

Network Architecture I

Project Report

Sri Harsha Chennavajjala(16210893)

Teja Garidepally(16183523)

Raj Kiran Reddy Munnangi(16210167)

Introduction:

In this project, we have developed a simple TCP client and server programs using GENI for simple message exchanges and simple file transfers.

Initial setup:

- Geni account creation
- Slice creation
- Resource allocation.

Part I: GENI/Socket programming Warm-up

- Client and server java code
- Initial lookup for part 2.


Requirements:

- Geni account and a slice to add resources where we can work on.
- SSH Keypairs for logging in to nodes and running scripts.
- Java scripts for communication between client and server.

Steps involved:

Step 1:

After logging in to GENI Portal, we have downloaded the SSH keypairs for authentication purposes.



GENI Portal

Home

Tools

Partners

Help

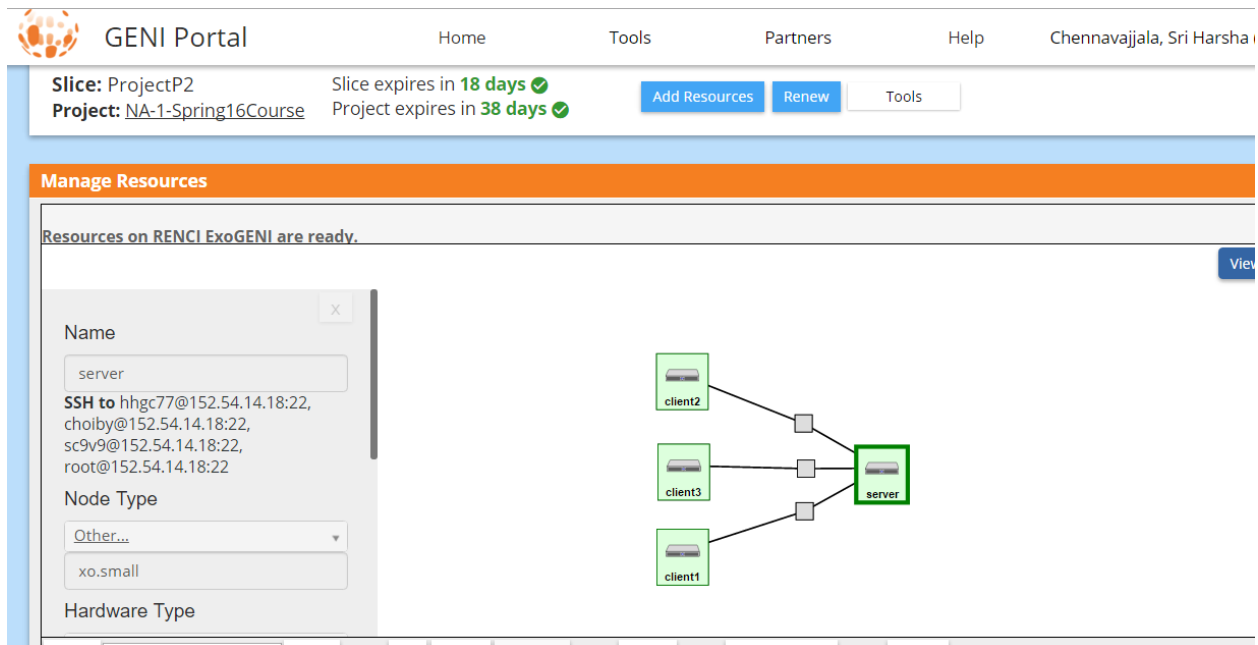
Chennavajjala, Sri Harsha (UMKC-Studen

SSH Keys

Name	Description	Public Key	Private Key	PuTTY	Edit	Delete
id_geni_ssh_rsa 48:96:12:7e:a3:fb:9e:df:79:b1:05:6b:1a:14:06:78	Generated SSH keypair	Download Public Key	Download Private Key	Download PuTTY Key	Edit	Delete

Step 2:

Created a slice “ProjectP2” and added four resources i.e four VM’s. One of the VMs acts as server and the remaining three will act as clients. In the next step, I’ve established connection between these four resources.



