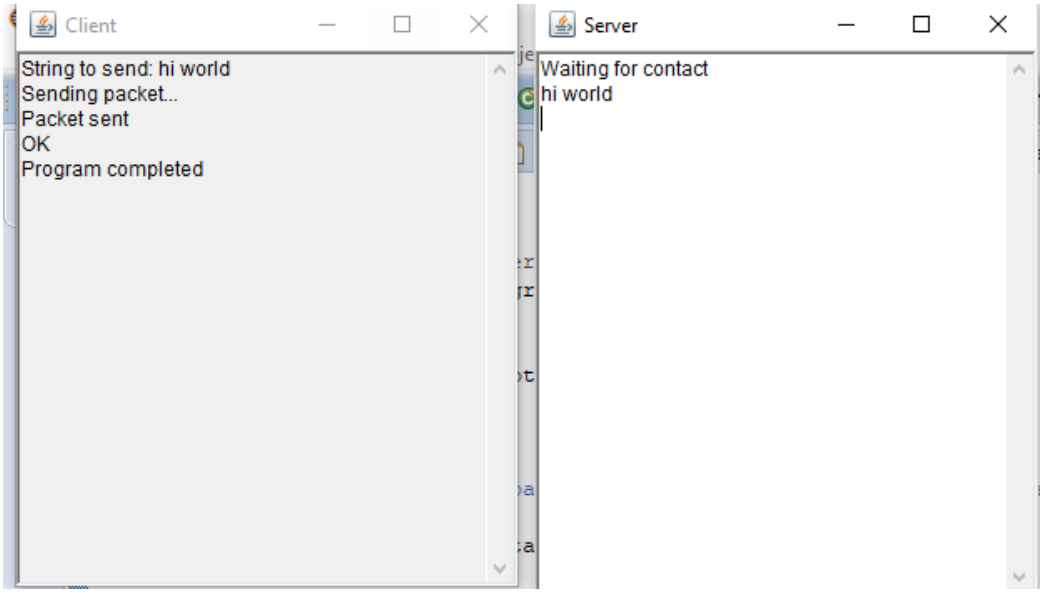
CS 2031: Telecommunications

Assignment #1

Jakub Slowinski: 16319781

My report for the first telecoms assignment is as follows:

This is all my program could manage after the first week. The longer I spent on this project, the more it flourished as a program and became more complete.

