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

# CS2031 Telecommunication II
# Assignment #1: Publish-Subscribe

Patrick Dillon Ryan, 17340382

December, 2018

## Table of Contents

Introduction ..... 2
Theory of Topic – Sockets & Header ..... 3
Flow Control ..... 3
Sockets ..... 4
Header ..... 4
Datagram ..... 4
Threads ..... 5
Implementation ..... 5
Subscriber ..... 6
Publisher ..... 7
Broker ..... 7
Discussion ..... 8
Summary ..... 8

# Introduction

The task which was given was to design a protocol that forwards messages from a publisher to a broker which in turn distributes these messages to a number of subscribers. This assignment was given to us with the aim of growing our knowledge of sockets, datagram packets and threads and to design a protocol. I had to decide on packet layout and packet handling, for the communication between a number of nodes and to learn to describe my solution as part of a report.

The program should transmit a packet from a client to a server which contains a string which has been entered by the user. The server responds to that incoming packets with a packet which contains the string “OK”. The client opens a port on port number 5000, creates a packet from the input which the user gave. It then sends this packet to another port on the local machine with the port number 50001.

The program should have the following features:

1. Subscribers that accept a topic as input, send a subscription message to a broker and print published messages that have been forwarded by a broker.
2. Publishers that accept a topic and a message as input and transmit a message with the topic and message to a broker.
3. A broker that receives subscriptions, maintains lists of subscribers to various topics and forwards incoming publish-messages to the subscribers with matching topics.
4. Subscribers may choose to unsubscribe from topics.
5. Messages from publishers should have a sequence number and subscribers should print messages in the order of sequence numbers.
6. The broker and the subscribers may implement acknowledgements and the publisher may wait for an acknowledgement from a broker before proceeding to accept input of another topic and message.

The approach I took to tackling this problem was by creating 4 different classes. A Subscriber(client) class, a Publisher(server) class, a Broker class and a Node class. The program let either a subscriber subscribes to a topic or it let a publisher publish about a certain topic.

# Theory of Topic – Sockets & Header

## *Flow Control*

The main focus of telecommunications in this assignment is on flow control. Flow control is about a sender and receiver. It focuses on how the sender can send frames to the receiver without overflowing the receiver. It is rare that the receiver can receive the frames as quick as the sender can send them, so a compromise is needed. This is achieved by limiting the rate at which the frames are sent and also by the use of acknowledgements.

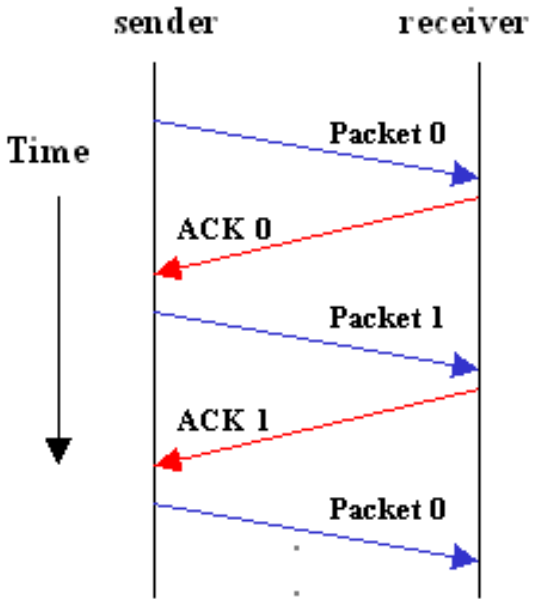


Figure 1: Stop & Wait Flow Control

The most basic form of flow control is the Stop & Wait Flow Control protocol. It involves a sender and receiver. The sender sends the frame. Once the receiver gets the frame it sends an ACK to say that it has received it correctly. Only once the ACK has been received will the sender send the next frame. It is the simplest setup, but it is also the most inefficient as it will not send the next frame until it receives the ACK for the frame that it has just sent. There are three other types of flow control. They are Stop & wait ARQ, Selective Repeat and Go Back N. They all have their own specific trade offs.

