**Foundations of Computer Networks**

**Project 1**

Components of the project:

The project is mainly divided into 2 parts:

1. TCP (Transmission Control Protocol)
2. UDP (User Datagram Protocol)

The project consists of server, proxy server and client classes for each protocol and 2 helper classes to handle time an return/set the time in the required format and to calculate the roundtrip time. I have written the code in Java

Basic Flow of the Project:

The following figure shows the communication between different ports of the Client, Server and Proxy Server.

Server

TCP Port

UDP Port

Client

TCP Port

UDP Port

Proxy

Proxy to Server TCP Port

Proxy TCP Port

Proxy to Server UDP Port

Proxy UDP Port

Architecture of the Project

Message format:

The client needs to communicate with the Server in a specific format.

1. If the Client **wants to read** the time stored at the Server, he needs to send either a **blank string** or specify a format in which he wants the time.
   1. This format can either be –z specified when the client starts communicating.
   2. If the String is blank, the Server will return the time in a Calender Date format
2. If the client wants to modify the time at the server, he will need to do the following things:
   1. Send the user id and password along with the data he needs to format. This user id and password will be user to authenticate at the server and modify the time.
   2. The message format that I have used for my project is:

Modify(value):userid(useridofclient):password(passwordoftheclient)

Eg. modify10:useridssa2923:passwordpass1234

Here, 10 is the value the client wants to set, his userid is ssa2923 and password is pass1234

**If the user does not provide valid credentials, the server will return saying the client is not authorized to modify the value.**

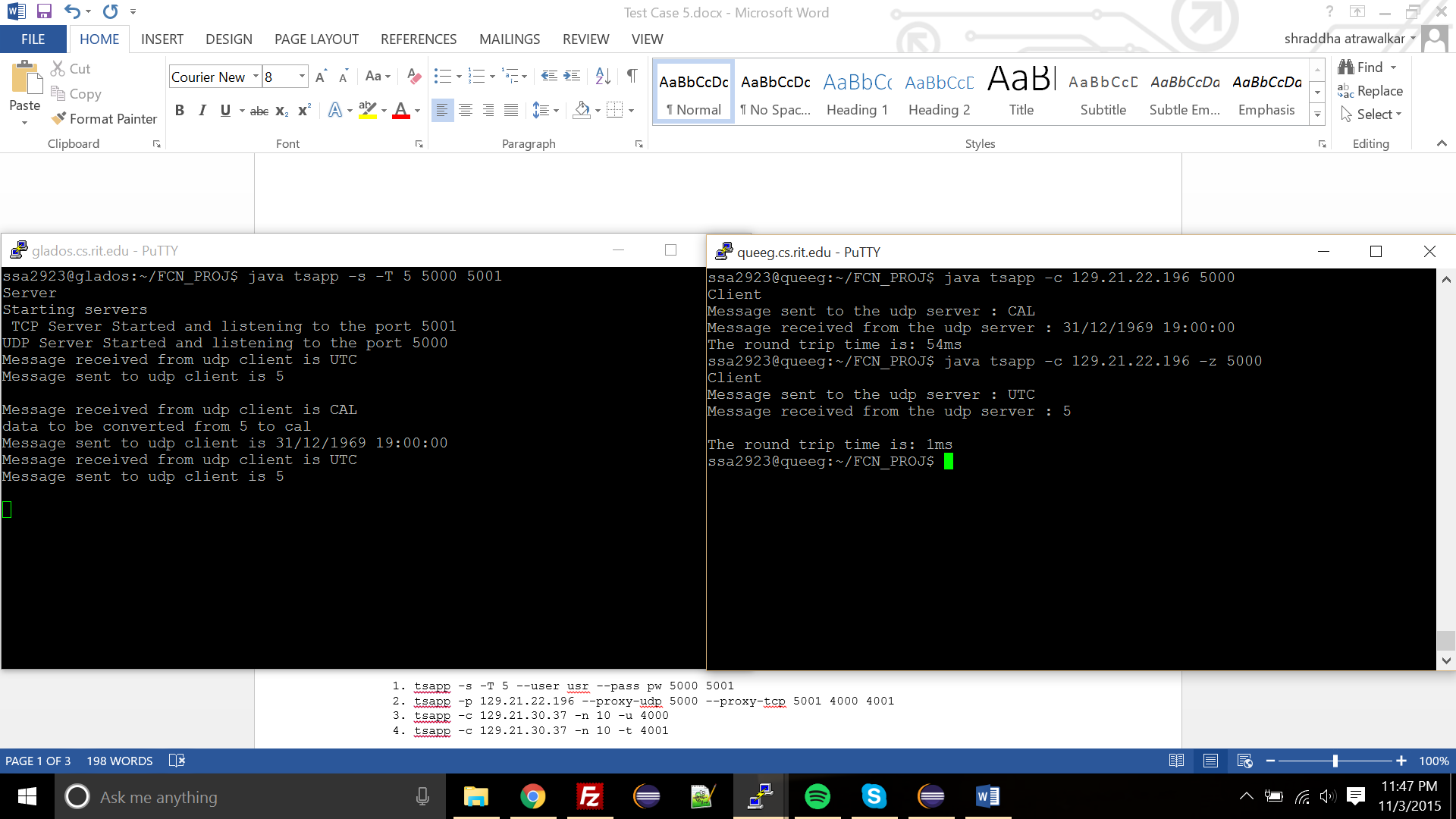
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Modify** | **Value:** | **Userid** | **Username:** | **Password** | **Password** |

