###### *CSE 473 – Introduction to Computer Networks*

Lab 1 Report – 70 Points

##### *Haiyu Wang*

***Part A (15 points).*** Paste a copy of the source code for MapServer.java here. Use the pre-formatted paragraph style for the code sections (Courier, 10) and make sure that no lines wrap around. Don’t forget to include appropriate comments in your code.

**import** java.io.\*;

**import** java.net.\*;

**import** java.util.\*;

/\*\*

\* author: Haiyu Wang

\* usage: java MapServer

\*

\* have a simple storage servive with three operations: get, put,remove

\* receive a packet from client and decide the operation

\* send back certain packet back to client

\*\*/

**public** **class** MapServer {

**public** **static** **void** main(String args[]) **throws** Exception {

// create Hashmap as storage service

HashMap<String,String> Map = **new** HashMap<String,String>();

//get address and port

InetAddress myIp = **null**;

**int** port = 30123;

**if** (args.length > 0) myIp = InetAddress.*getByName*(args[0]);

**if** (args.length > 1) port = Integer.*parseInt*(args[1]);

//create socket

DatagramSocket sock = **new** DatagramSocket(port,myIp);

// create incoming packets

**byte**[] buf = **new** **byte**[1000];

DatagramPacket inpkt = **new** DatagramPacket(buf, buf.length);

**while** (**true**) {

// wait for incoming packet

sock.receive(inpkt);

// get strings from buf and split them

String str = **new** String(buf,0,inpkt.getLength(),

"US-ASCII");

String[] strSplit = str.split(":");

//get client address from incoming packet

SocketAddress clientAdr = inpkt.getSocketAddress();

String value = **null**;

// check the first string and decide the operation

**switch** (strSplit[0]){

// operation get

**case** "get": {

// check the format

**if** (strSplit.length>2){

value = "Error:unrecognizable input: "

+"put a copy of the input's"

+" packet payload here";

}

**else**{

value = Map.get(strSplit[1]);

**if** (value == **null**)

value = "no match";

**else**

value = "Ok:"+value;

}

**break**;

}

// operation put

**case** "put": {

// check the format

**if** (strSplit.length>3){

value = "Error:unrecognizable input: "

+"put a copy of the input's"

+" packet payload here";

}

**else**{

**if** (Map.containsKey(strSplit[1]))

value = "updated:"+strSplit[1];

**else**

value = "Ok";

Map.put(strSplit[1],strSplit[2]);

}

**break**;

}

//operation remove

**case** "remove": {

// check the format

**if** (strSplit.length>2){

value = "Error:unrecognizable input: "

+"put a copy of the input's"

+" packet payload here";

}

**else**{

**if** (Map.containsKey(strSplit[1])){

Map.remove(strSplit[1]);

value = "Ok";

}

**else**

value = "no match";

}

**break**;

}

// Error info

**default** :{

value = "Error:unrecognizable input: "

+"put a copy of the input's"

+" packet payload here";

**break**;

}

}

// trans string to byte

**byte**[] buffer = value.getBytes();

// create outcoming packet

DatagramPacket outpkt = **new** DatagramPacket(buffer,

buffer.length,

clientAdr);

sock.send(outpkt);

// and send it back

}

}

}

***Part B (15 points).*** Paste a copy of the source code for MapClient.java here.

**import** java.io.\*;

**import** java.net.\*;

/\*\*

\* author: Haiyu Wang

\* usage: java MapClient serverName port string

\* test: TestScript

\*

\* Send a packet to the named server:port containing the given string.

\* Wait for reply packet and print its contents.

\*\*/

**public** **class** MapClient {

**public** **static** **void** main(String args[]) **throws** Exception {

// get server address and port

InetAddress serverAdr = InetAddress.*getByName*(args[0]);

**int** port = Integer.*parseInt*(args[1]);

// create socket

DatagramSocket sock = **new** DatagramSocket();

// encode byte array with US-ASCII

// create outpkt and send

**byte**[] outbuf = args[2].getBytes("US-ASCII");

DatagramPacket outpkt = **new** DatagramPacket(outbuf,

outbuf.length,

serverAdr,

port);

sock.send(outpkt); // send packet to server

// create buffer and packet for reply

**byte**[] inbuf = **new** **byte**[1000];

DatagramPacket inpkt = **new** DatagramPacket(inbuf,

inbuf.length);

sock.receive(inpkt); // wait for reply

// get content of buf and trans to String and print

String reply = **new** String(inbuf,

0,

inpkt.getLength(),

"US-ASCII");

System.***out***.println(reply);

sock.close();

}

}

***Part C (10 points).*** Paste a copy of the output from testScript when both client and server are run on the same computer.

