Nova Southeastern University
College of Computing and Engineering

Assignment 4

CISC650 Computer Networks

Fall 2019

Due Date: 11/24/2019 11:59 PM EST

Total Points: 100

Part 1. Text reading

Chapter 7, Chapter 8, Chapter 9

Part 2. Textbook questions

Chapter 7 [25 points]

7.1 What are the services that can be provided using IEEE 802.11?

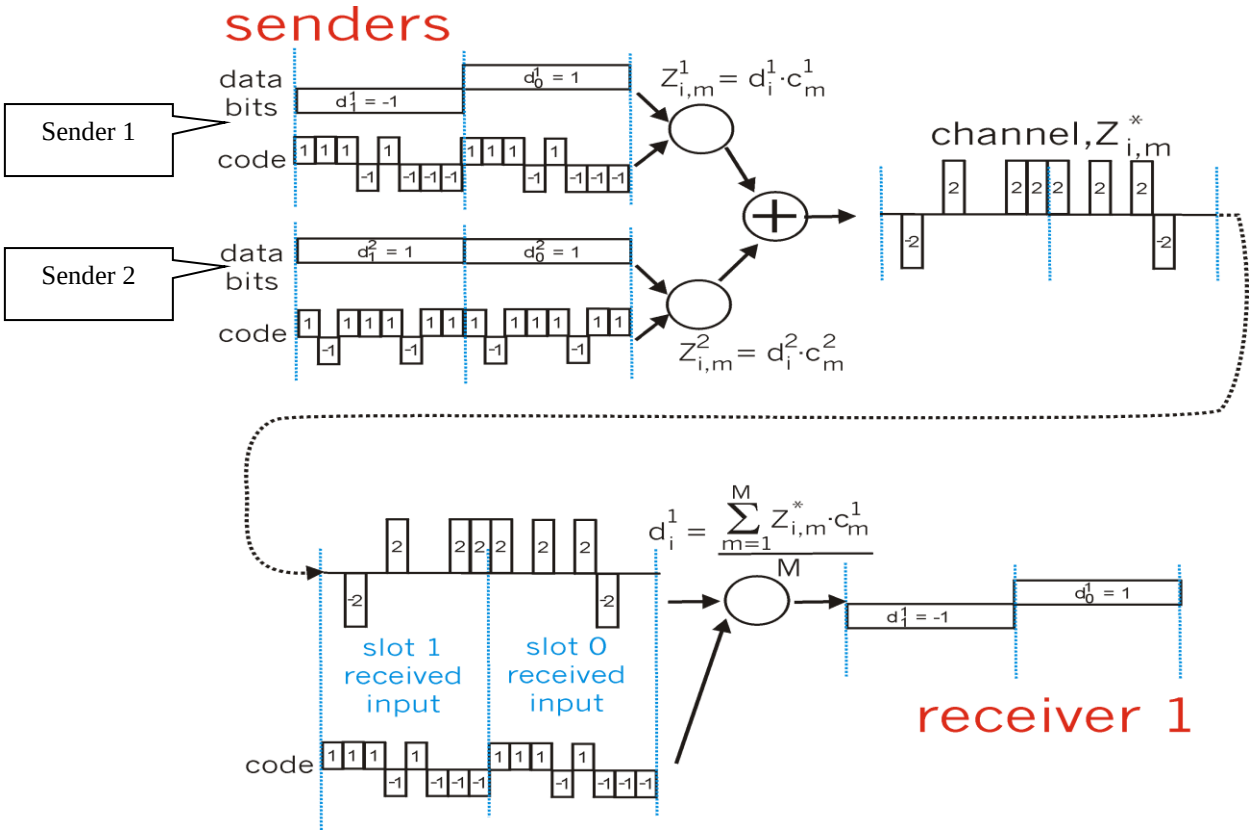
7.2 What are the differences between the infrastructure mode and the ad-hoc mode in wireless networks?

