CNT4007C Computer Network Fundamentals, Spring 2015
Programming Assignment 2

Shubhi Jain
1159-7408
shubhijain106@ufl.edu

INTRODUCTION

Running Socket Programs Compile network, server and client programs and then deploy network program code on a machine which is going to act as a server and sender and receiver program, which are going to act as clients. If required, both client and server programs can run on the same machine.

COMPILE AND EXECUTION FORMAT:

Linux platform,coding in JAVA.

Network:

```
java network [port_number]
```

Server:

```
java server [serverURL][port_number]
```

Client:

```
java client [serverURL] [port_number]
```

```
[Java]
sand> javac *.java
rain> javac *.java
storm> javac *.java
sand> java network 21234
rain> java client sand.cise.ufl.edu 21234
storm> java server storm.cise.ufl.edu 21234
```

INSTRUCTIONS:
File has been included in the client programme and not specified explicitly as an argument.
First compile network programme,then server programme and then client programme.
Output is altered and differs for better understanding.

When compiling network layer
Compile RandomJava.java
Compile network.java
Compile networkThread.java

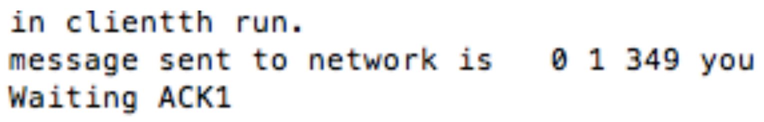
When compiling server layer
Compile server.java
Compile serverThread.java

When compiling client layer
Compile client.java
Compile ChatClientThread.java

CODE STRUCTURE:

Message Format

Packet Format



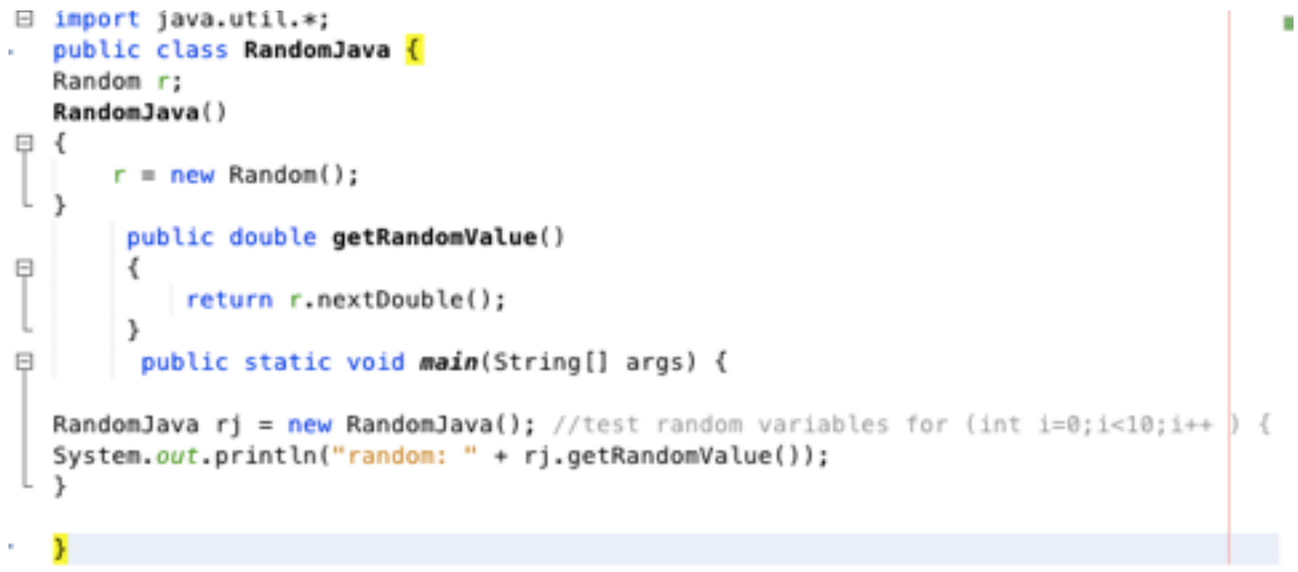
```
in clientth run.
message sent to network is   0 1 349 you
Waiting ACK1
```

