DCN704 – Collaborative Communications
Laboratory Report

Lab # 6 (5%): Instant Messaging and Presence Public Servers

Student Name: Fatjon Dauti

Background:

XMPP, the Extensible Messaging and Presence Protocol, is currently an Internet standard. Public XMPP servers are available everywhere in the world. They can be used for free and allow connections without paying any fees. As Cisco Jabber requires an active Cisco account, you can set an independent and decentralized service.
An XMPP service needs a network that is able to accept user's registration to specific channels. Each channel is able to accept messages from users in an open manner. Channels can be password protected or open.
Users can join a channel using a Jabber ID, a chat account, or by means of a web client through a secure TLS (Transport Layer Security) connection.

Objective:

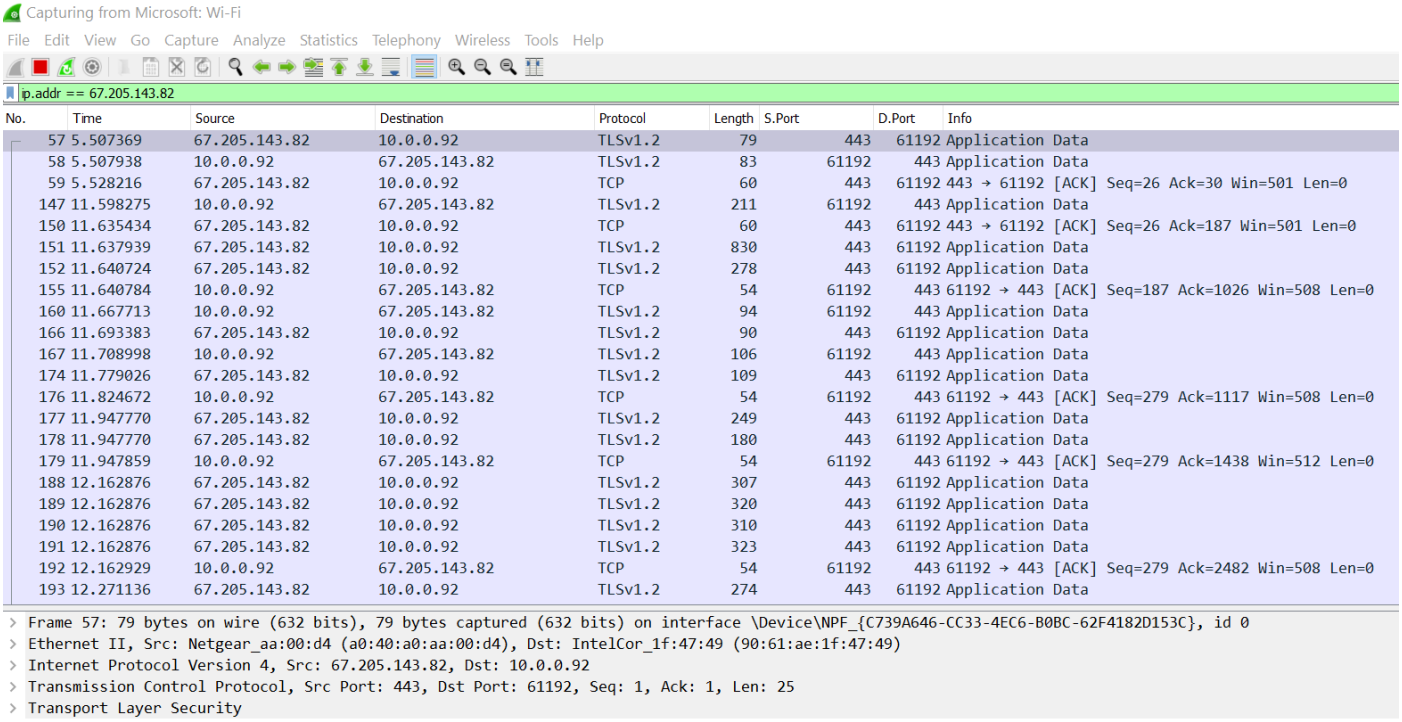
1. To use a public IRC (Internet Relay Chat) server to create an IMP (Instant Messaging and Presence) network and define channels that can be used by users to communicate.
2. Create different user accounts belonging to the same channel, through a web client, in order to communicate via the public chat.
3. Capture IRC traffic generated by the web clients and identify the chat sequences between the registered users, as well as the secured procedures running under TLS protocol.

Procedure:

Perform the following procedures and answer the questions as indicated.