7.3 Why are acknowledgements used in 802.11 but not in a wired Ethernet?

7.4 Suppose the IEEE 802.11 RTS and CTS frames were as long as the standard DATA and ACK frames. Would there be any advantages to using the CTS and RTS frames? Why or why not?

7.5 What is the difference between a permanent address and a care-of address? Who assigns a care-of address?

7.6 The following picture is a copy of a Figure in the textbook. It shows a CDMA example that supports two senders. Suppose that the receiver wanted to receive the data being sent by sender 2. Show (by calculation) that the receiver is indeed able to recover sender 2's data from the aggregate channel signal by using sender 2's code.



Chapter 8 [25 points]

8.1 What is the most important difference between a symmetric key system and a public key system?

8.2 In what way does a hash function provide a better message integrity check than a checksum (such as Internet Checksum)?

8.3 Can you "decrypt" a hash of a message to get the original message? Explain.

8.4 Suppose that Bob receives a PGP message from Alice. How does Bob know for sure that Alice created the message?

8.5 Consider WEP for 802.11. Suppose that the data is 10010100 and the keystream is 10110010. What is the resulting ciphertext?

8.6. Using the monoalphabetic cipher in the textbook, encode the message "This is a server." Decode the message "rmij'u uamu xyj."

8.7 Consider the RSA algorithm with $p=5$ and $q=17$.
a. What are n and z?
b. Let e be 7. Is this an acceptable choice? Why? If not, can you suggest another option?
c. Based on your answer for part b), find d such that $de=1 \pmod z$ and $d<85$.

Chapter 9 [25 points]

9.1 Generalize the basic approaches we used for making the best out of best effort service for real-time interactive multimedia applications.

9.2 There are two types of redundancy in video. Describe them and discuss how they can be exploited for efficient compression.

9.3 Assume an Internet phone application generates packets only during talk spurts. During a talk spurt the sender generates bytes at a rate of 1000 bytes per second, and every 50 msecs the sender gathers bytes into chunks. Assume that RTP is used that will add a header to each chunk. In addition UDP and IP will be used. Suppose all headers (including RTP, UDP and IP) have a total length of h and an IP datagram is emitted every 40 msecs. Find the transmission rate in bits per second for the datagram generated by one side of the application.

9.4 Consider the procedure described in "Adaptive Playout Delay" for estimating average delay d_i. Let $r_3 - t_3$ be the most recent sample delay, let $r_2 - t_2$ be the next most recent sample delay, and so on. For a given audio application, suppose three packets have arrived at the receiver with sample delays $r_1 - t_1$, $r_2 - t_2$, and $r_3 - t_3$. Express the estimate of delay d in terms of u and the three samples.

9.5 This chapter describes several FEC schemes. Briefly summarize them . Both schemes increase the transmission rate of the stream by adding overhead. Does interleaving also increase the transmission rate?

9.6 Compare the procedure described in "Adaptive Playout Delay" for estimating average delay with the procedure in Chapter 3 ("Estimating the Round-Trip Time") for estimating round-trip time. What do the procedures have in common? How are they different?

9.7 Is it possible for a CDN to provide worse performance to a host requesting a multimedia object than if the host has requested the object directly from the distant origin server? Please explain.

9.8 What is the difference between end-to-end delay and packet jitter? What are the causes of packet jitter?

9.9 Summarize how the token buckets and WFQs can be used together to provide policing mechanisms.

Part 3. Practical assignment

3.1 Ping Application [15 points]

This assignment consists of programming a Pinger client. Please submit the following items in a ZIP file.

1) Java or Python source code;
2) Instructions on how to install and run your program;
3) A brief design document explaining your solution.

Note: I shall not provide remedial help concerning coding problems that you might have. Students are responsible for the setup of their own coding environment. Each student is also expected to debug their code.

Option 1: UDP Pinger Lab Using Java

In this assignment, you will study a simple Internet ping server written in the Java language, and implement a corresponding client. The functionality provided by these programs is similar to the standard ping programs available in modern operating systems, except that they use UDP rather than Internet Control Message Protocol (ICMP) to communicate with each other. (Java does not provide a straightforward means to interact with ICMP.)