WHYdeMacBook-Air:javatest a1100$ /Users/a1100/Desktop/javatest/testScript

java MapClient WHYdeMacBook-Air.local 30123 put:foo:bar

Ok

java MapClient WHYdeMacBook-Air.local 30123 put:who:hah

Ok

java MapClient WHYdeMacBook-Air.local 30123 put:goodbye:world

updated:goodbye

java MapClient WHYdeMacBook-Air.local 30123 get:foo

Ok:bar

java MapClient WHYdeMacBook-Air.local 30123 get:who

Ok:hah

java MapClient WHYdeMacBook-Air.local 30123 remove:who

Ok

java MapClient WHYdeMacBook-Air.local 30123 get:who

no match

java MapClient WHYdeMacBook-Air.local 30123 get:goodbye

Ok:world

java MapClient WHYdeMacBook-Air.local 30123 got:goodbye

Error:unrecognizable input: put a copy of the input's packet payload here

java MapClient WHYdeMacBook-Air.local 30123 pat:goodbye:world

Error:unrecognizable input: put a copy of the input's packet payload here

java MapClient WHYdeMacBook-Air.local 30123 get:bar

no match

java MapClient WHYdeMacBook-Air.local 30123 put:foo:"toast is tasty"

updated:foo

java MapClient WHYdeMacBook-Air.local 30123 get:foo

Ok:"toast is tasty"

***Part D (10 points).*** Paste a copy of the output from testScript when the client and server are run on different computers.

ACCOUNTS+haiyu.wang@URB218-05 ~/desktop/javatest

$ /cygdrive/h/Desktop/javatest/testScript.cmd

H:\desktop\javatest>echo java MapClient URB218-06 30123 "put:foo:bar"

java MapClient URB218-06 30123 "put:foo:bar"

H:\desktop\javatest>java MapClient URB218-06 30123 "put:foo:bar"

Ok

H:\desktop\javatest>echo java MapClient URB218-06 30123 "put:who:hah"

java MapClient URB218-06 30123 "put:who:hah"

H:\desktop\javatest>java MapClient URB218-06 30123 "put:who:hah"

Ok

H:\desktop\javatest>echo java MapClient URB218-06 30123 "put:goodbye:world"

java MapClient URB218-06 30123 "put:goodbye:world"

H:\desktop\javatest>java MapClient URB218-06 30123 "put:goodbye:world"

Ok

H:\desktop\javatest>echo java MapClient URB218-06 30123 "get:foo"

java MapClient URB218-06 30123 "get:foo"

H:\desktop\javatest>java MapClient URB218-06 30123 "get:foo"

Ok:bar

H:\desktop\javatest>echo java MapClient URB218-06 30123 "get:who"

java MapClient URB218-06 30123 "get:who"

H:\desktop\javatest>java MapClient URB218-06 30123 "get:who"

Ok:hah

H:\desktop\javatest>echo java MapClient URB218-06 30123 "remove:who"

java MapClient URB218-06 30123 "remove:who"

H:\desktop\javatest>java MapClient URB218-06 30123 "remove:who"

Ok

H:\desktop\javatest>echo java MapClient URB218-06 30123 "get:who"

java MapClient URB218-06 30123 "get:who"

H:\desktop\javatest>java MapClient URB218-06 30123 "get:who"

no match

H:\desktop\javatest>echo java MapClient URB218-06 30123 "get:goodbye"

java MapClient URB218-06 30123 "get:goodbye"

H:\desktop\javatest>java MapClient URB218-06 30123 "get:goodbye"

Ok:world

H:\desktop\javatest>echo java MapClient URB218-06 30123 "got:goodbye"

java MapClient URB218-06 30123 "got:goodbye"

