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

Lab 1 Report – 70 Points

##### *Your name here: Gai Ashkenazy*

***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.HashMap;  
  
public class MapServer {  
 public static void main(String args[]) throws Exception {  
  
 //get port number or set it to default  
 //check if port number given is valid  
 int port;  
 if(args.length > 0) {  
 int argumentPort = Integer.*parseInt*(args[0]);  
 if (argumentPort > 0 && argumentPort < 65536) {  
 port = argumentPort;  
 } else {  
 port = 30123;  
 }  
 } else {  
 port = 30123;  
 }  
 // open a UDP socket on port given  
 DatagramSocket sock = new DatagramSocket(port);  
 System.*out*.println("Server running on port " + sock.getLocalPort());  
 //init hash map, out packet  
 HashMap<String, String> map = new HashMap<>();  
 DatagramPacket outPkt;  
  
 while (true) {  
 // create a DatagramPacket for receiving packets  
 byte[] buf = new byte[1000];  
 DatagramPacket pkt = new DatagramPacket(buf, buf.length);  
 // response array to construct the response to client  
 byte[] response = new byte[1000];  
  
 sock.receive(pkt); // wait for incoming packet (blocking)  
  
 //get clients port and address  
 InetAddress clientAdd = pkt.getAddress();  
 int clientPort = pkt.getPort();

//take the packets information and process it  
 String message = new String(buf);  
 String [] splitedMessage = message.split(":", 3);  
 String command = splitedMessage[0]; String key; String value;  
  
 // PUT command  
 if(command.equals("put") && splitedMessage.length > 2) {  
 key = splitedMessage[1];  
 value = splitedMessage[2];  
  
 //create response  
 if(map.containsKey(key)){  
 response = ("updated:" + key).  
 getBytes("US-ASCII");  
 } else {  
 response = "Ok".getBytes("US-ASCII");  
 }  
  
 //insert the key and value to the map, if needed this will  
 // update and existing key  
 map.put(key, value);  
 //create response packet  
 outPkt = new DatagramPacket(response,response.length,  
 clientAdd, clientPort);  
 //send response  
 sock.send(outPkt);  
 //GET command  
 } else if (command.equals("get") && splitedMessage.length > 1){  
 key = splitedMessage[1];  
  
 //create response to client  
 if(map.containsKey(key)){  
 response = ("ok:" + map.get(key)).  
 getBytes("US-ASCII");  
 } else {  
 response = "no match".getBytes("US-ASCII");  
 }  
  
 //create response packet  
 outPkt = new DatagramPacket(response,response.length,  
 clientAdd, clientPort);  
 //send response  
 sock.send(outPkt);  
 // REMOVE command  
 } else if (command.equals("remove") && splitedMessage.length > 1){  
 key = splitedMessage[1];

//create response to client  
 if(map.containsKey(key)){  
 response = "Ok".getBytes("US-ASCII");  
 //remove key from map if key is in map  
 map.remove(key);  
 } else {  
 response = "no match".getBytes("US-ASCII");  
 }  
 //create response packet  
 outPkt = new DatagramPacket(response,response.length,  
 clientAdd, clientPort);  
 //send response  
 sock.send(outPkt);  
 //if command is NOT VALID  
 } else {  
 String payload = new String(pkt.getData(), 0, pkt.getLength());  
 response = ("Error: unrecognizable input: {" + payload + "}").  
 getBytes("US-ASCII");  
 //create response packet  
 outPkt = new DatagramPacket(response,response.length,  
 clientAdd, clientPort);  
 //send response  
 sock.send(outPkt);

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

import java.io.\*;  
import java.net.\*;  
  
*/\*\*  
 \* usage: UdpEchoClient serverName port string  
 \*  
 \* 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, port number and command to server, if they  
 // don't exist enter default values  
 InetAddress serverAdr =(args.length == 0)? null :  
 InetAddress.*getByName*(args[0]);  
 //port  
 int port = (args.length <= 1) ? -1 : Integer.*parseInt*(args[1]);  
 //server command  
 String command;  
 if (args.length <= 2) {  
 command = "";  
 } else if (args.length <= 3) {  
 command = args[2];  
 } else if (args.length <= 4) {  
 command = args[2] + ":" + args[3] + ":" ;  
 } else if (args.length <= 5) {  
 command = args[2] + ":" + args[3] + ":" + args[4];  
 } else {  
 command = "";  
 }  
// open datagram socket  
 DatagramSocket sock = new DatagramSocket();  
 // build packet addressed to server containing server command  
 byte[] outBuf = command.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 (this is blocking)  
  
 // 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.

