[2015 Advanced Computer Networks Homework 5]

Motivation

Implement an Ftp-like network system.

Specification

The system is a multi-client server structure, based on UDP protocol stop-and-wait process, file transmission program. Moreover, checksum method and retransmit mechanism for error control are necessary.

Offering files

myftp.c, myftp.h, myftpClient.c, myftpServer.c , testfile

Part1:

1. You must run your program on mininet.
2. To start server, you must have super user authority.
   It's possible to receive duplicate broadcast messages, because of multiple NICs (network interface controller also called network interface card) on the server. To make sure the correctness of processing, specify NIC is needed.
3. Execution:
   Server side:

   ```
   root@yin:~/tcpip/HW5# ./myftpServer 44301 testfile
   network interface = h1-eth0
   network port = 44301
   MyFtp Server start !
   share file : testfile
   wait client!
   ```

   Client side:

   ```
   root@yin:~/tcpip/HW5# ./myftpClient 44301 testfile
   ```

   ※ Port number is XX + student ID last 3 numbers, XX for Bachelor Degree is 22, Master Degree 1st year is 33, Master Degree 2nd year is 44, and Doctor Degree is 55.

   ※ "testfile " is the name of the file to transmit.
4. Whether server side or client side, your program also needs **Time out** mechanism for error control (Note: Limit is 20sec).
   When server side transmits:

   ```
   file transmisssion start
   wait client time out
   ```

When client side connects:

```
root@yin:~/tcpip/HW5# ./myftpClient 44301 testfile
  [Timeout]
        No Server Answer!!
```

When client side downloads:

```
  [file transmisssion - start]
        download to file : client_testfile
        wait server time out
```

You have to resend packets until either client connects server successfully or timeout. For more details, please refer to Part2.

5. Server side's procedure:
   a. Get and specify device name:
      getIFname(), initServAddr()

```
root@yin:~/tcpip/HW5# ./myftpServer 44301 testfile
network interface = h1-eth0
network port = 44301
MyFtp Server start !
share file : testfile
wait client!
```

   b. Start listen to port:
      listenClient()
      When receive broadcast messages (**struct bootServerInfo**) from client side, find the file of the request. Send server address and transmit port number back to client side if the file exists.

```
share file : testfile
wait client!
  [Request]
        file : testfile
        from : 10.0.0.2
  [Reply]
        file        : testfile
        servAddr    : 10.0.0.2
        connectPort : 44012
```

      ※ Transmit port number is 44012
   c. After build connection between server and client successfully, follow (description1) steps to start transmitting file:startMyftpServer()

※ Maximum data size in DATA packet is 512 bytes, larger is not allowed.

※ After sending a DATA packet (block = 1), must receive a ACK packet (block = 1) to process next data transmit step.

※ Checksum mechanism (description2) for packet authorization is necessary, send ERROR packet and ask for data retransmit if checksum is not correct.

※ Server must be able to serve other clients when the other client is already in transmit data processing.

6. Client side's procedure:

a. Setup client:
   initCliAddr()
b. Send broadcast messages (**struct bootServerInfo**)to search server that apply the request:
   findServerAddr()

```
  [Receive Reply]
        Get MyftpServer servAddr : 10.0.0.1
        Myftp connectPort        : 44012
```

c. Follow the steps (description 1), to send request and receive the specify file:
   startMyftpClient()

※ Checksum mechanism (description2) for packet authorization is necessary, send ERROR packet and ask for data retransmit if checksum is not correct.

※ Download file should be named as client_filename,
   ex: testfile →client_testfile.testfile2→client_testfile2

7. Packet format：

Packet	Opcode	Checksum
FRQ	FRQ(2)	Checksum(2) filename
DATA	DATA(2)	Checksum(2) Block() Data(512)
ACK	ACK(2)	Checksum(2) Block()
ERROR	ERROR(2)	Checksum(2) Block()

8. Final process screen should be:
   Server :

```
root@yin:~/tcpip/HW5# ./myftpServer 44301 testfile
network interface = h1-eth0
network port = 44301
MyFtp Server start !
share file : testfile
wait client!
  [Request]
        file : testfile
        from : 10.0.0.2
  [Reply]
        file        : testfile
        servAddr    : 10.0.0.2
        connectPort : 44012
wait client!
  [file transmisssion - start]
        send file : <testfile> to  10.0.0.2
  [file transmisssion - finish]
        33085699 bytes sent
wait client!
```

Client :

```
root@yin:~/tcpip/HW5# ./myftpClient 44301 testfile
  [Receive Reply]
        Get MyftpServer servAddr : 10.0.0.1
        Myftp connectPort        : 44012
  [file transmisssion - start]
        download to file : client_testfile
        get file : <testfile> from 10.0.0.1
  [file transmisssion - finish]
        33085699 bytes received
```