H:\desktop\javatest>java MapClient URB218-06 30123 "got:goodbye"

Error:unrecognizable input: put a copy of the input's packet payload here

H:\desktop\javatest>echo java MapClient URB218-06 30123 "pat:goodbye:world"

java MapClient URB218-06 30123 "pat:goodbye:world"

H:\desktop\javatest>java MapClient URB218-06 30123 "pat:goodbye:world"

Error:unrecognizable input: put a copy of the input's packet payload here

H:\desktop\javatest>echo java MapClient URB218-06 30123 "get:bar"

java MapClient URB218-06 30123 "get:bar"

H:\desktop\javatest>java MapClient URB218-06 30123 "get:bar"

no match

H:\desktop\javatest>echo java MapClient URB218-06 30123 "put:foo:\"toast is tasty\""

java MapClient URB218-06 30123 "put:foo:\"toast is tasty\""

H:\desktop\javatest>java MapClient URB218-06 30123 "put:foo:\"toast is tasty\""

updated:foo

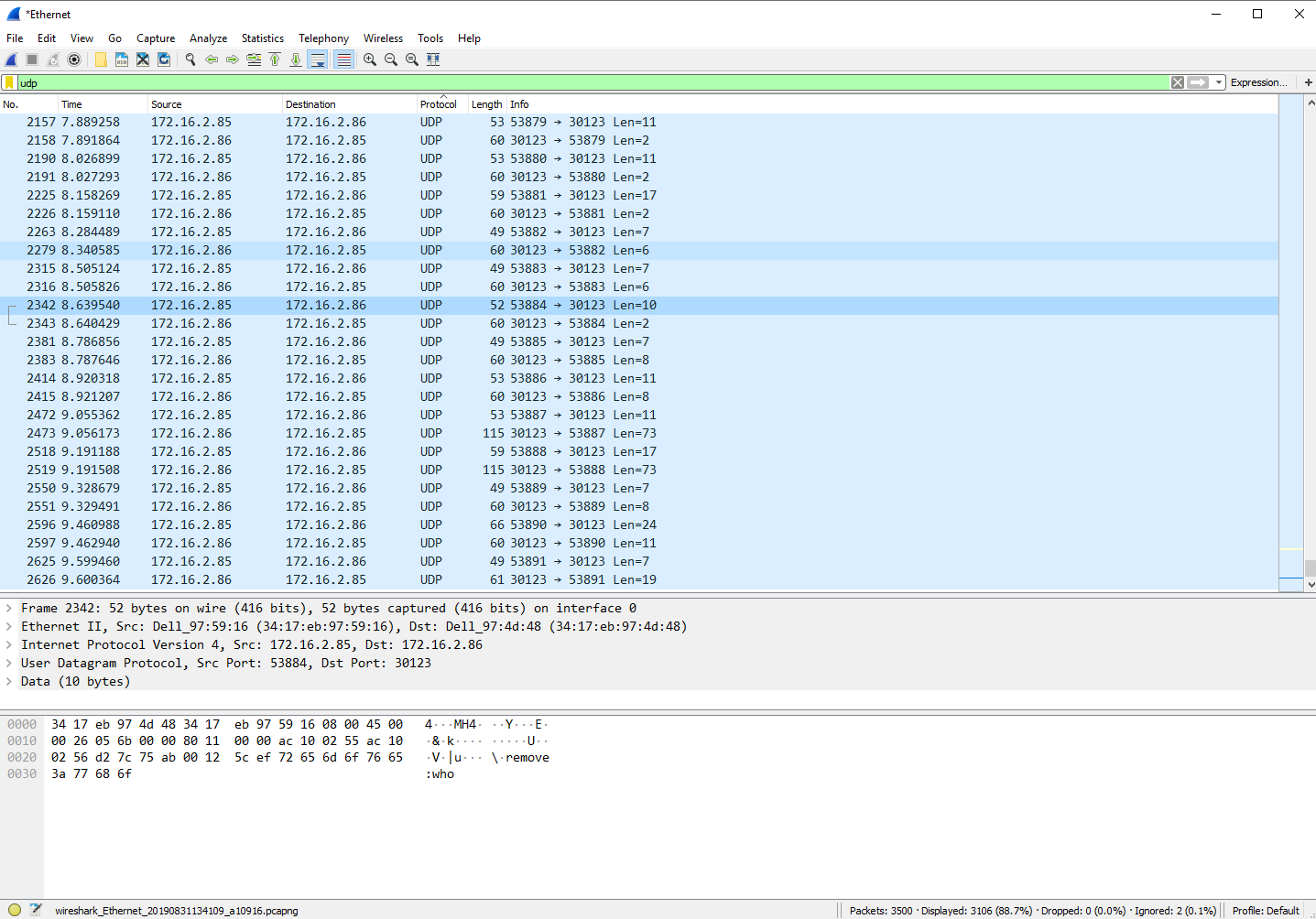
H:\desktop\javatest>echo java MapClient URB218-06 30123 "get:foo"