C:\Users\GuyPc\Downloads\CSE473S - Computer Networks\cse473s-f19-students-gai.ashkenazy\lab1>testScript.cmd 172.27.32.34

C:\Users\GuyPc\Downloads\CSE473S - Computer Networks\cse473s-f19-students-gai.ashkenazy\lab1>echo java MapClient 172.27.32.34 30123 put foo bar

java MapClient 172.27.32.34 30123 put foo bar

C:\Users\GuyPc\Downloads\CSE473S - Computer Networks\cse473s-f19-students-gai.ashkenazy\lab1>java MapClient 172.27.32.34 30123 put foo bar

Ok

C:\Users\GuyPc\Downloads\CSE473S - Computer Networks\cse473s-f19-students-gai.ashkenazy\lab1>echo java MapClient 172.27.32.34 30123 put who hah

java MapClient 172.27.32.34 30123 put who hah

C:\Users\GuyPc\Downloads\CSE473S - Computer Networks\cse473s-f19-students-gai.ashkenazy\lab1>java MapClient 172.27.32.34 30123 put who hah

Ok

C:\Users\GuyPc\Downloads\CSE473S - Computer Networks\cse473s-f19-students-gai.ashkenazy\lab1>echo java MapClient 172.27.32.34 30123 put goodbye world

java MapClient 172.27.32.34 30123 put goodbye world

C:\Users\GuyPc\Downloads\CSE473S - Computer Networks\cse473s-f19-students-gai.ashkenazy\lab1>java MapClient 172.27.32.34 30123 put goodbye world

Ok

C:\Users\GuyPc\Downloads\CSE473S - Computer Networks\cse473s-f19-students-gai.ashkenazy\lab1>echo java MapClient 172.27.32.34 30123 get foo

java MapClient 172.27.32.34 30123 get foo

C:\Users\GuyPc\Downloads\CSE473S - Computer Networks\cse473s-f19-students-gai.ashkenazy\lab1>java MapClient 172.27.32.34 30123 get foo

ok:bar

C:\Users\GuyPc\Downloads\CSE473S - Computer Networks\cse473s-f19-students-gai.ashkenazy\lab1>echo java MapClient 172.27.32.34 30123 get who

java MapClient 172.27.32.34 30123 get who

C:\Users\GuyPc\Downloads\CSE473S - Computer Networks\cse473s-f19-students-gai.ashkenazy\lab1>java MapClient 172.27.32.34 30123 get who

ok:hah

C:\Users\GuyPc\Downloads\CSE473S - Computer Networks\cse473s-f19-students-gai.ashkenazy\lab1>echo java MapClient 172.27.32.34 30123 remove who

java MapClient 172.27.32.34 30123 remove who

C:\Users\GuyPc\Downloads\CSE473S - Computer Networks\cse473s-f19-students-gai.ashkenazy\lab1>java MapClient 172.27.32.34 30123 remove who