Format of the message sent by the client to the server

Example Test Cases:

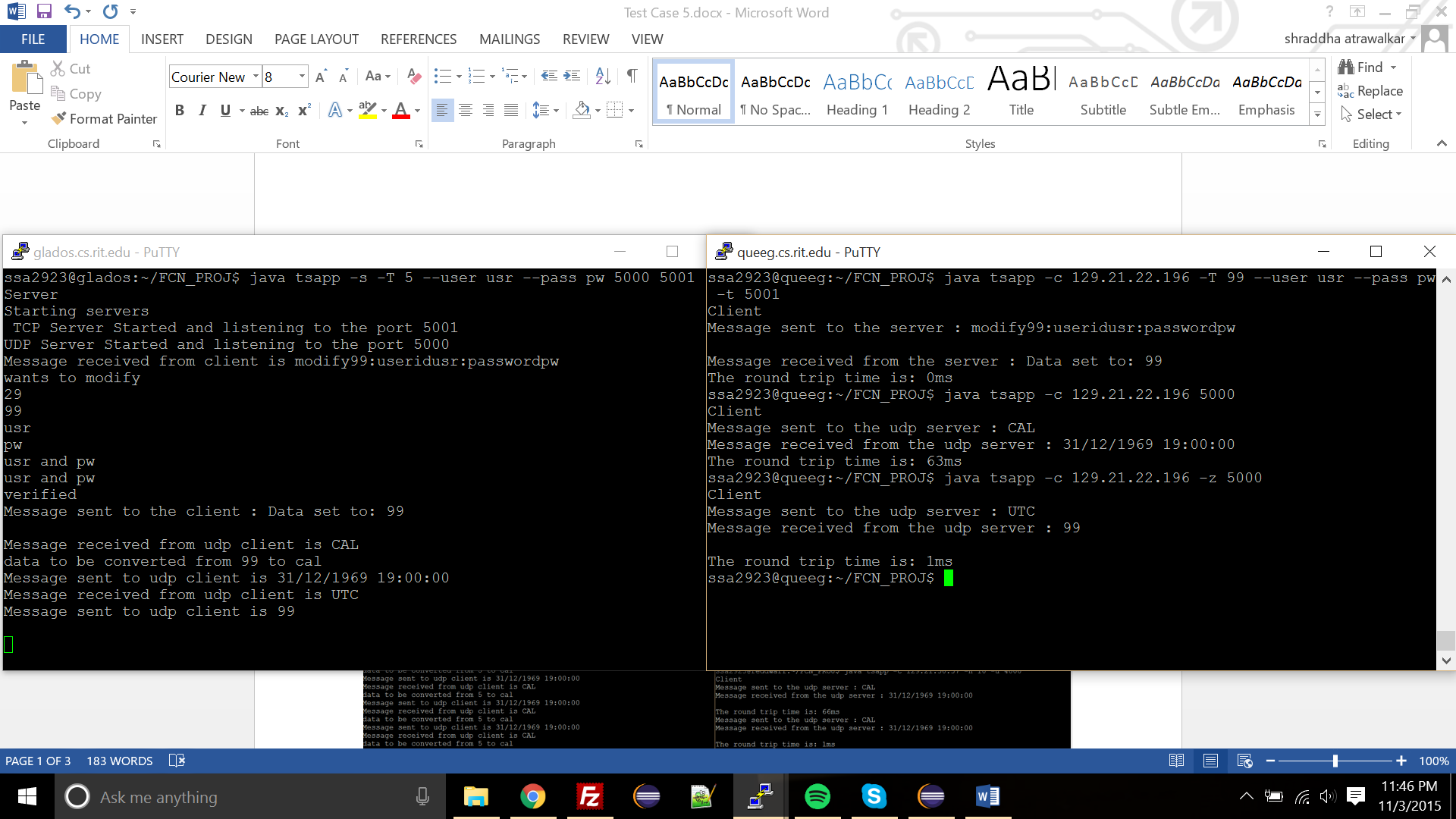
Test Case 1

 1. tsapp -s -T 5 5000 5001  
