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

CSU33031 Computer Networks
IoT Publish-/Subscribe Protocol

September 14, 2021

1 Introduction

The focus of this assignment is to learn about protocol development and the information that is kept in a header to support functionality of a protocol. Protocol design is always a balancing act between introducing functionality that relies on additional header information and the overhead that the additional header information introduces.

2 Protocol Details

The aim of the protocol is to provide a publish-/subscribe-mechanism for processes based on UDP datagrams. There are two simplistic scenarios that form the very basic requirement for your protocol: 1) the reporting of sensor data to a number of subscribers and 2) the issuing of instructions to actuators. An example of the first case is the collection of measurements from temperature sensors in a building and their display at a dashboard or through various applications. The dashboard may subscribe to measurements from all sensors in a building, whereas an application of an office manager may just display measurements from a number of rooms. An example for the second case would be the control of air conditioning (AC) units in a building from a dashboard or from mobile phone applications. The control of certain AC units may be limited to control from the dashboard, whereas an application of an office manager maybe limited to the control of AC units in certain rooms.

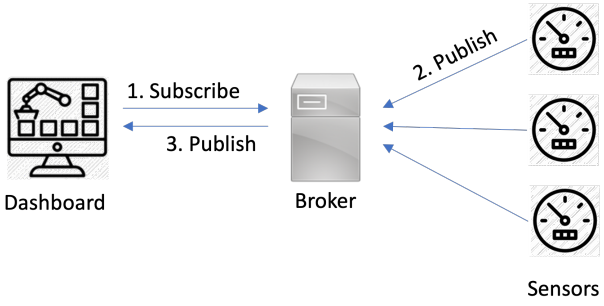

Figure 1: A dashboard process subscribes at a Broker for data with a given topic e.g. "temperature". Data that is published by any of the temperature sensors after the subscription will be forwarded to the dashoard.

So, the basic functionality that a publish/subscribe protocol has to provide is the subscription to topics by nodes to a broker and the publication of information from nodes to topics at the broker. The description of the two scenarios above shows that there are a number of additional functionalities that a protocol may need to provide such as security, caching, etc. The Message Queuing Telemetry Transport (MQTT) Protocol [6] is

an example of a publish-subscribe protocol that you could look at for functionalities that may be interesting for you to include in your protocol. In the end, it is up to you what functionalities you implement in your protocol. In the deliverables, you need to describe these functionalities and justify why you included them in your protocol.

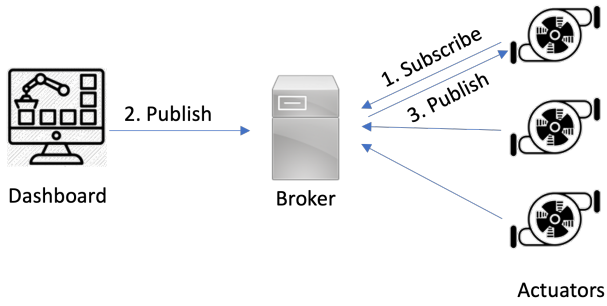

Figure 2: Actuators subscribe at a Broker for instructions with a given topic e.g. "blinds". Data that is published by the Dashboard to this topic will be forwarded to the actuators.

The following flow diagrams, figures 3 and 4, are examples of visualizations of network traffic between components. They show the sequence of messages exchanged between components with the start of their transmission and their arrival at the receiver.

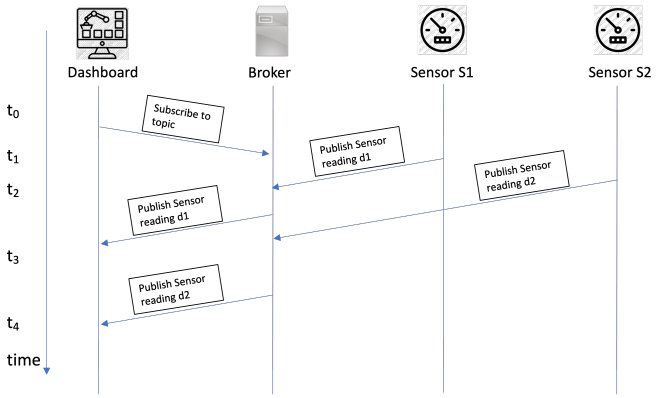

Figure 3: A dashboard process subscribes at a Broker for data with a given topic e.g. "temperature". Data that is published by any of the temperature sensors after the subscription will be forwarded to the dashoard.

Please use Flow Diagrams like these to document the communication between components of your implementation in your videos and in your final report.
The protocol can be implemented in a programming language of your choosing. One of the conditions is that the communication between the processes is realized using UDP sockets and Datagrams. Please avoid Python 2.7 because the implementation of Datagram sockets in the obsolete versions is based the transfer of strings instead of binary arrays.
The easiest way to start with the development of your solution is possibly to connect your components through the localhost interface of a machine; however, at the end, you will need to be able to demonstrate that your protocol can connect components located at a number of hosts. There are a number of platforms that support the simulation of topologies or provide virtual infrastructures e.g. Docker [2], Mininet [5], Kubernetes [1], etc. For someone starting with socket programming and networking, I would suggest to use a platform such as Docker or Mininet; for someone already familiar with these concepts, I would suggest to implement their solution using Kubernetes. However, these are only suggestion and you need to make the decision how to implement your solution.