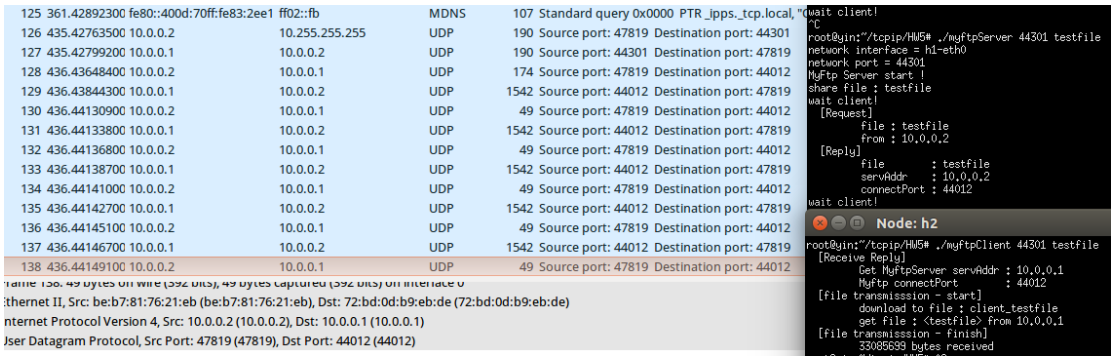
9. Please **print screen** your server and client program, like No.8. Make TA sure that you run your program on mininet.
10. Make sure your Ftp-like program could transmit file size larger than **32M**.
11. You can use Linux command “diff” to check whether client_testfile and testfile are same.

Part2：

Follow the No.4 in Part 1. When data is being downloaded, control the filter of mininet switch UDP port to result in packet loss scenario. You have to control the mininet switch to drop UDP packets and then close the filter before timeout. Your program must enable packet to be resent.

※ Please **print screen** your control switch commands and programs.

In generally:



Part3：

Execute your client and connect with TA on homework server.

※ Please confirm your server & client works at first, and then connect with TA.

※ Note that broadcast is different between mininet and physical machine.

※ Please use Computer Science and Engineering IP to test.ex:140.117.(170、171、172、174、176.....) .xxx

※ Port by default is **44301**. Transmit port number is 44301 + Random no. (1 ~ 999).

※ File name is **testfile2**.

Description 1:

A file need many DATA packet transitions.

Client		Server	
Sendto	→	Recvfrom	//Client broadcast to Server
Recvfrom	←	Sendto	// Server replies

Sendto(FRQ)	→	Recvfrom	//Send request
Recvfrom	←	Sendto(DATA,block=1)	//Start transmission
Sendto(ACK, block=1)	→	Recvfrom	
Recvfrom	←	Sendto(DATA, block=2)	
Sendto(ACK, block=2)	→	Recvfrom	
……			
Recvfrom	←	Sendto(DATA, block=n)	
Sendto(ACK, block=0)	→	Recvfrom	//Finish transmission

Description 2:

Before sending a packet out, fill the checksum field by 0 first, then fill in the correct number by checksum method.
When receiving a packet, check packet with its total size by checksum method. Return 0, for correct, otherwise it is wrong.
About checksum method, it seems to say take 2 bytes (short int) at a time and plus all together to get summation, then turn it into 1' Complement to get checksum number. Please search ICMP for more detail information.

Description 3:

You may write the program as you want, but follow the step in (description 1) or part2 and part3 will failed work otherwise.

Upload:

1. Please compress your homework into **zip** or **tar** archive.
2. Naming rules: "**StudentID_TCPIP_HW5.zip**".
   For example: M033040001_TCPIP_HW5.zip
3. Upload your homework to **National Sun Yat-sen Cyber University**.
4. **Deadline: 2015/11/18(Tue.) 23:59.**

Rules:

1. Please use **C** language in this homework and run your program on **Ubuntu 14.04**.
2. Please provide Makefile to compile your homework; otherwise, you will get **ZERO**.
3. **Do not copy homework of others (classmates, senior etc).** If it happened, you will get **ZERO** whether you are either the owner of the homework or the copycat.
4. You have to deeply understand what your program do because TA will ask you something about your program during the demo.
5. If you have any question, please send email to net_ta@net.nsysu.edu.tw or come to EC5018, but TA does not help to debug.
6. If you do not submit your assignment on time, you will not hand in the delayed homework and get **ZERO** as well. If you have trouble, please advise it in advance by email. Moreover, time and place for demo will be announced later.