CS2031 Telecommunications II

Assignment #1: Publish-Subscribe

Chloe Conneely 17323080

Contents

1. Introduction
2. Components

- Subscriber

- Broker

- Publisher

- The Packet

1. Running the Program
2. Reflection

- Extra Components

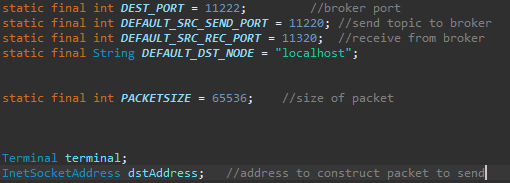
- Closing

1. Introduction

I was tasked with designing a protocol that forwards messages from a publisher to subscribers via a broker. The purpose of this exercise being to gain an understanding of sockets, datagram packets and threads by designing a protocol focusing on packet layout, packet handling and communication between nodes. This report describes the components and processes of my implementation of the protocol.

2. Components

Subscriber: The subscriber class contains the destination port which is the port number to where the subscriber sends subscription requests to the broker, the port from which subscriber sends a topic and the port through witch a subscriber receives a message from the broker. The terminal allows for a terminal to appear on the screen to accept input and display the outputs. The address is where the packet to send will be constructed.



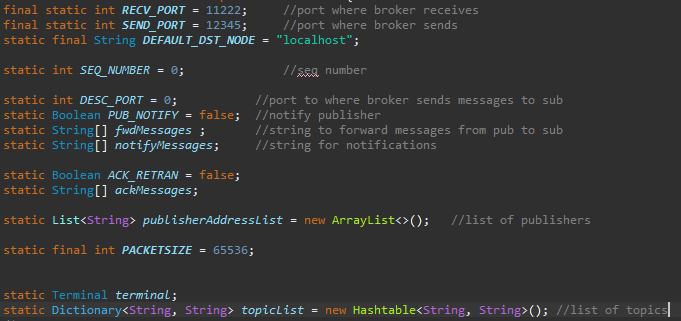
Methods:

onReceipt passes the packet as its parameter to open the packet received from the broker. Then it creates a buffer to extract the data and displays the data in the terminal. This is the method that acts as the receiver.

inputTopic creates a packet to send to the broker. The packet is composed of a type marker “S-” meaning subscriber followed by its origin port and the topic to which the subscriber wants to subscribe. This method informs the user that they have successfully subscribed to a topic by printing a message saying so. This is the method which acts as the sender.

Main creates the terminal for the subscriber class and creates threads to allow for the subscriber to simultaneously send and receive messages.

Broker: The broker contains the port from which it will receive subscription requests from the subscriber and receive messages from the publisher along with the port from which it will send messages to the subscriber through the publisher. The sequence number keeps track of messages in order of oldest to newest. The destination port is the port to where the broker forwards messages from the publisher to the subscribers. The Booleans notify and ack determine if the broker has received the packet. The string “fwdMessages” contains the message to be forwarded to subscribers sent from the publisher through the broker. The sting “notifyMessages” contains the acknowledgement of the broker receiving the packet. The terminal creates a terminal to allow the user to keep track of the broker’s actions. The arrayList “publisherAddressList” contains a list of publishers. The hashtable “topicList” maintains a list of all subscribers and the topics to which they have subscribed.



Methods

onReceipt receives a packet from either the publisher or the subscriber.

parsePacket creates a buffer to extract a packet. Once the packet is extracted it creates a string called messages. The method then determines from where the packet was sent by checking the first character in the string is from either the publisher or the subscriber. If the packet is from the publisher, the method confirms receiving the packet and then checks the topic list to send on to the packet’s subscribers. If the packet from a subscriber wanting to subscribe to a topic, the method adds them to the list of subscribers.

addTopicList adds a new subscriber to the list of subscribers by getting the date and time of the subscription and adding a time stamp to the subscription to prevent the subscription from being overwritten by a future subscription request. This method also allows for a subscriber to unsubscribe from a topic by checking the list for the topic in the list and removing it. Then it prints a message in the terminal saying the un-subscription was successful.

checkTopicList searches the list of subscriptions for the relevant subscribers to the topic. The method then removes the timestamp and copies the message into a string in preparation for sending along with getting the destination port.

