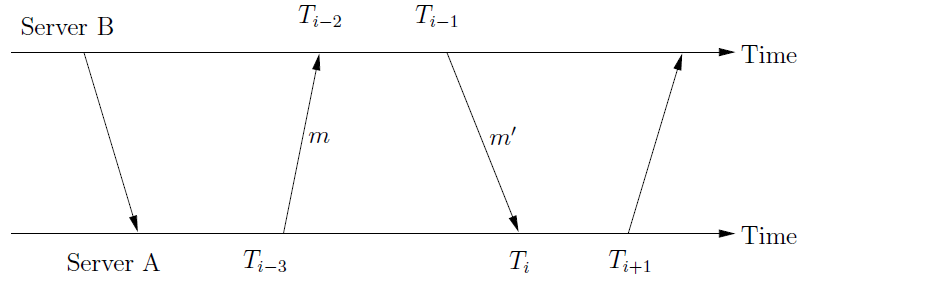
**1(1) This problem is pertaining to the NTP protocol. Explain the meaning of offset Oi between two NTP servers.**

The offset Oi is calculated between two NTP servers when they exchange a pair of messages. It is calculated for each pair of messages. The offset Oi is based on the actual offset (difference) between the clocks on the two NTP servers and the transmission delay for the two messages.



For example from the above figure where NTP server (A) want to synchronize the clock relative to NTP server (B)’s clock.

If A’s clock is at 12.0

If B’s clock is at 12.1

The actual offset (O) between the clocks is 0.1

The two message s m and m` between servers have t and t` transmission time. The total transmission delay (di) is t+t`.

Message between A and B is m, where A’s clock is Ti-3 and B’s clock is Ti-2.

Message between B and A is m`, where B’s clock is at Ti-1 and A’s clock is at Ti

Message m’s offset value O=Ti-2-Ti-3-t

Message m` offset value O=Ti-1-Ti+t`

From the above equations t+t’=Ti-2-Ti-3+Ti-Ti-1, where the offset value gets cancelled which is very negligible.

O=oi+ (t’-t)/2 where Oi= Ti-2-Ti-3+Ti-Ti-1/2 assuming t, t`>=0

di=t+t`

From the above two equations, it shows Oi is based on O and di.

Therefore solving above two equations, it concludes to O =Oi+di/2-t and O=Oi-di/2+t`

**1) (2) Why the values of t and t′ cannot be measured? Why we can obtain**

**the value of t + t′ accurately?**

From the above question I got the two equations

t=Ti-2-Ti-3-o t’=Ti-Ti-1+o

t and t’ depends on servers local times and the offset value. The offset cannot be accurate all the time, so it is difficult to calculate t and t’ individually. If I add the above equation, the offset o gets cancelled and t+t’ depends on local clock values of the servers.

**1) (3) The propagation delay di plays a significant role in the accuracy of**

**NTP protocol. Explain using your own words why.**

The propagation delay is the measure of accuracy in NTP protocol. The servers are arranged in levels called strata. Each level synchronizes the clock with higher level servers. The servers with higher strata number are less accurate. Therefore the lower level servers synchronizes with primary servers which are directly connected to the UTC .Errors will be there at each level, so the propagation delay play a very important role. Propagation delay is considered in all algorithms where servers synchronizing their clocks with external clocks. Even the mathematical expressions prove that di is very important.

t+t’=Ti-2-Ti-3+Ti-Ti-1

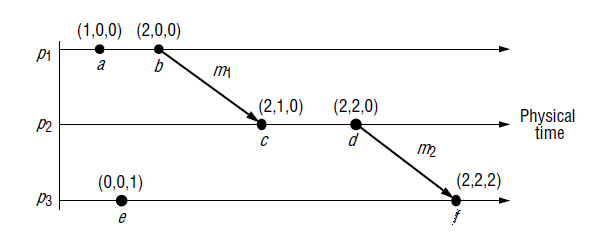
O =Oi+di/2-t and O=Oi-di/2+t`

From the above equations it shows that NTP protocol is completely dependent on offset and transmission delay. The offset is considered negligible. Therefore propagation delay plays a significant role in NTP protocol.

