Telecommunications II Assignment

DOCUMENTATION

Koh Li Hang | 18313798 | Computer Science Year 2

Table Of Content

Introduction

The task of this assignment is to design a protocol that will forward messages between several nodes. There will be at least a C&C (Command & Control), a Broker and at least a Worker. And the task is to handle the messages sent from the C&C to the Broker and distributes these messages to the Workers.

While working on this assignment, we are mean to understand and know sockets, datagram packets and threads and to design our own protocol. After this, we should have a deeper understand on the telecommunications.

Stop & Wait ARQ

In this assignment, we are mean to design a protocol to transmit the data pack to each node. By using the sample code given in the beginning as a template (from TCD Blackboard), we are using Stop & Wait ARQ to design our protocol. In my codes, every time a message is send from a node to another node there will be an ack send back from the receiver. But since I'm simulate the whole process in the same machine, so there won't be a time out situation show up. Therefore, in my case, I am not considered about the time out situation.

Definition

The Stop & Wait ARQ, which also referred to as alternating bit protocol, is a method used in connection-oriented communication. It is the simplest automatic repeat-request (ARQ) mechanism, and offers error and flow control in a basic way which might be not very efficient. In the telecommunications, it is used in Data Link and Transport Layers.

How does it work

Assume there are two nodes in a same channel, a sender and a receiver. We are sending data pack from sender to the receiver.

In the beginning, the data will be breaks into frames (in my case the data is sorted into a byte array) and fill with 0 and 1.

A stop & wait ARQ sender only send one frame at a time, and it won't send any further frames until an acknowledgement (ACK) signal is sent back from a receiver.

In a certain time, if the ACK does not reach the sender the sender will sends the same frame again. This situation is known as timeout.

The frame sent by the sender will included its sequence number, error detection bytes and the data (or depend on how you design the data pack).

For the receiver, after the frame arrived. The receiver will check if it was the sequence number expected to check whether the frame is corrupted.

An ACK will only be send after the receiver checking process, and the next frame sequence number will be sent together too. The sender got the ACK and it will do the same thing again.

Besides that, flow control in stop & wait ARQ is a process of managing the rate of data transmission between the sender and the receiver to prevent a fast sender from overwhelming a slow receiver.

From above, we know that only one frame is sent at a time and to send the next frame will required an ACK from the receiver to send again.

For error control, there will be an error detection method on the receiver side. It will only send an ACK after the frame passed through all the check process.

And to deal with lost frame, after a certain time the sender will resend the frame again.

Advantages

- Can be used for noisy channels
- Has both flow and error control mechanism
- Has timer implementation
- Easy to understand

Disadvantages

- Not efficiency at all
- No pipelining because it needs to wait for the ack respond

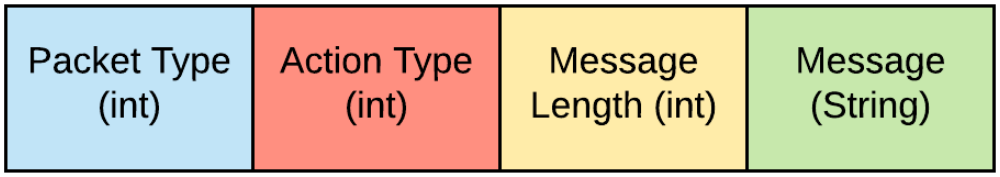
Workers

The workers have several features in this program, it is supposed to send a volunteer request to the broker to become one of the workers will accept the work description from the Command & Control application. And from the work description the workers will know how many times it should going to print. After the work has been done, the worker will have a choice to report its progress back to the broker.

Design concept – Data Packet

To transmit the data between the nodes, I design different data packets for each case. For the workers, there will have 2 type of situation that the message is being send to the broker which are the volunteer request and the ack reply. The ack reply data design will be mentioned afterwards.

My data packet design for the workers



Packet Figure 1

In the first slot, it is the Packet Type slot. It will take an integer that is a constant in both sides to distinguish the type of this data packet.

In second slot, it will be the Action Type slot. There are 3 type of actions will be taken by the workers. A Yes or No action for the volunteering process and a Work Done action for the ack being send after the work has been done. Similarly, it also takes an integer to represent different action.

For the last 2 slots, they are related. The third slot will store the length of the message and the fourth slot is the message. For the reason on putting the length of message into the data packet is that while unpacking the data packet on the receiver side, it will require the length of the message to generate a byte array with the size of the message. So instead of generating a huge byte array that can handle every message received, generating a suitable size of array might be a better way to save the memory. For the type of these 2 slots are integer for the message length and string for the message.

Pseudo Code & Explanation

onReceipt function Pseudo Code

```
onReceipt(DatagramPacket packet){
if(receivedData())
    if(isAck(packet))
    {
        checkReply(packet);
    }
    else if(isWork(packet))
    {
        startWorking(packet);
        reportBack();
    }
}
```

onReceipt function Explanation

In my codes, whenever the worker received a data packet it will be passing through this function. So, if there's a data packet this function will check the type of the packet. Depends on the type of packet, if it is an ack then it will check the reply from the broker show the result to the user. If it is a work description then it will start to work on it. And after the work has been done the worker will have to decide whether to report back the progress to the broker.

reportBack function Pseudo Code

```
reportBack(DatagramPacket packet, String name){
    choice = askForReportBackDecision();
    if(choice)
    {
        sendAck(packet, name);
        print("Progress sent.");
    }
    else{
        print("Progress not sent.")
    }
}
```

reportBack function Explanation

Basically, this function is asking the user for the decision to send the work progress of the workers back to the broker. If the user decides to send the report then it will call the sendAck function and send the progress. After that, it will also print a message to show the progress is sent. If the user chooses to not sent, then it will just simply print a message to show the progress is not sent.

sendMessage function Pseudo Code

```
sendMessage(){
    print("Would you want to work?(Y/N)")
    while(sending)
    {
        input = getUserInput();
        if(input.equals("N"))
        {
            sendQuitResquest();
            sending = !sending;
        }
        else if(input.equals("Y"))
        {
            sendJoinRequest();
            sending = !sending;
        }
        else{
            print("only (Y/N)")
        }
    }
}
```

sendMessage function Explanation

The aim of this function is to send a volunteer request to the broker, therefore the broker will be able to add it into the workers list. So, in the beginning it will print a question to ask the willingness to work of user.

