Elton Sia A00800541

Thilina Ratnayake

COMP 5D

COMP 7005 Project

Dec 2, 2014

**Table of Contents**

**Summary ……………………………………………………………………………………………………….. 3**

**Instructions ……………………………………………………………………………………………………. 4**

**Finite State Machine ………………………………………………………………………………………. 5**

**Psuedo Code ………………………………………………………………………………………………….. 6**

**Testing …………………………………………………………………………………………………………. 13**

**Summary**

The project includes 3 main java classes: Host A, Host B, and Network. Both Host A and Host B can be used as a transmitter or receiver, however only 1 can be a transmitter while the other would be a receiver.

When a host sends, it creates a packet and a window and fills that window with packets and sends it over the network. The network will then process those packets one by one and randomly chooses which to drop. After it processes which packets to drop, it sends it over to the second host where the second host will receive it and send it back to the network( at this point, it is now an ack that is being sent back). The network will then process the acked packet to be dropped and sends it to the first host. When the first host receive the ack packet, it then removes the acked packet and moves on to the next set of packets to be sent. For example, Host A sends packets 0, 1, 2, 3, and 4 to the network. The network then process which packets to be dropped, let’s say the network dropped packets 2 and 3. The network then sends 0, 1, and 4 to Host B and Host B receives and sends it back to the network. The network then process which packets to be dropped, let’s assume that nothing gets dropped. The network then sends 0, 1, and 4 to Host A. Host A processes the ack packets and removes it from the window. Host A then resends 2, and 3, along with 5, 6, and 7 into a new window. The program will keep on running until the EOT has been reached. Currently, we have set it to send the 15th packet as the EOT.

For timers, our window is an array list that holds packets to be sent, and in flight. When the host is in send mode, we send out every packet in the window. Once the last packet in a window has been sent, a timer begins. Every-time a packet is received, that packet is removed from the array list (window). If the array is empty, that indicates that all packets have been received and the timer is cancelled. If the timer expires, the array is checked; If there are still packets in the array when the timer has expired; that indicates packets that have not been received and therefore have timed out.

A log file is created for Host A, Host B and Network when the program ends. We also have a configuration file for the users to follow. The only thing that is needed to be changed is the IP address in the configuration file.

**Instructions**

1. In the config file, make sure that the IP addresses for each machine are correct
   1. hostIPA is the IP address of HostA
   2. hostIPB is the IP address of hostB
   3. hostAToNt is the IP address of Network
   4. hosBToNet is the IP address of Network
   5. Other than the specified changes that is needed from above, the default value will make the program run successfully.
2. Start up Host A
3. Start up Host B
4. Start Up Network
   1. Type in “a” if you would like HostA to be the sender and HostB to be the receiver
   2. Type in “b” if you would like HostB to be the sender and HostA to be the receiver.
5. The program will finish when an End of session is seen for the sender, and an EOT packet sent from the sender.

**Finite State Machine**

**Psuedo Code**

**Network:**

Main()

{

Ask user to input who will be the sender

Ask user to input the bit error rate that they would like

Create a new thread that listens to host A

Create a new thread that listens to host B

Join the threads so that the main will have to wait till the thread is done before stopping it.

}

sendControlSignal()

{

Get the IP addresses of host A and host B

Create socket for host A to use

Create socket for Host B to use

Create packet to send the command to host B and host B

Send to commands to Host A and Host B

}

HostThread()

{

Creates a thread for Host A

Creates a thread for Host B

}

Run()

{

Create a random number generator

Create socket for inbound packets

Create socket for outbound packets

Create byte array for receiving data

Create byte array for sending data

Receive packet that is currently in the socket

Evaluate that packet whether it will be dropped or not

If it is dropped, go to next packet, but if not, forward that packet using the outbound socket to the receiver

}

paxType

{

Change the type of packet from numbers to words for readability

}

**Packet:**

Packet()

{

The structure of the packet

}

Getters and setters for each type in the structure to be called by the other classes.