## *Sockets*

A socket is one endpoint of a two-way communication link between two programs running on the network. A socket is bound to a port number so that the TCP (Transmission Control Protocol) layer can identify the application that data is destined to be sent to.

Each TCP (Transmission Control Protocol) connection is identified uniquely by its two endpoints. An endpoint is a combination of a port number and an IP (Internet Protocol) address.

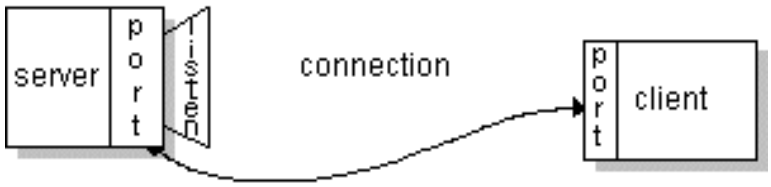


Figure 2: Example of a socket

## *Header*

A header is a unit of information that precedes a data object. A header is part of the data packet and contains transparent information about the file or the transmission.

## *Datagram*

A Datagram is a self-contained, independent message sent over the network where the content, arrival and arrival time are not guaranteed.

A packet sent over a perfect channel does not contain any information about its source or its destination. The channel contains the information.

A Datagram packet must contain the full address of its destination and its source depending on weather the Datagram is being sent or being received.

The difference between a Datagram socket and a datagram packet is that a Datagram socket is a communication link used to send datagrams between applications whereas a Datagram packet is a message sent between applications via a Datagram socket.

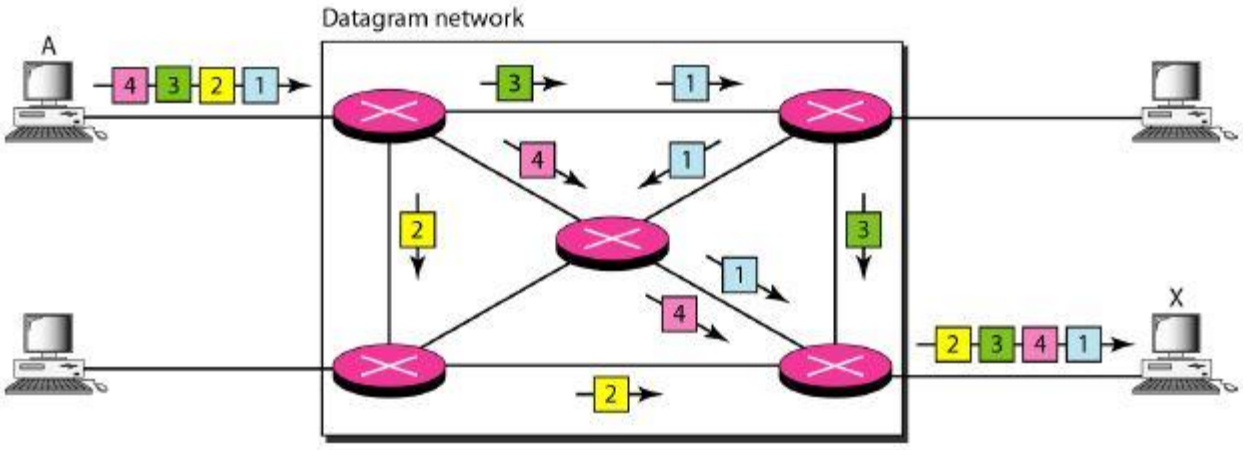


Figure 3: Example of Datagram Networks

### *Threads*

A thread is a sequence of instructions that run independently of the program and of any other threads. Using threads, a multi-threaded server program can accept a connection from a client, start a thread from that communication, and continue listening for requests from other clients