Acknowledgement Format

ACK1—- 1 0
ACK0—- 0 0
ACK2—- 2 0

Network Program

```
RandomJava r=new RandomJava();
    double t;
    t = r.getRandomValue();
```



```
import java.util.*;
public class RandomJava {
Random r;
RandomJava()
{
    r = new Random();
}
        public double getRandomValue()
        {
            return r.nextDouble();
        }
         public static void main(String[] args) {

RandomJava rj = new RandomJava(); //test random variables for (int i=0;i<10;i++ ) {
System.out.println("random: " + rj.getRandomValue());
}

}
```

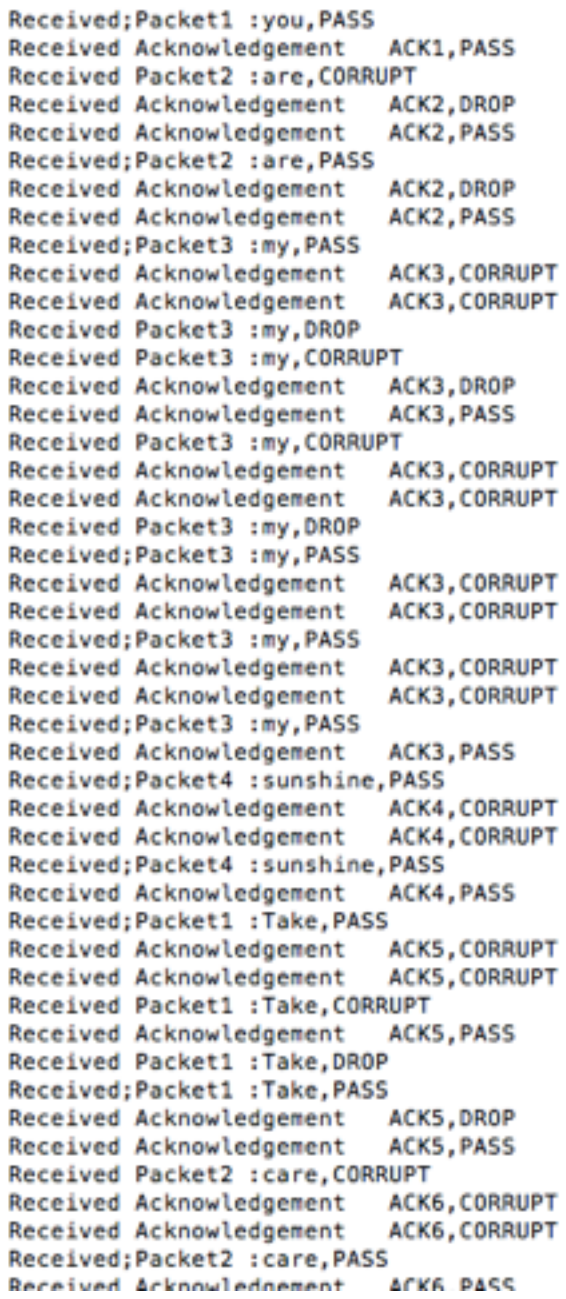
```
if(t<0.5 && input.length()>5) //PASS

else if(t>=0.5 && t<=0.75 && input.length()>5)//CORRUPT

else if(t>0.75 && input.length()>5)//DROP
```

In order to emulate the lossy channel with bit errors, we let the Network choose an operation from the following three options: PASS, CORRUPT or DROP whenever it needs to relay a packet/ACK. In this implementation, the Network picks one of the three operations RANDOMLY with probability 0.5 for PASS, 0.25 for both CORRUPT and DROP.

