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

Lab 1 Report – 70 Points

##### *Your name: Richard Wu (464493)*

***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.

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Name: Richard Wu

Student ID: 464493

Date: 2019/9/1

This program system includes a client program(MapClient) and a server

program (MapServer). The server will store a set of key and value pairs and

the client can send instructions to server to get, put, or remove any pair

stored. The format of using the client program is listed as follows: the

first argument is the name of the host; the second argument is the server's

port number; the third argument is either "get", "put", or "remove"; if the

instruction is "get", the fourth argument is the key; if the instruction is

"put", the fourth argument is the key and the fifth argument is the value;

if the instruction is "remove", the fourth argument is the key. For the

server side, the default port number used is 30123 unless the first argument

specifies the port number. When it receives an instruction, it will send

back a feedback to the client.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

import java.net.\*;

import java.util.\*;

public class MapServer {

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

// Default port number

int port = 30123;

// Set an optional port number

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

// Create the UDP socket

DatagramSocket sock = new DatagramSocket(port,null);

// Create the receiver packet

byte[] receiverBuf = new byte[1000];

DatagramPacket receiverPkt = new DatagramPacket(receiverBuf,

receiverBuf.length);

// Create HashMap to store the data

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

// Start receiving data

while (true) {

// Wait for incoming packet

sock.receive(receiverPkt);

// Turn the data into string

String dataString = new String(receiverPkt.getData(), 0,

receiverPkt.getLength(), "US-ASCII");

// Manipulate the data

String[] dataStrings = dataString.split(":");

// Return string

String returnString;

// get instruction

if("get".equals(dataStrings[0]) &&

dataStrings.length == 2) {

String value = map.get(dataStrings[1]);

if(value == null) {

returnString = "no match";

} else {

returnString = "ok:"+value;

}

}

// put instruction

else if("put".equals(dataStrings[0]) &&

dataStrings.length == 3) {

String previous = map.put(dataStrings[1], dataStrings[2]);

if(previous == null) {

returnString = "Ok";

} else {

returnString = "updated:"+dataStrings[1];

}

}

// remove instruction

else if("remove".equals(dataStrings[0]) &&

dataStrings.length == 2) {

String removed = map.remove(dataStrings[1]);

if(removed == null) {

returnString = "no match";

} else {

returnString = "Ok";

}

} else { // invalid instruction

returnString = "Error:unrecognizable\ninput:"+dataString;

}

// Create the sender packet

DatagramPacket senderPkt = new DatagramPacket(

returnString.getBytes(),

returnString.length(),

receiverPkt.getAddress(),

receiverPkt.getPort());

// Send the packet

sock.send(senderPkt);

}

}

}

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

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Name: Richard Wu

Student ID: 464493

Date: 2019/9/1

This program system includes a client program(MapClient) and a server

program (MapServer). The server will store a set of key and value pairs and

the client can send instructions to server to get, put, or remove any pair

stored. The format of using the client program is listed as follows: the

first argument is the name of the host; the second argument is the server's

port number; the third argument is either "get", "put", or "remove"; if the

instruction is "get", the fourth argument is the key; if the instruction is

"put", the fourth argument is the key and the fifth argument is the value;

if the instruction is "remove", the fourth argument is the key. For the

server side, the default port number used is 30123 unless the first argument

specifies the port number. When it receives an instruction, it will send

back a feedback to the client.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

import java.net.\*;

public class MapClient {

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

// Check the argument number

if (args.length < 4) {

System.out.println("The number of argument is incorrect");

return;

}

// Get the name of the post

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

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

// Create a UDP datagram socket

DatagramSocket sock = new DatagramSocket();

// Build the instruction string

String outString = null;

// get or remove

if(args.length == 4) {

outString = args[2] + ":" + args[3];

} else if(args.length == 5) { // put

outString = args[2] + ":" + args[3] + ":" + args[4];

}

// Build the output packet

byte[] outBuf = outString.getBytes();

DatagramPacket outPkt = new DatagramPacket(outBuf,outBuf.length,

serverAdr,port);

sock.send(outPkt); // Send packet to server

// Prepare for the input packet

byte[] inBuf = new byte[1000];

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

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

// Print the data in the input packet

String reply = new String(inBuf, 0, inPkt.getLength(), "US-ASCII");

System.out.println(reply);

sock.close(); // Close the socket

}

}

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

D:\WorkingStation\CSE473\_code\cse473s-f19-students-yuanpei.wu\lab1>testScript.cmd RichardWu

D:\WorkingStation\CSE473\_code\cse473s-f19-students-yuanpei.wu\lab1>echo java MapClient RichardWu 30123 put foo bar

java MapClient RichardWu 30123 put foo bar

D:\WorkingStation\CSE473\_code\cse473s-f19-students-yuanpei.wu\lab1>java MapClient RichardWu 30123 put foo bar

Ok

D:\WorkingStation\CSE473\_code\cse473s-f19-students-yuanpei.wu\lab1>echo java MapClient RichardWu 30123 put who hah

java MapClient RichardWu 30123 put who hah

D:\WorkingStation\CSE473\_code\cse473s-f19-students-yuanpei.wu\lab1>java MapClient RichardWu 30123 put who hah

Ok

D:\WorkingStation\CSE473\_code\cse473s-f19-students-yuanpei.wu\lab1>echo java MapClient RichardWu 30123 put goodbye world

java MapClient RichardWu 30123 put goodbye world

D:\WorkingStation\CSE473\_code\cse473s-f19-students-yuanpei.wu\lab1>java MapClient RichardWu 30123 put goodbye world