1. **[0.5 marks]** Be sure to run the most recent version of Wireshark on your computer. Starting this step, you must capture every action you perform. Close all the other applications that can generate traffic over the network. Be sure that you only capture traffic pertaining to this activity. Insert the screenshot from Wireshark as you started capturing traffic.

   In the screenshot I'm filtering by the IP of "demo.thelounge.chat"



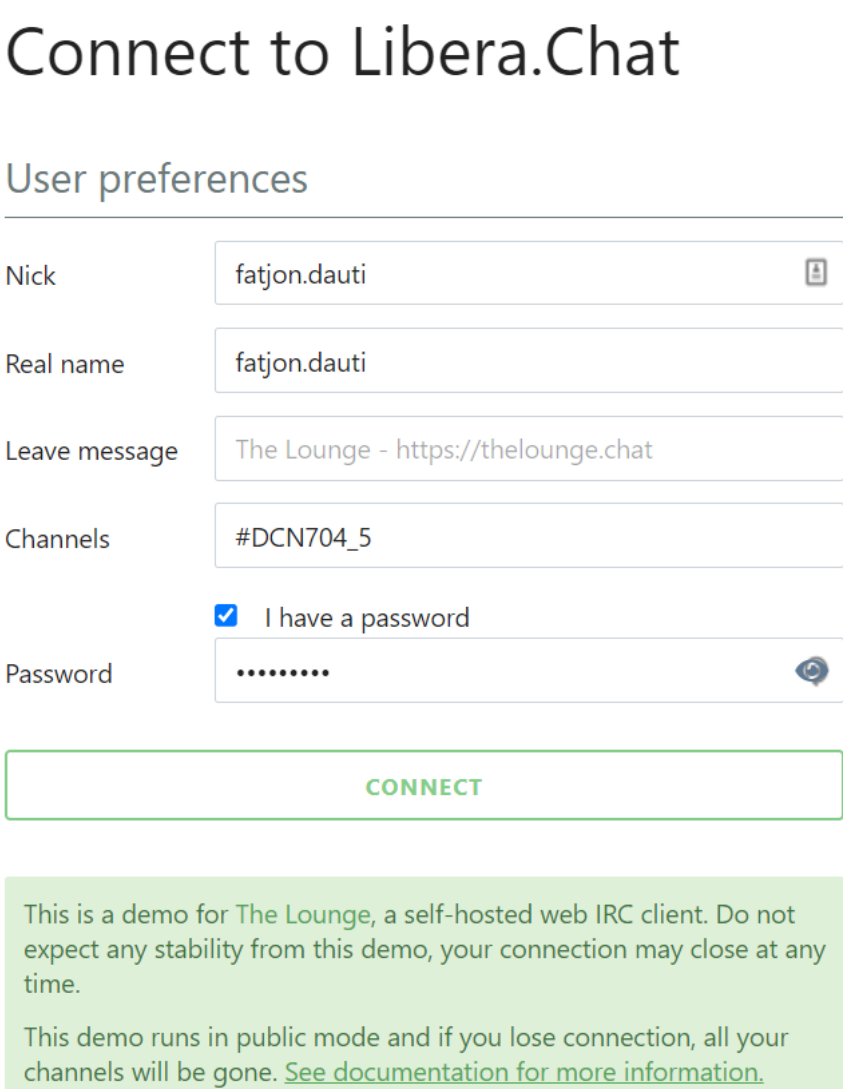
2. **[0.5 marks]** Now, you will use a **public IM Server**. You can use any public IM Server, or select https://www.koderoot.net/ which creates its own IMP network using the server irc.koderoot.net. The network name must be <Group#> .

   What transport layer port number is the server using? TCP port 6667

3. **[0.5 marks]** Create your own user profile; use your Seneca College username (firstname.lastname) as nickname, and username. Do not forget your password. Create at least one channel named **#DCN704_<Group#>.** Do not click on the **Connect** button.

   (It does not support "." in the nick name)