After that, it will go into a while loop and keep asking the user input until the user has entered Y or N. For the workers who don't want to work, although it might not currently in the broker workers list it will still send out a quit job message to the broker.

The broker will have a mechanism to check this worker is on its workers list and if it is inside the worker list the broker will remove it.

If the workers want to work or still want to work for the broker, it will send a join request to the broker. If the worker hasn't joined the broker workers list before the broker will add it into the list; but if the list has already had this worker, the list remained unchanged.

setName function Pseudo Code

```
setName(){
    name = getUserInput();
    terminal.setName(name);
}
```

setName function Explanation

This function is used to set up the worker's name. After getting the worker's name from the user, it will call the terminal's setName function to set up the title of the terminal.

sendProgress function Pseudo Code

```
sendProgress(SocketAddress address, String name){
    progress = setUpPacket(name);
    progess.setSocketAddress(address);
    socket.send(progress);
}
```

sendProgress function Explanation

This function is used to send it's progress to the broker after received a data packet from the broker. In my case, this is used in the reportBack function. It takes a datagram packet and a string as the parameters. Firstly, it will set up a new datagram packet which is the acknowledgement we are going to send. And takes the string and set up the whole packet. For the datagram packet the function passed into this function is used to get its socket address, therefore the ack will be able to set up the socket address by using the packet's socket address. After all of this, the static socket will send out it ack to the broker.

Broker

The broker features will more than the features in Workers and C&C and will be more complex too, since it is design to handle the messages being send from workers and C&C.

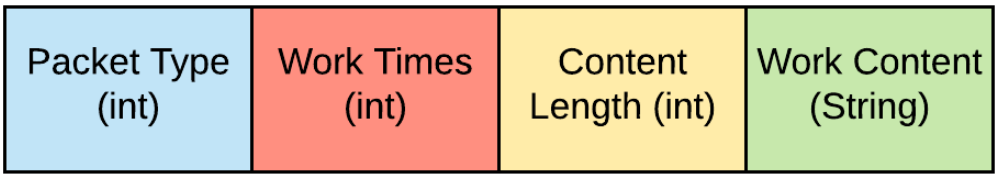
For the workers, the broker is able to collect the workers request and put them into a worker list. The list will be use to assign the work to the workers when a work description has been sent from the C&C. And it also able to get the acknowledgement after the work has been done by the workers.

For the C&C, the broker will need store the work description, required workers and work times from C&C. Therefore, the order in which data from workers or C&C arrive in succession does not affect the data being transmitted on both of them.

Design concept – Data Package

Similar with workers, broker will also need to transmit 2 different type of the data packets which are the work description to the required amount of the workers and acknowledgement to workers and C&C. But unlike workers and C&C there will be no input from the user directly to the broker, its role is more like a router.

My data packet design for the broker



Packet Figure 2

This data packet is designed almost the same compare with the one mentioned above. There are also 4 slots in this packet, and type of contents are also the same too. But the slots contents are slightly different.

The first slot still remains the same – the type of the packet, which is really important without it we won't be able to move to the next step since the packet is unknown for us.

The second slot will be storing the amount of the work that the workers need to be done. The is decided by the C&C, every worker will be doing the same amount of works assigned by the C&C.

For the third and fourth slots are also related too. The third slot will be storing the length of the work content and the fourth slot will be storing the work content in strings.

onReceipt function Pseudo Code

```
onReceipt(DatagramPacket packet){
    if(receivedData())
    {
        if(isFromC&C(packet))
        {
            extract&setUpC&CPacket(packet);
            if(!isWorkersListEmpty())
            {
                workers = sendWork(socketAddressList,
                        workTimes, workers, content);
            }
            sendAck(packet);
        }
        else if(isFromWorkers(packet))
        {
            extract&setUpWorkerPacket(packet);
            checkWorkersAction(action);

            if(isWorkAvailable() && workers!=0)
            {
                workers = sendWork(socketAddressList,
                        workTimes, workers, content);
            }
            sendAck(packet);
        }
        else{
            print("unexpected packet");
        }
    }
}
```

onReceipt function Explanation

The onReceipt function in broker works in a different way, because it will be required to have the ability to distinguish the workers packets and the C&C packets. And depends on the packet, there will be several actions that the broker needs to be taken.

First for all, after checking is there's a data packet is coming in, the function will check the type of the data packet. If it is a data packet from C&C, the broker will need to extract the data out and store them into a static value in the broker. So, these values can be used after this process.

After this, it will start to look on the workers list. If there is worker available the function will be starting to send out the works to the worker. From the codes above, the variable 'workers' is a static integer value which use to store the number of the workers that the C&C requested. So, instead of sending the work to all of the workers the C&C has the right to choose how many workers that it requested.

In the end of this process, the broker will send an ack back to the C&C, represent mission accomplished.

On the other hands, if it is a worker data packet the broker will need to take a completely different action. To begin with, it will extract the data first. Since there are 3 actions can be taken by the worker, the broker will need to check it before it moves to the next step.