java MapClient URB218-06 30123 "get:foo"

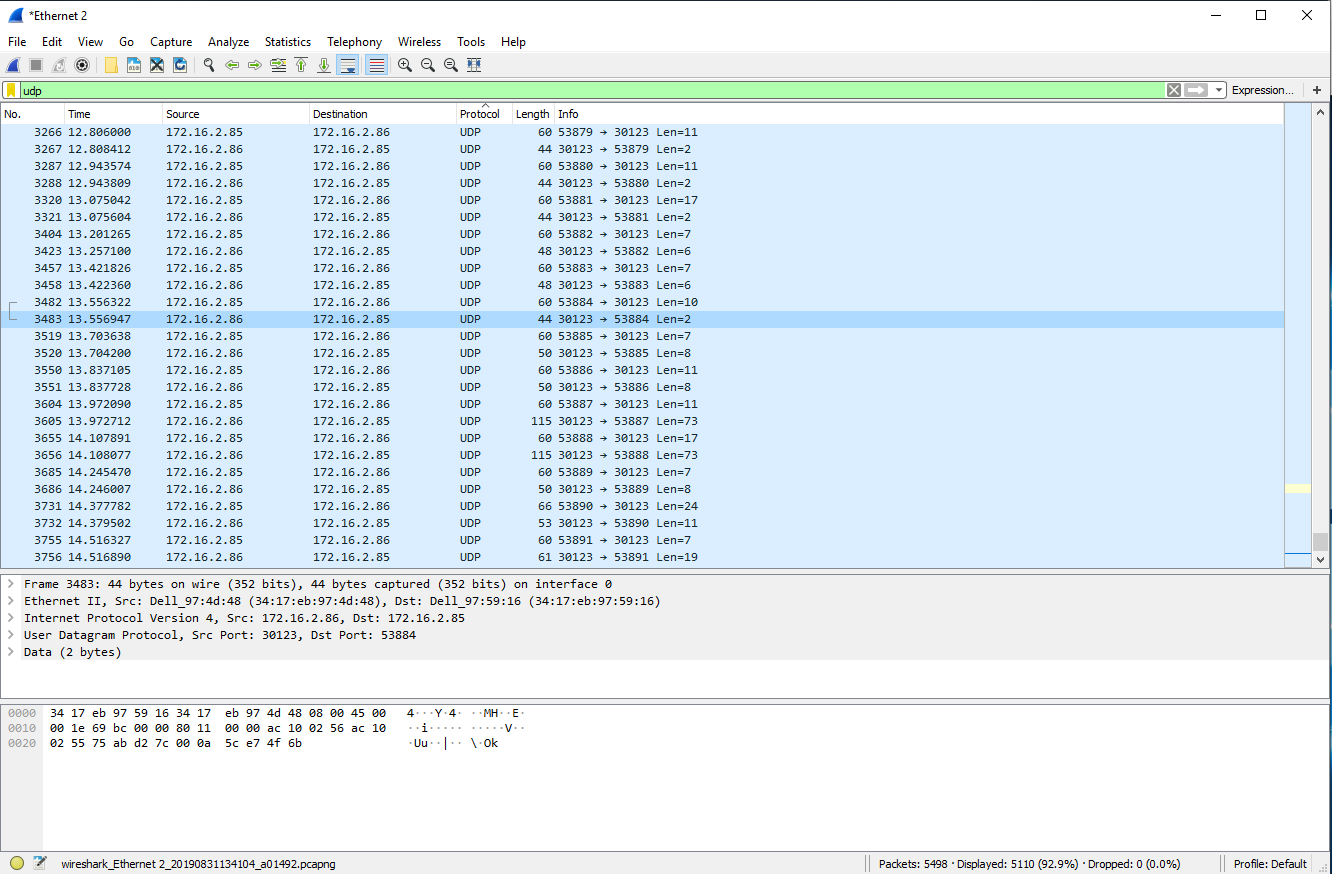
H:\desktop\javatest>java MapClient URB218-06 30123 "get:foo"