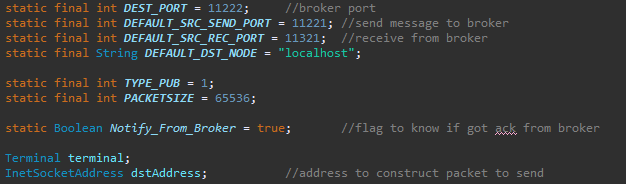
sendMessage takes the destination port along with the message to send and creates a new packet. This packet contains the type of sender, the port from which it was originally sent, the sequence number and the message.

start allows for the broker to act as an intermediary between the publisher and subscriber by being able to receive messages from both.

Main creates the broker terminal and starts the timer along with threads for the broker to simultaneously send and receive messages.

startTimerTask creates a timer to allow for timestamps to be processed on the packets from the publisher. The timestamps are used to organise messages from the publisher and prevent messages from being overwritten in the list of messages

Publisher: The publisher contains the port of the broker, the port from where it will send messages to the broker and the port where the publisher will receive any packets if necessary. The notification indicates whether the broker receives the packet. The terminal allows for a terminal to appear on the screen to display the outputs. The address is where the packet to send will be constructed.



Methods

onReceipt receives the acknowledgement from the broker determining whether or not the broker received the packet.

start creates a packet to send to the broker. The packet is composed of a type marker “P” meaning publisher followed by its origin port, the topic and the message to be sent. The method then sends the packet to the broker. This method also checks to see if the publisher gets an acknowledgement regarding the packet. This method resends the packet in the case where the broker does not receive the packet.

Main creates publisher the terminal along with an instance of publisher to allow for the publisher to send messages.

# The Packet

The packet for the publisher comprises of a type marker, the origin port, a timestamp, the topic and the message. The type marker indicates from where the packet is being sent. In this case, the sender is the publisher which is indicated by the type marker “P”. The origin port contains the port from where the packet was sent. The timestamp contains the time at which the packet was sent. The topic contains the topic being sent and the message contains the message being sent.

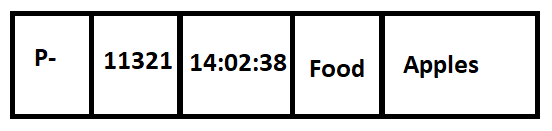


Fig. 2.1 Example of the Publisher Packet

The packet for the subscriber comprises of a type marker, the origin port and the topic. The type marker works the same as the it does for the publisher, the only change being that the type marker is now “S” for subscriber. The origin port and the topic work the same as for the publisher. The subscriber packet lacks a message section as all the subscriber packet is used for is to request a subscription to a topic.

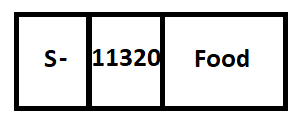


Fig. 2.2 Example of the subscriber packet

3. Running the Program

* The broker waits to receive a packet from either the publisher or subscriber as shown in figure 3.1.
* The subscriber sends a packet to the broker containing a subscription request as shown in figures 3.2 and 3.3.
* The broker receives the packet from the subscriber and displays in the window a message saying it received the subscription request as shown in figure 3.4.
* The publisher sends a packet to the broker containing a message pertaining to a topic as shown in figure 3.5 and 3.6.
* The broker receives the packet from the publisher and then sends it on to the subscriber along with sending a notification back to the publisher saying it received the packet as seen in figure 3.7.
* The subscriber receives the packet from the broker and displays it in the window as shown in figure 3.8.
* The program supports a situation where a subscriber has unsubscribed from a topic. The subscriber sends an un-subscription request to the broker as shown in figure 3.9.
* The broker removes the subscriber from the list of subscribers for the topic from which the subscriber has unsubscribed as shown in figure 3.10.
* When the publisher sends a message relating to the topic from which the subscriber has unsubscribed, the broker receives the packet but does not send it on as shown in figures 3.10 and 3.11.