For the volunteer request, it will add the worker to the workers list if it is already in the list, the list remained unchanged. For the quit request, the broker will check the workers list first, it won't do anything if it is not on the workers list, but if it is on the list the broker will remove it. And for the work done request, the broker will take the name of the worker and send it with an ack to C&C.

After that, it will check is there's any work available and the needed workers amount is not less than 1. If there's a situation, it will call the sendWork function and keep sending the work until the needed workers amount is equal to zero.

Finally, it will also send an ack back to the worker to show the packet is received.

If the type of the packet doesn't match any of those, it will just simply print out a error message.

sendAckToC&C function Pseudo Code

```
sendAckToC&C(SocketAddress address, String name){
    ack = setUpAck(name);
    ack.setSocketAdress(packet.getSocketAddress());
    socket.send(ack);
}
```

sendAckToC&C function Explanation

This function basically takes a socket address and a string as a parameter. It will create an ack data packet with the string and take the socket address to set the ack data packet sock address. Finally, send the ack to C&C. This function will be call when a worker has report it progress to the broker and the broker will call this function to send an ack including the worker's name to the C&C this is slightly different to the normal ack, that why I separate it from the sendAck function.

sendWork function Pseudo Code

```
sendWork(ArrayList<SocketAddress> socketAddressList,
        int workTimes, int neededWorkers, String workContent){
    while(neededWorkers!=0 && socketAddressList.size()!=0)
    {
        packet = setUpWorkDetails(workContent, workTimes);
        temp = socketAddressList.get(0);
        socketAddressList.remove(temp);
        packet.setSocketAdress(temp);
        socket.send(packet);
        workers--;
    }
    return workers;
}
```

sendWork function Explanation

For the sendWork function, it takes the most parameters compare with all of the functions in this assignment. We can see they are the socket address array list, the worktimes, the needed workers, and the work content. In the beginning, it goes into a while loop.

When the needed workers are not zero and also the size of the socket address list, the loop will keep going on.

Firstly, it will set up the data packet for broker to transmit the data to each worker by taking the work content and the work times. After that, it will get the first socket address from the list and remove it from the list. So, the same address won't be reuse.

It will send out the packet as soon as the socket address of the data packet has been set. Since the work has been sent to a worker, the amount of the needed workers will be deducted by 1.

And different from the others function, this function will return the amount of the needed workers. Since there might be a situation that the amount needed workers is greater than the available workers. So this will return the left of the needed workers and store in the static value in broker.

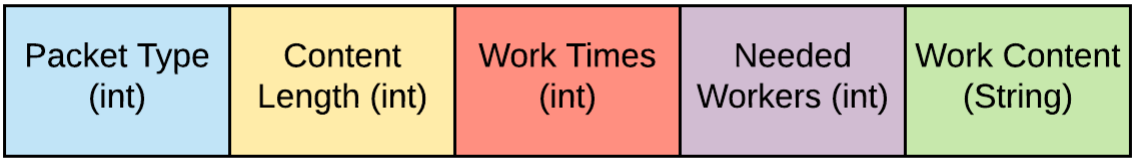
Command & Control (C&C)

The C&C concept is kind of simple, basically it will just handle on the work description given by the user, pack those data up and send to the broker. The hardest part is to think about how to design the data package, since this package is going to store the most data in this assignment.

Design concept – Data Package

The design concepts of the data packages are all similar but it might change a little bit depends on how many kinds of the data you are going to transmit. For the one in C&C, there are also 2 type of data package. One of them will be the acknowledgement which are the same with the others and a different one that use to transmit the work description getting from the user.

My data packet design for the C&C



Packet Figure 3

This data packet design has added a new slot, so instead of having 4 slots like those packets mentioned above, the data packet of the C&C has 5 slots.

There will still have the packet type slot, the length of the content slot, the work content slots and the work times slot. These four slots have been mentioned on the broker data packet design, but here is a new one which is the needed workers slot. Basically, it is just integer and from my design the slots from the first one to the fourth one are all set to a size of a byte. And the size of the work content slot is depending on the length of the string that the user input. But this might cause some problems on between slot 2 and slot 4. Since 1 byte can only store with an integer between 0 to 127. Therefore, when it is over this range, the data packet might be corrupted. So, I've set up a mechanism for the user input that the user shouldn't input the work time and the needed workers is below 0 or over 127. For now, this might be the best way for me to handle this error.

onReceipt function Pseudo Code

```
onReceipt(DatagramPacket packet){
    if(receivedData())
    {
        if(isAck(packet))
        {
            print("Respond received.");
        }
        else if(isProgress(packet))
        {
            whoHasDoneIt = extract(packet);
            print(whoHasDoneIt + "has done the work.");
        }
        else{
            print("unexpected packet.");
        }
    }
}
```

onReceipt function Explanation

For any onReceipt function in my codes, the first step is always checking the arrival of the data packet. After that, there will a checking data packet type process. If it is an acknowledgement, the terminal will simply print out “Respond received”.

Else if the packet is a progress packet which it will show which worker has reported the work progress, there will be an extract data process. Therefore, this function will be able to get the worker’s name from it and print out the result on the terminal.