The ping protocol allows a client machine to send a packet of data to a remote machine, and have the remote machine return the data back to the client unchanged (an action referred to as echoing). Among other uses, the ping protocol allows hosts to determine round-trip times to other machines.

You are given the complete code for the Ping server below. Your job is to write the Ping client.

Server Code

The following code fully implements a ping server. You need to compile and run this code. You should study this code carefully, as it will help you write your Ping client.

```
import java.io.*;
import java.net.*;
import java.util.*;

/*
 * Server to process ping requests over UDP.
 */
public class PingServer
{
  private static final double LOSS_RATE = 0.3;
  private static final int AVERAGE_DELAY = 100;  // milliseconds

  public static void main(String[] args) throws Exception
  {
    // Get command line argument.
    if (args.length != 1) {
      System.out.println("Required arguments: port");
      return;
    }
    int port = Integer.parseInt(args[0]);
```

```
  // Create random number generator for use in simulating
  // packet loss and network delay.
  Random random = new Random();

  // Create a datagram socket for receiving and sending UDP packets
  // through the port specified on the command line.
  DatagramSocket socket = new DatagramSocket(port);

  // Processing loop.
  while (true) {
    // Create a datagram packet to hold incomming UDP packet.
    DatagramPacket request = new DatagramPacket(new byte[1024], 1024);

    // Block until the host receives a UDP packet.
    socket.receive(request);

    // Print the recieved data.
    printData(request);

    // Decide whether to reply, or simulate packet loss.
    if (random.nextDouble() < LOSS_RATE) {
      System.out.println("   Reply not sent.");
      continue;
    }

    // Simulate network delay.
    Thread.sleep((int) (random.nextDouble() * 2 * AVERAGE_DELAY));

    // Send reply.
    InetAddress clientHost = request.getAddress();
    int clientPort = request.getPort();
    byte[] buf = request.getData();
    DatagramPacket reply = new DatagramPacket(buf, buf.length, clientHost, clientPort);
    socket.send(reply);

    System.out.println("   Reply sent.");
  }
}

/*
 * Print ping data to the standard output stream.
 */
private static void printData(DatagramPacket request) throws Exception
{
  // Obtain references to the packet's array of bytes.
  byte[] buf = request.getData();

  // Wrap the bytes in a byte array input stream,
  // so that you can read the data as a stream of bytes.
  ByteArrayInputStream bais = new ByteArrayInputStream(buf);

  // Wrap the byte array output stream in an input stream reader,
  // so you can read the data as a stream of characters.
```

```
    InputStreamReader isr = new InputStreamReader(bais);

    // Wrap the input stream reader in a bufferred reader,
    // so you can read the character data a line at a time.
    // (A line is a sequence of chars terminated by any combination of \r and \n.)
    BufferedReader br = new BufferedReader(isr);

    // The message data is contained in a single line, so read this line.
    String line = br.readLine();

    // Print host address and data received from it.
    System.out.println(
      "Received from " +
      request.getAddress().getHostAddress() +
      ": " +
      new String(line) );
  }
}
```

The server sits in an infinite loop listening for incoming UDP packets. When a packet comes in, the server simply sends the encapsulated data back to the client.

Packet Loss

UDP provides applications with an unreliable transport service, because messages may get lost in the network due to router queue overflows or other reasons. In contrast, TCP provides applications with a reliable transport service and takes care of any lost packets by retransmitting them until they are successfully received. Applications using UDP for communication must therefore implement any reliability they need separately in the application level (each application can implement a different policy, according to its specific needs).

Because packet loss is rare or even non-existent in typical campus networks, the server in this lab injects artificial loss to simulate the effects of network packet loss. The server has a parameter LOSS_RATE that determines which percentage of packets should be lost.

The server also has another parameter AVERAGE_DELAY that is used to simulate transmission delay from sending a packet across the Internet. You should set AVERAGE_DELAY to a positive value when testing your client and server on the same machine, or when machines are close by on the network. You can set AVERAGE_DELAY to 0 to find out the true round trip times of your packets.