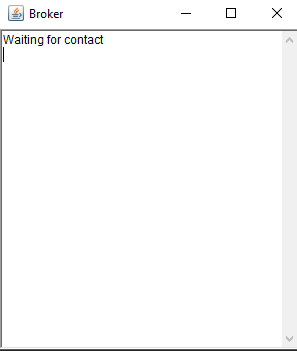


Fig. 3.1. Broker waiting for packet

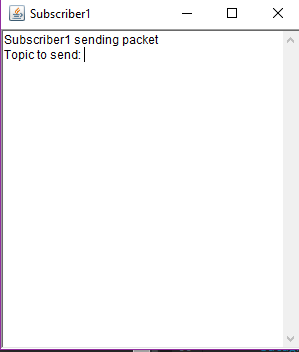


Fig. 3.2. Subscriber allowing for a user to send a message

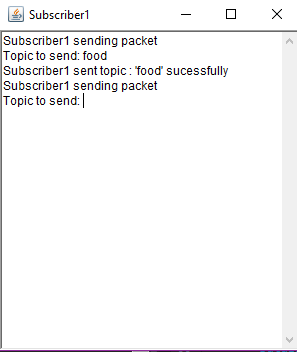


Fig 3.3. Subscriber sending a subscription message

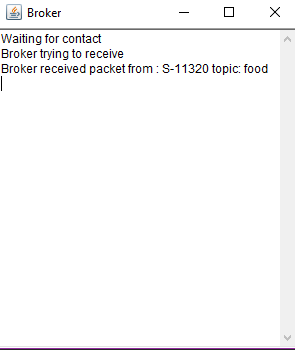


Fig. 3.4. Broker receiving subscription message from subscriber

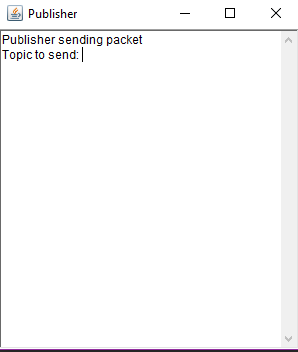


Fig. 3.5. Publisher allowing for a user to send a message

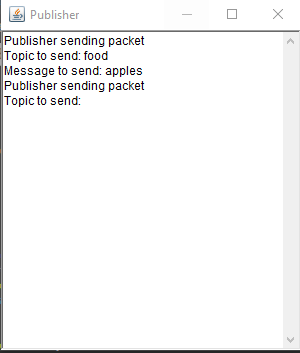


Fig. 3.6. Publisher sending message

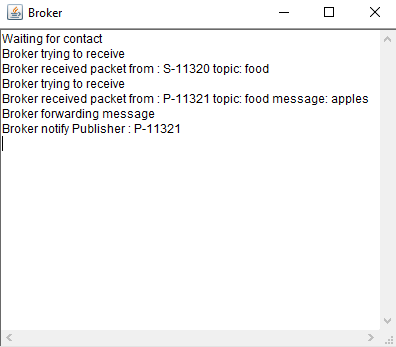


Fig 3.7. Broker receiving packet from publisher, sending packet onto subscriber and notifying publisher of receiving packet