**2) (1) The problem is pertaining to the concepts of logical time and clocks.**

**It was stated in class that for the vector timestamps, property P3 (if e! = e′ then L (e)! = L (e′)) may not always be true. Do you agree with this statement? Please explain your conclusion.**

In vector timestamp, each process has a vector clock to timestamp the events. Initially each process vector is initialized to zero. Before timestamping the event the vector clock increases by one. Vi[j]=Vi[j]+1.If the process receive the message with timestamp t,it sets Vi[j]=max(Vi[j],t[j]),called the merge operation.



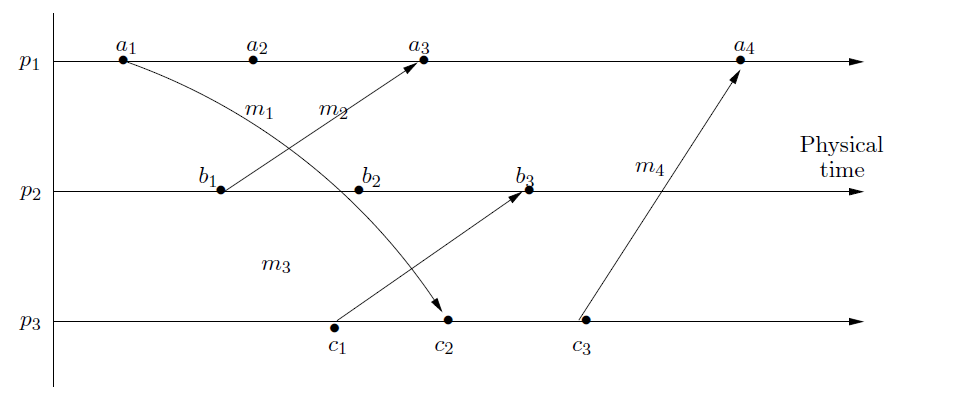
From the above diagram from class example. There are three processes. Initially each process vector clock is p1(0,0,0),p2(0,0,0),p3(0,0,0)

At the event ‘a’, before timestamping the event, it increments the p1 process clock in vector clock by one. Therefore event ‘a’ has vector clock (1, 0, 0).Event ‘b’ happens after a so the vector clock is (2, 0, 0).Event ‘c’ receives the message from ‘b’, which takes maximum of c’s timestamp and b’s vector value. Therefore c has (2, 1, 0).D happens after c, so it is (2, 2, 0).Event ‘f’ receives the message from d (2, 2, 0).Therefore f’s value is (2, 2, 2).

Relating to the problem, consider event ‘e’ which is concurrent with respect to a, b, c or d events. There is no relationship between e to other events. The vectors also cannot be compared. For example a (1, 0, 0) and e (0, 0, 1), from this no relationship can be made. Vectors cannot be compared with respect to e and other events. Even between b and therefore if e! = e′ then V (e)! = V (e) is always true in vector timestamps.

**2) (2) For the diagram shown in Fig. 1, assume that each process starts with a**

**vector (0, 0, 0). Process p1 always increases its component by 1, process p2 always increases its component by 2, and process p3 always increases its component by 10. Write down the complete vector timestamps for all the events in the diagram.**



Process p1 increase its component by 1

Process p2 increase its component by 2

Process p3 increase its component by 10

Complete vector timestamp for all the events in the diagram are

a1(1,0,0) a2(2,0,0) a3(3,2,0) a4(4,2,30)

b1 (0, 2, 0) b2 (0, 4, 0) b3 (0, 6, 10)

c1 (0, 0, 10) c2 (1, 0, 20) c3 (1, 0, 30)

3) (1) Modify the Java UDP example given in Chapter 4, p.152-153 as follows.

(1) The client will iterate in a loop. During each iteration, it will get a character string from standard input and send it to the server. The server will also iterate in a loop. During each iteration it will receive a character string and send it back to the client.

For this program, I modified the client code which is specified in p152.I didn’t modify the server. Client enters hostname and data using standard input. The data entered by the user is split into words and sent to server individually and the server reply’s it back to client.

