29_T3-CB02

COM Challenge 1: Protocol Overview.

Student: Johnson Domacasse
Date: 19 sept 2023
Student#: 4471709
Teacher: Hans van Heumen

Contents

Introduction

I started off by going through the provided material. I got a good understanding of node-red from the introduction slide. I then got a good understanding of how mqtt works by doing some research. Lastly I familiarize myself with the different communication models and I have decided on the request/response and publish/subscribe models.

Procedure:

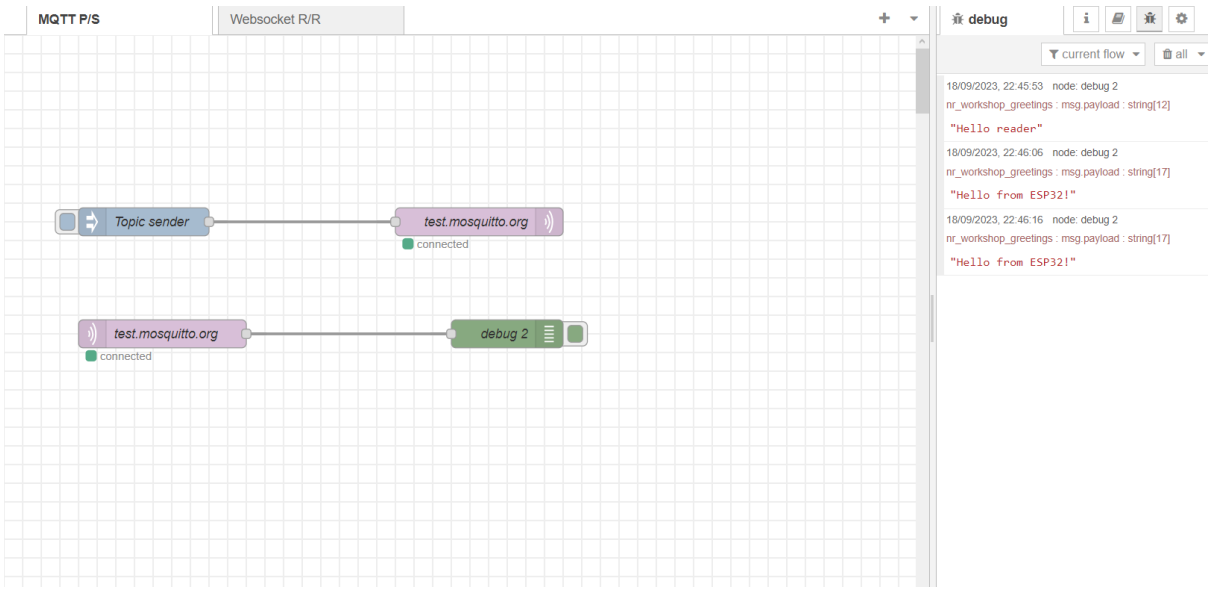
Q1: MQTT

I made a simple mqtt flow by connecting an inject node to an MQTT-out node. This is so that I can publish messages and my subscribers can read them. I also have an mqtt-in node connected to a debug node to read if a subscriber publishes a message.

For reference, the topic I will be using is: nr_workshop_greetings. So within the inject, I assign this as the topic and I can write anything for the message. For now I will go with “Hello reader”.