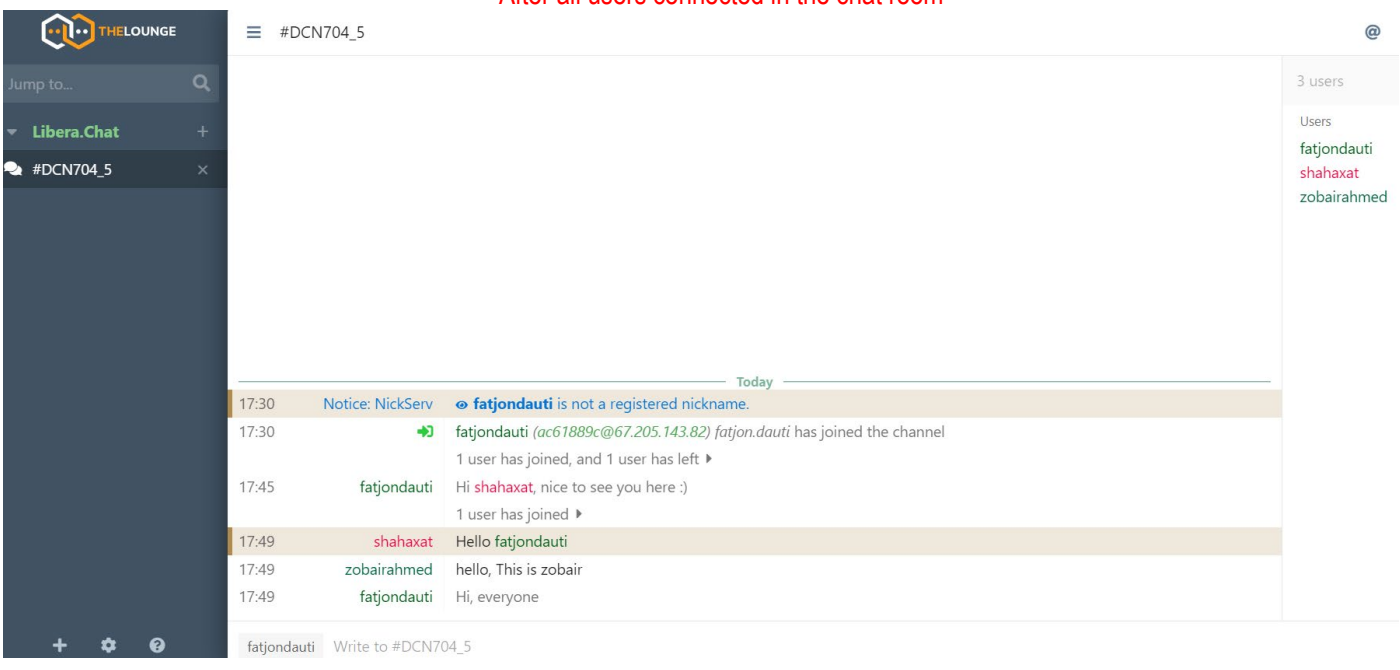
   Insert the screenshot of your network Settings and User Preferences here.



4. **[0.5 marks]** Now, click on the **Connect** button. Once you are connected, visit the (?) Help button and recognize all the possible keyboard shortcuts and commands available.

Insert the screenshot of "The Lounge" after everyone in the group create their own user. (replace the following screenshot with yours)

After all users connected in the chat room



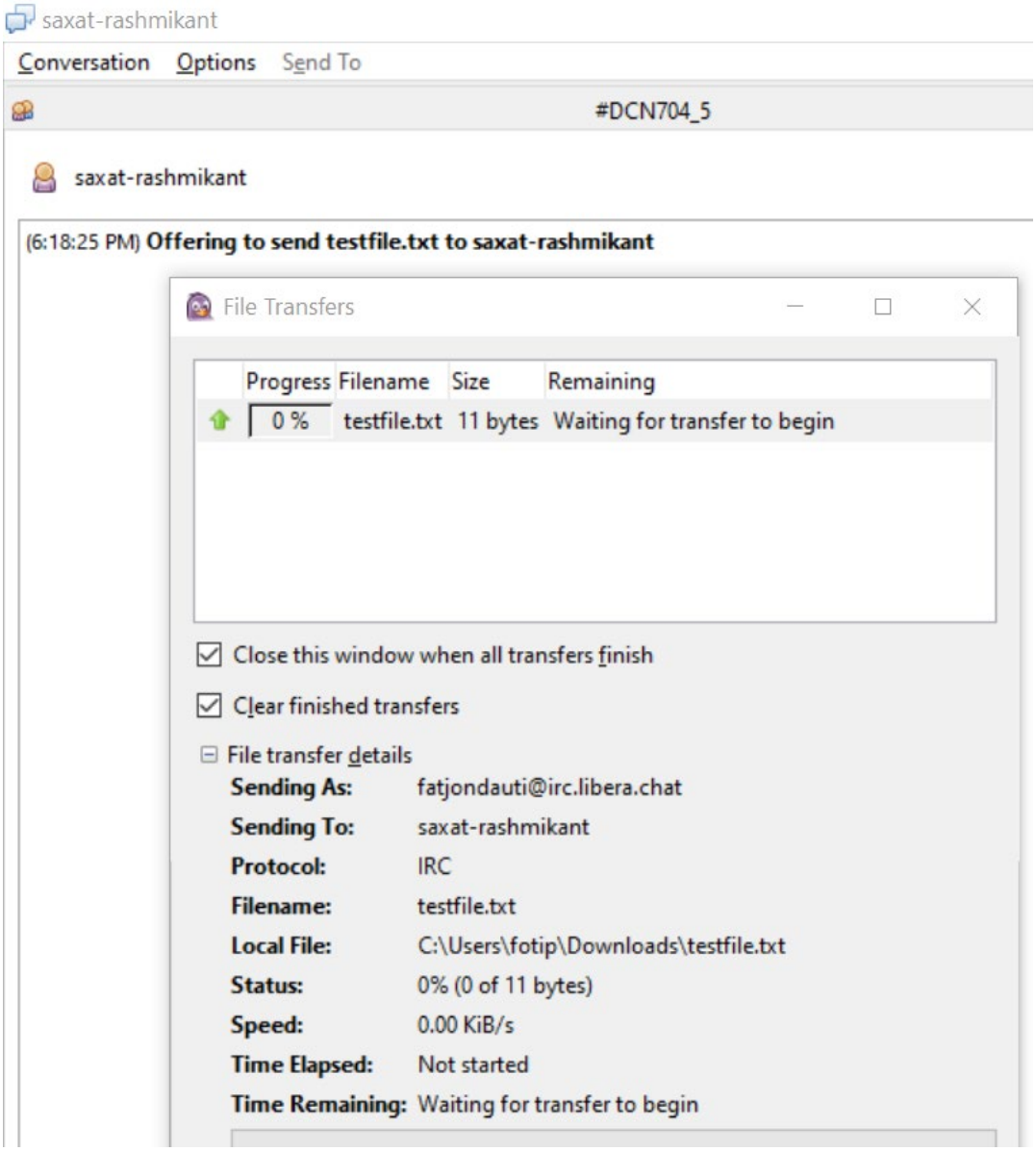
5. **[0.5 marks]** Write a greeting message to one of the users. Be sure that each member of the group send and receive messages.