And finally, for those unspecified data packets, this function will print out a” unexpected packet” on the terminal.

sendMessage function Pseudo Code

```
sendMessage(){
    // in a range between 0 and 128
    workTimes = (int)(isNumeric(getUserInput()));
    neededWorkers = (int)(isNumeric(getUserInput()));
    workContent = (String)getUserInput();
    packet = setUpPacket(workTimes, neededWorkers, workContent);
    packet.setSocketAddress(brokerAddress);
    socket.send(packet);
}
```

sendMessage function Explanation

The function will start to ask for the user input in the beginning. So, it will be asking for the work times for the workers, the needed workers and the work content from the user. I realized there will be a problem if the work times and the needed workers amount is over 127, because I set up the slot space for only 1 byte which can only store up an integer between 0 and 128. Instead of generating a bigger space for them, I simple set up a mechanism that only allow the user input an integer that is between 0 and 128. But there's still might have a problem when the work content's string size is over 127. Unfortunately, I don't have enough time to deal with it. This is a disadvantage of my data packet design and the worker's sendMessage function is having the same issue too. Because in their both packet, there is a content length slot. This will cause the worker could not have a name with over 128 character so as the work content from C&C.

isNumeric function Pseudo Code

```
isNumeric(String input){
    try{
        int check = Integer.parseInt(input);
    }catch(NumberFormatException){
        return false;
    }
    return true;
}
```

isNumeric function Explanation

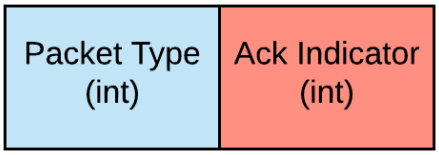
Basically, I implement this function to check for the user input for the work times and the number of needed workers. Since the input from the user will always be a string in this case. So, a function that can check the string is numeric might be an efficient way to deal with it.

There's a try and catch statement inside it, if it passed through the test it will return a true, and false if failed.

sendAck function for Workers, Broker and C&C

The sendAck function is the same in these different classes, expect for the specific one for the broker to send the progress back to the C&C. And for that one, it's kind of similar with the sendProgress function in workers.

My data packet design for the Ack



Basically, it is just a simple data packet which has a packet type slot and an ack indicator slot. Since it's just an acknowledgement to let the nodes knows the packet is received or not.

sendAck function Pseudo Code

```
sendAck(SocketAddress dstAddress){
    ack = setUpAck();
    ack.setSocketAddress(dstAddress);
    socket.send(ack);
}
```

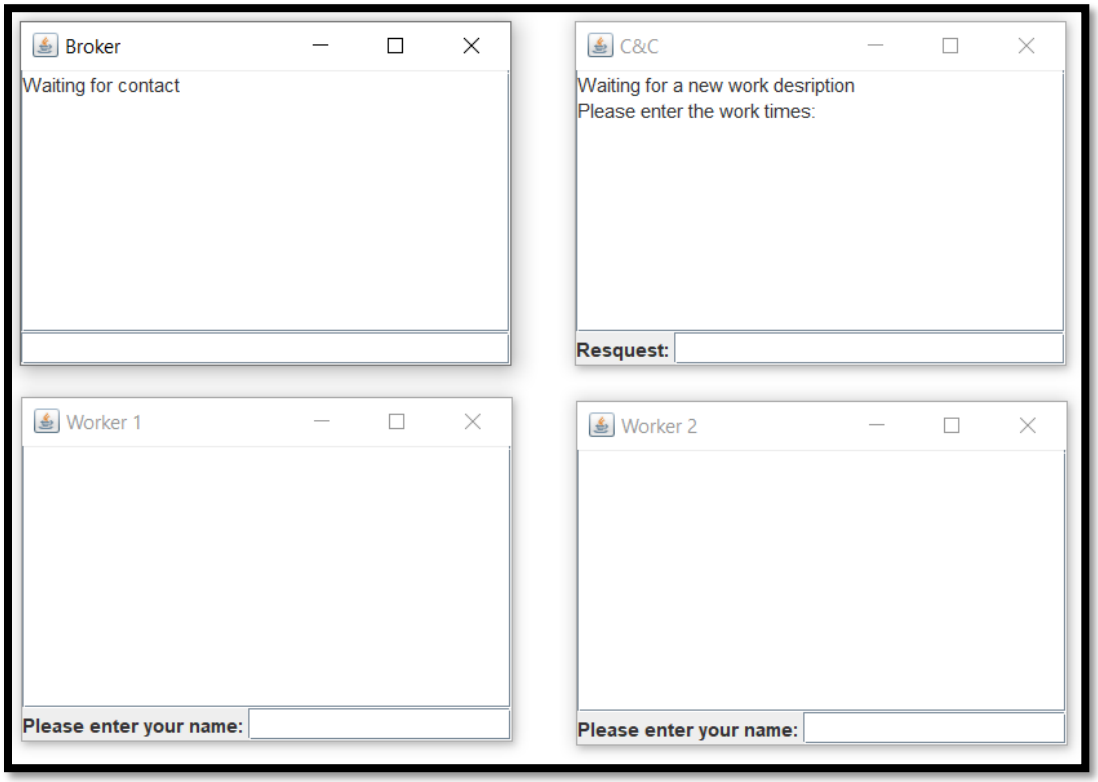
sendAck function Explanation

There's no much things to explain over here, this function only takes the destination socket address as parameter to set up the socket address for the ack. And the socket will send out the packet.