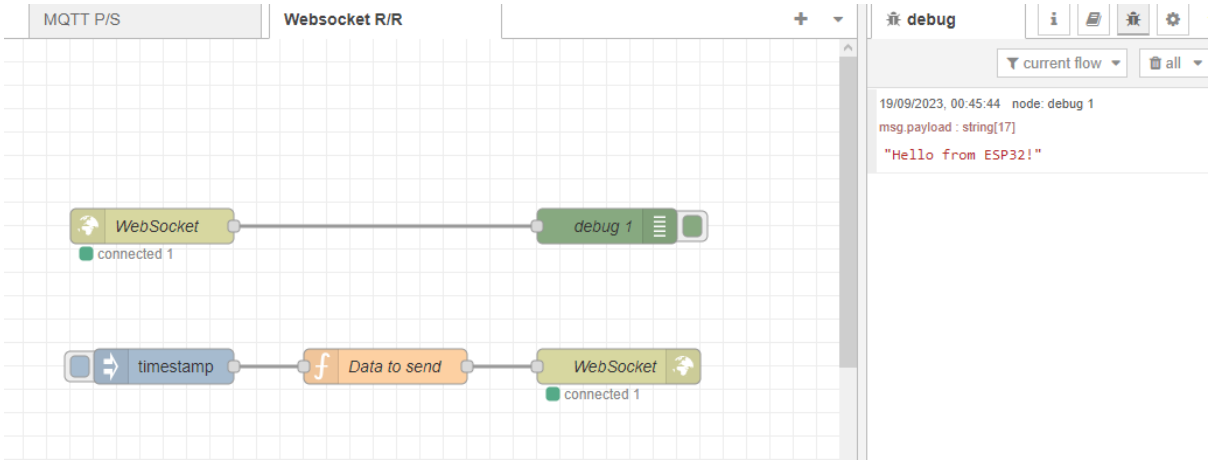
Within the mqtt-out node, I start setting up the server. For the sake of testing, I am using “test.mosquitto.org” with the port 1883 for my project. This is an MQTT 5 protocol. The same is done for the MQTT-in node. The only difference is I have to add the topic which is as before, “nr_workshop_greetings”.

I use my esp32 to act as the subscriber in this case. I first subscribe to the topic mentioned above. Now that it is added, any message that gets published (injected) will be received by all that are subscribed. I can also choose to publish a message of my own using the ESP32. In my program, I have a simple line that creates a string and sends this string to the node-red terminal. See below.



WebSocket

For this section I made use of the node-red application to both view and send messages like before. I also did some minor research to get some simple JavaScript programming. Like the previous model, I did so by both connecting a WebSocket-out node to an inject node to send some messages. I connected to WebSocket-in node to a debug node to read messages. In this case node-red will act as my server. Since it is my server I need to configure a path so that my esp32 can connect to it as well. In the WebSocket nodes I added the following in the path section: /ws/challenge. I also added a function node for easier understanding. Its function is to simply send data to the ESP32



I start by sending a message from my ESP32 to my WebSocket server. As you can see above.

This concludes what I have done for my first challenge.

Question 2:

In terms of advantages and disadvantages here is what I have found while working with these protocols and research done. What I have found is that for IoT applications like a microcontroller board that controls an LED MQTT would be advisable to use. This is because with MQTT you can have the devices perform their tasks despite the internet being poor quality. Unlike web sockets which will take longer. Web sockets should be used when you are operating with a project that requires good internet communication. Take a chat application for example. Here it would be advisable to use a WebSocket.

Question 3

In my research that I have done regarding keeping track of the lifespan of my nodes, I came up with the following conclusions. It is VERY important to have one of the methods in your projects for different reasons. To check if your network is still running, to moderate power consumption by knowing when the device is running or in sleep mode etc.. Different methods I looked up regarding keeping track of being alive are heartbeats and polling.

Key differences I found were the following:

Polling:

Polling has to do with the client repeatedly asking te server for information and the server provides this information. There can be some latency when receiving the information from the server because the client has to wait for the next polling interval. Lastly it is very resource intensive. Due to its frequency, it can consume a lot of traffic between client and server but also CPU cycles.

Polling is better used when you are checking for updates on a resource that isn't frequently changing. This way it can keep up with the polling intervals and also not spend too much resources.

Heartbeat:

The heartbeat has to do with the client and server establishing a connection and the server continuously sends "heartbeats" to the client over a period to show that it is available for use. Since the server is sending these signals, it is less resource intensive than Polling. Since there is no need for requests but the server sends these signals on its own. It is very low latency since the server automatically sends updates of the resource when it becomes available.

This method is best used when I need to work with real-time or near-real-time updates, want to minimize resource consumption or be notified immediately when one of my resources has been changed.

Conclusion

My overall conclusion is that it all depends on the use case. Whether you are making a chatting system, or you are making an IoT application for a company. It all depends on the use case when you are choosing which communication model you are choosing. Implementing the different models shouldn't necessarily be too difficult and can be quite simple. You just need to know what the use of the program will be.

The same thing counts for choosing your method to keep track of the lifespan of your connections. If you are building a system that works with real-time aspects, then you would pick the heartbeat method. If you are building a system that you don't have any control over the resource that is being monitored, then you would use Polling.

Along with the explanation above, I will provide my code base for question 1 for both WebSocket and MQTT.