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

Lab 1 Report

##### *Your name here 9/10/2013*

***Part A (15 points).*** Paste a copy of the source code for MapServer.java here. Use the “code” paragraph style and make sure that none of the lines wraps around. Don’t forget to include appropriate comments in your code.

paste your code here

import java.io.\*;

import java.net.\*;

import java.util.HashMap;

/\*\*

\* Name: Bingkun Guo

\* Date: 07/Sep/2013

\* Explanation and Protocal Document: This class represents a MapServer,

\* it holds a HashMap which stores (key,value) pairs.

\* When the server receivs a packet message from a client,

\* it will decompose the packet message by setting a ":"

\* delimiter, then the server will behave according

\* to the opeartion the client requests.

\* Here are the two operations:

\* get(k): returns the value part of the pair whose key is k

\* put(k,v): adds the pair (k,v) to the set, possibly replacing some other pair (k,x).

\* put(k): delete the key-value pair where key is k

\*

\* Example of messages sent from the client:

\* (1). get:this is the key string

\* After receiving this message, the server should return the value whose

\* key is "this is the key string".

\*

\* (2). put:another key string:and the corresponding value

\* After receiving this message, the server should add key value pair

\* ("another key string", "and the corresponding value")

\*

\* Example of messages sent from the server:

\* (1). ok:this is the value string

\* If the key-value pair exists, return the value for that key,

\* in this case, the value is "this is the value string"

\*

\* (2). no match

\* If the key-value pair doesn't exist, return "no match"

\*

\* (3) error: unrecognizable input: put a copy of the input packet’s payload here

\* If the request has a wrong format which the server doesn't recognize, then

\* send a erro message.

\*

\*\*/

public class MapServer {

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

HashMap<String,String> pairs = new HashMap<String,String>();

int port = 30123;

// change the port number if the user provides

if (args.length > 0) port = Integer.parseInt(args[0]);

DatagramSocket sock = new DatagramSocket(port);

// create two packets sharing a common buffer

byte[] buf = new byte[1000];

DatagramPacket pkt = new DatagramPacket(buf, buf.length);

while (true) {

// reset the packet data to make sure it has 1000 length

buf = new byte[1000];

pkt.setData(buf);

// wait for incoming packet

sock.receive(pkt);

// retrieve the packet data

String message = new String(pkt.getData(), 0, pkt.getLength(), "US-ASCII");

// split the message by setting ":" delimiter

String[] splitedMsg = message.split(":");

String reply = "";

String operation = splitedMsg[0];

// assign key, value after we're sure that the array has enough length

String key;

String value;

if (operation.equals("get") && splitedMsg.length == 2) {

key = splitedMsg[1];

// if the operation is get(),

// return the value of that key

if (pairs.get(key) != null) {

reply = "ok:" + pairs.get(key);

} else {

reply ="no match";

}

} else if (operation.equals("put")) {

// if the operation is put(),

// add a key pair or delete a key pair

if (splitedMsg.length == 2) {

key = splitedMsg[1];

if (pairs.keySet().contains(key)) {

pairs.remove(key);

}

reply = "ok";

} else if (splitedMsg.length == 3) {

key = splitedMsg[1];

value = splitedMsg[2];

pairs.put(key, value);

reply = "ok";

} else {

reply = "error:unrecognizable input:" + message;

}

} else {

reply = "error:unrecognizable input:" + message;

}

pkt.setData(reply.getBytes("US-ASCII"));

// and send it back

sock.send(pkt);

}

}

}

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

paste your code here

import java.io.\*;

import java.net.\*;

/\*\*

\* Name: Bingkun Guo

\* Date: 07/Sep/2013

\* Explanation: This class represents a client.

\* It will first collect the command line arguments

\* given by the user and format the payload with

\* these command line arguments using ":" delimiter.

\* Then send the packet to the corresponding server.

\* Here are the two operations:

\* get(k): returns the value part of the pair whose key is k

\* put(k,v): adds the pair (k,v) to the set, possibly replacing some other pair (k,x).