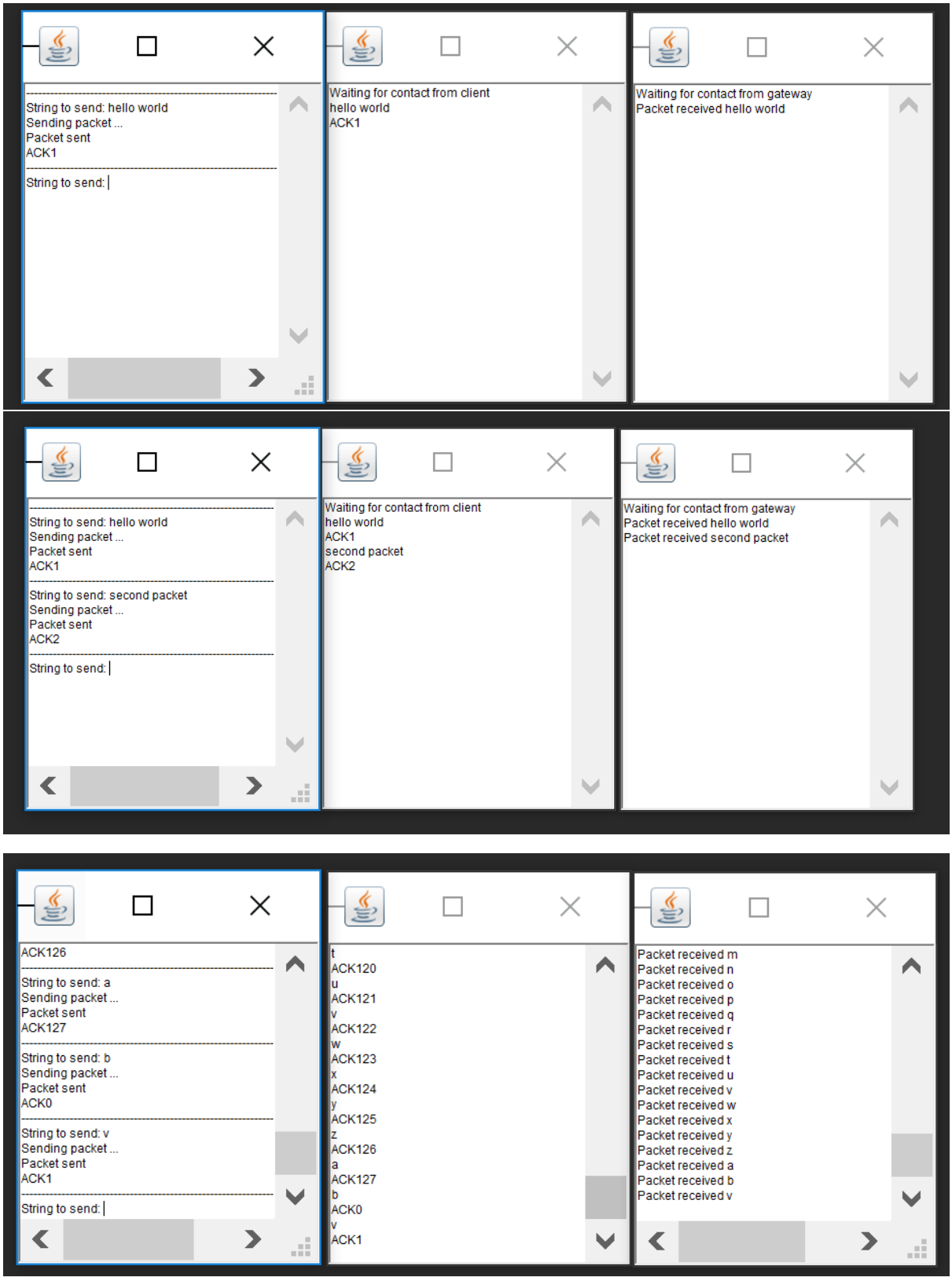


In my final solution, yhe client sends a message to the server, through the gateway. The server then acknowledges the message and then the client sends the next piece.

The Client opens a port on port number 50000, creates a packet from the input provided by the user and sends this packet to another port on the local machine with the port number 50001 which sends it to port 50002.

If the piece came in the wrong order, then the server acknowledges an error which allows the client to retransmit the failed message again.

My implementation allows a sender node to send a packet to a gateway with a packet that should be forwarded to a server. The server replies to the receipt of a packet with an acknowledgement. The packet from the sender carries a sequence number and the acknowledgement carries the sequence number that the server expects next. The server sends a negative acknowledgement to the client if it receives a packet with a segment number that it did not expect.



The client sends a packet which passes through the gateway to the server. The server sends back an acknowledgement asking for the next packet, ie. ACK1 means its expecting the second packet (packet no.1).

Client class:

The client operates on port 50000

My client class takes in a string which becomes the payload of the message being send to the server.

The client class puts the packet number into the first space of the header array (header[0]) and the receiver number (1) into the second space of the array (header[1]) which means that the intended target is the server.

```
header = new byte[PacketContent.HEADERLENGTH];
header[0] = packetNumber;
header[1] = recNumber;
```

The packet number goes from 0-127 as it is in signed bytes. The start function also saves the last send payload and last sent packet number in case it has to be resend in case of an error.

Upon receipt from the gateway from the server, the client checks did it receive back an error. If it receives an error then the client proceeds to resend the previous packet. If everything went according to plan, the client notifies itself to stop waiting and prints out the message received.

Gateway class:

The gateway operates on port 50001

My gateway class possessed 2 destination addresses, the client and the server. The gateway prints what it receives whether the information is coming from the server or the client. The gateway then extracts the header[1] of the packet which contains its destination. 1 to go to the server or any other values to return to the client. The gateway then proceeds to reroute the packet and send it to its correct destination.

Server class:

The server operates on port 50002

My server class upon receival of a packet compares the packet number to the previous one.

```
if (ack == 127)
    ack=0;
else
    ack++;
if (getNextPack() == 127)
{
    if (ack == 0)
        seq = Byte.toString(ack);
    else
    {
        error = true;
    }
}
else
{
    if (ack == getNextPack() + 1)
        seq = Byte.toString(ack);
    else
    {
        error = true;
    }
}
```

If the packet isn't the number after the last packet then the server will make the acknowledgment into a negative number signalling an error.
The server prints confirmation of receival and the message received if it came across successfully.
The server sends the acknowledgement back to the place it came from which should be the gateway.

Packet content class:

My packet content class serves as an interface. It possesses the toSting() and toDatagramPacket() methods. It also holds the length of the header as HEADERLENGTH.

String content class:

String content implements the packet content class. It returns the string and makes a datagram packet through.

Node class:

The listener function in the node class listens for incoming packets on a datagram socket and informs registered receivers about incoming packets. It listens for incoming packets and informs receivers upon arrival.

Errors:

No matter how much research I did I was not able to get multiple clients open at the same time.

I was also not able to resend the packet if it has not received an acknowledgement after a given time. I had experimented with the setSoTimeout method but after approximately 6 hours work I reached a dead end. You can see that my code still catches socket timeout exceptions even though they won't be thrown.

Conclusion:

I spent approximately 26 hours on this project including the report.

All in all this was a very challenging assignment to work on. I started with little to no knowledge of how sockets, datagram packets and treads work but just a solid understanding of how java works. As time went on I understood more and more of how what I was doing was designing a protocol, both packet layout and handling for the communication between client and server. This assignment made me see how the study of telecommunications can be practically applied to real life. Since it was done in java I had an environment that I was already familiar in yet a daunting new challenge of TCP and UDP ports being used. I think my gateway implementation worked way for me as I found an effective solution with not too many lines of code.