Ok

D:\WorkingStation\CSE473\_code\cse473s-f19-students-yuanpei.wu\lab1>echo java MapClient RichardWu 30123 get foo

java MapClient RichardWu 30123 get foo

D:\WorkingStation\CSE473\_code\cse473s-f19-students-yuanpei.wu\lab1>java MapClient RichardWu 30123 get foo

ok:bar

D:\WorkingStation\CSE473\_code\cse473s-f19-students-yuanpei.wu\lab1>echo java MapClient RichardWu 30123 get who

java MapClient RichardWu 30123 get who

D:\WorkingStation\CSE473\_code\cse473s-f19-students-yuanpei.wu\lab1>java MapClient RichardWu 30123 get who

ok:hah

D:\WorkingStation\CSE473\_code\cse473s-f19-students-yuanpei.wu\lab1>echo java MapClient RichardWu 30123 remove who

java MapClient RichardWu 30123 remove who

D:\WorkingStation\CSE473\_code\cse473s-f19-students-yuanpei.wu\lab1>java MapClient RichardWu 30123 remove who

Ok

D:\WorkingStation\CSE473\_code\cse473s-f19-students-yuanpei.wu\lab1>echo java MapClient RichardWu 30123 get who

java MapClient RichardWu 30123 get who

D:\WorkingStation\CSE473\_code\cse473s-f19-students-yuanpei.wu\lab1>java MapClient RichardWu 30123 get who

no match

D:\WorkingStation\CSE473\_code\cse473s-f19-students-yuanpei.wu\lab1>echo java MapClient RichardWu 30123 get goodbye

java MapClient RichardWu 30123 get goodbye

D:\WorkingStation\CSE473\_code\cse473s-f19-students-yuanpei.wu\lab1>java MapClient RichardWu 30123 get goodbye

ok:world

D:\WorkingStation\CSE473\_code\cse473s-f19-students-yuanpei.wu\lab1>echo java MapClient RichardWu 30123 got goodbye

java MapClient RichardWu 30123 got goodbye

D:\WorkingStation\CSE473\_code\cse473s-f19-students-yuanpei.wu\lab1>java MapClient RichardWu 30123 got goodbye

Error:unrecognizable

input:got:goodbye

D:\WorkingStation\CSE473\_code\cse473s-f19-students-yuanpei.wu\lab1>echo java MapClient RichardWu 30123 pat goodbye world

java MapClient RichardWu 30123 pat goodbye world

D:\WorkingStation\CSE473\_code\cse473s-f19-students-yuanpei.wu\lab1>java MapClient RichardWu 30123 pat goodbye world

Error:unrecognizable

input:pat:goodbye:world

D:\WorkingStation\CSE473\_code\cse473s-f19-students-yuanpei.wu\lab1>echo java MapClient RichardWu 30123 get bar

java MapClient RichardWu 30123 get bar

D:\WorkingStation\CSE473\_code\cse473s-f19-students-yuanpei.wu\lab1>java MapClient RichardWu 30123 get bar

no match

D:\WorkingStation\CSE473\_code\cse473s-f19-students-yuanpei.wu\lab1>echo java MapClient RichardWu 30123 put foo "toast is tasty"

java MapClient RichardWu 30123 put foo "toast is tasty"

D:\WorkingStation\CSE473\_code\cse473s-f19-students-yuanpei.wu\lab1>java MapClient RichardWu 30123 put foo "toast is tasty"

updated:foo

D:\WorkingStation\CSE473\_code\cse473s-f19-students-yuanpei.wu\lab1>echo java MapClient RichardWu 30123 get foo

java MapClient RichardWu 30123 get foo

D:\WorkingStation\CSE473\_code\cse473s-f19-students-yuanpei.wu\lab1>java MapClient RichardWu 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.

wuyuanpeidembp:lab1 wuyuanpei$ ./testScript RichardWu

java MapClient RichardWu 30123 put foo bar

Ok

java MapClient RichardWu 30123 put who hah

Ok

java MapClient RichardWu 30123 put goodbye world

Ok

java MapClient RichardWu 30123 get foo

ok:bar

java MapClient RichardWu 30123 get who

ok:hah

java MapClient RichardWu 30123 remove who

Ok

java MapClient RichardWu 30123 get who

no match

java MapClient RichardWu 30123 get goodbye

ok:world

java MapClient RichardWu 30123 got goodbye

Error:unrecognizable

input:got:goodbye

java MapClient RichardWu 30123 pat goodbye world

Error:unrecognizable

input:pat:goodbye:world

java MapClient RichardWu 30123 get bar

no match

java MapClient RichardWu 30123 put foo toast is tasty

updated:foo

java MapClient RichardWu 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 your screenshot here*

***A screenshot of a social media post

Description automatically generated***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*

A screenshot of a social media post

Description automatically generated

***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?

*The IP address of the server is 172.27.139.164; the IP address of the client is 172.27.159.10*

*The Ethernet address of the server is 9c:b6:d0:db:1c:b7*

*The Ethernet address of the client is f0:18:98:26:72:cd*

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?

*Client uses port number 62740*

*No, because the port number is dynamically assigned by the operating system for a UDP client program.*

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.

*f5 14, the most significant digit is f; the least significant digit is 4. 0xf514 matches 62740.*

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?

*52 bytes for packet 11. Remove command contains 10 bytes. UDP protocol contains 8 bytes. IP protocol contains 20 bytes. The rest is about Ethernet II, containing 14 bytes.*

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?

*The server received packet11 at the 1.002961 sec. It sent the reply at the 1.003454 sec. The difference is 0.000493 sec, meaning the time that the server code used to deal with its logic and prepare for a reply. The client sent packet11 at the 1.009187 sec. It received the reply at the 1.013261sec. The difference is 0.004074 sec. The time it takes to send the packets across the network is the difference between 0.004074 and 0.000493, which is 0.003581 sec.*