboard that is subscribed to a broker on the temperature topic which was successful as it received an acknowledgement. It has been published data from the temperature sensors and it tells the user whether these readings are adequate or not.

## 3.5 User interaction

My protocol handles a wide range of user interaction to allow for different publishing and subscribing scenarios.
The user interface also deals with erroneous user input.

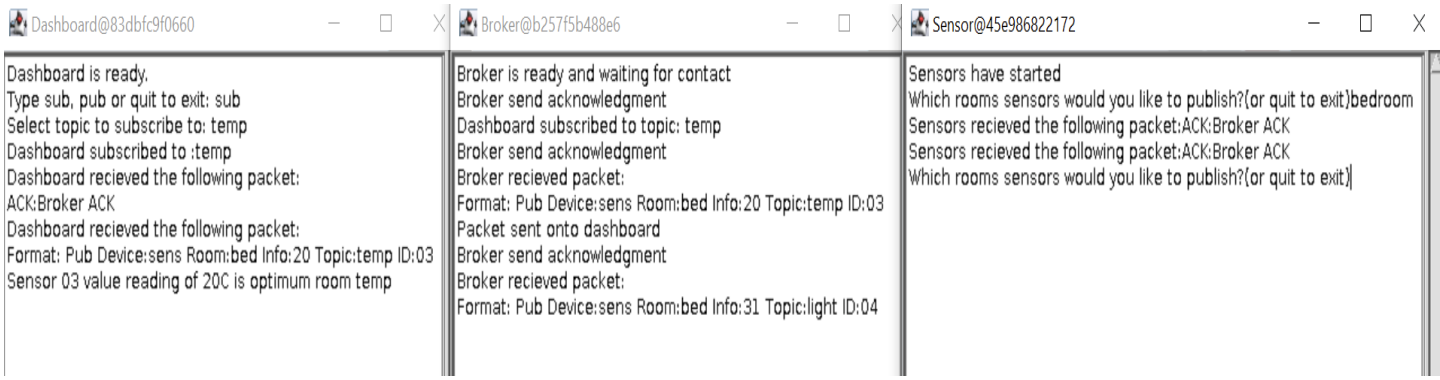


Figure 8: Shows the terminals from a sensor , broker and dashboard container. Dashboard has subscribed to the temp topic . Broker publishes data on any sensor with this topic which in this scenario is a sensor with ID 03.

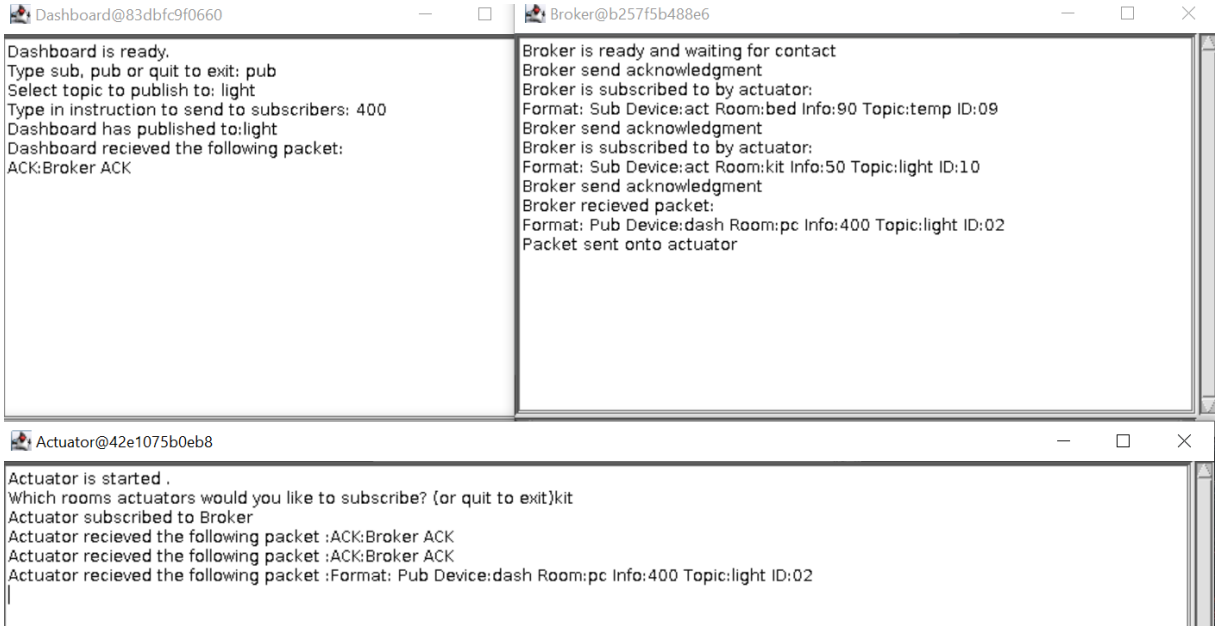


Figure 9 : Actuators from the kitchen are subscribed to the broker. Dashboard publishes an instruction of 400 to the light topic. Broker will forward these packets to the light actuator.

