CS339 Computer networking assignment4

Name: Chao Gao

Student ID: 5142029014

File Structure

- source folder, contain three implementations
- runnable folder, contain comiled java byte codes
- test folder, contain simple test code and test file
- report.pdf, this file

Description

This assignment is implemented on Mac OS X EI Capitan 10.11.4. I use java socket api to implement code.

In this project, I simply implement C/S model. The main function is that client sends the file to the server. When the transmission finishes, the program terminates.

How to use the application

Enter into runnable folder, open two terminals and type the follwing command respectively.(take the UDP as an example)

```
java UDPServer <port>
```

```
java UDPClient <hostname> <port> <filename>
```

The file will simply saved in the server side after the transmission.

RDT implemention

Here I implement *stop_and_wait* method.

Client(send side)

- add sequence number as header
- wait for ACK from receiver. If positive ACK is received(acknowledgementSequenceNum == sequenceNum), then send next packet, else resend current packet.
- resend the packet if unreceived after a specific time

Server(receive side)

- receive the packet, check the sequence number, if it is the same as the previous sequence number, resend the ACK. Else save the packet and send ACK.
- receive last packet, terminates

Result analysis

delay time

The total delay time of transmission is from first bit sending out to the last bit received. Therefore, I record start time in the client side and record end time in the server side, however, since UDP lose packets and the program may stuck, so I simply record end time in the client side and this may cause a little deviation.

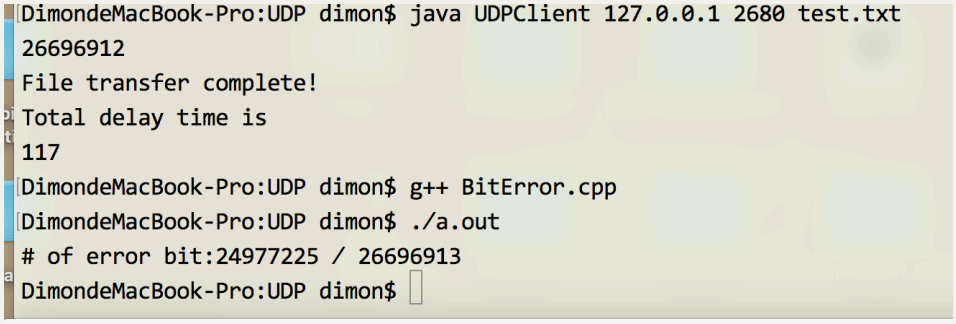
error test

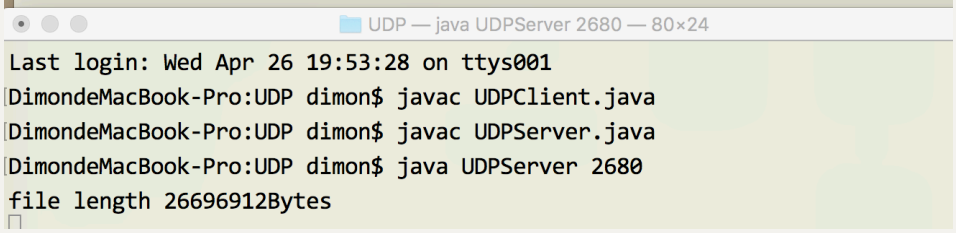
Here I simply write a C++ program to evaluate the reult file and the original file.

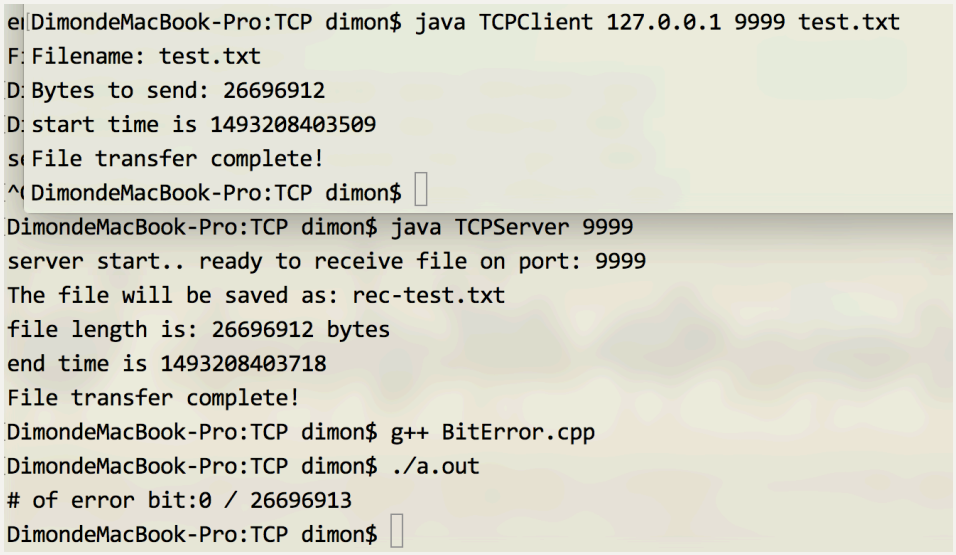
Use the **test.txt**(size 26.7MB)as the test case, I get the following result.

```
DimondeMacBook-Pro:RDT dimon$ javac RDTClient.java
DimondeMacBook-Pro:RDT dimon$ java RDTClient 127.0.0.1 2680 test.txt
[RDTClient] Started sending file
start time is 1493205705176
---------------------------------------
File size: 26696912 Bytes

Number of re-transmissions: 9764
DimondeMacBook-Pro:RDT dimon$
DimondeMacBook-Pro:RDT dimon$ javac RDTServer.java
DimondeMacBook-Pro:RDT dimon$ java RDTServer 2680
end time is 1493205706503
[RDTServer] File received and saved successfully
DimondeMacBook-Pro:RDT dimon$ g++ BitError.cpp
DimondeMacBook-Pro:RDT dimon$ ./a.out
# of error bit:0 / 26696913
```







	delay(ms)	error
TCP	209	0
UDP	117	24977225/26696913
RDT3.0	1327	0

Since the error in UDP is too large, I start to think about my C++ test program. Because the main problem in UDP is packet loss, and all these I count it into errors which makes the error so large. So I test the packet loss, and I get **4271(loss)/6571**.

It can be seen that UDP is fast but not reliable, and RDT 3.0 are reliable but needs much more delay time to ensure the reliability.

Problems and experience

- Understanding the RDT 3.0 is very important, after getting familiar with the whole mechnism of the protocol, I implement it easily.

- At first I find that TCP is not reliable because there are errors with my test code and test case, after digging into the code finally I find out that there are mistakes in the file-write part in the receiver side. OK… I am not reliable, TCP should not be the one to be blame.
- This assignment is quite challenging and I really gained a lot about these protocols, and I hope I can learn more in the coming lessons.