**Config:**

Config()

{

Create properties

Create input stream to read files from

Load the input stream into the property

}

Getter and setter for the property to be called by the other classes.

**HostA/HostB:**

Main()

{

Create a config object to be able to use the parameters set in the configuration file

Create socket to receive the first command from the network

Create an array list for the packets container

Create an array list for the packets that are acked

If the command received is 1 then go into send mode

If the command received is 2 then go into receive mode

Send packets while send mode is true

Receive packets while receive mode is true

}

ListenForCommand()

{

Wait for command to be received from the network

Once the command is received from the network, return it so that we can start sending receiving.

}

SEND()

{

Create packets to send

Add those packets to the window

Send those packets

}

prepPacket()

{

Create a byte array output stream

Create object output stream

write the packet into the object output stream

return the output stream

}

sendPackets()

{

Create a socket for sending packets

Check if the packet is an EOT , if its not, create a packet

Send the packet using the socket

Create a timer for the packet

Close the socket

}

sendPacket()

{

Create a socket for sending the EOT packet

Check if the sequence number is the EOT, if it is, then change the type of the packet to an EOT type

Create a packet of type EOT

Send the EOT packet

Create a timer for the EOT packet

Close the socket

}

RECEIVE()

{

Create a byte array to receive the data

Create a socket for listening

Create packet

Receive whatever is on the socket put that into the packet

Create byte array input stream

Put the byte array input stream into the object input stream

Cast the object input stream into a packet

Call removeInWindow()

Call checkArray()

}

RECEIVEACK()

{

Create byte array for receiving acks

Create socket to listen for ports

Create packet

Receive packets from the socket and put it into the created packet

Check if the packet is not of type EOT, if its not then convert the packet into an ACK

Create socket

Send the packet to the new socket

Close the socket

}

removeInWindow()

{

If EOT exists, remove any packets after the EOT packet inside the window

}

checkArray()

{

Checks if the array (window) is empty

If it is empty, stop the timer

Call postConversation

}

TIMEOUT()

{

Cancel the timer

Call postConversation

}

postConversation()

{

Send a last message

}

**timeOut:**

timeOut()

{

Create string to find out which hose to time out

}

Run()

{

If the host is host A then timeout host A

Otherwise timeout Host B

}

**Testing**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| # | Name | Resources | Expected | Actual | Result | Fig |
| 1 | Host A (transmitter) sends packet to Host B (receiver) | HostA, network, HostB | Host A is able to send packet to Host B | As expected | Pass | 1.1, 1.2 |
| 2 | Host B (transmitter) sends packet to Host A( receiver) | HostB, network, HostA | Host B is able to send packet to Host A | As expected | Pass | 2.1, 2.2 |
| 3 | Network drops packets | HostA, network, HostB | Network randomly drops packets | As expected | Pass | 2.3 |
| 4 | Host A sends packet to Host B and Host B sends back an “Ack” | HostA, network, HosB | Host A sends a packet and Host B sens back an “Ack” | As expected | Pass | 1.1, 1.3 |
| 5 | Host B sends packet to Host A and Host A sends back an “Ack” | Host B, network, Host A | Host B sends a packet and Host a sends back an “Ack” | As expected | Pass | 2.1, 2.5 |
| 6 | The “Syn” that has been “Acked” is deleted from the array list | HostA, network, Host B | Any “Syn” that has been “Acked” should be deleted | As expected | Pass | 1.4 |
| 7 | The “Syn” that has not been “Acked” is resent | Host A, network, Host B | Any “Syn” that has been not been “Acked” should be resent until it gets back an “Ack” | As expected | Pass | 2.4 |