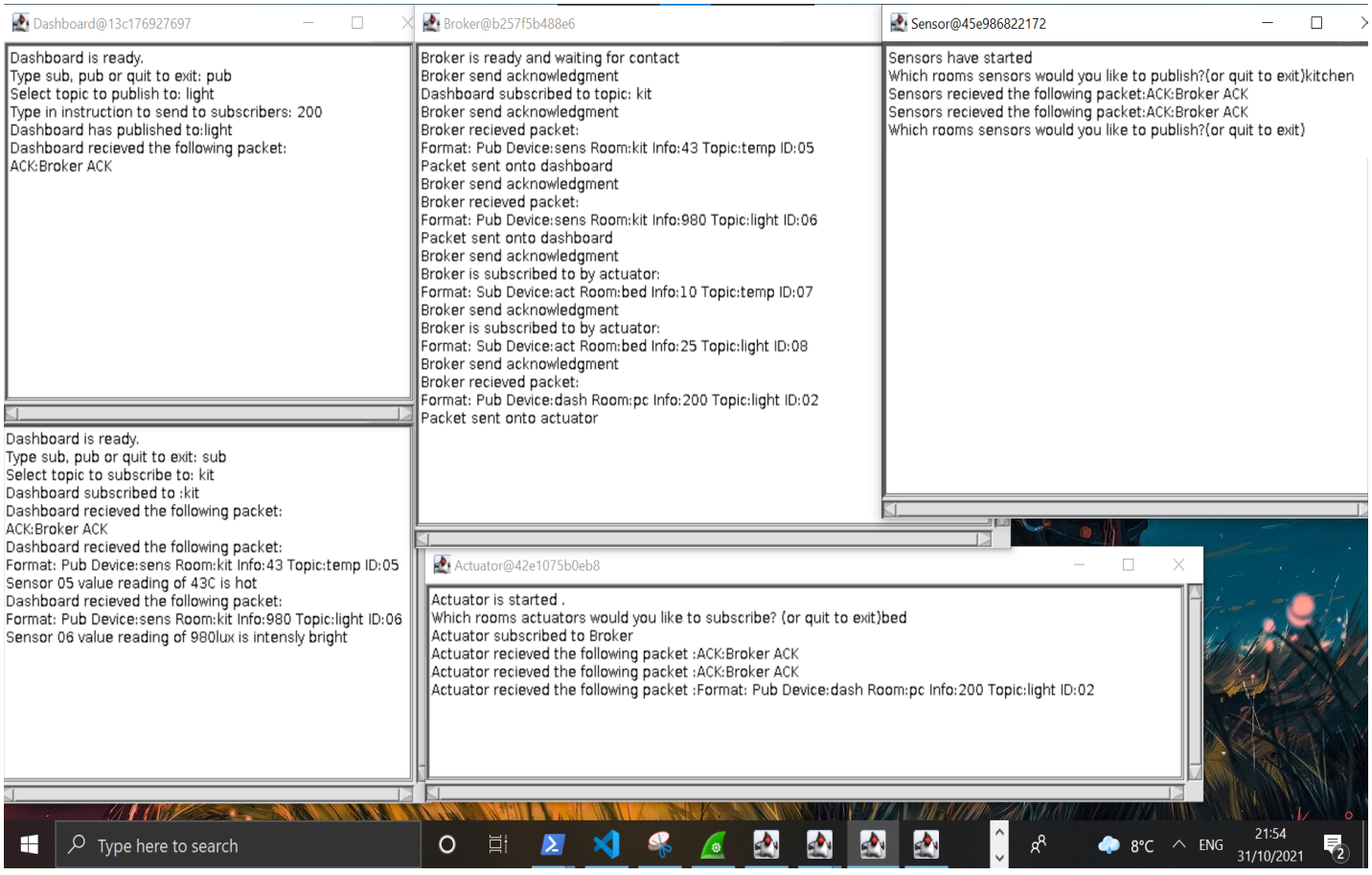


Figure 10 : There are two dashboards one that is publishing to the light topic and another that is subscribed to the bedroom subtopic. Broker forwards the relevant packets to the correct Actuators and Dashboard .