   The screenshot above, was taken with each member exchanging messages

6. **[0.5 marks]** Send a file to each one of the users. Be sure that the file can be downloaded from the chat space hyperlink.

Insert the screenshot of The Lounge once you have received the file on your computer.

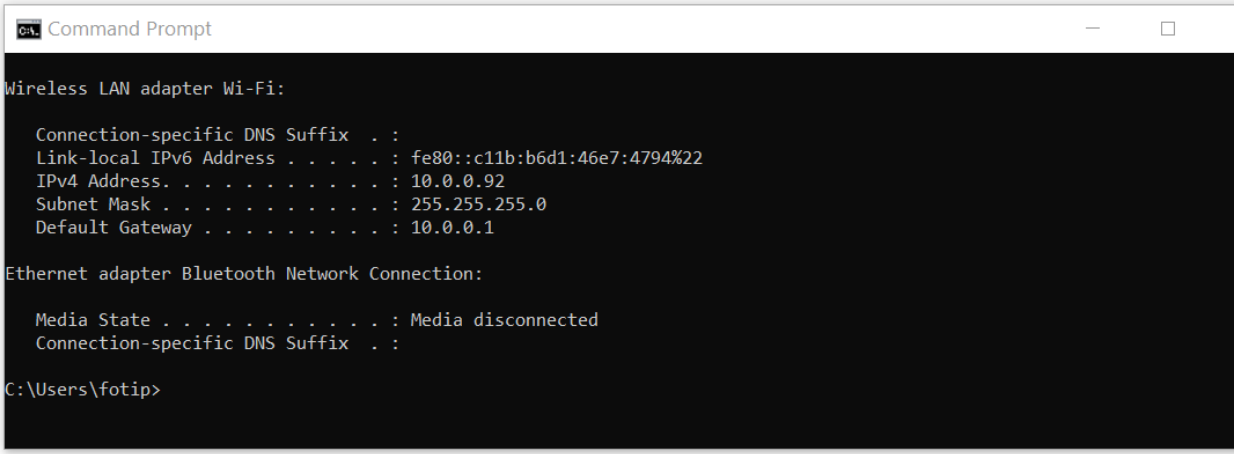
File transer is not supported in the Lounge webclinet, we tried to use the Pidgin desktop IRC client to send a file, as seen below, but it still didn't succeeded, it hangs like below:



7. **[0.5 marks]** Stop the capture on Wireshark and answer the following questions. Insert here the screenshot of the Wireshark capture (be sure that you use **xmpp as display filter.)**

xmpp

No.	Time	Source	Destination	Protocol	Length	S.Port	D.Port	Info
8449	802.775481	173.255.245.243	10.0.0.92	XMPP/XML	173	5222	54769	UNKNOWN PACKET
229…	1642.861483	173.255.245.243	10.0.0.92	XMPP/XML	108	5222	54769	UNKNOWN PACKET
291…	2362.660817	10.0.0.92	173.255.245.243	XMPP/XML	144	54769	5222	UNKNOWN PACKET
336…	3082.870130	173.255.245.243	10.0.0.92	XMPP/XML	108	5222	54769	UNKNOWN PACKET




```
> Frame 8449: 173 bytes on wire (1384 bits), 173 bytes captured (1384 bits) on interface \Device\NPF_{C739A646-CC33-4EC6
> Ethernet II, Src: Netgear_aa:00:d4 (a0:40:a0:aa:00:d4), Dst: IntelCor_1f:47:49 (90:61:ae:1f:47:49)
> Internet Protocol Version 4, Src: 173.255.245.243, Dst: 10.0.0.92
> Transmission Control Protocol, Src Port: 5222, Dst Port: 54769, Seq: 2250, Ack: 2574, Len: 119
> [27 Reassembled TCP Segments (2368 bytes): #448(119), #453(54), #1017(119), #1021(54), #1474(119), #1477(54), #1896(11
> XMPP Protocol
```

8. **[0.5 marks]** Determine how much the protocols from Layer 2 to Layer 4 were used during all the captured sequence.

   Based on the details in the protocol hierarchy statistics for the capture file we have:

   a. Level 2 (Data Link Layer): Ethernet 100% (ARP 2.5%)

   b. Level 3 (Network Layer): IP 97.4% , (IPv6 0.1%)

   c. Level 4 (Transport Layer): TCP 74.1%, UDP 23.3%

9. **[0.5 marks]** From the Expert Information utility, indicate explicitly which protocols were responsible for the chat sequence and what kind of sequences correspond to each one.

   The protocols responsible for the chat sequence are XMPP, TCP and TLS
   With TLS no chat sequence can be seen since messages are encrypted.
   When used with Pidgin chat client, protocol IRC is also responsible, and chat sequence is in plain text because no encryption was configured.

33437	3054.137056	10.0.0.92	130.239.18.120	IRC	104	59086	6667 Request (PRIVMSG)
33496	3059.278865	130.239.18.120	10.0.0.92	IRC	160	6667	59086 Response (PRIVMSG)

```
> Frame 33437: 104 bytes on wire (832 bits), 104 bytes captured (832 bits) on interface \Device\NPF_{C739A646-CC33-4EC6-B0BC-6
> Ethernet II, Src: IntelCor_1f:47:49 (90:61:ae:1f:47:49), Dst: Netgear_aa:00:d4 (a0:40:a0:aa:00:d4)
> Internet Protocol Version 4, Src: 10.0.0.92, Dst: 130.239.18.120
> Transmission Control Protocol, Src Port: 59086, Dst Port: 6667, Seq: 492, Ack: 6108, Len: 50
v Internet Relay Chat
  v Request: PRIVMSG #DCN704_5 :let me try sending you a file
      Command: PRIVMSG
    v Command parameters
        Parameter: #DCN704_5
      Trailer: let me try sending you a file
```

10. **[0.5 marks]** Analyzing the endpoints, indicate the four addresses with the highest amount of TCP packets transmitted and received; also indicate the four addresses with the highest amount of UDP packets transmitted and received; determine if the addresses indicated are the same four in both cases (TCP and UDP).

    Statistics - Endpoints

Ethernet · 20 | IPv4 · 205 | IPv6 · 4 | TCP · 664 | UDP · 1295

Address	Port	Packets	Bytes	Tx Packets	Tx Bytes	Rx Packets	Rx Bytes
206.80.249.9	443	5,001	4703k	3,275	4544k	1,726	
31.13.80.53	443	3,636	1400k	2,038	939k	1,598	
104.18.22.110	443	3,574	655k	2,055	472k	1,519	
10.0.0.92	49317	1,651	239k	739	93k	912	

Ethernet · 20 | IPv4 · 205 | IPv6 · 4 | TCP · 664 | UDP · 1295

Address	Port	Packets	Bytes	Tx Packets	Tx Bytes	Rx Packets	Rx Bytes
172.217.165.10	443	2,579	397k	1,334	203k	1,245	
10.0.0.92	55256	1,971	238k	931	119k	1,040	
255.255.255.255	34569	1,939	557k	0	0	1,939	
172.217.1.10	443	1,470	315k	730	163k	740	

No, they are not, but as it can be seen other address are thrown in the mix since is difficult to stop other flows of traffics when running wireshark on windows.