 2. tsapp -c 129.21.22.196 5000



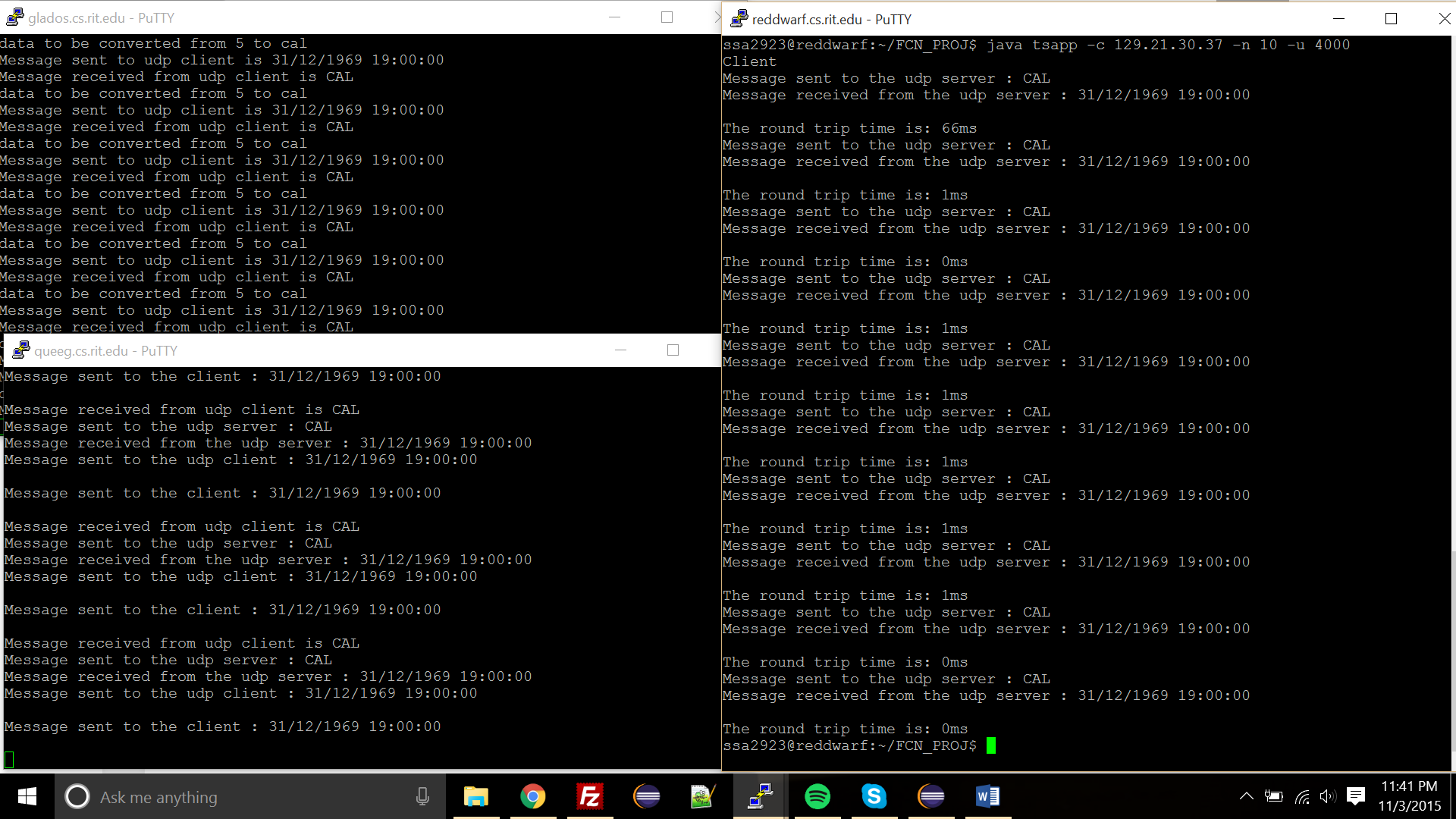
Test Case 2

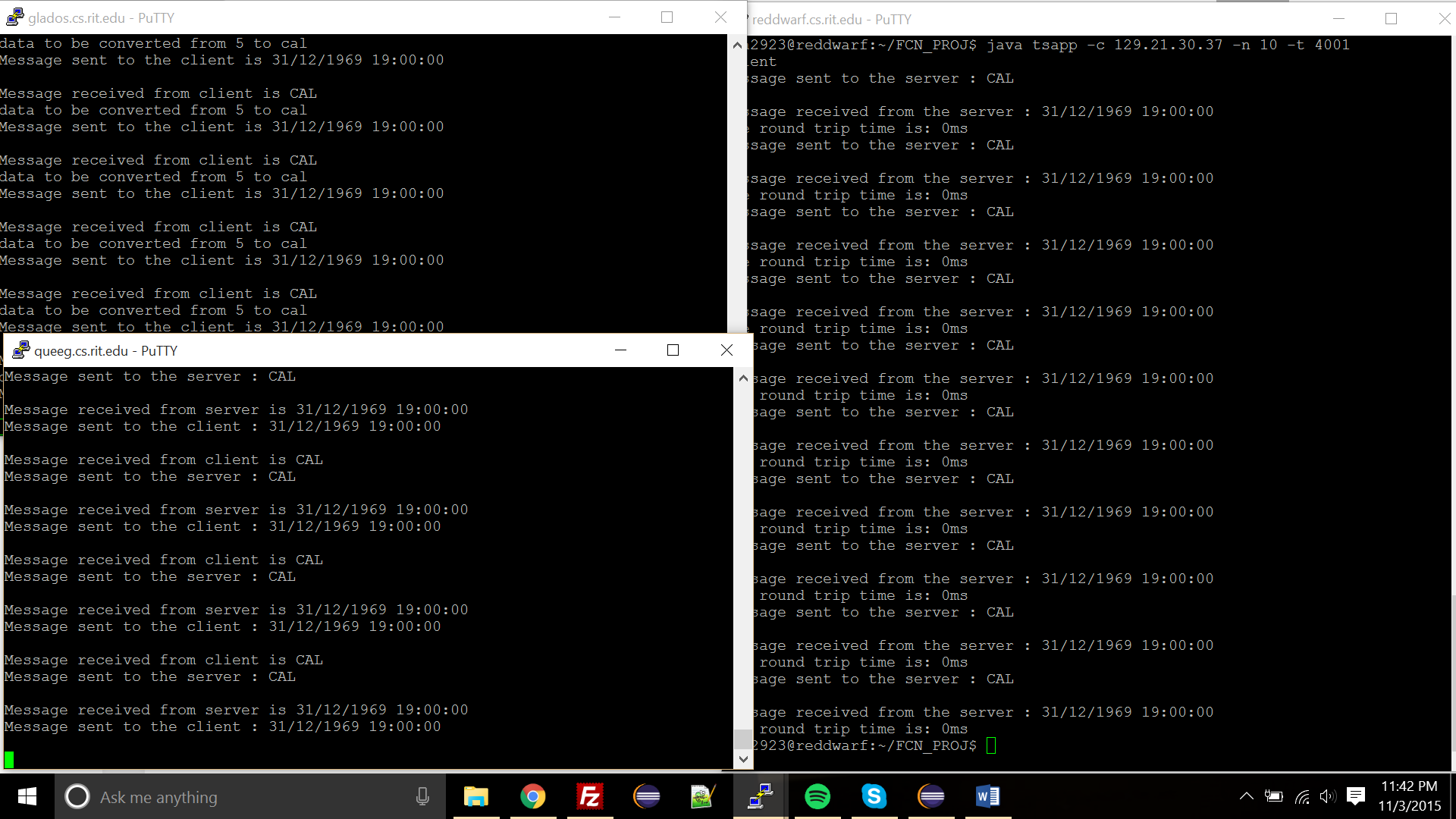
    1. tsapp -s -T 5 --user usr --pass pw 5000 5001  
    2. tsapp -c 129.21.22.196 -T 99 --user usr --pass pw -t 5001  
    3. tsapp -c 129.21.22.196 5000