Step 3:

Logged in to our client and server nodes using SSH command to the client IP address and server IP address.

Screenshots showing client and server login:

```
sc9v9@server: ~  
harsha@chakram-pc:~$ ssh sc9v9@152.54.14.18  
Welcome to Ubuntu 14.04.3 LTS (GNU/Linux 3.13.0-68-generic x86_64)  
  
* Documentation:  https://help.ubuntu.com/  
  
System information as of Tue Apr 12 13:09:16 EDT 2016  
  
System load:  0.01          Users logged in:   1  
Usage of /:   15.2% of 10.80GB IP address for eth0: 10.103.0.15  
Memory usage: 12%          IP address for eth1: 10.1.1.1  
Swap usage:   0%           IP address for eth2: 10.1.1.5  
Processes:    86           IP address for eth3: 10.1.1.3  
  
Graph this data and manage this system at:  
https://landscape.canonical.com/  
  
85 packages can be updated.  
49 updates are security updates.  
  
Last login: Tue Apr 12 12:32:18 2016 from 134.193.105.179  
$ bash  
sc9v9@server:~$
```

Server Login Screenshot

```
sc9v9@client1: ~
harsha@chakram-pc:~$ ssh sc9v9@152.54.14.17
Welcome to Ubuntu 14.04.3 LTS (GNU/Linux 3.13.0-68-generic x86_64)

 * Documentation:  https://help.ubuntu.com/

System information as of Tue Apr 12 12:31:53 EDT 2016

System load:  0.0                Processes:            81
Usage of /:   15.2% of 10.80GB    Users logged in:     1
Memory usage: 11%                IP address for eth0: 10.103.0.14
Swap usage:   0%                IP address for eth1: 10.1.1.2

Graph this data and manage this system at:
https://landscape.canonical.com/

85 packages can be updated.
49 updates are security updates.

Last login: Tue Apr 12 12:31:53 2016 from 134.193.105.179
$ bash
sc9v9@client1:~$
```

Client1 login screenshot

Step 4:

Part 1 contains two tasks, communication through simple messages and file exchanges.

For task 1(a), communication through simple messages we developed two java files ChatClient.java and ChatServer.java.

The mechanism in ChatClient.java and ChatServer.java is as follows:

- Running ChatServer.java starts server and waits for the client to connect to it.
- After connection establishment client can send message to server and server can send message to client.
- Server or client needs to send “bye from server” or “bye from client” message to terminate the connection.

Below is screenshots of simple chat between client and server:

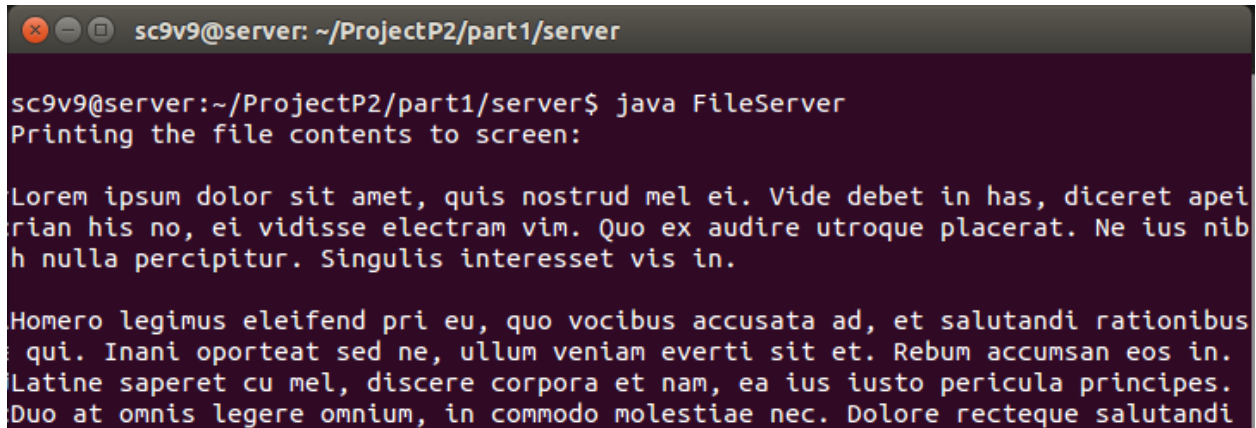
```
sc9v9@client1: ~/ProjectP2/part1/chat
sc9v9@client1:~/ProjectP2/part1/chat$ java ChatClient
Type and press Enter key
Hello from Client - C1
From Server: Hi C1!!!, How are you?
I'm fine server, How are you?
From Server: Good, What's up?
I gotta go, bye
From Server: Ok. Take care
bye from client
Exiting the application
sc9v9@client1:~/ProjectP2/part1/chat$

sc9v9@server: ~/ProjectP2/part1/chat
sc9v9@server:~/ProjectP2/part1/chat$ java ChatServer
Started listening on port 12345
Socket: Socket[addr=/152.54.14.17,port=51902,localport=12345]
Client C1 connected!!!
From Client: Hello from Client - C1
Hi C1!!!, How are you?
From Client: I'm fine server, How are you?
Good, What's up?
From Client: I gotta go, bye
Ok. Take care
bye from server
Exiting the application
sc9v9@server:~/ProjectP2/part1/chat$
```

Client chat window

Server chat window

For task 1(b) we have developed two applications FileClient and FileServer, running FileClient sends a text file to server and server first displays the contents of received file and saves the file in local system.

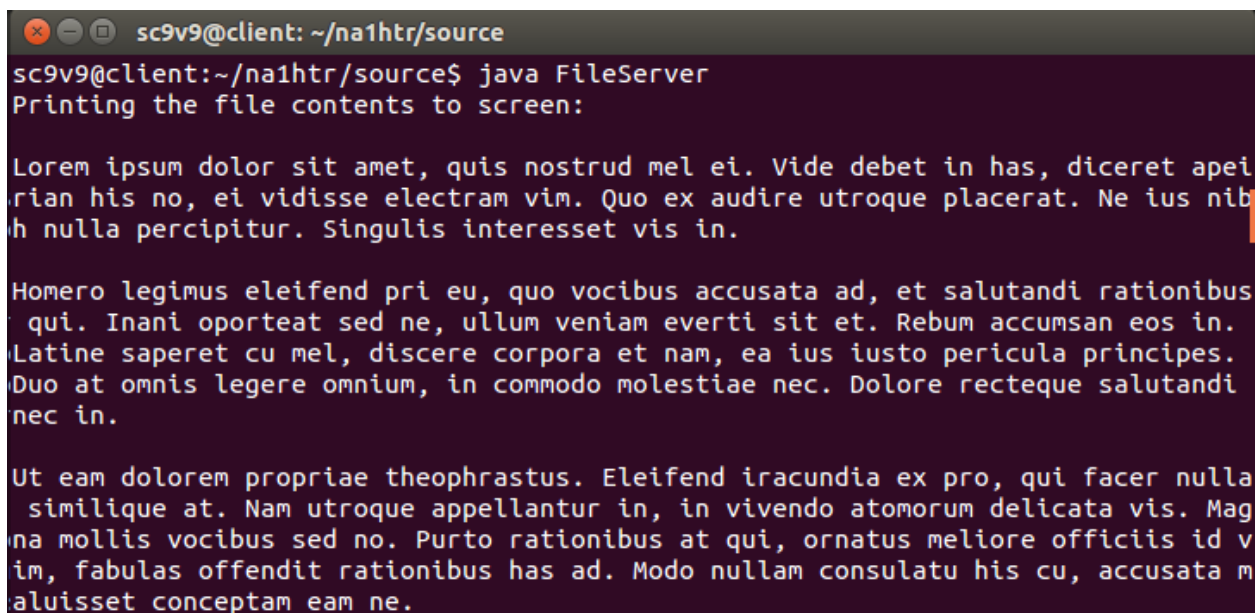
A terminal window with a dark background and light-colored text. The title bar shows 'sc9v9@server: ~/ProjectP2/part1/server'. The prompt is 'sc9v9@server:~/ProjectP2/part1/server\$'. The command 'java FileServer' has been executed. The output shows 'Printing the file contents to screen:' followed by two paragraphs of Lorem Ipsum text.

```
sc9v9@server: ~/ProjectP2/part1/server
sc9v9@server:~/ProjectP2/part1/server$ java FileServer
Printing the file contents to screen:

Lorem ipsum dolor sit amet, quis nostrud mel ei. Vide debet in has, diceret apei
rian his no, ei vidisse electram vim. Quo ex audire utroque placerat. Ne ius nib
h nulla percipitur. Singulis interesset vis in.

Homero legimus eleifend pri eu, quo vocibus accusata ad, et salutandi rationibus
qui. Inani oporteat sed ne, ullum veniam everti sit et. Rebum accumsan eos in.
Latine saperet cu mel, discere corpora et nam, ea ius iusto pericula principes.
Duo at omnis legere omnium, in commodo molestiae nec. Dolore recteque salutandi
```