\* put(k): delete the key-value pair where key is k

\*

\* Example of messages sent from the client:

\* (1). get:this is the key string

\* After receiving this message, the server should return the value whose

\* key is "this is the key string".

\*

\* (2). put:another key string:and the corresponding value

\* After receiving this message, the server should add key value pair

\* ("another key string", "and the corresponding value")

\*

\* Example of messages sent from the server:

\* (1). ok:this is the value string

\* If the key-value pair exists, return the value for that key,

\* in this case, the value is "this is the value string"

\*

\* (2). no match

\* If the key-value pair doesn't exist, return "no match"

\*

\* (3) error: unrecognizable input: put a copy of the input packet’s payload here

\* If the request has a wrong format which the server doesn't recognize, then

\* send a erro message.

\*

\*\*/

public class MapClient {

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

// get server address using first command-line argument

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

// get server port number

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

// get operation

String operation = args[2];

// open datagram socket

DatagramSocket sock = new DatagramSocket();

// build packet addressed to server containing second argument,

// encoded using US-ASCII Charset, then send it

String message = operation + ":";

for (int i = 3; i < args.length; i++) {

if (i != args.length - 1) {

message += args[i] + ":";

} else {

message += args[i];

}

}

byte[] outBuf = message.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, then receive it

byte[] inBuf = new byte[1000];

DatagramPacket inPkt = new DatagramPacket(inBuf,inBuf.length);

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

// print buffer contents and close socket

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.

paste your output here

D:\workspace\CSE473\_FL13\_guobingkun\lab1>java MapClient localhost 6789 put foo bar

ok

D:\workspace\CSE473\_FL13\_guobingkun\lab1>java MapClient localhost 6789 put who hah

ok

D:\workspace\CSE473\_FL13\_guobingkun\lab1>java MapClient localhost 6789 put goodbye world

ok

D:\workspace\CSE473\_FL13\_guobingkun\lab1>java MapClient localhost 6789 get foo

ok:bar

D:\workspace\CSE473\_FL13\_guobingkun\lab1>java MapClient localhost 6789 get who

ok:hah

D:\workspace\CSE473\_FL13\_guobingkun\lab1>java MapClient localhost 6789 put who

ok

D:\workspace\CSE473\_FL13\_guobingkun\lab1>java MapClient localhost 6789 get who

no match

D:\workspace\CSE473\_FL13\_guobingkun\lab1>java MapClient localhost 6789 get goodbye

ok:world

D:\workspace\CSE473\_FL13\_guobingkun\lab1>java MapClient localhost 6789 got goodbye

error:unrecognizable input:got:goodbye

D:\workspace\CSE473\_FL13\_guobingkun\lab1>java MapClient localhost 6789 pat goodbye world

error:unrecognizable input:pat:goodbye:world

D:\workspace\CSE473\_FL13\_guobingkun\lab1>java MapClient localhost 6789 get bar

no match

D:\workspace\CSE473\_FL13\_guobingkun\lab1>java MapClient localhost 6789 put foo "toast is tasty"

ok

D:\workspace\CSE473\_FL13\_guobingkun\lab1>java MapClient localhost 6789 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.

paste your output here

D:\workspace\CSE473\_FL13\_guobingkun\lab1>java MapClient 192.168.2.9 6789 put foo bar

ok

D:\workspace\CSE473\_FL13\_guobingkun\lab1>java MapClient 192.168.2.9 6789 put who hah

ok

D:\workspace\CSE473\_FL13\_guobingkun\lab1>java MapClient 192.168.2.9 6789 put goodbye world

ok

D:\workspace\CSE473\_FL13\_guobingkun\lab1>java MapClient 192.168.2.9 6789 get foo

ok:bar

D:\workspace\CSE473\_FL13\_guobingkun\lab1>java MapClient 192.168.2.9 6789 get who

ok:hah

D:\workspace\CSE473\_FL13\_guobingkun\lab1>java MapClient 192.168.2.9 6789 put who

ok

D:\workspace\CSE473\_FL13\_guobingkun\lab1>java MapClient 192.168.2.9 6789 get who