Test Case 3

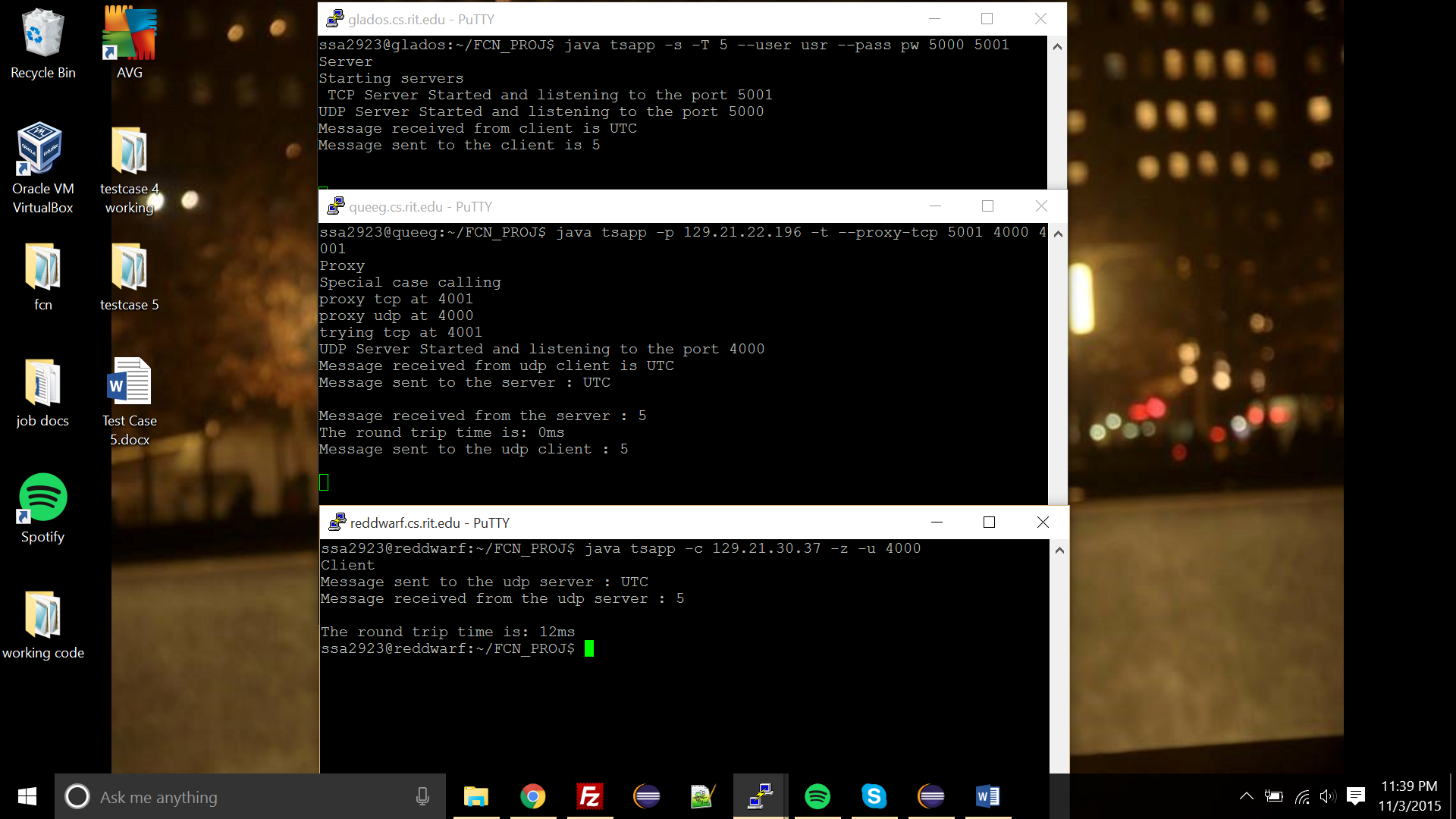
    1. tsapp -s -T 5 --user usr --pass pw 5000 5001  
    2. tsapp -p 129.21.22.196 --proxy-udp 5000 --proxy-tcp 5001 4000 4001  
    3. tsapp -c 129.21.30.37 -n 10 -u 4000  
    4. tsapp -c 129.21.30.37 -n 10 -t 4001





Test Case 4

1. tsapp -s -T 5 --user usr --pass pw 5000 5001
2. tsapp -p 129.21.22.196 -t --proxy-tcp 5001 4000 4001
3. tsapp -c 129.21.30.37 -z -u 4000



    Test Case 5

1. tsapp -s -T 5 --user usr --pass pw 5000 5001  
    2. tsapp -p 129.21.22.196 -t --proxy-tcp 5001 4000 4001  
    2. tsapp -p 129.21.30.37 -u --proxy-udp 4000 3000 3001  
    3. tsapp -c 129.21.37.16 -u 3000  
    4. tsapp -c 129.21.37.16 -n 250 -u 3000  
    4. tsapp -c 129.21.37.16 -T 99 --user usr --pass pw -t 3001  
    5. tsapp -c 129.21.37.16-n 25 -u 3000

