RDT3.0 REPORT

王誉锡 5131309040 15058617922@163.com

1. How to run the application

1.1 RTD3.0 PART

1. cd rdt3.0
2. execute ./winsock_server.exe
3. cd rdt3.0/rec
4. execute ./winsock_client.exe
5. $ ./winsock_client.exe
   Connect to where (IP_ADDRESS):
   127.0.0.1
   PORT:
   10234
   Input the file name:
   1.pdf
6. input the server IP, port and the file name you want
7. server will send you the file you requested like below
8. $ ./winsock_server.exe
   A Connection: 127.0.0.1
   Client request 1.pdf
   1.pdf sent
   This used 0.085 Second
9. and then you can keep request other file

1.2 TCP PART

It's similar with the rdt3.0 part.

1. cd clientserver/server, execute ./winsock_server.exe
2. cd clientserver/client, execute ./winsock_client.exe
3. input IP, port and filename
4. Connect to where (IP_ADDRESS):
   127.0.0.1
   PORT:
   10234
   Connected!
   Input the file name:
   1.pdf
5. then server will response you request
6. after you get the file you can do next query

2. High-level structure of RDT code

2.1 SENDER SIDE(SERVER)

Main function part

```
main (){
    init some socket stuffs()

    while(true) {
        1. get the requested file name()
        2. start a receive thread()
        2. start a finite state machine thread()
        while(file sending is not finished);
    }
}
```

Receive thread part

```
while (not turn off thread signal && recv some thing) {
    set receive state = true
    while(receive state); // wait until the msg has been processed
}
```

Finite state machine

```
while(true) {
    if state = 0 {
        send(0, pkt)
        start timer
        state = 1
        continue
    }
    if state = 1 {
        if rcvpkt is not right ACK or timerout continue
        else extract msg write to local and clear receive state
        state = 2
        continue
    }
    if state = 2 {
        send(1, pkt)
        start timer
        state = 3
        continue
    }
    if state = 3 {
        if rcvpkt is not right ACK or timerout continue
        else extract msg write to local and clear receive state
        state = 0
        continue
    }
}
```

also other functions omitted here include add checksum, check the corruption.

2.2 RECEIVE SIDE(CLIENT)

Main function

```
main(){
    1. init socket stuff
    2. connect to server
    while(true) {
        1. get the file name
        2. send the file request to server
        // start fsm
        while(receive some thing) {
            check msg
            if msg is not legal then send a illegal ACK continue
            else
                send a legal ACK
                change the state = 1-state
            if msg == eof break;
        }
    }
}
```

3. Test

3.1 MULTI-QUERY TEST

First test is on the normal multi-query test, it works good.

```
Connect to where (IP_ADDRESS):
127.0.0.1
PORT:
10234
Input the file name:
1.pdf
fileName 1.pdf
Input the file name:
2.pdf
fileName 2.pdf
Input the file name:
3.pdf
fileName 3.pdf
```

server side

```
A Connection: 127.0.0.1
Client request 1.pdf
timeout
timeout
1.pdf sent
This used 0.085 Second
```

```
A Connection: 127.0.0.1
Client request 2.pdf
timeout
2.pdf sent
This used 0.116 Second

A Connection: 127.0.0.1
Client request 3.pdf
timeout
timeout
timeout
3.pdf sent
This used 2.783 Second
```

3.2 TIME OUT TEST

Because I used char[256] as one message, thus it's very fast that only take about 0,001s. Thus I set the timeout as 2ms. One thing I can test is to sleep after the timer start like this:

```
      timer_start = clock();
      state = 3;
      Sleep(10);
      continue;
```

and then compile it again

```
g++ winsock_server.cpp -o winsock_server -lws2_32
```

result present below

```
...
timeout * 1000...
timeout
timeout
timeout
timeout
timeout
timeout
1.pdf sent
This used 7.804 Second
```

It can still receive the file correctly.

4. Problems and experiences

1. It very hard to debug the multi-thread project especially the finite state machine part
2. Using gdb tool to debug is very convenienet
3. Implement the FSM after step by step, start from small number of state is very useful trick