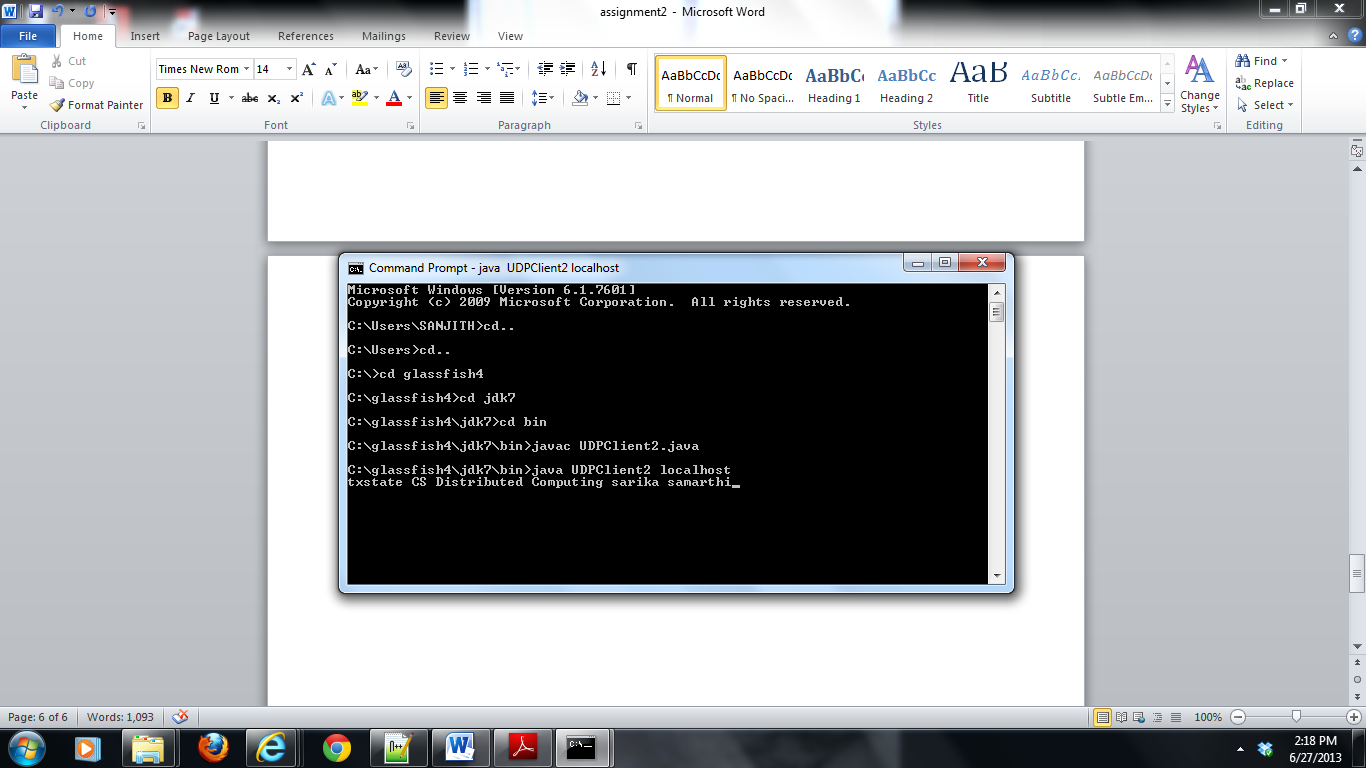
No modification to server program.

Here are the screen shots for the output.