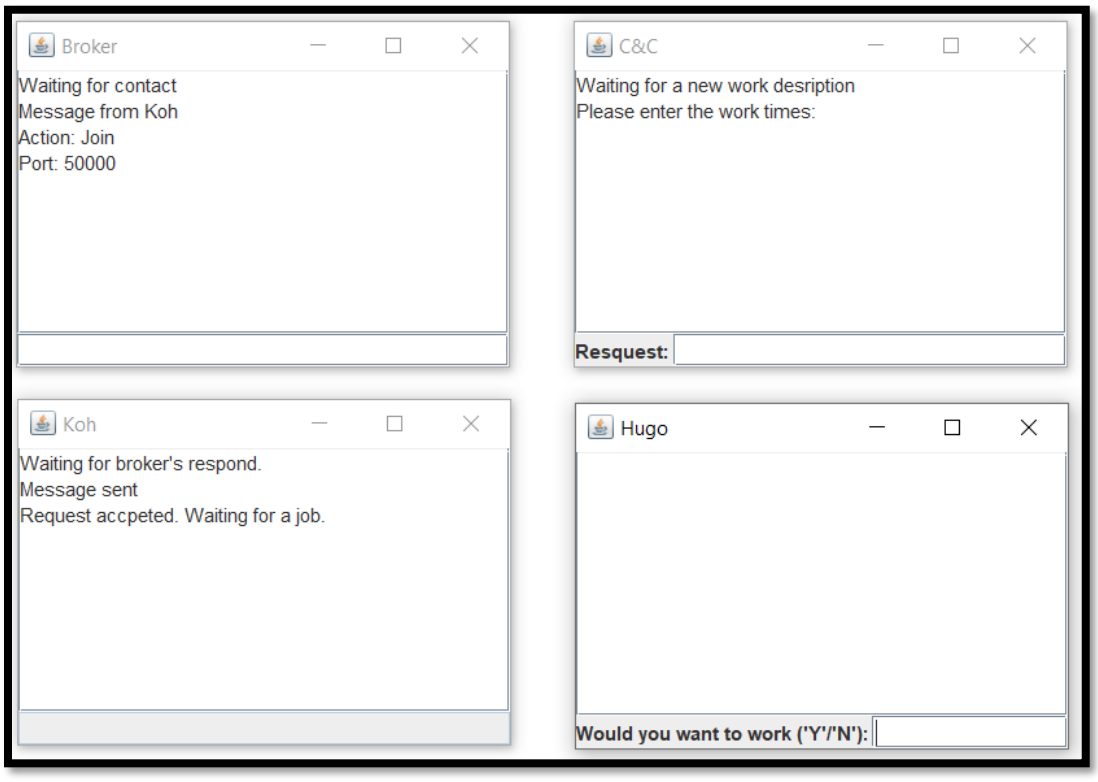
For the ack indicator, it is just basically a constant that can be access in all classes to identify the ack.

Codes Operation Process Display

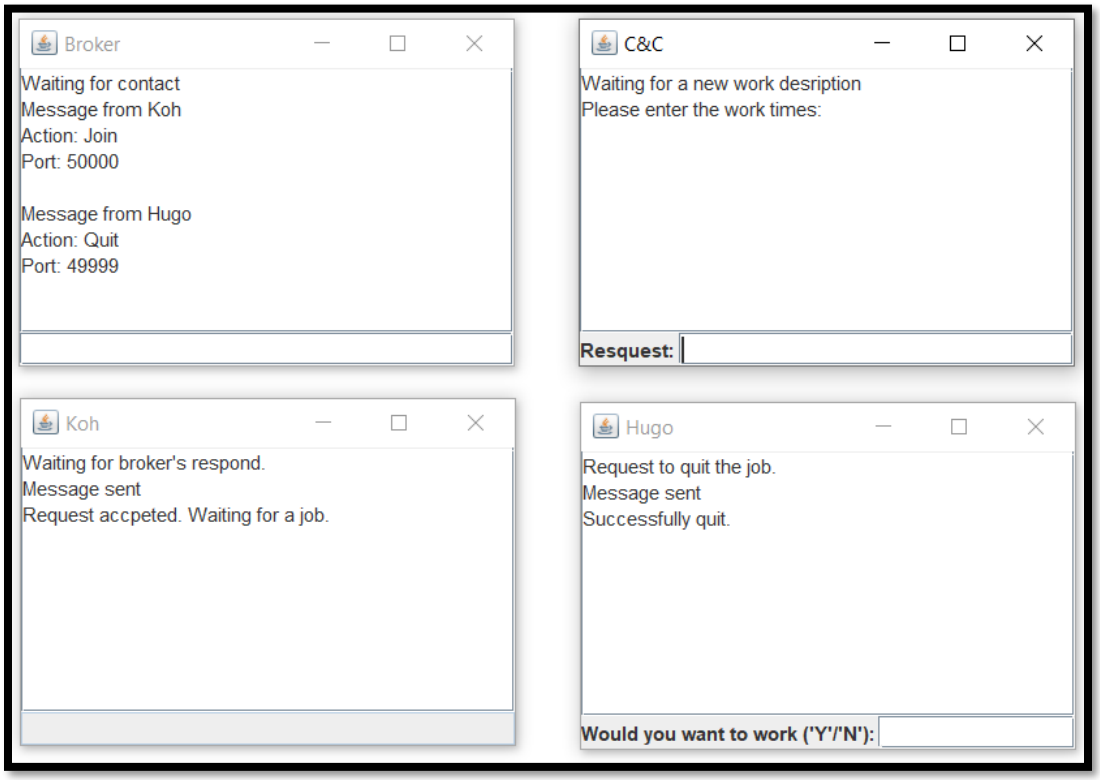
1) The starting interface for the assignment



