Telecommunications Assignment – 1

Tanmay Kaushik (18308341)
Teacher's Assistant – Luke Hackett

Introduction

The aim of this assignment is to design a protocol that forwards messages from a Command & Control (C&C) to a Broker which distributes work to various Workers. This assignment has helped me learn more about the concepts of sockets, datagram packets and threads.

Design

C and C sends work description to Broker.

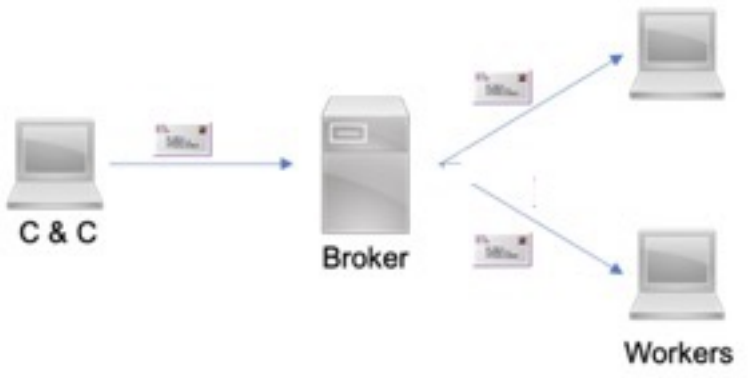
In this process, I made an arraylist in the broker class where work sent from C and C is stored. This makes the process of distributing work very simple and efficient, especially in the case when work has been sent by C and C but there are no available workers. To tackle this problem, I store the work and when any worker becomes available, work is sent to that worker.

Workers send messages to Broker indicating that the Workers are free and can accept work.

To make this work, I have made two workers with different ports. When any worker sends a message to broker, the broker saves the worker's id and port (in an arraylist called workerList). This arraylist indicates the number of workers available and their address.

Broker distributes work to workers.

Under this case, all the available workers are distributed work. This operation works in such a way that the worker from the list is chosen and is allotted the work. When the work has been allotted, that worker is removed from the list and the next worker comes in line for work.

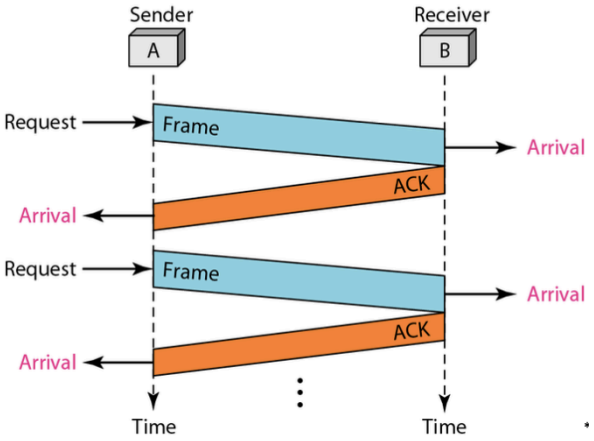


Theory

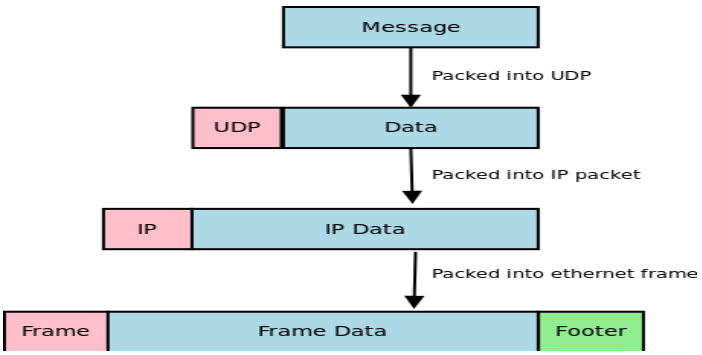
Stop and Wait ARQ is a type of error handling protocol. In this protocol, an acknowledgement is sent to the sender if the message has been transmitted successfully or not.
The propagation in my program takes place in the following way:

- The sender sends data to the receiver.
- After receiving the data, an ack is sent back to the sender.
- After receiving the ack, the sender can send the send the data again. However, if the ack is not received, the sender cannot send new data.