Ok:"toast is tasty"

***Part E (10 points***). Paste a screenshot of the *Wireshark* window at the client computer below, showing the packets transferred when you run *testScript*. Make sure that the top portion of the window shows all packets sent and received. Also select packet number 11, and in the middle portion of the window expand the sections for the User Datagram Protocol and for the Data part of the packet. Make sure that all text is clearly legible (you may need to adjust the size of the *Wireshark* window when you do the capture, to ensure that everything is legible in the report).

**

Paste a screenshot of the *Wireshark* window at the server computer below, showing the packets transferred when you run *testScript*. In this case select packet number 12, and again, make sure that all relevant data is visible.



***Part F (10 points).*** Answer the following questions using the Wireshark output.

1. What is the IP address of the host on which the server runs? What is the IP address of the host on which the client runs? What are the Ethernet addresses of the two hosts?

*IP address:*

*Server: 172.16.2.86, client: 172.16.2.85*

*Ethernet address:*

*Server: Dell\_97:4d:48 (34:17:eb:97:4d:48), Client: Dell\_97:59:16 (34:17:eb:97:59:16)*

1. What port number does the client use in your session when packet #11 is sent? Is this same port number used when the other packets are sent? Do you understand why?

*Packet #11 send from port number: 53884.*

*It ‘s not the same port number in other session. Because in UDP the client can choose different ports to send packet in different session, the port numbers will change when sending different packets*

1. The bottom section of the *Wireshark* output shows the contents of the packet as a series of 8 bit hexadecimal values. Find the 4 hex digits that correspond to the client’s port number (hint, click on the port number in the middle section of the window) for packet number 11. What are these hex digits. Which hex digit is the most significant? Which is the least significant? Verify that the hex value represented by these 4 hex digits matches the number you observed.

*The 4 hex digits are d2 7c 75 ab. After converting to decimal numbers, d2 7c (hex) is 53884, and 75 ab (hex) is 30123. 75 ab is more significant since it is the Server’s port number, while d2 7c is less important*

1. How many bytes are shown in the window for packet number 11? How many of these bytes are associated with the actual *remove* command? How many are associated with the UDP protocol? How many are associated with the IP protocol? What about the rest?

*The total number of the bytes of packet #11 is 52.*

*The remove command has 10 bytes.*

*The UDP protocol has 8 bytes.*

*The IP protocol has 20 bytes.*

*The rest are the bytes of Ethernet address.*

1. At what time (according to *Wireshark*) did the server receive packet 11? At what time did it send the reply? What is the difference between these two times? When did the client send packet 11 and when did it receive the reply? What is the difference between these two times? What does this tell you about the time taken to send the two packets across the network?

*Server:*

*Received packet #11 at 13.556322, and sent reply at 13.556947*

*Difference: the Server received the packet and after 0.000625s, it sent the reply. This time included process of decoding packet, deciding operation, operating, repacking the packet and so on.*

*Client:*

*Sent packet #11 at 8.639540, and received reply at 8.640429.*

*Difference: the time between these two process is 0.000889, which contains the time of waiting for the Server’s reply and the times of getting packet from wires*

*It tells me that the process of packet delivery is like the following:*

*Server:*

1. *The Server receives a packet and deals with it;*
2. *After a certain time (this is the time showing above), it will send reply to client.*

*Client:*

1. *The Client send a packet to the Server and wait;*
2. *It will wait for the reply of the Server and get the reply from nets.*