If the chosen operation is PASS, the packet is forwarded to the receiver without any change.

If the operation is DROP, sent ACK2 to the sender.

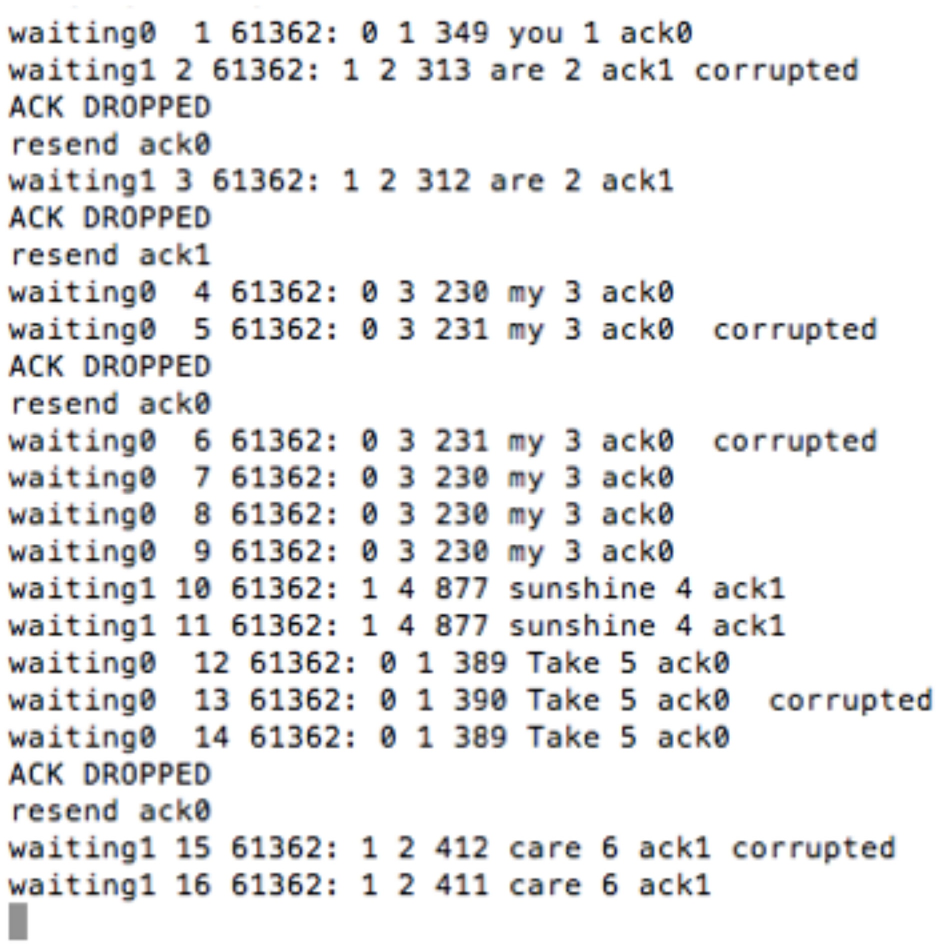
If CORRUPT, added 1 to the checksum field.



```
Received;Packet1 :you,PASS
Received Acknowledgement   ACK1,PASS
Received Packet2 :are,CORRUPT
Received Acknowledgement   ACK2,DROP
Received Acknowledgement   ACK2,PASS
Received;Packet2 :are,PASS
Received Acknowledgement   ACK2,DROP
Received Acknowledgement   ACK2,PASS
Received;Packet3 :my,PASS
Received Acknowledgement   ACK3,CORRUPT
Received Acknowledgement   ACK3,CORRUPT
Received Packet3 :my,DROP
Received Packet3 :my,CORRUPT
Received Acknowledgement   ACK3,DROP
Received Acknowledgement   ACK3,PASS
Received Packet3 :my,CORRUPT
Received Acknowledgement   ACK3,CORRUPT
Received Acknowledgement   ACK3,CORRUPT
Received Packet3 :my,DROP
Received;Packet3 :my,PASS
Received Acknowledgement   ACK3,CORRUPT
Received Acknowledgement   ACK3,CORRUPT
Received;Packet3 :my,PASS
Received Acknowledgement   ACK3,CORRUPT
Received Acknowledgement   ACK3,CORRUPT
Received;Packet3 :my,PASS
Received Acknowledgement   ACK3,PASS
Received;Packet4 :sunshine,PASS
Received Acknowledgement   ACK4,CORRUPT
Received Acknowledgement   ACK4,CORRUPT
Received;Packet4 :sunshine,PASS
Received Acknowledgement   ACK4,PASS
Received;Packet1 :Take,PASS
Received Acknowledgement   ACK5,CORRUPT
Received Acknowledgement   ACK5,CORRUPT
Received Packet1 :Take,CORRUPT
Received Acknowledgement   ACK5,PASS
Received Packet1 :Take,DROP
Received;Packet1 :Take,PASS
Received Acknowledgement   ACK5,DROP
Received Acknowledgement   ACK5,PASS
Received Packet2 :care,CORRUPT
Received Acknowledgement   ACK6,CORRUPT
Received Acknowledgement   ACK6,CORRUPT
Received;Packet2 :care,PASS
Received Acknowledgement   ACK6,PASS
```