## 3.6 Packet Encoding

There are two types of packetClasses in my implementation. They are both inherited from the packetContent class.

First is the simple acknowledgement packet that simply contains the information that the Broker has been published and subscribed towards successfully handled by the AckPacketContent class. Acknowledgement packets have a fixed byte making it easy to identify in wireshark as they are significantly less bytes than publish or subscribe packets.

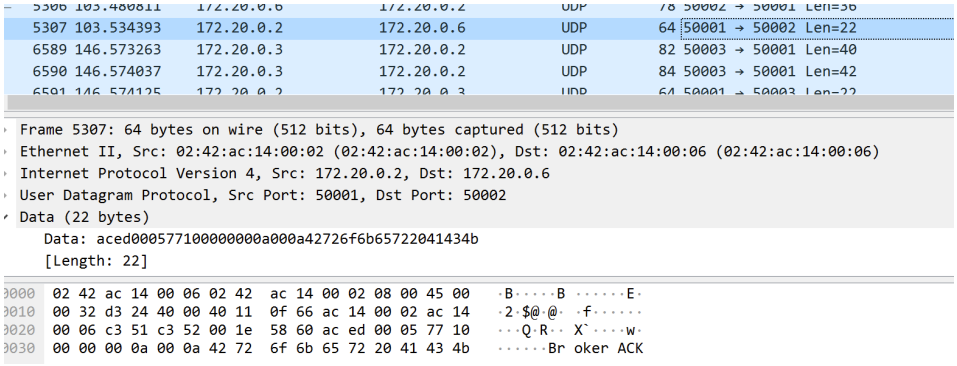


Figure 11: This screenshot of the pcap file from wireshark shows an acknowledgement(ACK) packet. The payload has a fixed length of 22 bytes and contains the message “Broker ACK”.

Secondly there is the SendPacketContent class which contains the data publishers and subscribers send. This class handles encoding the header into UDP datagram packets so that my protocol can send the payload safely.

```
tempSensor = new SendPacketContent("Pub","sens","kit","43","temp","05").toDatagramPacket();
tempSensor.setSocketAddress(dstAddress);
socket.send( tempSensor);
```

Figure 12 : This image shows how the sensor publishes datagram packets towards the Broker . The tempSensor is a datagramPacket that will contain information on the temperature sensors containerd in the room . This specific sensor is sensor 05 which is the kitchen's temperature sensor which currently has a reading of 43.

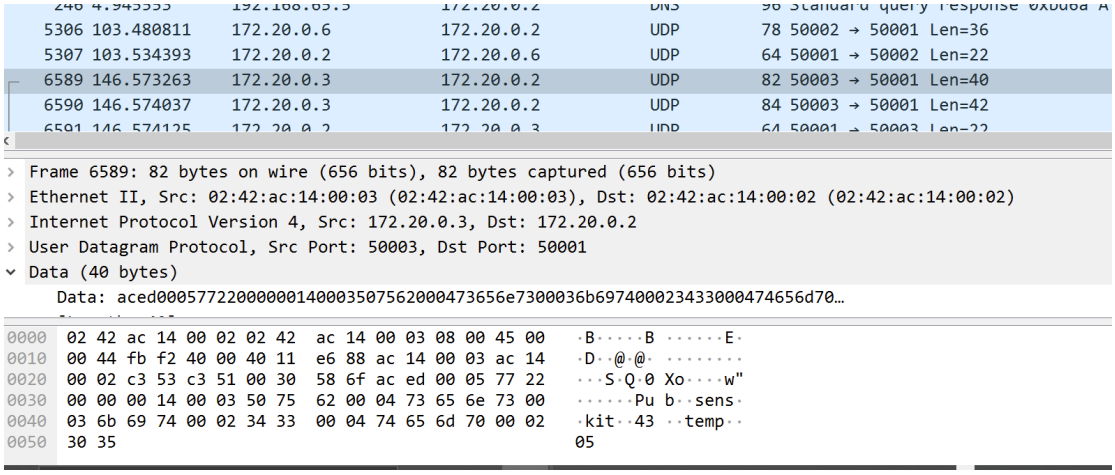


Figure 13 : Shows how sensor 05 in the kitchen sensing for temperature topic is viewed in a pcap file. The header is enclosed in the data of the udp

The following table explains meanings of the acronyms I used to encrypt my packet to reduce my byte overhead in the payload.