## Implementation

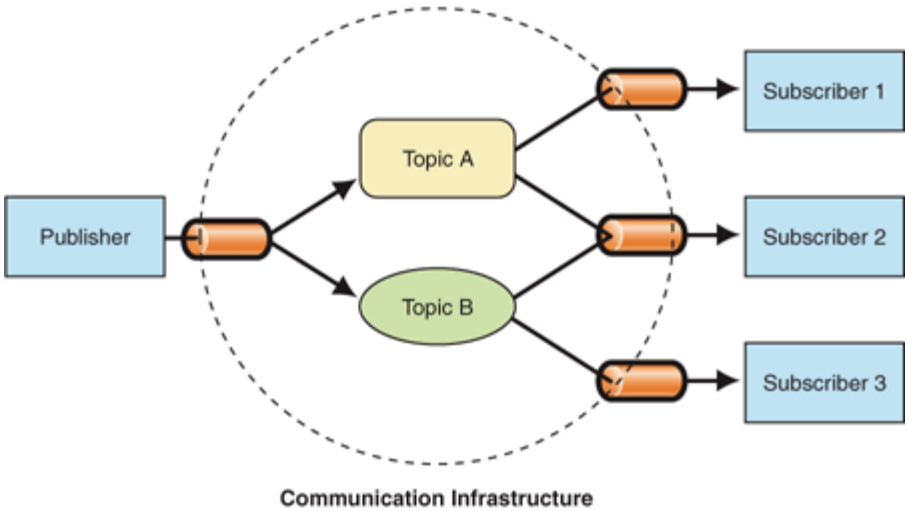


Figure 1: How the publisher is linked to the subscribers

The goal of this assignment was to create a connection between a Publisher, a Broker and a Subscriber. The publisher can publish about a certain topic and a subscriber can subscribe to a certain topic. The broker is the middleman making it all happen.

## *Subscriber*

My program allows for a subscriber to enter the topic that they wish to subscribe too. It will then Show that the packet is sent. Once the packet is sent it show a message that the packet has been then sent. And shows if it has been received correctly or not.

To achieve this, I used an Array List of all the packet topics to easy store all the information.

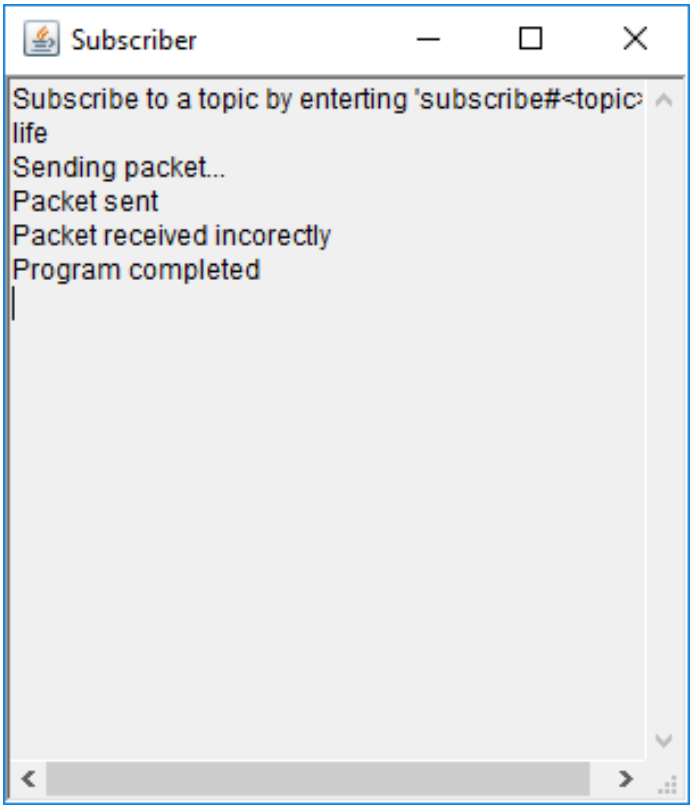


Figure 2: A screenshot of the GUI for the Subscriber

As can be seen for the above image my packet was not sent correctly. My subscriber never got to work 100% but I feel as if there was not much more needed to get it working fully.

