Nova Southeastern University
Graduate School of Computer and Information Sciences

Assignment 2
CISC 650 Computer Networks
Fall 2013

Note: Please include your name and the "Certification of Authorship" form in EVERY document (except for the programming assignment) that you submit. Thanks.

Part 1. Text reading
Chapter 3, Chapter 4

Part 2. Textbook questions

Chapter 3. [40 points]

3.1 Describe why an application developer might choose to run over TCP rather than UDP.

Reliable data transfer
Elastic throughput
Not time sensitive

3.2 Suppose host A is sending host B a large file over a TCP connection. If the acknowledge number for a segment of this connection is *y*, then the acknowledge number for the subsequent segment will necessarily be *y+ 1*. Is this true or false? Why?

False

Note: The answer is true only when 1) the data field of the segment in response to the first one has a length of 1; 2) the cumulative acknowledgement mechanism does not come into play; and 3) there are no retransmitted packets (either from A or from B) before the subsequent segment is sent. The example shown in the textbook is just a special case.

3.3 Suppose 5 TCP connections are present over some bottleneck link of rate X bps. All connections have a huge file to send (in the same direction over the bottleneck link). The transmissions of the files start at the same time. What is the transmission rate that TCP would like to give to each of the connections?

X/5

3.4 How to identify a UDP socket? How to identify a TCP socket? Are these data fields same? Why?

UDP: dest IP address, dest port number
TCP: source IP address; source port number; dest IP address; and dest port number
TCP needs acknowledge and retransmission

3.5 UDP and TCP use 1's complement for their checksums. Suppose you have the following three 8-bit words: 01010100, 01111000, 11001100. What is the 1's complement of the sum of these words? Show all work. Why UDP takes the 1's complement of the sum, that is, why not just use the sum?

```
  0 1 0 1 0 1 0 0
+ 0 1 1 1 1 0 0 0
  1 1 0 0 1 1 0 0

  1 1 0 0 1 1 0 0
+ 1 1 0 0 1 1 0 0
  1 0 0 1 1 0 0 1
```

(the extra 1 is wrapped around)

One's complement = 0 1 1 0 0 1 1 0.

The major advantage of using one's complement is that the sender and the receiver can use the same algorithm for checksum generation and error-checking. This reduces the cost of hardware/software products.

3.6 Suppose Client A initiates a Telnet session with server S. Provide possible source and destination port numbers for:

a. The segment sent from S to A.
b. The segment sent from A to S.

a) S to A: 23 1234
b) A to S: 1234 23

For a), the destination port number could be any number from 1024 to 65535. Answers to b) are similar except that they should match the answers for a).

3.7 Compare two pipelining protocols shown in the textbook – go-back-N and selective repeat.

See textbook.

3.8 In our textbook, protocol rdt 3.0 shows a data transfer protocols that uses only acknowledges. As an alternative, consider a reliable data transfer protocol that uses

negative acknowledgements. Suppose the sender sends data only infrequently. Will a NAK-only protocol be preferable to protocol that uses ACKs? Why? Suppose the sender has a lot of data to send and the end-to-end connection experiences few losses. In the second case, would a NAK-only protocol be preferable to a protocol that uses ACKs? Why?

In a NAK only protocol, the loss of packet *x* is only detected by the receiver when packet *x+1* is received. That is, the receivers receives *x-1* and then *x+1,* only when *x+1 is* received does the receiver realize that *x* was missed. If there is a long delay between the transmission of x and the transmission of *x+1,* then it will be a long time until *x* can be recovered, under a NAK only protocol.

On the other hand, if data is being sent often, then recovery under a NAK-only scheme could happen quickly. Moreover, if errors are infrequent, then NAKs are only occasionally sent (when needed), and ACK are never sent – a significant reduction in feedback in the NAK-only case over the ACK-only case.

3.9 Let us assume that the roundtrip delay between sender and receiver is constant and known to the sender. Would a timer still be necessary in protocol rdt 3.0, assuming that packets can be lost? Please explain.

A timer would still be necessary in the protocol rdt 3.0. If the round trip time is known then the only advantage will be that, the sender knows for sure that either the packet or the ACK (or NACK) for the packet has been lost, as compared to the real scenario, where the ACK (or NACK) might still be on the way to the sender, after the timer expires. However, to detect the loss, for each packet, a timer of constant duration will still be necessary at the sender.