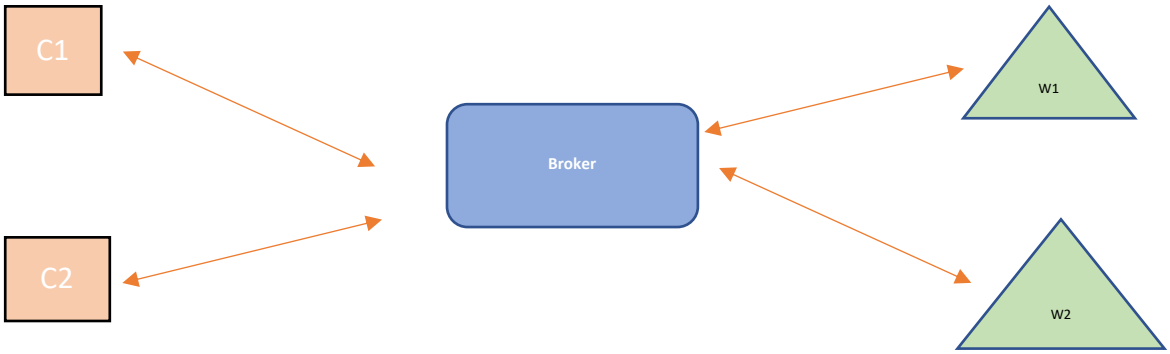
This creates a problem when the ack sent by the receiver is lost. In that case, the sender retransmits the data resulting into duplication of data. To prevent that, one can number the frames and acks to overcome the problem of duplication.



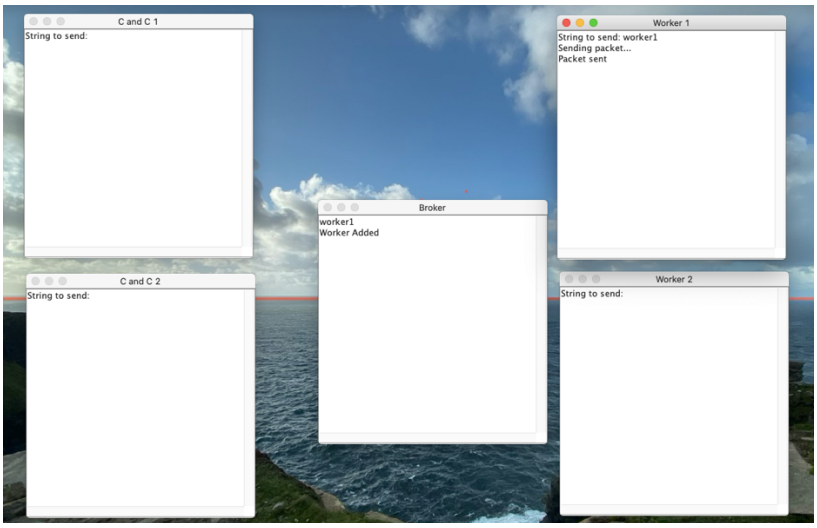
In the project, the concept of UDP is also used. UDP stands for User Datagram Protocol.
The application writes a message to UDP socket, which is then encapsulated in a UDP datagram, which is further encapsulated in an IP datagram and is sent to the destination.
This is how we are establishing communications amongst the computers in the command and control program.