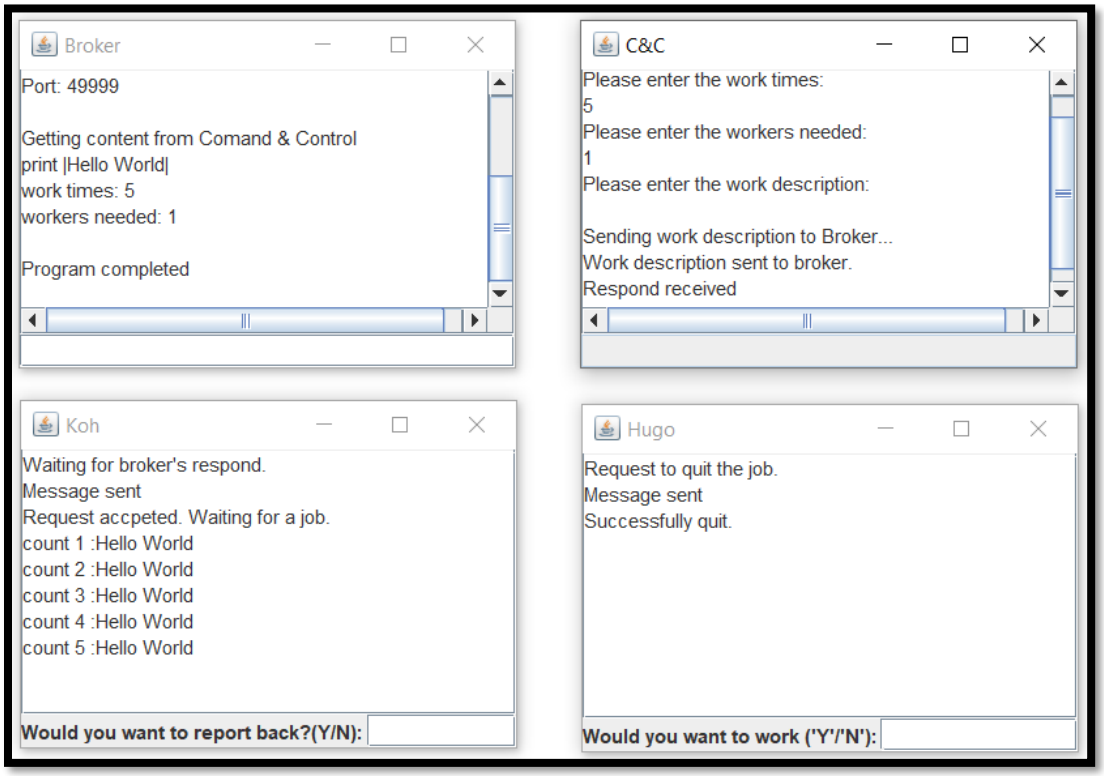
2) One worker has joined the broker while the other one still deciding.



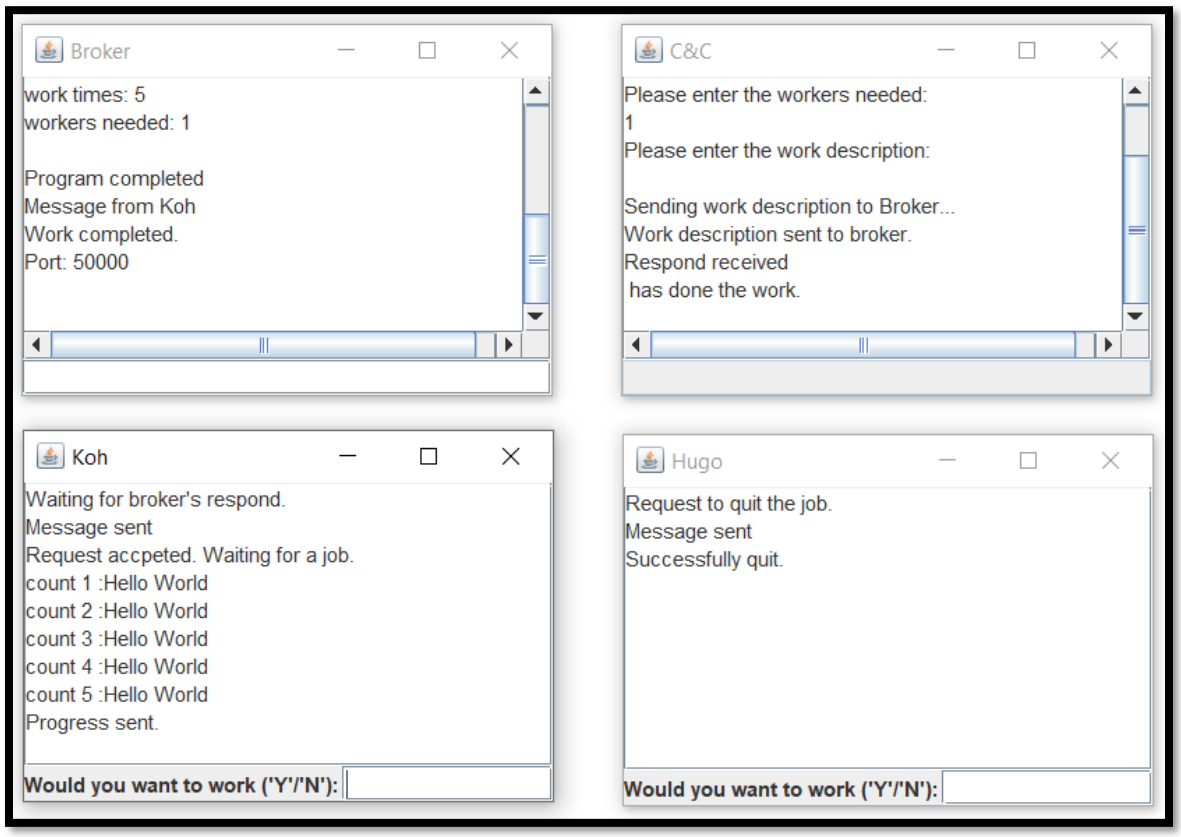
3) The second worker decided not to join the broker, but the request still send to the broker. This action is to remove that worker might be already in the list. And the terminal will ask again for the decision of the worker.



4) In this phase, the C&C has sent the work description to the broker and the broker will soon assign the work to the worker. As you can see the work has been done and it is asking the decision of the worker to send the progress back.