Server displaying the received file contents

A terminal window with a dark background and light-colored text. The title bar shows 'sc9v9@client: ~/na1htr/source'. The prompt is 'sc9v9@client:~/na1htr/source\$'. The command 'java FileServer' has been executed. The output shows 'Printing the file contents to screen:' followed by three paragraphs of Lorem Ipsum text, including the third paragraph that was not in the previous screenshot.

```
sc9v9@client: ~/na1htr/source
sc9v9@client:~/na1htr/source$ java FileServer
Printing the file contents to screen:

Lorem ipsum dolor sit amet, quis nostrud mel ei. Vide debet in has, diceret apei
rian his no, ei vidisse electram vim. Quo ex audire utroque placerat. Ne ius nib
h nulla percipitur. Singulis interesset vis in.

Homero legimus eleifend pri eu, quo vocibus accusata ad, et salutandi rationibus
qui. Inani oporteat sed ne, ullum veniam everti sit et. Rebum accumsan eos in.
Latine saperet cu mel, discere corpora et nam, ea ius iusto pericula principes.
Duo at omnis legere omnium, in commodo molestiae nec. Dolore recteque salutandi
nec in.

Ut eam dolorem propriae theophrastus. Eleifend iracundia ex pro, qui facer nulla
similique at. Nam utroque appellantur in, in vivendo atomorum delicata vis. Mag
na mollis vocibus sed no. Purto rationibus at qui, ornatus meliore officiis id v
im, fabulas offendit rationibus has ad. Modo nullam consulatu his cu, accusata m
aluisset conceptam eam ne.
```

Server displaying the file contents after received from client

Server appends the one more line to received file and sends the updated file back to client. Client now displays the file on screen after receiving the full updated file.

```
sc9v9@server: ~/ProjectP2/part1/server
Ut eam dolorem propriae theophrastus. Eleifend iracundia ex pro, qui facer nulla
similique at. Nam utroque appellantur in, in vivendo atomorum delicata vis. Mag
na mollis vocibus sed no. Purto rationibus at qui, ornatus meliore officiis id v
im, fabulas offendit rationibus has ad. Modo nullam consulatu his cu, accusata m
aluisset conceptam eam ne.

Eu his ipsum utinam ceteros, iusto nostro splendide cu mea. Omnis electram an qu
i, primis maiestatis mnesarchum mel no, nisl molesti
Appending the text to the file
Sending file back to client
sc9v9@server:~/ProjectP2/part1/server$
```