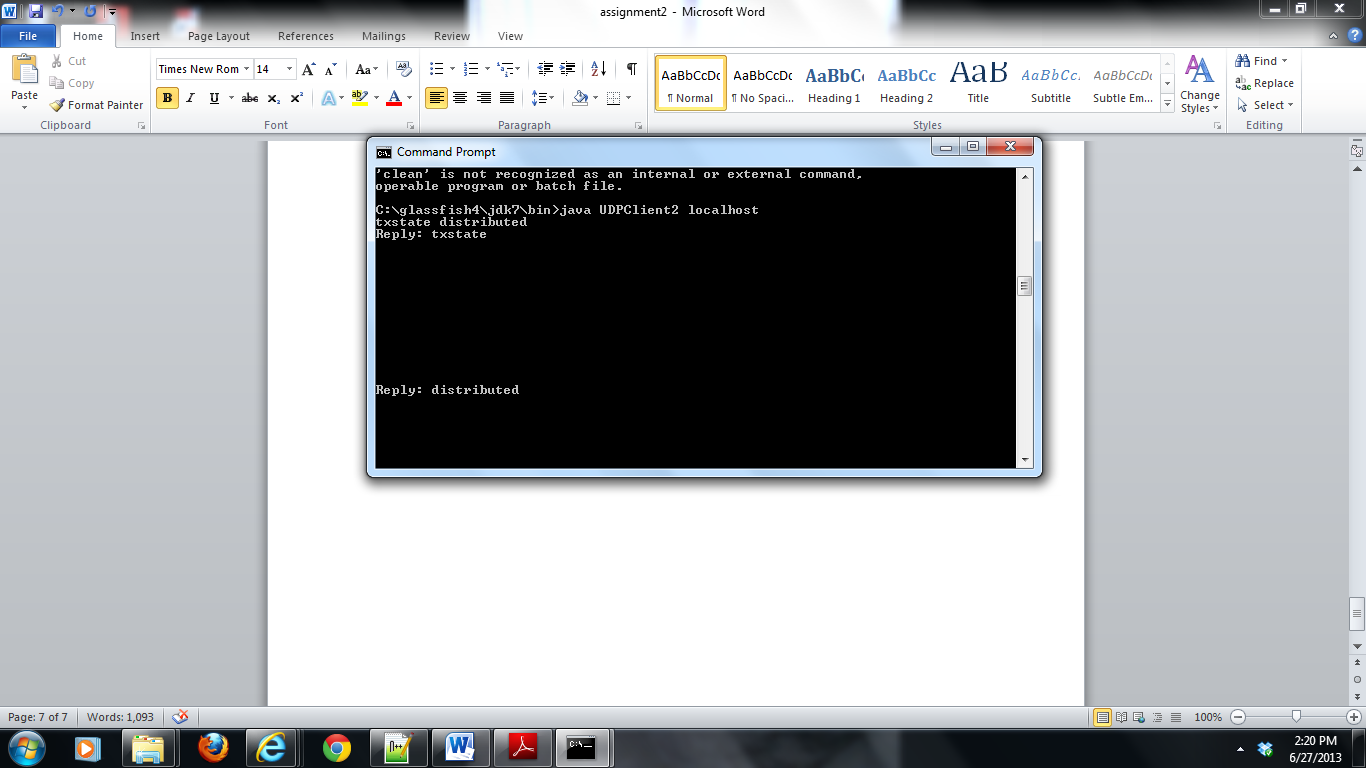
**Client: javac UDPClient2.java**

**Run: java UDPClient2 localhost**

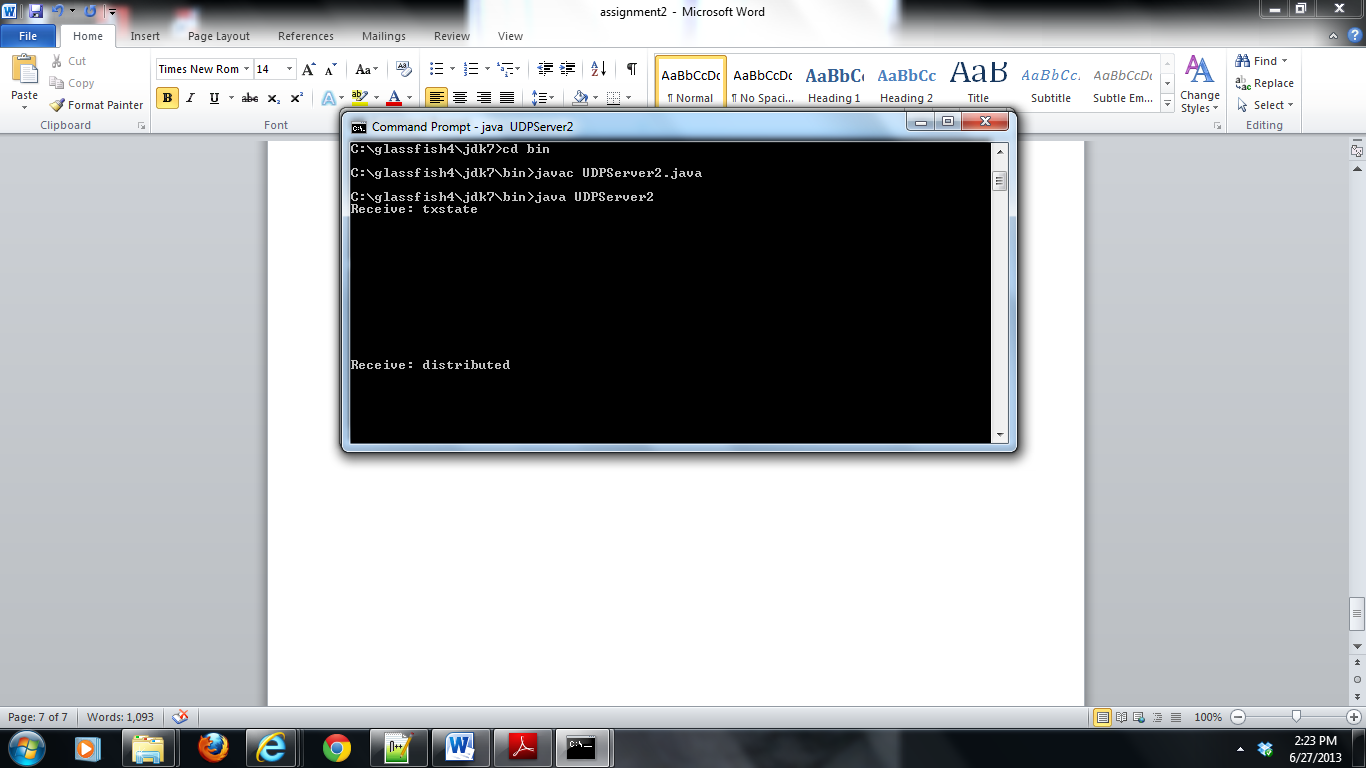
**Txstate distributed**



Client gets the reply in separate words,one after the other.



Server receives individual words and reply back to client.



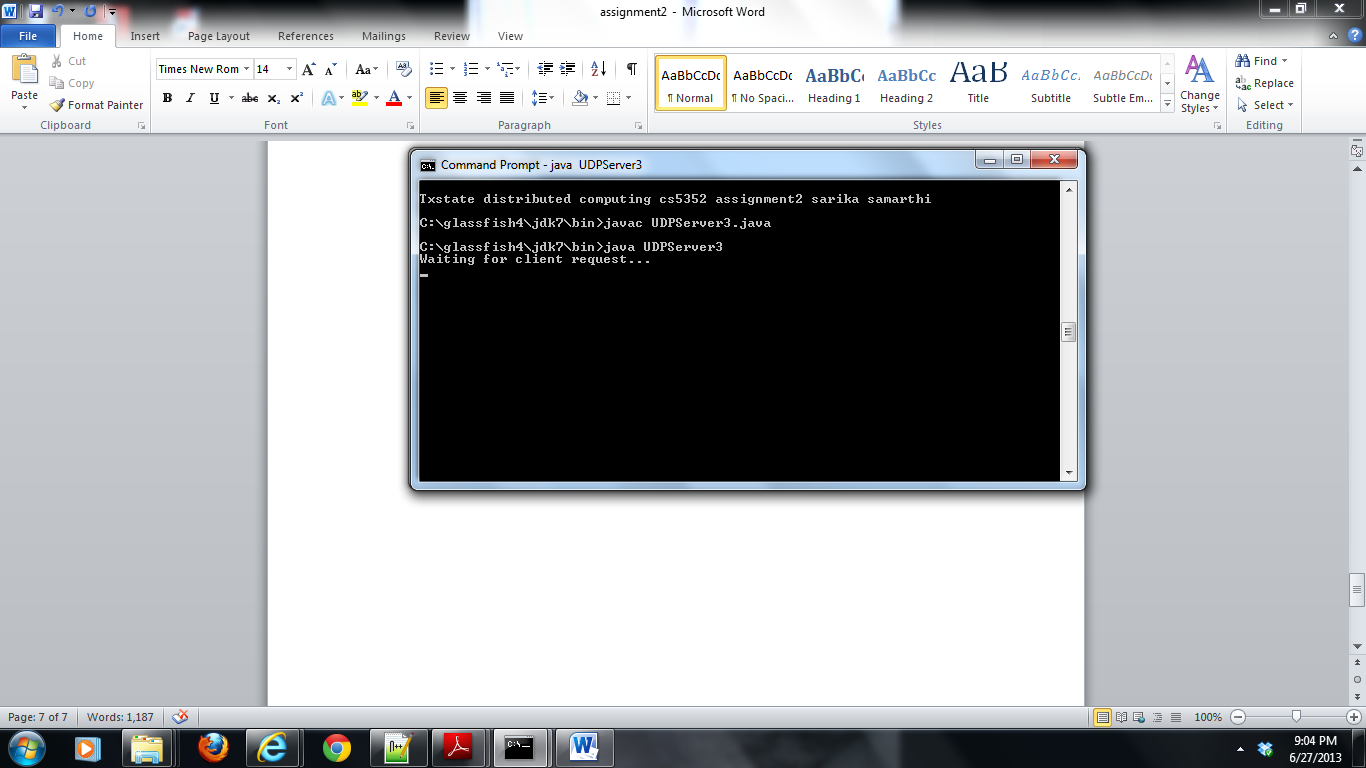
**3) (2) In this second modification, the client will still send a character string to the server. However, that character string is the name of a file on the server’s host. After receiving that file name, the server will try to open the file and send each line of the file back to the client. If the server cannot open the file, it will issue an error message back to the client. The client should just print each line of replies, plus the IP number of replying computer on its standard output.**