Ok

C:\Users\GuyPc\Downloads\CSE473S - Computer Networks\cse473s-f19-students-gai.ashkenazy\lab1>echo java MapClient 172.27.32.34 30123 get who

java MapClient 172.27.32.34 30123 get who

C:\Users\GuyPc\Downloads\CSE473S - Computer Networks\cse473s-f19-students-gai.ashkenazy\lab1>java MapClient 172.27.32.34 30123 get who

no match

C:\Users\GuyPc\Downloads\CSE473S - Computer Networks\cse473s-f19-students-gai.ashkenazy\lab1>echo java MapClient 172.27.32.34 30123 get goodbye

java MapClient 172.27.32.34 30123 get goodbye

C:\Users\GuyPc\Downloads\CSE473S - Computer Networks\cse473s-f19-students-gai.ashkenazy\lab1>java MapClient 172.27.32.34 30123 get goodbye

ok:world

C:\Users\GuyPc\Downloads\CSE473S - Computer Networks\cse473s-f19-students-gai.ashkenazy\lab1>echo java MapClient 172.27.32.34 30123 got goodbye

java MapClient 172.27.32.34 30123 got goodbye

C:\Users\GuyPc\Downloads\CSE473S - Computer Networks\cse473s-f19-students-gai.ashkenazy\lab1>java MapClient 172.27.32.34 30123 got goodbye

Error: unrecognizable input: {got:goodbye:}

C:\Users\GuyPc\Downloads\CSE473S - Computer Networks\cse473s-f19-students-gai.ashkenazy\lab1>echo java MapClient 172.27.32.34 30123 pat goodbye world

java MapClient 172.27.32.34 30123 pat goodbye world

C:\Users\GuyPc\Downloads\CSE473S - Computer Networks\cse473s-f19-students-gai.ashkenazy\lab1>java MapClient 172.27.32.34 30123 pat goodbye world

Error: unrecognizable input: {pat:goodbye:world}

C:\Users\GuyPc\Downloads\CSE473S - Computer Networks\cse473s-f19-students-gai.ashkenazy\lab1>echo java MapClient 172.27.32.34 30123 get bar

java MapClient 172.27.32.34 30123 get bar

C:\Users\GuyPc\Downloads\CSE473S - Computer Networks\cse473s-f19-students-gai.ashkenazy\lab1>java MapClient 172.27.32.34 30123 get bar

no match

C:\Users\GuyPc\Downloads\CSE473S - Computer Networks\cse473s-f19-students-gai.ashkenazy\lab1>echo java MapClient 172.27.32.34 30123 put foo "toast is tasty"

java MapClient 172.27.32.34 30123 put foo "toast is tasty"

C:\Users\GuyPc\Downloads\CSE473S - Computer Networks\cse473s-f19-students-gai.ashkenazy\lab1>java MapClient 172.27.32.34 30123 put foo "toast is tasty"