When the Network gets a packet or ACK, it will print on the screen the packet type(packet or ACK), ID , content and the operation it picks.

Receiver Program

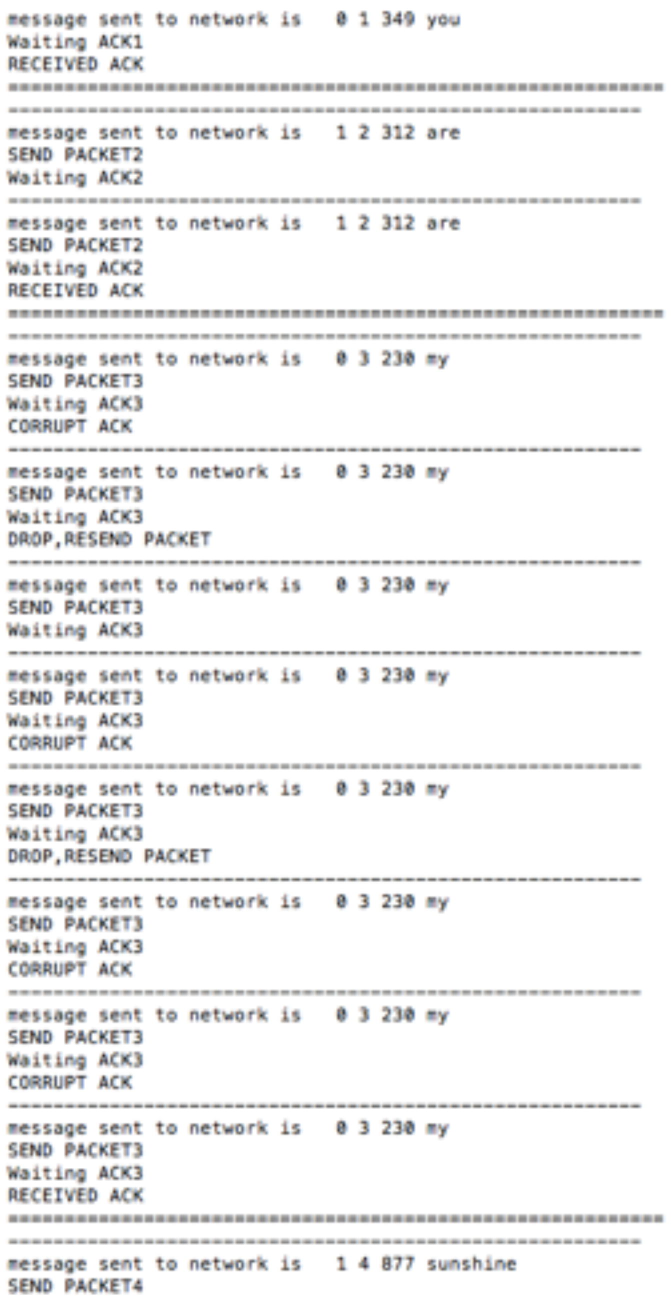

```
waiting0  1 61362: 0 1 349 you 1 ack0
waiting1 2 61362: 1 2 313 are 2 ack1 corrupted
ACK DROPPED
resend ack0
waiting1 3 61362: 1 2 312 are 2 ack1
ACK DROPPED
resend ack1
waiting0  4 61362: 0 3 230 my 3 ack0
waiting0  5 61362: 0 3 231 my 3 ack0  corrupted
ACK DROPPED
resend ack0
waiting0  6 61362: 0 3 231 my 3 ack0  corrupted
waiting0  7 61362: 0 3 230 my 3 ack0
waiting0  8 61362: 0 3 230 my 3 ack0
waiting0  9 61362: 0 3 230 my 3 ack0
waiting1 10 61362: 1 4 877 sunshine 4 ack1
waiting1 11 61362: 1 4 877 sunshine 4 ack1
waiting0  12 61362: 0 1 389 Take 5 ack0
waiting0  13 61362: 0 1 390 Take 5 ack0  corrupted
waiting0  14 61362: 0 1 389 Take 5 ack0
ACK DROPPED
resend ack0
waiting1 15 61362: 1 2 412 care 6 ack1 corrupted
waiting1 16 61362: 1 2 411 care 6 ack1
▌
```