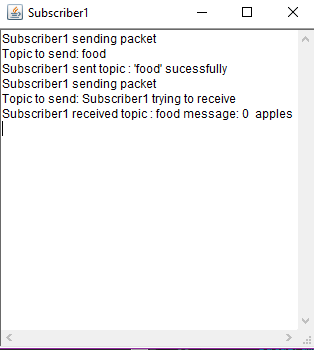


Fig. 3.8 Subscriber receiving packet from broker

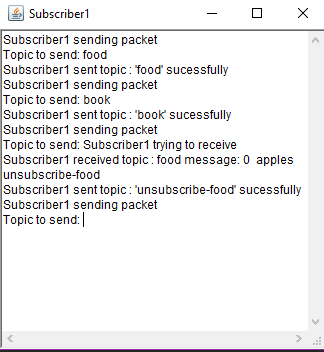


Fig. 3.9. Subscriber unsubscribes from topic

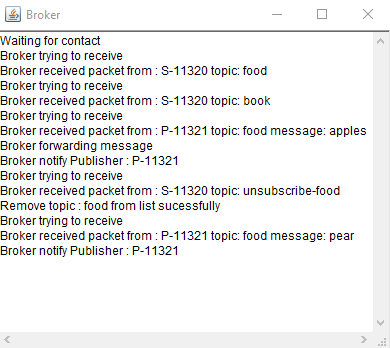


Fig. 3.10. Broker removes subscriber from the subscriber list

and doesn’t send on the packet from publisher

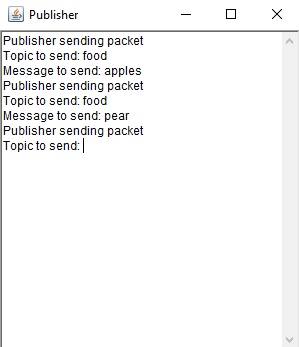


Fig. 3.11. Publisher sends topic to which subscriber is no longer subscribed

4. Reflection

# Extra Components

Timestamp: I had to include the timestamp in my publisher packet as prior to this, the previous message from the publisher kept being overwritten by the newest message.

Multiple publishers: I attempted to add a second publisher to the protocol who would send messages in the same manner as the original publisher. I was successful in my attempt as the second publisher sends packets in the same way as the original publisher. Images of the running of the second publisher are show below in figures 4.1-4.3

Thread.sleep(): Thread.sleep() forces the packet to wait a specified time before sending a packet. I chose the time of one second. I use thread.sleep() as a means of controlling the sending of a packet as without it, an error occurs in the sending of the packet

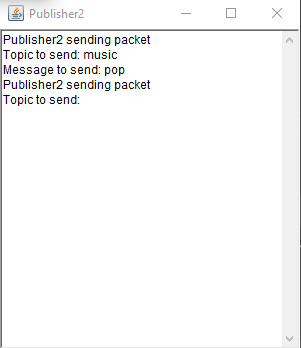


Fig. 4.1. Publisher2 sending a message

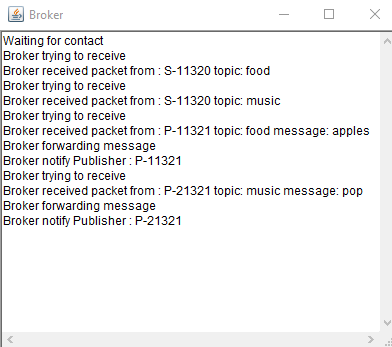


Fig. 4.2. Broker receiving packet from publisher2, sending packet onto subscriber and notifying publisher2 of receiving packet

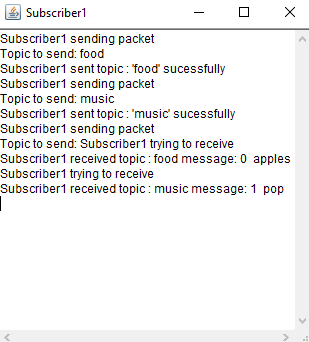


Fig. 3.8 Subscriber receiving packet from publisher2 through broker

Closing

I have designed a protocol that forwards messages from a publisher to a subscriber via a broker. The protocol makes extensive use of sockets, packets and threads to send and receive messages. The subscriber sends a subscription request to and receives messages from the broker in order of sequence number. The subscriber also has the option to unsubscribe from a topic. The publisher sends a message for a topic to the broker which contains a sequence number and time stamp for processing. The broker receives messages from the subscriber, informs the publisher that it received the message and sends the messages onto the subscribers of the topic in question along with maintaining a list of subscribers. I used the sample files provided as a starting point for the protocol which, while it was helpful in terms of finding a direction for the implementation, it means the protocol may be obsolete in its design. I did not make use of containers as I felt I would not understand how it worked in time for completing this assignment. I estimate I spent about 150 hours on this assignment between the implementation and the documentation.