updated:foo

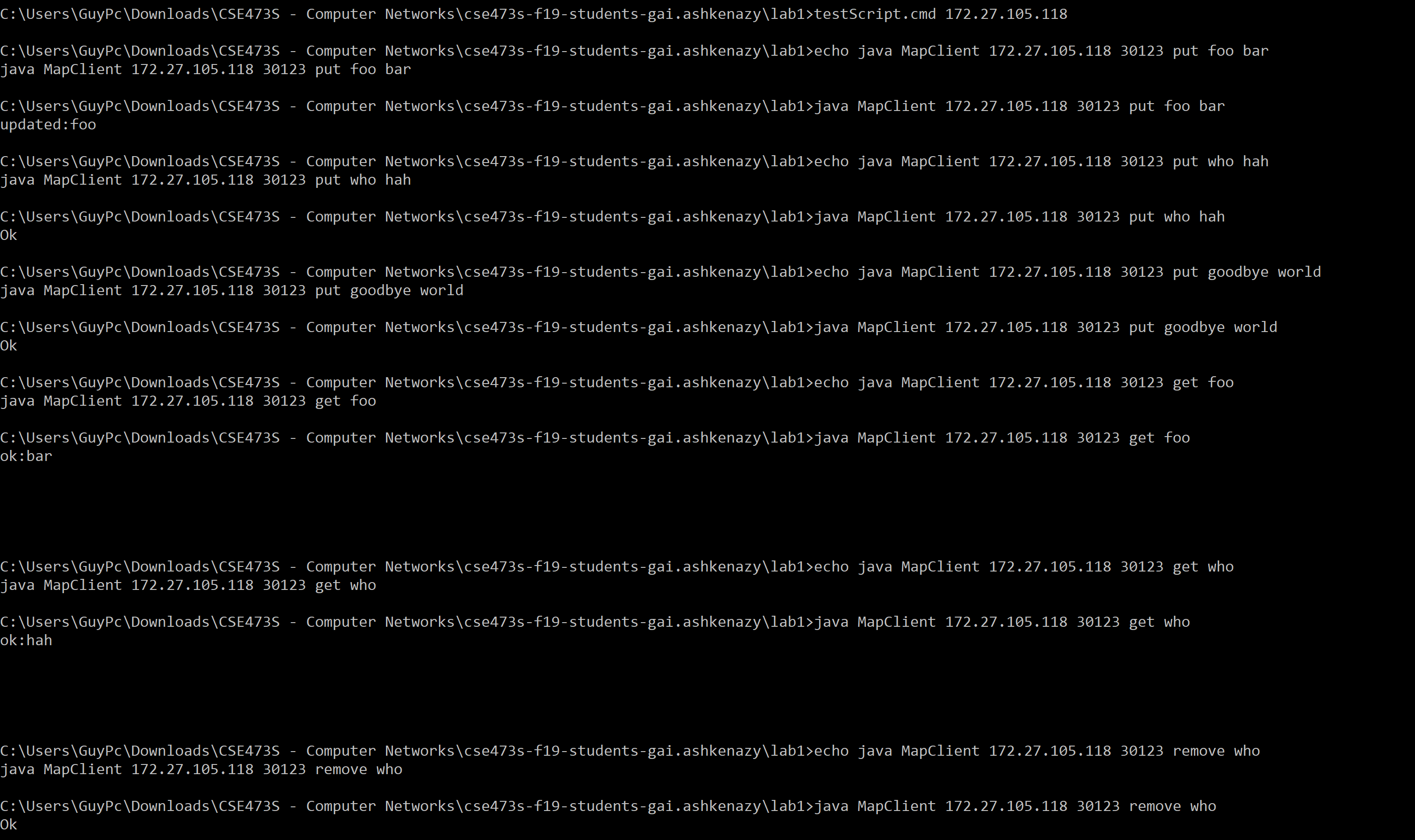
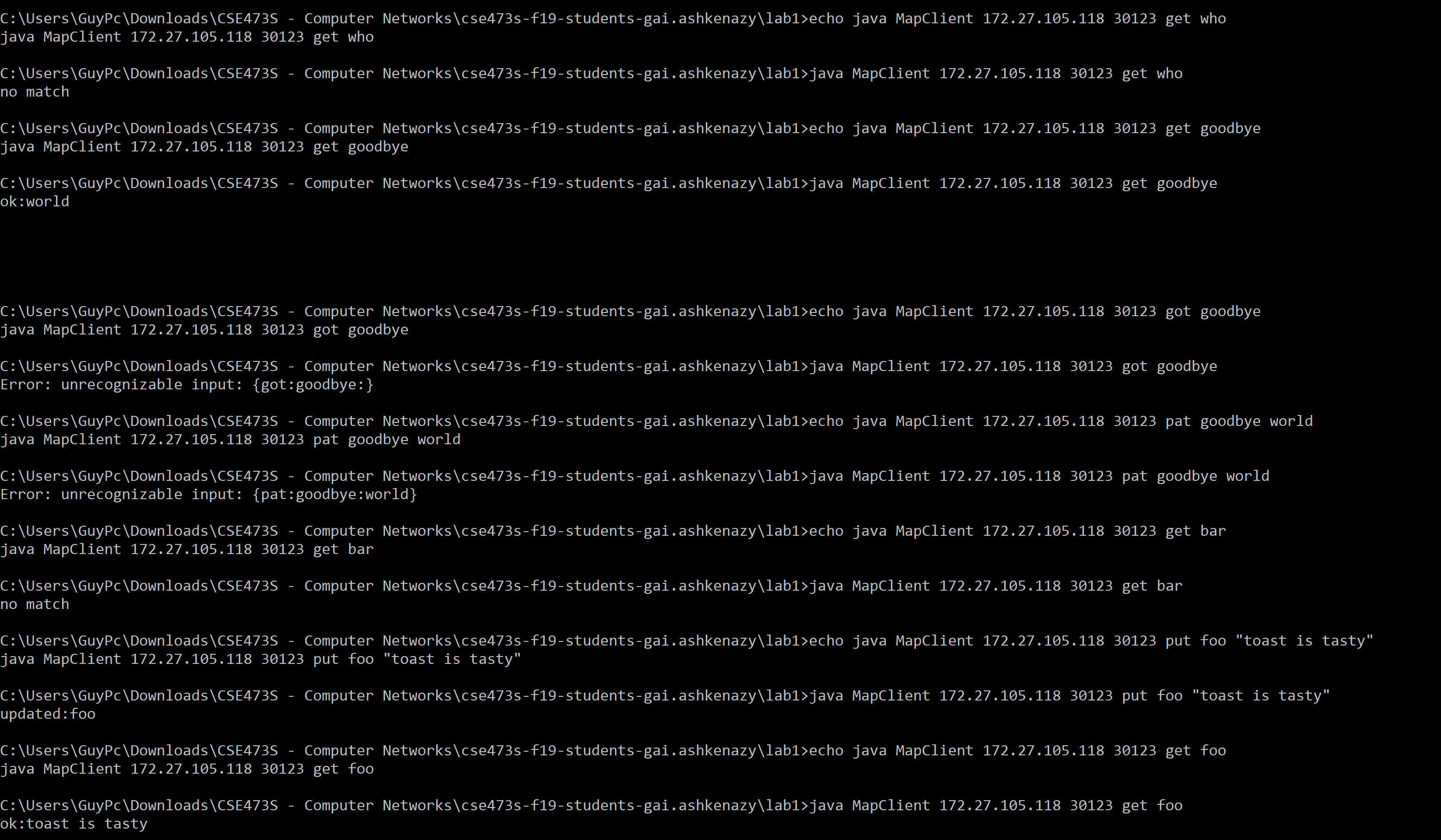
C:\Users\GuyPc\Downloads\CSE473S - Computer Networks\cse473s-f19-students-gai.ashkenazy\lab1>echo java MapClient 172.27.32.34 30123 get foo