no match

D:\workspace\CSE473\_FL13\_guobingkun\lab1>java MapClient 192.168.2.9 6789 get goodbye

ok:world

D:\workspace\CSE473\_FL13\_guobingkun\lab1>java MapClient 192.168.2.9 6789 got goodbye

error:unrecognizable input:got:goodbye

D:\workspace\CSE473\_FL13\_guobingkun\lab1>java MapClient 192.168.2.9 6789 pat goodbye world

error:unrecognizable input:pat:goodbye:world

D:\workspace\CSE473\_FL13\_guobingkun\lab1>java MapClient 192.168.2.9 6789 get bar

no match

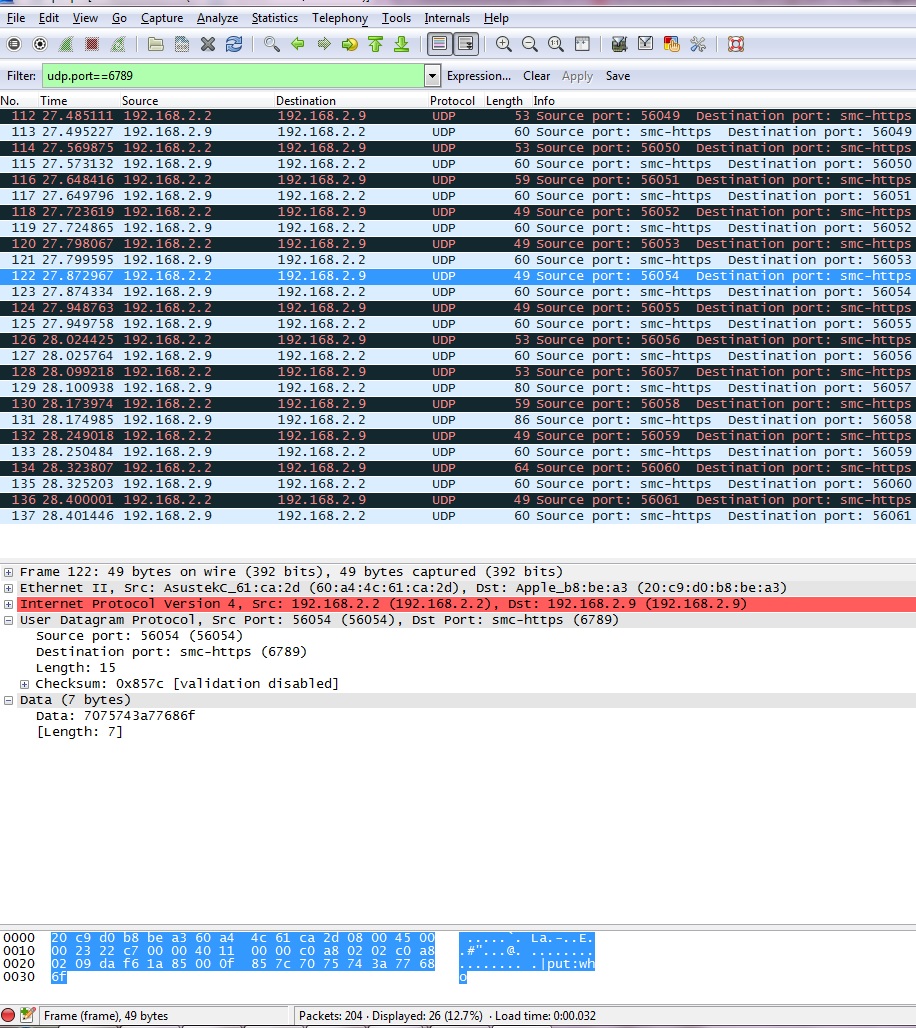
D:\workspace\CSE473\_FL13\_guobingkun\lab1>java MapClient 192.168.2.9 6789 put foo "toast is tasty"