3.10 Briefly discuss the basic mechanisms adopted by TCP congestion control.

Additive increase, multiplicative decrease
Slow start
Congestion avoidance
Fast retransmission

Chapter 4 [40 points]

4.1 Describe two major network-layer functions in a datagram network.

Forwarding and routing. Forwarding is about moving a packet from a router's input link to the appropriate output link. Routing is about determining the end-to-routes between sources and destinations.

4.2 Describe how packet loss can occur at input and outputs of a router. Is it possible to eliminate packet loss at these ports? If so, how? If not, please explain.

Packet loss occurs if queue size at the input port grows large because of slow switching fabric speed and thus exhausting router's buffer space. It can be eliminated if the switching fabric speed is at least ***n*** times as fast as the input line speed, where ***n*** is the number of input ports.

Packet loss can occur if the queue size at the output port grows large because of slow outgoing line-speed.

4.3 Suppose an application generates chunks of 160 bytes of data every 20 msec, and each chunk gets encapsulated in a TCP segment and then an IP datagram. What percentage of each datagram will be overhead, and what percentage will be application data?

Length of header fields: 20 (TCP header) + 20 (IP header) = 40 bytes
Length of data fields: 160 bytes
Length of the datagram: 40 + 160 = 200 bytes

Overhead: 40/200 = 20%
Application data: 160/200 = 80%

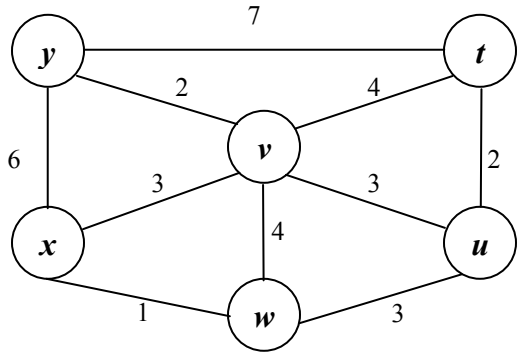
4.4 Consider a datagram network using 8-bit host addresses. Suppose a router uses longest prefix matching and has the following forwarding table:

Prefix Match	**Interface**
11	0
110	1
otherwise	2

For each of the 3 interfaces, give the associated range of destination host addresses and the number of addresses in the range.

Destination Address Range	**Link Interface**
11100000 through 11111111 (32 addresses)	0
11000000 through 11011111 (32 addresses)	1
00000000 through 10111111 (192 addresses)	2

4.5 Consider the following network. With the indicated link costs, use Dijkstra's shortest-path algorithm to computer the shortest path from *x* to all network nodes. Show how the algorithm works by computing a table similar to the textbook example. In cases when several candidate nodes have the same minimal costs, choose a node according to its alphabetical order.



Step	*N'*	*D(t),p(t)*	*D(u),p(u)*	*D(v),p(v)*	*D(w),p(w)*	*D(y),p(y)*
0	x	∞	∞	3,x	1,x	6,x
1	xw	∞	4,w	3,x	1,x	6,x
2	xwv	7,v	4,w	3,x	1,x	5,x
3	xwvu	6,u	4,w	3,x	1,x	5,x
4	xwvuy	6,u	4,w	3,x	1,x	5,x
5	xwvuyt	6,u	4,w	3,x	1,x	5,x

4.6 Consider the count-to-infinity problem in the distance vector routing. Will the problem occur if we decrease the cost of a link? Why?

NO, this is because that decreasing link cost won't cause a loop (caused by the next-hop relation of between two nodes of that link). With the distance vector algorithm one major advantage is that "good news travels fast" - the algorithm will converge quickly when the cost of a link is decreased.

4.7 IPv6 adopts a fixed-length 40 byte IP header. What is the major advantage of this approach compared to that in IPv4?

Fast processing, less overhead

4.8 Suppose an ISP owns the block of addresses of the form 200.200.128.0/19. Suppose it wants to create four subnets from this block, with each block having the same number of IP addresses. What are the prefixes (of form a.b.c.d/x) for the four subnets?

Four equal size subnets: 200.200.128.0/21, 200.200.136.0/21, 200.200.144.0/21, 200.200.152.0/21

4.9 Why are different inter-AS and intra-AS protocols used in the Internet?

Refer to the textbook for details.