java MapClient 172.27.32.34 30123 get foo

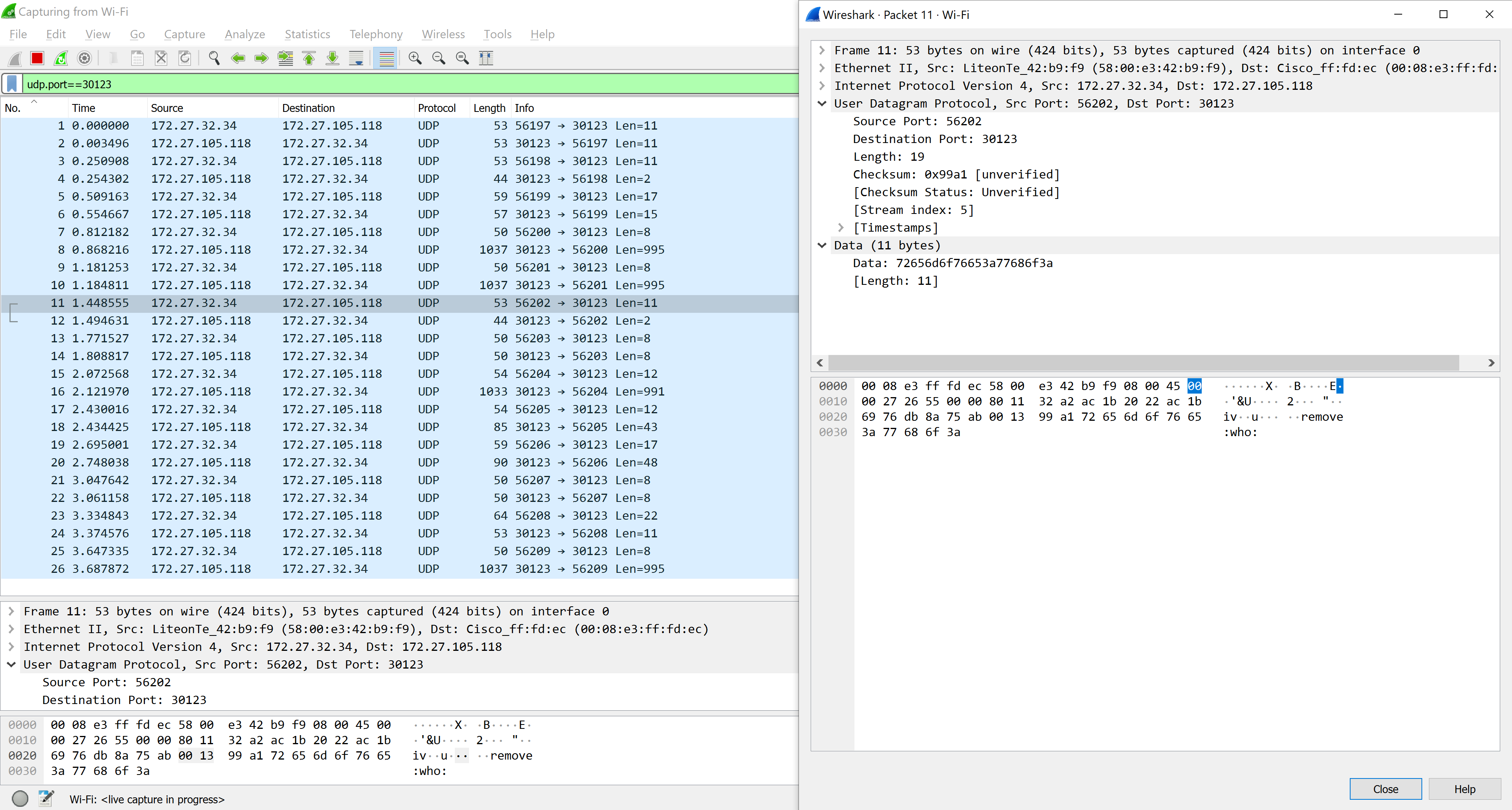
C:\Users\GuyPc\Downloads\CSE473S - Computer Networks\cse473s-f19-students-gai.ashkenazy\lab1>java MapClient 172.27.32.34 30123 get foo