| **Title** | **Byte overhead** | **Acronyms** |
|---|---|---|
| Format | 3 bytes | publish (Pub) and subscribe (Sub) |
| Device | 3 to 4 bytes | sensor (sens) , dashboard (dash), and actuator (act) |
| Room | 2 to 3 bytes | Computer room (pc) , bedroom (bed) and kitchen (kit) |
| Topic | 4 to 5 bytes | Temperature (temp) and light |
| Packet Info | 2 to 5 bytes | A numeric value to represent degrees celsius and lux. |
| ID | 2 bytes | 2 digit value |

This the list of all ID values I used and the specific device they correspond towards.

| **ID** | **Device** | **Topic** | **Room** |
|---|---|---|---|
| 01 | Broker | | |
| 02 | Dashboard | | PC |
| 03 | Sensor | Temperature | bedroom |
| 04 | Sensor | Light | bedroom |
| 05 | Sensor | Temperature | kitchen |
| 06 | Sensor | Light | kitchen |
| 07 | Actuator | Temperature | bedroom |
| 08 | Actuator | Light | bedroom |
| 09 | Actuator | Temperature | kitchen |
| 10 | Actuator | Light | kitchen |

# 4. Discussion

Using docker allowed me to simulate running my components on different hosts . It also guarantees that the code can be executed on other laptops  as well.

Having multiple different components that are able to subscribe and publish increases the sophistication of my protocol. Instead of just having publishers and subscribers I made specific devices that carry this action out . This also allowed me to demonstrate subscription and publication from multiple different angles satisfying the assignment specifications .

Including the device that sends the packet helps decipher the packet in Wireshark and also enables  a broker to know what device is trying to publish and subscribe towards it. A unique ID allows publishers to send messages to specific devices it wants to without knowing its other information. Enabling the dashboard to subscribe to a room or topic means it can get data from multiple relevant sensors and see if the measurement is adequate. I have also ensured that the header will never be too big by preventing user input that is too long from being accepted.

I decided to use acronyms to reduce my byte overhead . Further encryption such as using numbers to denote rooms and topics would reduce the payload overhead further but would also reduce readability in my code. I believe given the functionality my current header can implement, the current byte overhead is justifiable.

Sending acknowledgments (ACK) helps increase the security of my protocol and by making it have a short unique length it is easy to distinguish between ACK and publish/subscribe packets .

Running multiple sensors in my sensor process allows me to lower the amount of containers and terminals open. In addition to this, separating  sensors by room adds more complexity in my protocol by allowing dashboards to now subscribe to subtopics about rooms. This also allowed me to control how many packets were being published at a time which was useful to see in Wireshark as it let me easily identify relevant packets.

I made the dashboard also provide useful feedback to the user once it decided to subscribe. Having the dashboard inform the user on whether temperature or light value was adequate allows the user to now know what instruction to publish to actuators to make a room more hospitable.

My protocol is heavily interactive which allows me to demonstrate the many different subtopics available to be selected from. Adding user input did increase the likelihood of human error and the need for error handling.Nevertheless it makes the protocol more unique and allows for more rigorous testing that all components work.

## 5.Summary

By adding four separate components to my protocol I am able to demonstrate publishing and subscribing with a high amount of user customization options. To enable me to do this I carefully chose what information to contain in my header and used acronyms to lower the byte overhead.

All together I was able to demonstrate a publishing and subscribing protocol for a house with multiple rooms and devices.

## 6.Reflection

Given the time frame and that I was a beginner in socket programming and using Docker I am pleased with how my protocol has turned out and how much I have learned. It meets the requirements of the assignment by demonstrating publishing and subscribing of multiple components.

However if I had more time I would have liked to have created more subtopics to my header in order to increase the functionality and complexity of my code.  It would also have been nice to add more security features to my protocol to encrypt the data sent .

Having sensors that publish periodically over a specified time frame would also be an interesting feature to include. Making the data contained within my sensors update and not be hardcoded values would make the protocol more realistic as well.

Furthermore I would like to implement my code in Kubernetes if I had to do this assignment again.

## 7.References

[1]Socket progamming in java
https://www.geeksforgeeks.org/socket-programming-in-java/ visited Sep 2021

[2] Docker hello world program
https://tcd.blackboard.com/webapps/cmsmain/webui/courses/CSU33031-202122/Resources/Docker?action=frameset&subaction=view&pid=pid-2016681-dt-content-rid-11717451_1 visited Sep 2021

[3]What is the ideal room temperature
https://www.viessmann.co.uk/heating-advice/what-is-the-ideal-room-temperature
visited Oct 2021

[4]Lighting levels by room https://www.thoughtco.com/lighting-levels-by-room-1206643
visited Oct 2021