## *Publisher*

I was able to get my publisher class to send the packet which the publisher wanted to publish. It then showed a message to say that the packet had been sent.

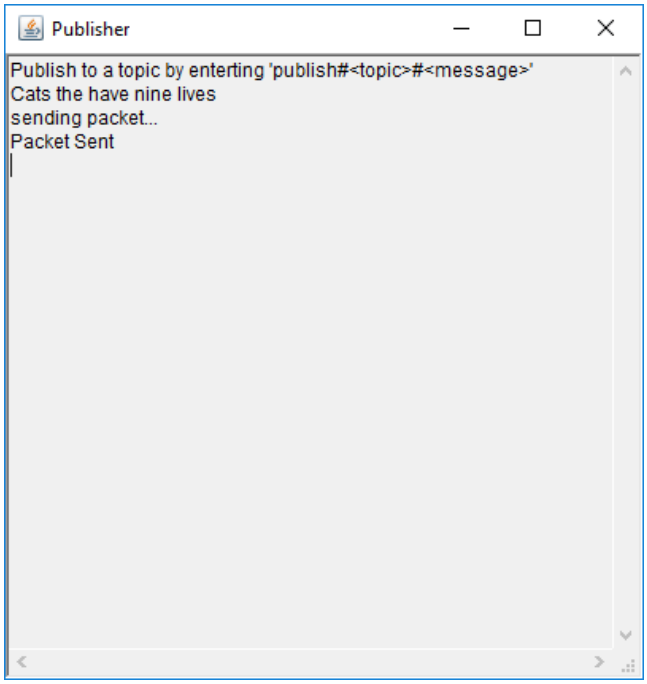


Figure 3: A screenshot from the publisher

The above image shows a screenshot saying that the packet from the publisher has been sent

## *Broker*

For my broker class it was used as a connection between the subscriber and the publisher. It created the link between showing the subscriber what the publisher had published. I used a hash map for the list of subscribers and also for the list of publishers. The program establishes contact between the publisher and subscriber

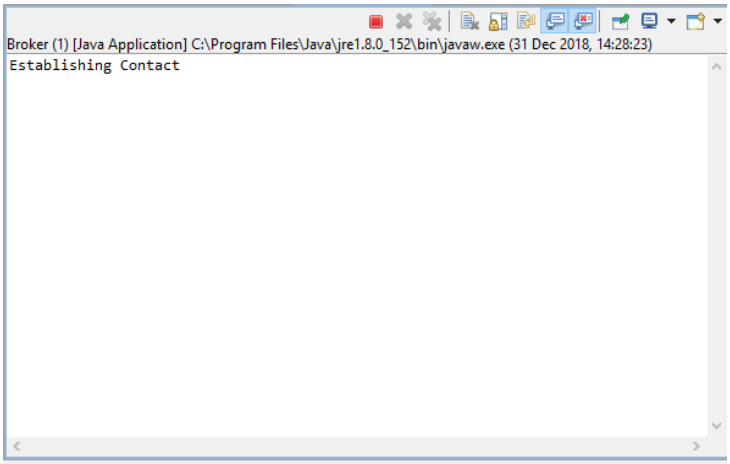


Figure 4: Broker class showing the connection has been established

## Discussion

This assignment took me the course of two weeks to complete and unfortunately due to my own bad planning between exams and life I ran out of time, I completely misjudged how long this would take me and as a result I don't feel as if I reached my full potential in completing this assignment. The project did however teach me a lot about how packets are sent and the full implementation of flow control. I found the assignment quite interesting as it thought me a lot about telecommunications and gave me a good understanding of sockets, datagram packets and threads.

## Summary

In this report, I have described my implementation of a partially working publisher-broker-subscriber model. It has also described many much of the key principals of telecommunications. Although the program did run, I feel like I didn't achieve my full potential in creating the publisher-broker-subscriber model due to time constraints. I am however happy overall with my implementation of my program