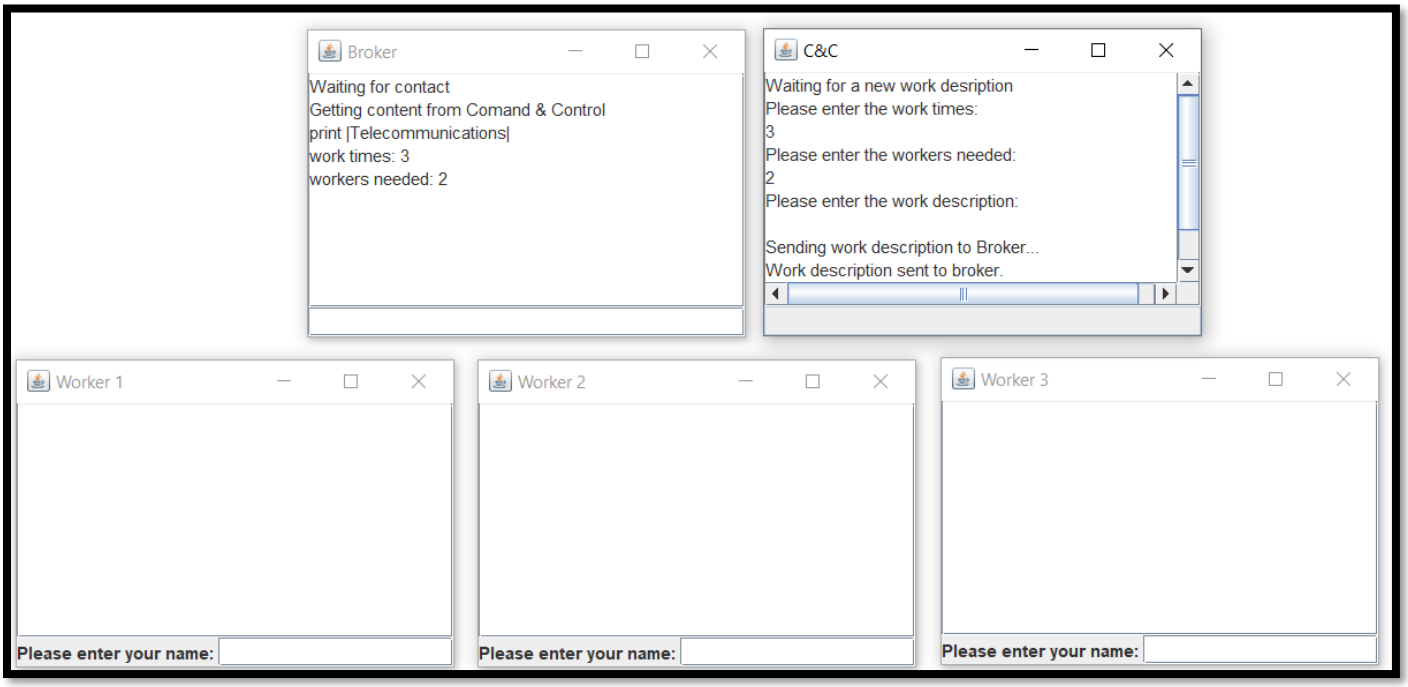


5) The worker decided to send the progress back, and the broker has received the respond. After that, it also sent back a message to the C&C. The C&C has received the message, but for some reason it couldn't print the worker's name out. It just printed out " has done the work", I have tried to fixed this issue but unfortunately I couldn't find the reason.