For the second program the example from page 152 and 153 where client remains the same and modification are done to the server is.Client sends the file name and server returns the file content to client.

With my program the text file is in the same directory as the server program,which I placed in java bin directory which has direct access to javac.exe and java.exe.

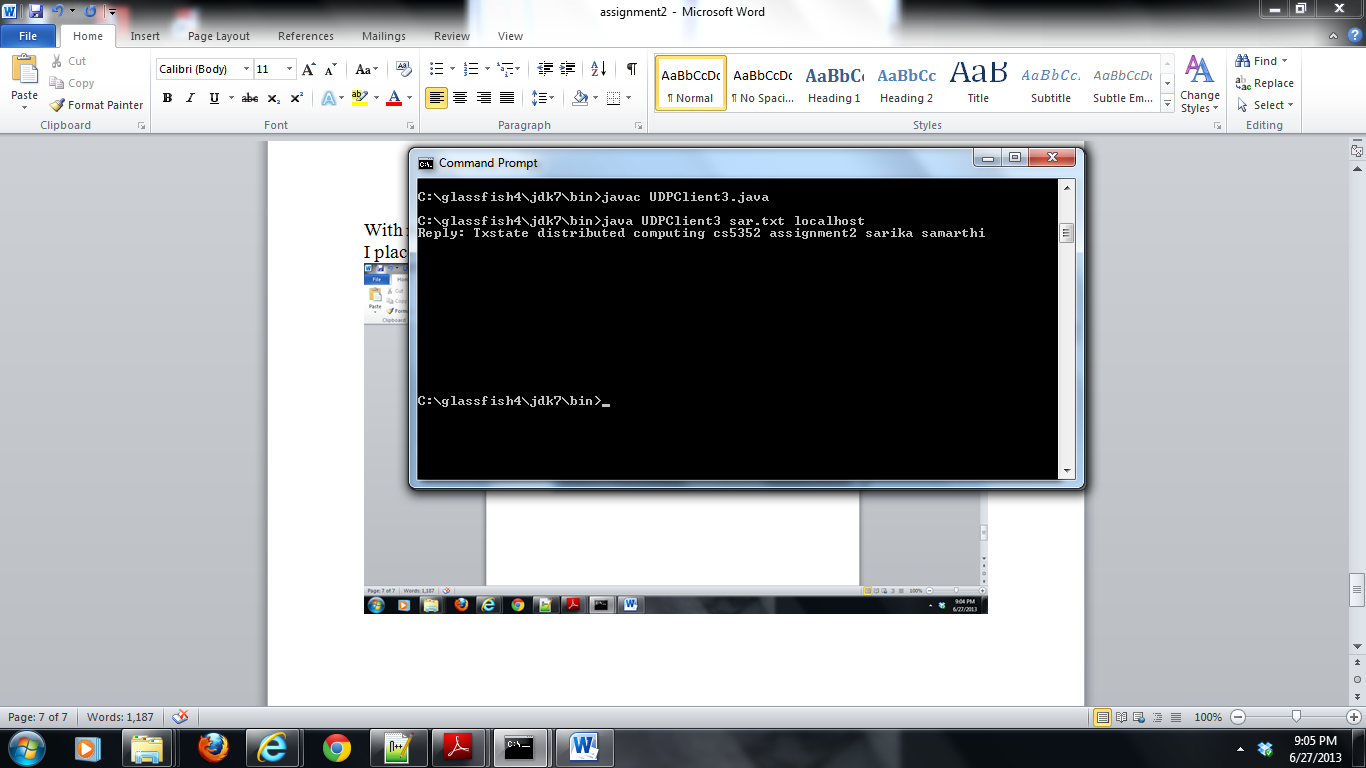