ok

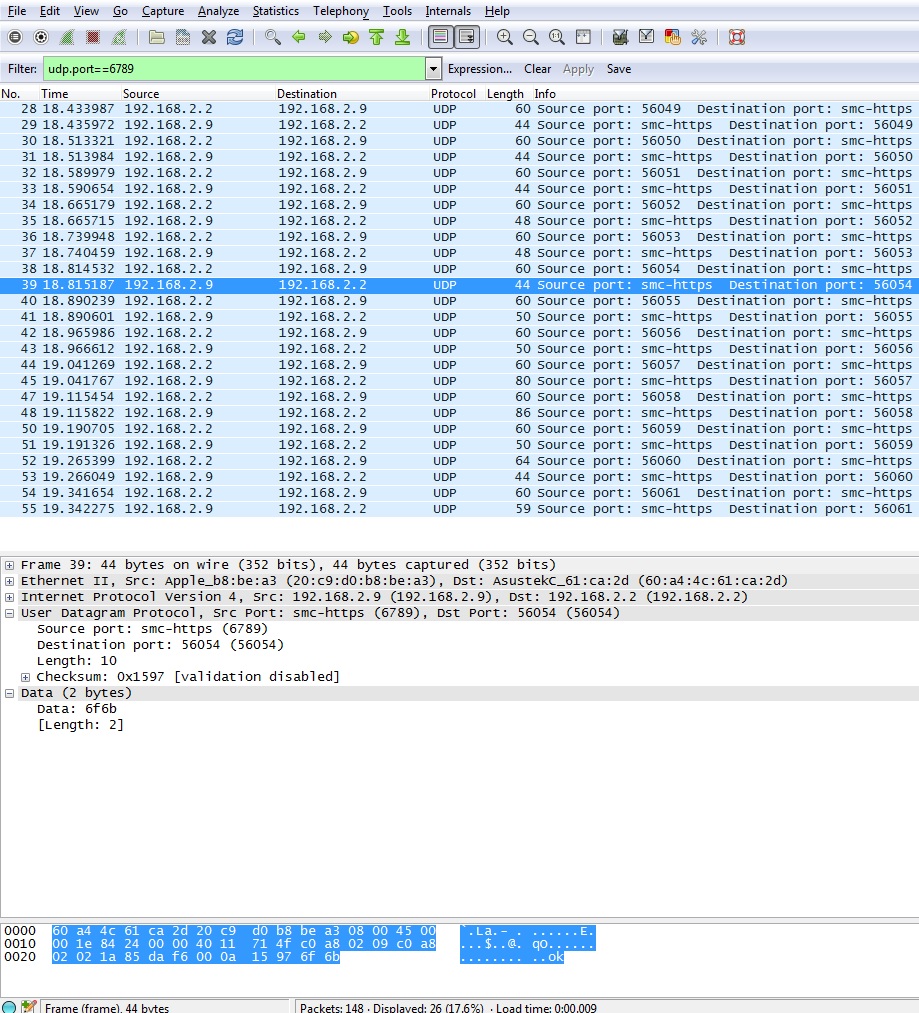
D:\workspace\CSE473\_FL13\_guobingkun\lab1>java MapClient 192.168.2.9 6789 get foo

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 your screenshot here*

******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.

*paste your screenshot here*

***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?

Server IP Address:192.168.2.9 Server Ethernet Address: 20:c9:d0:b8:be:a3

Client IP address: 192.168.2.2 Client Ethernet Address: 60:a4:4c:61:ca:2d

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

The port number used by the client is 56054. No, because the commands were separately executed, thus the operating system were executing different applications and each application has its own port number.

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 client’s port number is da f6, “d” is the most significant, “6” is the least significant. The decimal port number = d \* 16^3 + a \* 16^2 + f \* 16 + 6 = 53248 + 2560 + 240 + 6 = 56054 which matches the port number I observed.

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

49 bytes are shown in the window for packet number 11.

7 bytes are associated with the actual *put* command.

8 bytes are associated with the UDP protocol.

20 bytes are associated with the IP protocol.

The rest are for the Ethernet protocol.

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 at 18.814532 and sent the reply at 18.815187, the difference is 0.000655s

Client sent at 27.872967 and received the reply at 27.874334, the difference is 0.001367s

Time taken to send the two packets across the network = (0.001367 – 0.000655) = 0.000712s