Fig 1.1

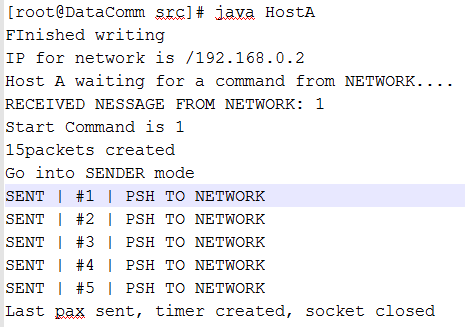


Fig 1.2

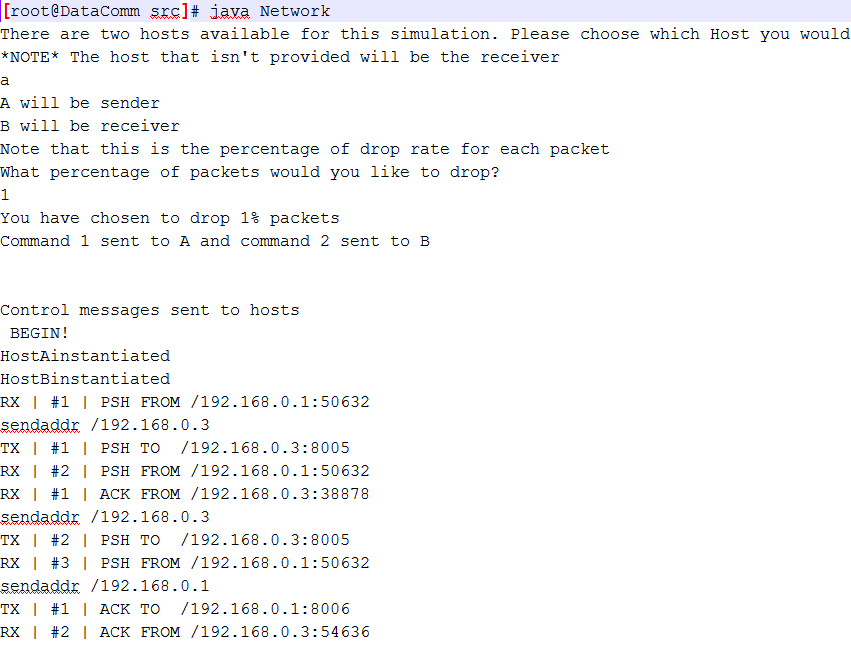


Fig 1.3

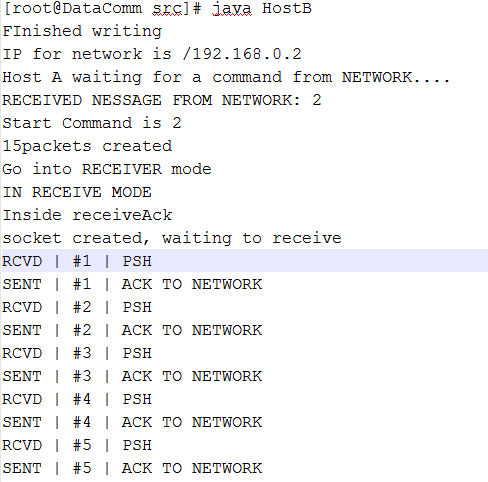


Fig 1.4

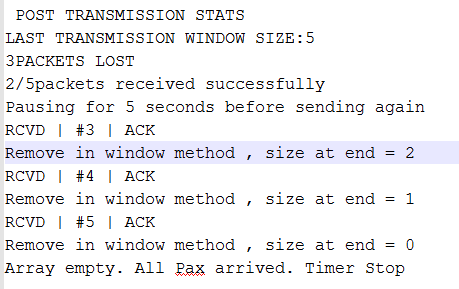