**Server: javac UDPServer3.java**

**RUN:java UDPServer3**

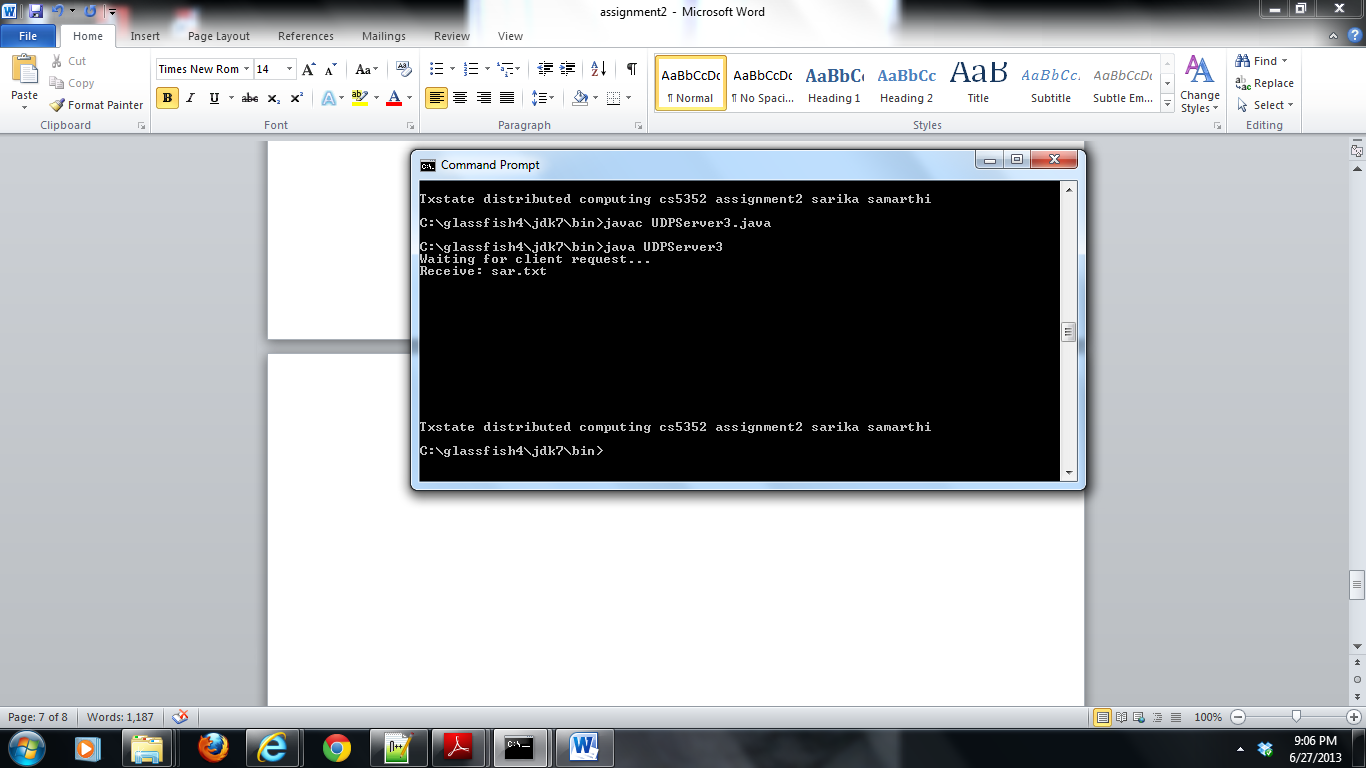


**Client: javac UDPClient3.java**

**RUN: java UDPClient3 sar.txt localhost**



**Server side output: It receives sar.txt file and prints the file content as well.**



**I uploaded all the files for question 3 programs.Each one has one client and one server.I also uploaded sar.txt file and also class files and Read me file.**