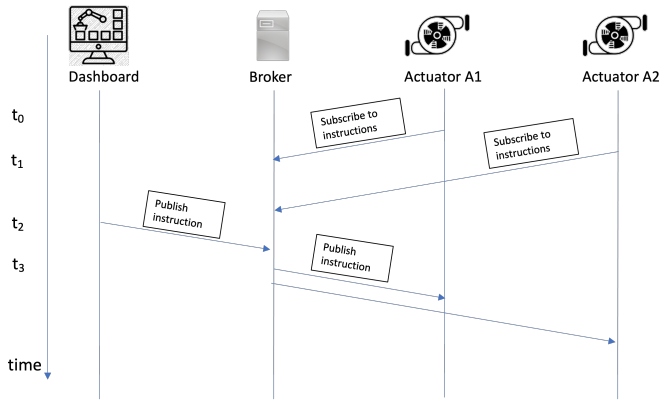

Figure 4: Actuators subscribe at a Broker for instructions with a given topic e.g. "blinds". Data that is published by the Dashboard to this topic will be forwarded to the actuators.

3 Deliverables & Submission Details

The deliverables for this assignment are split into 3 parts: 2 videos, a report describing your solution and the files making up the implementation of your solution. The deadline for the submission of these deliverables are given in Blackboard.
One component of the deliverables at every step is the submission of captures of network traffic. These captures should be in the form of PCAP files [3]. Programs such as Wireshark [4], tshark, etc offer functionality to capture network traffic from interfaces

3.1 Part 1: Video & PCAP file

The video of part 1 should demonstrate the initial design of your solution and a capture of network traffic between the components of your solution and the files of your traffic capture in pcap format. In the video, you should explain the setup of the topology that you are using and the information that makes up the header information in your traffic captures.
The submission process for this part consists of two steps: 1) Submitting the PCAP file or files that you captured from your network traffic, and 2) submitting the video for this step.
The video is to be no longer than 4 minutes; content past the 4 minute-mark will be ignored during marking. Videos with voice-over using text-to-speech (TTS) will not be accepted and marked with 0.

3.2 Part 2: Video & PCAP file

The video of part 2 should demonstrate the current state of your solution, the functionality that you have implemented so far, and a capture of network traffic between the components of your solution and the files of your traffic capture in pcap format. In the video, you should explain the basic implementation fo your protocol and the information that is being exchanged between the components of your solution.
The submission process for this part consists of two steps: 1) Submitting the PCAP file or files that you captured from your network traffic, and 2) submitting the video for this step.
Videos with voice-over using text-to-speech (TTS) will not be accepted and marked with 0. The video is to be no longer than 4 minutes; content past the 4 minute-mark will be ignored during marking.

3.3 Final Deliverable

The final deliverable should include a report that describes the components of your solution and their functionality, the protocol that you implemented and the communication between the components of your solution, the topology that you used to run you solution and how your solution was executed.

The submission process for this part consists of three steps: 1) Submitting the PCAP file or files that you captured from your network traffic, 2) submitting the source code and any files that may be necessary to execute your solution, and 3) submitting the report about your solution.
The files that contain the implementation and the report should be submitted through Blackboard. Every file should contain the name of the author and the student number. The source files of the implementation should be submitted as an archived file e.g. ".zip" or ".tar.gz". The report should be submitted as either word- or pdf-document. The deadline for the submission is given in Blackboard.
The report may be submitted to services such as TurnItIn for plagiarism checks.

4 Marking Scheme

The contribution of the assignment to the overall mark for the module is 30% or 30 points. The submission for part 1 and 2 will be each marked out of 5 points and the submission for the final part will be marked out of 20 points. The mark for the final deliverable will be split into 50% for the functionality of your solution and 50% for the documentation through the report.

References

[1] The Kubernetes Authors. Kubernetes project page. https://kubernetes.io, visited Sep 2021.

[2] Docker. Docker project page. https://www.docker.com, visited Sep 2021.

[3] Wikipedia Editors. pcap - wikipedia page. https://en.wikipedia.org/wiki/Pcap, visited Aug 2021.

[4] Wireshark Foundation. Wireshark project page. https://www.wireshark.org, visited Sep 2021.

[5] Mininet. Mininet project page. http://mininet.org, visited Sep 2021.

[6] OASIS Message Queuing Telemetry Transport (MQTT) TC. Message queuing telemetry transport (mqtt) protocol. https://docs.oasis-open.org/mqtt/mqtt/v5.0/os/mqtt-v5.0-os.pdf, visited Aug 2021, March 2019.