Part 3. Practical assignment [20 points]

Solutions for Java Mail User Agent: Simplified Version

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

public class EmailAgent
{
   public static void main(String[] args) throws Exception
   {
      // Establish a TCP connection with the mail server.
      Socket socket = new Socket("smtp.nova.edu", 25);

      // Create a BufferedReader to read a line at a time.
      InputStream is = socket.getInputStream();
      InputStreamReader isr = new InputStreamReader(is);
      BufferedReader br = new BufferedReader(isr);

      // Read greeting from the server.
      String response = br.readLine();
      System.out.println(response);
      if (!response.startsWith("220")) {
         throw new Exception("220 reply not received from server.");
      }

      // Get a reference to the socket's output stream.
      OutputStream os = socket.getOutputStream();

      // Send HELO command and get server response.
      String command = "HELO x\r\n";
      System.out.print(command);
      os.write(command.getBytes("US-ASCII"));
      response = br.readLine();
      System.out.println(response);
      if (!response.startsWith("250")) {
         throw new Exception("250 reply not received from server.");
      }

      // Send MAIL FROM command.
      command = "MAIL FROM: lwei@nova.edu\r\n";
      System.out.print(command);
      os.write(command.getBytes("US-ASCII"));
      response = br.readLine();
      System.out.println(response);
```

```
      if (!response.startsWith("250")) {
         throw new Exception("250 reply not received from server.");
      }

      // Send RCPT TO command.
      command = "RCPT TO: lwei@nova.edu\r\n";
      System.out.print(command);
      os.write(command.getBytes("US-ASCII"));
      response = br.readLine();
      System.out.println(response);
      if (!response.startsWith("250")) {
         throw new Exception("250 reply not received from server.");
      }

      // Send DATA command.
      command = "DATA\r\n";
      System.out.print(command);
      os.write(command.getBytes("US-ASCII"));
      response = br.readLine();
      System.out.println(response);
      if (!response.startsWith("354")) {
         throw new Exception("354 reply not received from server.");
      }

      // Send message data.
      os.write("SUBJECT: test message\r\n\r\n".getBytes("US-ASCII"));
      os.write("Hi there,\r\n".getBytes("US-ASCII"));
      os.write("\r\n".getBytes("US-ASCII"));
      os.write("This lab is too hard.\r\n".getBytes("US-ASCII"));

      // End with line with a single period.
      os.write(".\r\n".getBytes("US-ASCII"));
      response = br.readLine();
      System.out.println(response);
      if (!response.startsWith("250")) {
         throw new Exception("250 reply not received from server.");
      }

      // Send QUIT command.
      command = "QUIT\r\n";
      System.out.print(command);
      os.write(command.getBytes("US-ASCII"));
      response = br.readLine();
      System.out.println(response);
   }
}
```

Solutions for Python Mail User Agent

```
from socket import *

# Message to send
msg = '\r\nI love computer networks!'
```

```
endmsg = '\r\n.\r\n'

# Choose a mail server (e.g. Google mail server) and call it mailserver
mailserver = 'smtp.gmail.com'

# Create socket called clientSocket and establish a TCP connection with
mailserver
clientSocket = socket(AF_INET, SOCK_STREAM)

# Port number may change according to the mail server
clientSocket.connect((mailserver, 587))
recv = clientSocket.recv(1024)
print recv
if recv[:3] != '220':
      print '220 reply not received from server.'

# Send HELO command and print server response.
heloCommand = 'HELO gmail.com\r\n'
clientSocket.send(heloCommand)
recv1 = clientSocket.recv(1024)
print recv1
if recv1[:3] != '250':
      print '250 reply not received from server.'

# Send MAIL FROM command and print server response.
mailfrom = 'MAIL FROM: <alice@gmail.com>\r\n'
clientSocket.send(mailfrom)
recv2 = clientSocket.recv(1024)
print recv2
if recv2[:3] != '250':
      print '250 reply not received from server.'


# Send RCPT TO command and print server response.
rcptto = 'RCPT TO: <bob@yahoo.com>\r\n'
clientSocket.send(rcptto)
recv3 = clientSocket.recv(1024)
print recv3
if recv3[:3] != '250':
      print '250 reply not received from server.'

# Send DATA command and print server response.
data = 'DATA\r\n'
clientSocket.send(data)
recv4 = clientSocket.recv(1024)
print recv4
if recv4[:3] != '354':
      print '354 reply not received from server.'

# Send message data.
clientSocket.send('SUBJECT: Greeting To you!\r\n')
clientSocket.send('test again')
clientSocket.send(msg)
```

```
# Message ends with a single period.
clientSocket.send(endmsg)
recv5 = clientSocket.recv(1024)
print recv5
if recv5[:3] != '250':
      print '250 reply not received from server.'

# Send QUIT command and get server response.
quitcommand = 'QUIT\r\n'
clientSocket.send(quitcommand)
recv6 = clientSocket.recv(1024)
print recv6
if recv6[:3] != '221':
      print '221 reply not received from server.'
```