Fig 2.1

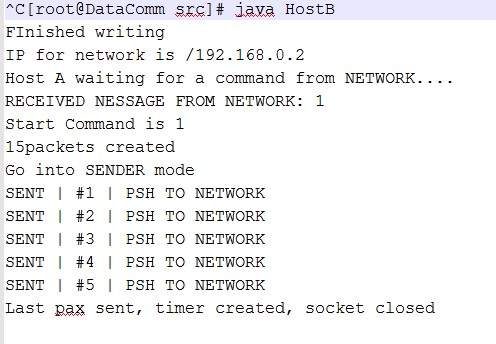


Fig 2.2

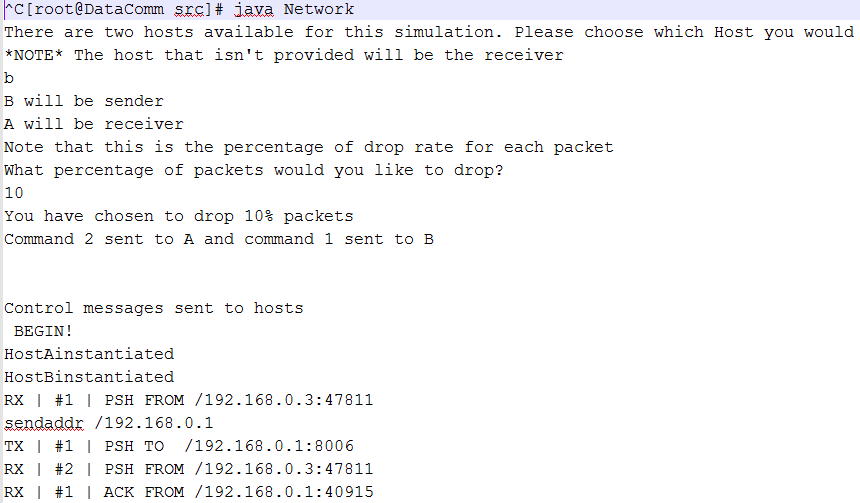


Fig 2.3

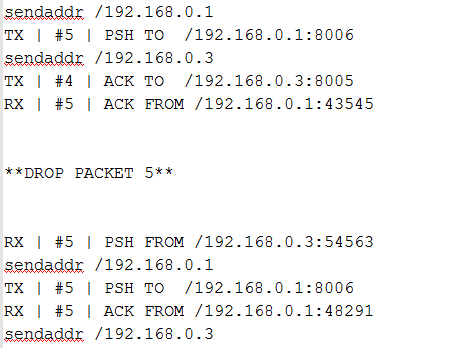


Fig 2.4

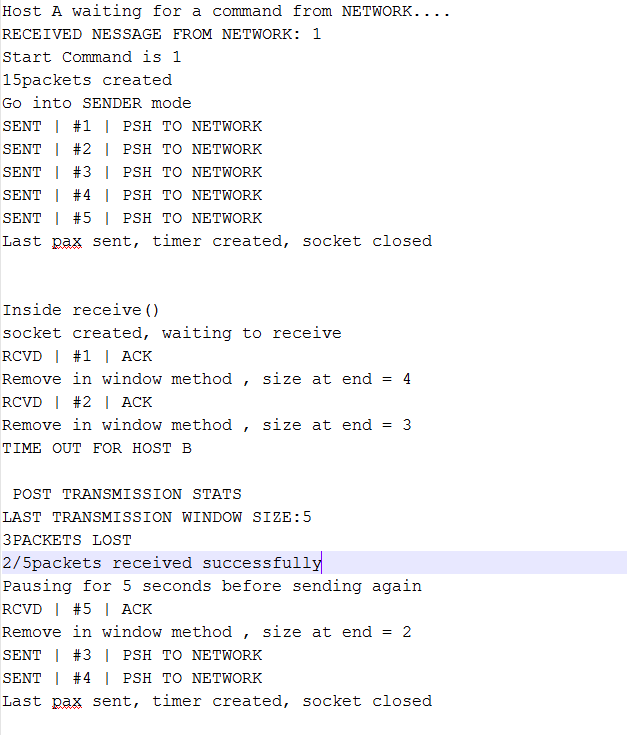


Fig 2.5