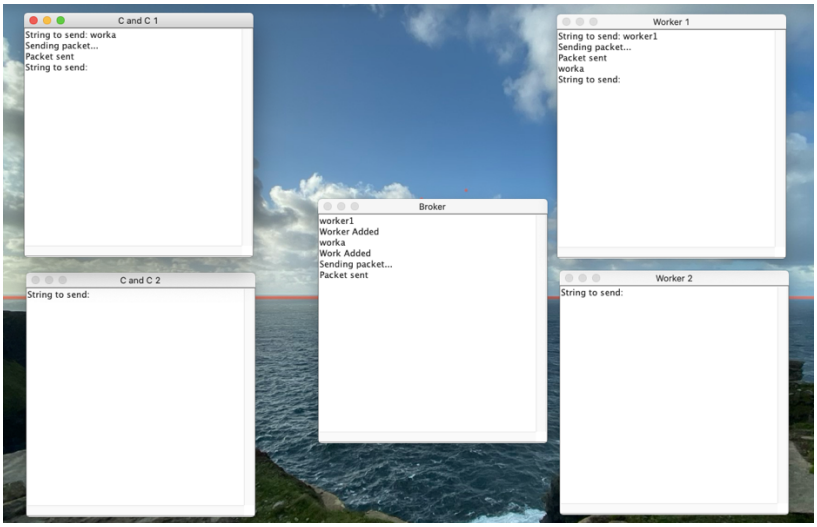
Diagram



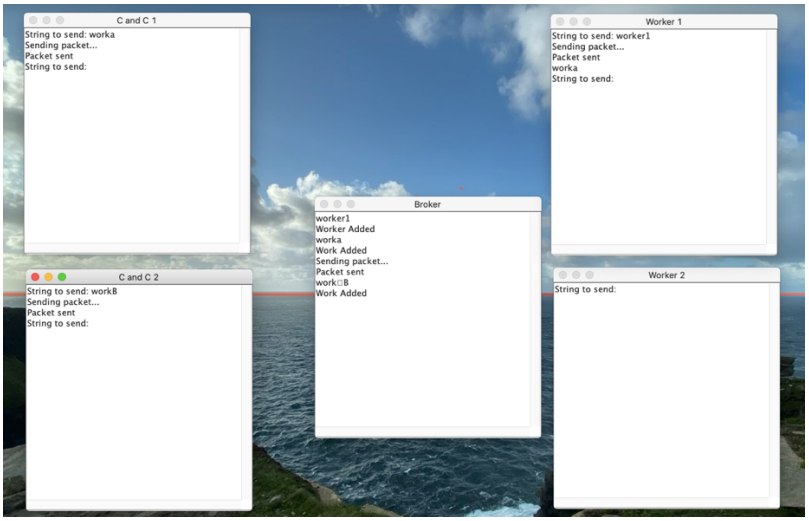
Snapshots and Explanation (for 2 C&Cs, 1 Broker, 2 Workers)



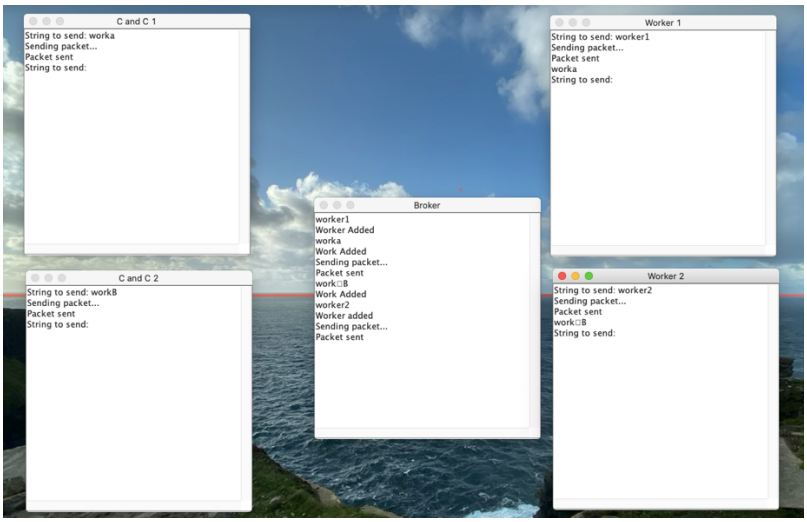
Worker 1 sends the message that it is available to accept work. Worker added to the list.



C and C 1 sends work to broker. Broker forwards that work to the available worker- Worker1.



C and C 2 sends work- “Work B” to broker. But there are no available workers, so work is added to the list.



Worker 2 sends message to broker that it is available. Broker gets the work- “Work B” from the list and sends it to Worker 2.

Working and Explanation

C and Cs and Workers send message to Broker and Broker distributes the work among the workers based on their availability.

Broker has two array-lists. When a Worker sends a message to Broker saying it’s available, that Worker’s Port is stored in the array-list (called workerList). When C and C sends a message to Broker, it stores the packet which contains the work in a separate array-list (called workList).

