*CSE 473 – Introduction to Computer Networks*

Lab 1 Report

*David McDonnel 9/9/2015*

***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.

paste your code here

//////////////////////////////////////////////////////////////

// Name: David McDonnel //

// WUID: 420325 //

// Date: 9/8/2015 //

// Description: MapServer //

// - Input: port - optional argument to specify port # //

// //

// - Behavior: This program runs a server, //

// accepts input from a client, //

// performs the specified operations on the //

// argument(s), //

// stores arguments to or updates a hashmap //

//////////////////////////////////////////////////////////////

import java.io.\*;

import java.net.\*;

import java.util.HashMap;

public class MapServer {

private HashMap<String,String> hMap;

public MapServer(){

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

}

/\*\*

\*input String key

\*perform hashmap get on input

\*return success message

\*\*/

public String get(String key){

String ans = hMap.get(key);

return ans == null ? "no match" : "success:"+ans;

}

/\*\*

\*input String key, String value

\*perform hashmap put operation on inputs

\*return success messsage

\*\*/

public String put(String key, String value){

boolean b = hMap.containsKey(key);

String ans = hMap.put(key,value);

return b ? "updated:"+key : "success";

}

/\*\*

\*input String key

\*perform hashmap remove on input

\*return success message

\*\*/

public String remove(String key){

String ans = hMap.remove(key);

return ans!=null ? "success":"no match";

}

/\*\*

\*input String key, String value

\*swap two hashmap key value pairs

\*return success message

\*\*/

public String swap(String key1,String key2){

boolean b = hMap.containsKey(key1)

&& hMap.containsKey(key2);

String tmp = hMap.get(key1);

hMap.put(key1,hMap.get(key2));

hMap.put(key2,tmp);

return b ? "success":"no match";

}

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

InetAddress myIp = null; int port = 31357;

myIp = InetAddress.getLocalHost();

//If port present in args then set

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

DatagramSocket sock = new DatagramSocket(port,myIp);

// create two packets sharing a common buffer

byte[] buf = new byte[1000];

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

String[] input;

String response;

int length;

String data;

MapServer ms = new MapServer();

while (true) {

// wait for incoming packet

sock.receive(pkt);

//parse incoming packet

data = new String(pkt.getData(),0,pkt.getLength());

input = data.split(":");

length = input.length;

//handle all cases of operation input

switch (input[0]){

case "get":

response=length==2?ms.get(input[1]):"bad";

break;

case "put":

response=length==3?ms.put(input[1],input[2]):"bad";

break;

case "remove":

response=length==2?ms.remove(input[1]):"bad";

break;

case "swap":

response=length==3?ms.swap(input[1],input[2]):"bad";

break;

default:

response="bad";

break;

}

if(response=="bad"){

response = "error:unrecognizable input:"+data;

}

pkt.setData(response.getBytes());

// 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

//////////////////////////////////////////////////////////////

// Name: David McDonnel //

// WUID: 420325 //

// Date: 9/8/2015 //

// Description: MapClient //

// - Input: This program take 4-5 arguments //

// name - name of host server runs on //

// ,port - port number server listens on //

// ,operation - get, put, swap, or remove //

// ,arg1 - first argument to be operated on //

// ,arg2 - second argument to be operated on //

// //

// - Behavior: This program connects to a server, //

// then sends a request with an operation //

// and one or two arguments //

//////////////////////////////////////////////////////////////

import java.io.\*;

import java.net.\*;

/\*\*

\* usage: MapClient name port operation arg1 arg2

\* Input:

\* name - String

\* port - integer

\* operation - String

\* arg1 - String

\* arg2 - String

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

\* Wait for reply packet and print its contents.

\*\*/

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]);

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

// open datagram socket

DatagramSocket sock = new DatagramSocket();

// build packet addressed to server containing formatted arguments,

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

String msg = "";

for (int i = 2; i < args.length; i++){

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

}

msg = msg.substring(0,msg.length()-1);

byte[] outBuf = msg.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");

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 code here

java MapClient Davids-MacBook-Pro.local 31357 put foo bar

success

java MapClient Davids-MacBook-Pro.local 31357 put who hah

success

java MapClient Davids-MacBook-Pro.local 31357 put goodbye world

success

java MapClient Davids-MacBook-Pro.local 31357 get foo

success:bar

java MapClient Davids-MacBook-Pro.local 31357 get who

success:hah

java MapClient Davids-MacBook-Pro.local 31357 swap who foo

success

java MapClient Davids-MacBook-Pro.local 31357 get foo

success:hah

java MapClient Davids-MacBook-Pro.local 31357 get who

success:bar

java MapClient Davids-MacBook-Pro.local 31357 remove who

success

java MapClient Davids-MacBook-Pro.local 31357 get who

no match

java MapClient Davids-MacBook-Pro.local 31357 get goodbye

success:world

java MapClient Davids-MacBook-Pro.local 31357 got goodbye

error:unrecognizable input:got:goodbye

java MapClient Davids-MacBook-Pro.local 31357 pat goodbye world

error:unrecognizable input:pat:goodbye:world

java MapClient Davids-MacBook-Pro.local 31357 get bar

no match

java MapClient Davids-MacBook-Pro.local 31357 put foo toast is tasty

error:unrecognizable input:put:foo:toast:is:tasty

java MapClient Davids-MacBook-Pro.local 31357 get foo

success:hah

java MapClient Davids-MacBook-Pro.local 31357 swap who

error:unrecognizable input:swap:who

java MapClient Davids-MacBook-Pro.local 31357 get

error:unrecognizable input:get

java MapClient Davids-MacBook-Pro.local 31357 put foo

error:unrecognizable input:put:foo

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

paste your code here

***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.

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?

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?

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.

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?

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?