When a packet arrives, the receiver performs the checksum and examine the serial number and ID to see if this packet is wanted and not corrupted.If corrupted sends an ACK expecting the sender to send the same packet again.

Every time the receiver gets a packet, the screen displays: its current state, total number of received packets so far (including corrupted), prints whole packet, and the proper ACK to be transmitted and whether the packet was corrupted or not.

Also if its ACK was dropped by the network layer it receives a message of DROP and resends it ACK to Network.

Sender Program



```
message sent to network is   0 1 349 you
Waiting ACK1
RECEIVED ACK
==========================================================
--------------------------------------------------------
message sent to network is   1 2 312 are
SEND PACKET2
Waiting ACK2
--------------------------------------------------------
message sent to network is   1 2 312 are
SEND PACKET2
Waiting ACK2
RECEIVED ACK
==========================================================
--------------------------------------------------------
message sent to network is   0 3 230 my
SEND PACKET3
Waiting ACK3
CORRUPT ACK
--------------------------------------------------------
message sent to network is   0 3 230 my
SEND PACKET3
Waiting ACK3
DROP,RESEND PACKET
--------------------------------------------------------
message sent to network is   0 3 230 my
SEND PACKET3
Waiting ACK3
--------------------------------------------------------
message sent to network is   0 3 230 my
SEND PACKET3
Waiting ACK3
CORRUPT ACK
--------------------------------------------------------
message sent to network is   0 3 230 my
SEND PACKET3
Waiting ACK3
DROP,RESEND PACKET
--------------------------------------------------------
message sent to network is   0 3 230 my
SEND PACKET3
Waiting ACK3
CORRUPT ACK
--------------------------------------------------------
message sent to network is   0 3 230 my
SEND PACKET3
Waiting ACK3
CORRUPT ACK
--------------------------------------------------------
message sent to network is   0 3 230 my
SEND PACKET3
Waiting ACK3
RECEIVED ACK
==========================================================
--------------------------------------------------------
message sent to network is   1 4 877 sunshine
SEND PACKET4
```

When it receives a DROP message from the Network, it realises that this is the fake timeout and then resends the same packet.

Every time an ACK or DROP is received, the sender displays on screen: message it had sent to Network, packet sent , packet received and proper action.