The code for this is shown in the two pictures attached below:

```
public synchronized void onReceipt(DatagramPacket packet) {
    StringContent content= new StringContent(packet);
    //this.notify();
    terminal.println(content.toString());


    int Port = packet.getPort();
    String message = content.toString();

    switch(Port) {


    case Constant.Worker1_PORT:
        if(message.equalsIgnoreCase("busy")) {
            for(int i=0; i<workerList.size();i++) {
                if(workerList.get(i)==Port) {
                    workerList.remove(i);
                    terminal.println("Worker Removed");
                }
            }
        }else {
            boolean isPortThere = false;
            for(int i=0; i<workerList.size();i++) {
                if(workerList.get(i)==Port) {
                    isPortThere = true;
                }
            }
            if(isPortThere==false) {
                workerList.add(Port);
                terminal.println("Worker Added");
            }

        }

        sendMessageToWorker();
    break;
```

```
    case Constant.Worker2_PORT:
        if(message.equalsIgnoreCase("busy")) {
            for(int i=0; i<workerList.size();i++) {
                if(workerList.get(i)==Port) {
                    workerList.remove(i);
                    terminal.println("Worker Removed");
                }
            }
        }else {
            boolean isPortThere = false;
            for(int i=0; i<workerList.size();i++) {
                if(workerList.get(i)==Port) {
                    isPortThere = true;
                }
            }
            if(isPortThere==false) {
                workerList.add(Port);
                terminal.println("Worker added");
            }

        }

        sendMessageToWorker();
        break;

    case Constant.CandC1_PORT:
        this.notify();
        workList.add(packet);
        terminal.println("Work Added");
        sendMessageToWorker();
        break;

    case Constant.CandC2_PORT:
        this.notify();
        workList.add(packet);
        terminal.println("Work Added");
        sendMessageToWorker();
        break;

    default: System.out.println("Port Not Found");
    }

}
```

Whenever the broker receives a message, it notifies the sender. Then it calls method called sendMessageToWorkers(). This method gets the available worker and the work from the list and sends the work to that worker. The ports of the workers are stored in the array-list workerList.

Please find the method below:

```
public void sendMessageToWorker() {
    if(!workerList.isEmpty() && !workList.isEmpty()) {
        //byte[] data= null;
        DatagramPacket packet= null;

        if(workList.get(0)!=null) {

        packet= (DatagramPacket) (workList.get(0));
        workList.remove(0);

        terminal.println("Sending packet...");

        InetSocketAddress dstAddress = new InetSocketAddress(Constant.DEFAULT_DST_NODE, workerList.get(0));
        workerList.remove(0);
        packet.setSocketAddress(dstAddress);
        //packet= new DatagramPacket(data, data.length, dstAddress);

        try {
            socket.send(packet);
        } catch (IOException e) {
            e.printStackTrace();
        }
        terminal.println("Packet sent");
        //this.wait();
        }
    } return;
}
```

In case a worker wants to withdraw its availability, they can type in “busy” and their address from the available workerList will be removed.

```
    case Constant.Worker1_PORT:
        if(message.equalsIgnoreCase("busy")) {
            for(int i=0; i<workerList.size();i++) {
                if(workerList.get(i)==Port) {
                    workerList.remove(i);
                    terminal.println("Worker Removed");
                }
            }
        }else {
            boolean isPortThere = false;
            for(int i=0; i<workerList.size();i++) {
                if(workerList.get(i)==Port) {
                    isPortThere = true;
                }
            }
            if(isPortThere==false) {
                workerList.add(Port);
                terminal.println("Worker Added");
            }

        }

        sendMessageToWorker();
    break;
```

Advantages and Features

- In my project there are two C&Cs, one Broker and two Workers. Multiple number of C&Cs and Workers can be added easily to the system.
- Lists is maintained in the Broker class for keeping a track of workers and work.
- Each C&C, Broker and Worker have been allotted a separate port number.
- When worker sends message to broker this means that the worker is available to work. However, if the worker wants to withdraw its availability, the worker can type in “busy”.
- An acknowledgement is sent to the sender after every message exchange.
- Work is not dependent on the workers and workers are not dependent on the work. If the workers are not available, the work is stored for later; and when the workers become available that work is distributed.

Disadvantages

- Currently, the C and C does not wait for the acknowledgement from the broker. However, the workers do.
- In my project, currently the retransmission of the message is not fully functional.

Reflection

This project has helped me understand a lot about data transmission, working of packets and sockets. This has been the first project for me that involves sockets and data transmission and I found this very interesting. The

project looked a bit confusing for some time, but by the end, I understood a lot about the working and managed to add quite a few features. This assignment was time consuming, but it made my fundamentals strong and cleared a lot of my doubts. I spent about 8-10 hours on the project.I also want to point out the constant help and guidance that was provided to me by the teacher and teacher assistants. They helped me a lot by telling me on how to improve my project and helped me figure out a lot of errors.