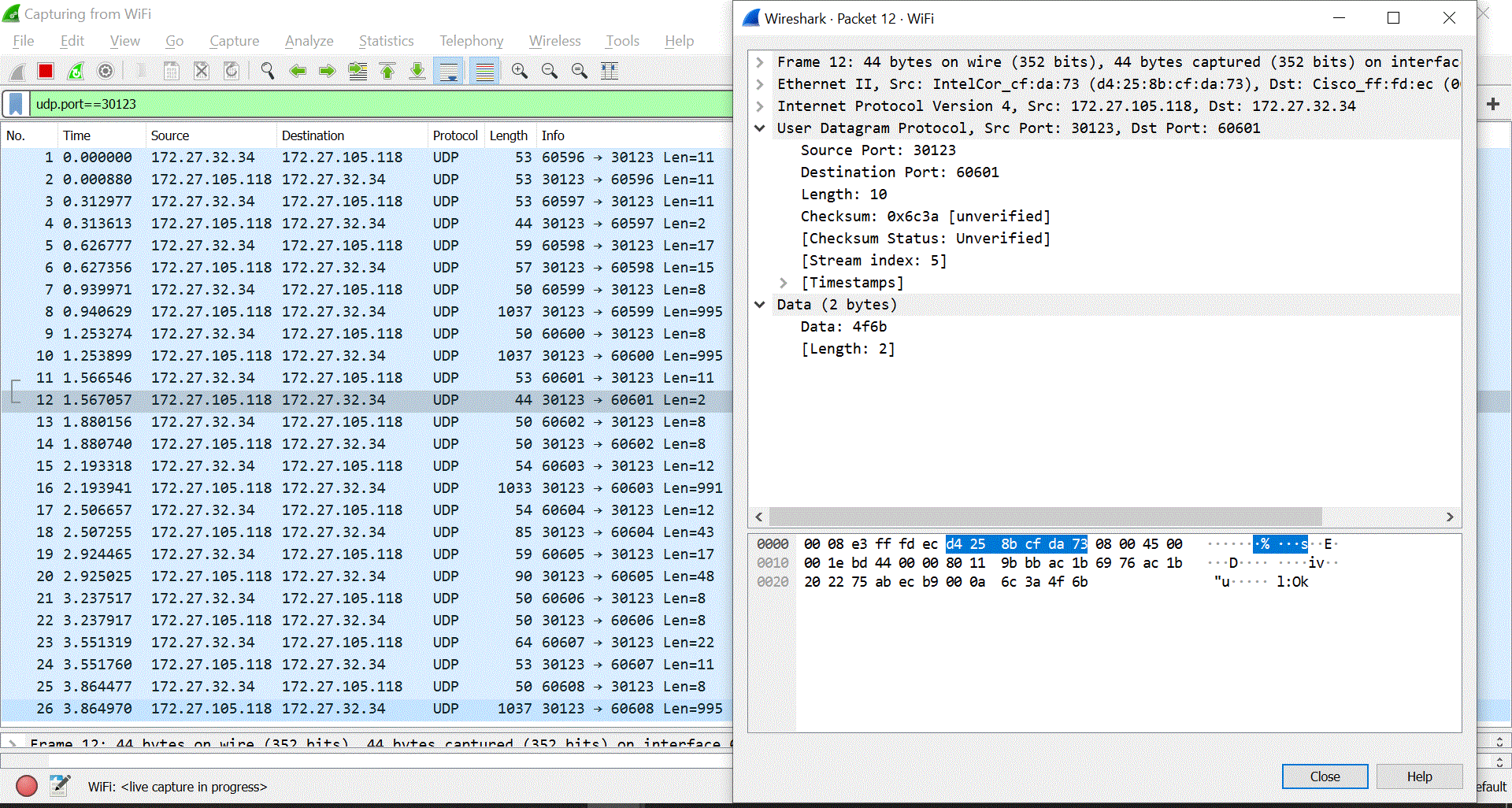
ok:toast is tasty

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

***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?

*Server: 172.27.105.118 Client: 172.27.32.34*

*Address Server: LiteonTe\_42:b9:f9 (58:00:e3:42:b9:f9) Address Client: Cisco\_ff:fd:ec (00:08:e3:ff:fd:ec)*

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?

*51591, the port is changing. This is because every time the test script runs java MapClient it initializes the port to a new port (usually the port changes by 1).*

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.

*Hex port number: c987, hex digits is a number represented in base 16 (from 0-f).*

*Like any numbering system the most significant digit is the left most number, in this case ‘c’, which is 12 in base 10, and because it is the 4th digit its value is*

*And the least significant is 7, in base 10 it is 7.*

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 packet is 53 bytes. The remove command is 11 bytes. UDP protocol is 8 bytes, and IP header is 20 bytes. The rest, 14 bytes, is 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: Sep 1, 2019 18:38:11.747775000 Central Daylight Time.

Sent: Sep 1, 2019 18:38:11.748259000 Central Daylight Time.

Difference: 0.000484 seconds.

*Client:*

Received: Sep 1, 2019 18:38:10.170124000 Central Daylight Time.

Sent: Sep 1, 2019 18:38:10.173605000 Central Daylight Time.

Difference: 0.003481 seconds.

This is a bit odd. Because this shows that the client gets the response before the server sent it, which is of cures not possible. I checked it three times and still got same answers. This could have happened because I ran the program on different computers and their time could be different.

What I can conclude is that between sending and receiving a packet it takes in this case about 0.4-3 milliseconds.