Compiling and Running Server

To compile the server, do the following:

```
javac PingServer.java
```

To run the server, do the following:

```
java PingServer port
```

where port is the port number the server listens on. Remember that you have to pick a port number greater than 1024, because only processes running with root (administrator) privilege can bind to ports less than 1024.

Note: if you get a class not found error when running the above command, then you may need to tell Java to look in the current directory in order to resolve class references. In this case, the commands will be as follows:

```
java -classpath . PingServer port
```

Your Job: The Client

You should write the client so that it sends 10 ping requests to the server, separated by approximately one second. Each message contains a payload of data that includes the keyword PING, a sequence number, and a timestamp. After sending each packet, the client waits up to one second to receive a reply. If one second goes by without a reply from the server, then the client assumes that its packet or the server's reply packet has been lost in the network.

Hint: Cut and paste PingServer, rename the code PingClient, and then modify the code.

You should write the client so that it starts with the following command:

```
java PingClient host port
```

where host is the name of the computer the server is running on and port is the port number it is listening to. Note that you can run the client and server either on different machines or on the same machine.

The client should send 10 pings to the server. Because UDP is an unreliable protocol, some of the packets sent to the server may be lost, or some of the packets sent from server to client may be lost. For this reason, the client can not wait indefinitely for a reply to a ping message. You should have the client wait up to one second for a reply; if no reply is received, then the client should assume that the packet was lost during transmission across the network. You will need to research the API for DatagramSocket to find out how to set the timeout value on a datagram socket.

When developing your code, you should run the ping server on your machine, and test your client by sending packets to localhost (or, 127.0.0.1). After you have fully debugged your code, you should see how your application communicates across the network with a ping server run by another member of the class.

Message Format

The ping messages in this lab are formatted in a simple way. Each message contains a sequence of characters terminated by a carriage return character (r) and a line feed character (n). The message contains the following string:

PING sequence_number time CRLF

where sequence_number starts at 0 and progresses to 9 for each successive ping message sent by the client, time is the time when the client sent the message, and CRLF represent the carriage return and line feed characters that terminate the line.

Option 2: UDP Pinger Lab Using Python

You can implement your ping client using Python. You should write a simple Python program that follow a step-by-step process to establish a UDP connection with a server through sockets, and send out ping requests. The other requirements are very similar to those by choosing Java. One interesting point to mention is that, that you can use one programming to write your server program, and use a different language to write your client program.

3.2 Nmap 101 [10 points]

Network Mapped (Nmap, https://nmap.org/) is a network scanning and host detection tool that is very useful during several steps of security testing. Nmap is not limited to merely gathering information and enumeration, but it is also powerful utility that can be used as a vulnerability detector or a security scanner. Nmap is a multipurpose tool, and can be run on many different operating systems including Windows, Linux, BSD, and Mac. More specifically, it can be used to:

- Detect the live host on the network (host discovery)
- Detect the open ports on the host (port discovery or enumeration)
- Detect the software and the version to the respective port (service discovery)
- Detect the operating system, hardware address, and the software version
- Detect the vulnerability and security holes (Nmap scripts)

Nmap is a very common tool, and it is available for both the command line interface and the graphical user interface.

For this assignment, you are required to download and use Nmap to scan two hosts: 1) your own computer; and 2) remote website located at http://gaia.cs.umass.edu/. Write a report with the following components.

1) Scan the hosts and report the Operating System (including version), the server application if any (including version), the IP address of the target host, DNS server of the target domain, open ports and services (if available) running on these ports.
2) List any possible vulnerabilities related to the services identified in the previous step.

3) Show the output of the traceroute program (executed within Nmap) to access http://gaia.cs.umass.edu/.
4) Explore additional features of the tool and demonstrate any findings you think useful.
5) Based on your experience, how will you evaluate the tool? In what aspects could the tool be improved?

Please attach necessary screenshots for all answers above.