Server appending new line to the received file

```
sc9v9@client1: ~/ProjectP2/part1/client
sc9v9@client1:~/ProjectP2/part1/client$ java FileClient
Getting updated file from the server....
Printing the file contents to screen:

Lorem ipsum dolor sit amet, quis nostrud mel ei. Vide debet in has, diceret apei
rian his no, ei vidisse electram vim. Quo ex audire utroque placerat. Ne ius nib
h nulla percipitur. Singulis interesset vis in.

Homero legimus eleifend pri eu, quo vocibus accusata ad, et salutandi rationibus
```

Client getting the updated file from server

```
sc9v9@client1: ~/ProjectP2/part1/client
aluisset conceptam eam ne.

Eu his ipsum utinam ceteros, iusto nostro splendide cu mea. Omnis electram an qu
i, primis maiestatis mnesarchum mel no, nisl molesti
This line is added by the Server
^Csc9v9@client1:~/ProjectP2/part1/client$ ls -ltr
total 48
```

Updated file at the client end

Step 5:

Part 2 contains four tasks - development of client-server chat application

Part 2(a): In this part, the chat server will start listening to the client. Once the client is connected, the server displays all the messages received from client. If the client types “exit”, both the client and server quits.

<pre>sc9v9@client1: ~/ProjectP2/part2/chat sc9v9@client1:~/ProjectP2/part2/chat\$ java ChatClient21 Type and press Enter key Hello from Client - C1 How are you? This is second message I gotta go exit Exiting the application sc9v9@client1:~/ProjectP2/part2/chat\$</pre>	<pre>sc9v9@server: ~/ProjectP2/part2/chat2 sc9v9@server:~/ProjectP2/part2/chat2\$ java ChatServer21 Started listening on port 12345 Client C1 connected!!! From Client: Hello from Client - C1 From Client: How are you? From Client: This is second message From Client: I gotta go Client asked to exit. Exiting... sc9v9@server:~/ProjectP2/part2/chat2\$</pre>
--	--

Client chat window

Server chat window

Part 2(b):

This part is similar to part 2(a) but the server remains active and waits to get new connection from client.

Here we test this scenario, with two clients C1, C2 and a Server. First client C1 connects to server sends some messages and exits the application. The server remains active. After sometime, client C2 connects to server and sends some messages. In the end, client C2 exits. Below screenshot shows the workflow.

```
sc9v9@server: ~/ProjectP2/part2/chat2
sc9v9@server:~/ProjectP2/part2/chat2$ java ChatServer22
Started listening on port 12345
```

Server started and waiting for the clients to connect

<pre>sc9v9@client1: ~/ProjectP2/part2/chat sc9v9@client1:~/ProjectP2/part2/chat\$ java ChatClient22 Type and press Enter key Hello from Client - C1 Message 1 from C1 exit Exiting the application sc9v9@client1:~/ProjectP2/part2/chat\$</pre>	<pre>sc9v9@server: ~/ProjectP2/part2/chat2 sc9v9@server:~/ProjectP2/part2/chat2\$ java ChatServer22 Started listening on port 12345 Client C1 connected!!! From Client: Hello from Client - C1 From Client: Message 1 from C1 Client C1 exited. Waiting for new client... From Client: exit</pre>
---	---

Client C1 chat window

Server chat window

Client C1 connected and send messages and exited. Server waiting for new clients

```
sc9v9@client1: ~/ProjectP2/part2/chat
sc9v9@client1:~/ProjectP2/part2/chat$ java ChatClient22
Type and press Enter key
Hello from Client - C1
Message 1 from C1
exit
Exiting the application
sc9v9@client1:~/ProjectP2/part2/chat$

sc9v9@server: ~/ProjectP2/part2/chat2
sc9v9@server:~/ProjectP2/part2/chat2$ java ChatServer22
Started listening on port 12345
Client C1 connected!!!
From Client: Hello from Client - C1
From Client: Message 1 from C1
Client C1 exited. Waiting for new client...

From Client: exit
Client C2 connected!!!
From Client: Hello from Client - C2
From Client: Hi, How are you?
Client C2 exited. Waiting for new client...

From Client: exit

```

Once a new client C2 connected to the server, the server displays the message “Client C2 connected!!!”