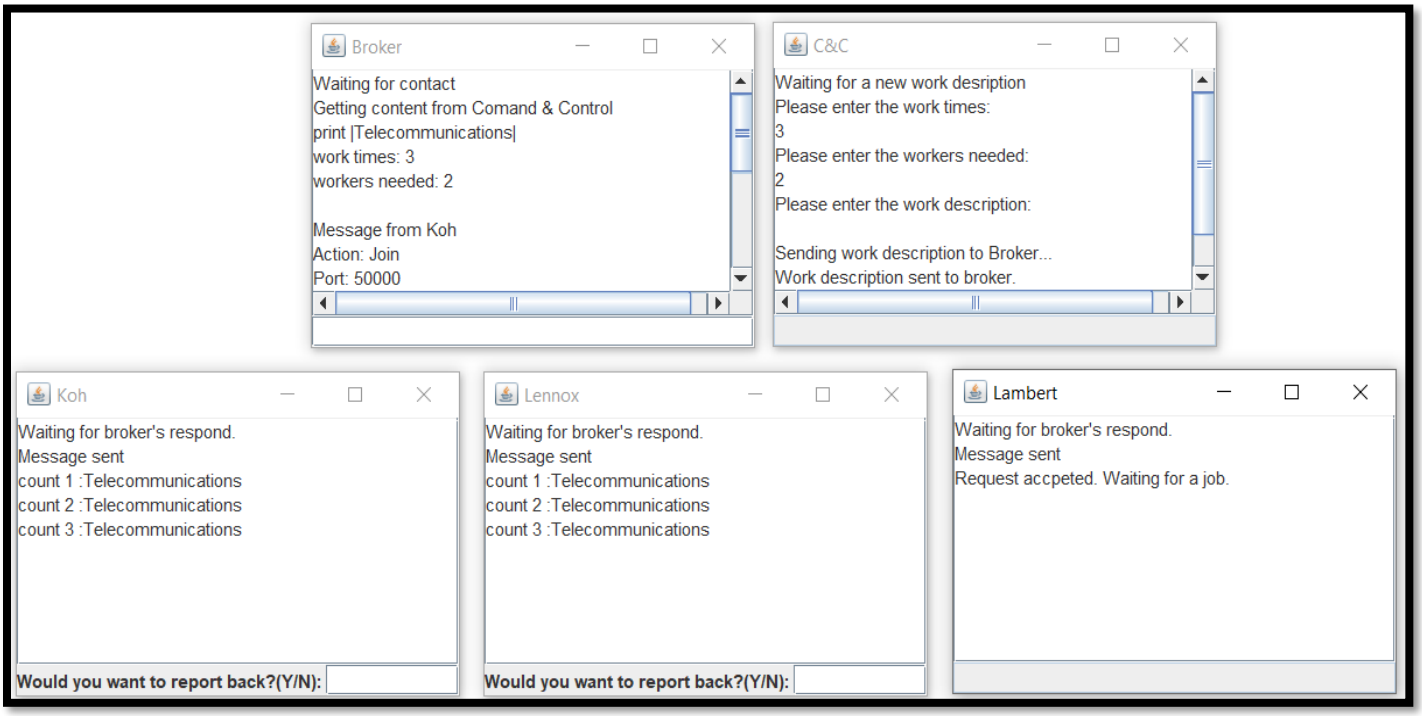


6) This is another example; it is going to show the C&C will be the starting point and the workers in this example are not fully used.

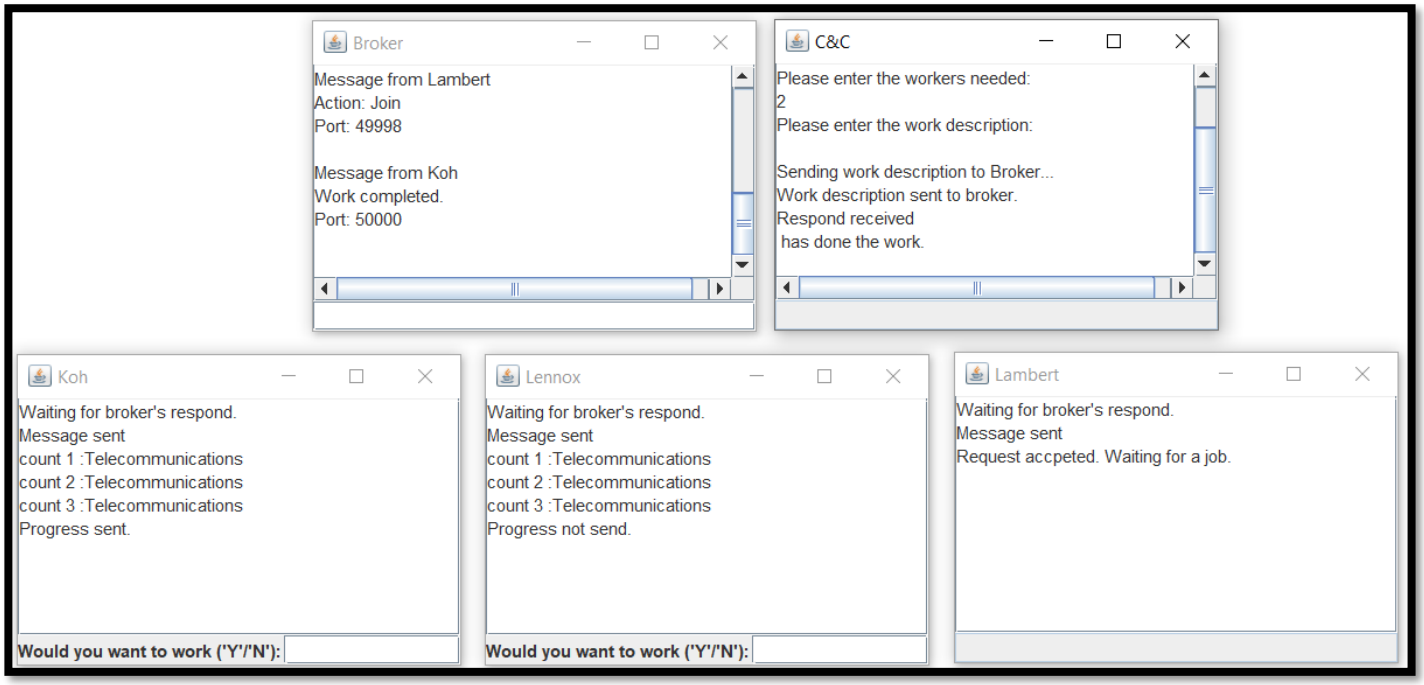
In the image below, the C&C has sent out the work description to the broker. Currently, there's no worker available to start working.



7) From the next image, we can see although the 3 workers have sent out the volunteer request but there are only 2 workers received the work description. And the third worker is still waiting for a work. This is because the C&C only needs 2 works. Therefore, the third worker will not receive a work from the broker.



8) From the last image, the C&C still having the same issue that the worker's name didn't show up. And actually, for some reason the broker will only receive the progress report from 1 worker. This is also one of the problems in my codes, I was thinking the broker will stop running after received a single respond from the worker. But I don't really have enough time to fix it which it's really a pity.



Reflection

From this assignment, it really gave us an opportunity for us to get in touch of the network layer in telecommunications. From understanding the protocol works, and learn how to design the data packet. And also, the aim of this assignment, to get know of the sockets, datagram packets, and the threads.

To be honest, I am totally lost in the beginning. Even though the lecturer has given the sample codes on Blackboard, but I'm still confused on what we should do. After spending time on researching the information about this assignment, and attending every telecommunication lecture I finally starting to understand what is going on. But in that time, I think is a little bit late because it is already Week 3. Luckily, I still came out with a decent report and codes at the end.

It takes a really long time for me to understand the concept of designing a data packet. And how the sockets and threads work. I learn how to use these features from my friends, since these things were mentioned a little bit on the lecture.

I got satisfaction after finished this assignment, and for the next assignment I will start it early so that I will have enough time to understand and work on it.

Estimation of time spent

For this assignment, it is kind of hard to estimate the total time spent. But I would say I spent over 60-70 hours on it. This is including the time I was doing the research about it.