The client C2 sends messages and the messages will be displayed on server. Finally C2 disconnects from server by sending “exit” message.

Part 2(c):

In this case, the server should be capable of serving multiple clients simultaneously. All the messages from the clients will be displayed on server side.

```
sc9v9@client1: ~/ProjectP2/part2/new
sc9v9@client1:~/ProjectP2/part2/new$ java ConfUser 152.54.14.18 12345
Enter your name.
C1
Welcome C1 to our chat room.
To leave enter '/quit' in a new line.
Hello
<C1> Hello
How are you?
<C1> How are you?

```

Client C1 sending messages to Server

```
sc9v9@server: ~/ProjectP2/part2/new
sc9v9@server:~/ProjectP2/part2/new$ java ConfServer 12345
<C1> Hello
<C1> How are you?

```

Server displays all the messages typed by client C1

Next C2 enters the chat, and sends messages to server. Once the clients are done with chat, they can exit the application by typing “/quit”.

The image shows two terminal windows side-by-side. The left window is titled 'sc9v9@client2: ~/ProjectP2/part2/new' and shows a client running 'java ConfUser 152.54.14.18 12345'. The client prompts for a name, then says 'Welcome C2 to our chat room.' and 'To leave enter '/quit' in a new line.' The client then sends 'Hello', 'This is from C2', and '/quit'. The right window is titled 'sc9v9@server: ~/ProjectP2/part2/new' and shows a server running 'java ConfServer 12345'. The server displays messages from both client C1 and client C2: '<C1> Hello', '<C1> How are you?', '<C2> Hello', '<C2> This is from C2', and '<C2> /quit'. It also shows 'User C2 left' and '<C1> /quit'.

C2 chat window

Server chat window

In the above screenshot, server is displaying the messages from both client C1 and client C2 simultaneously.

Part 2(d):

In this part, the messages sent by the clients will be passed through server and will be distributed to all the clients. So, if a client types a message, the message will be displayed on all the other clients.

Here I've used two clients C2, C3 and a server. Below screenshots demonstrates the workflow.

C3 chat window

Server chat window

The image shows three terminal windows. The top-left window is titled 'sc9v9@client3: ~/ProjectP2/part2/new' and shows a client running 'java ConfUser 152.54.14.18 12345'. The client prompts for a name, then says 'Welcome C3 to our chat room.' and 'To leave enter '/quit' in a new line.' The client then sends 'Hello', 'Hi', 'C2 is great', 'no C3 is great', 'noway', 'yes way', and '/quit'. The top-right window is titled 'sc9v9@server: ~/ProjectP2/part2/new' and shows a server running 'java ConfServer 12345'. The server displays messages from both client C2 and client C3: '<C3> Hello', '<C2> Hi', '<C2> C2 is great', '<C3> no C3 is great', '<C2> noway', '<C3> yes way', '<C3> /quit', and 'User C3 left'. The bottom window is titled 'sc9v9@client2: ~/ProjectP2/part2/new' and shows a client running 'java ConfUser 152.54.14.18 12345'. The client prompts for a name, then says 'Welcome C2 to our chat room.' and 'To leave enter '/quit' in a new line.' The client then sends 'Hi', 'C2 is great', 'C2 is great', 'no C3 is great', 'noway', 'yes way', and '/quit'. It also shows 'C2Exited'.

C2 chat window

In the above screenshot all the messages from C2 and C3 were displayed on server. Also all the messages of C2 were displayed on C3 and vice versa.

Source code Link:

<https://drive.google.com/open?id=0B8RcAeWxxOPWNI1nSUwtbIVMVzg>