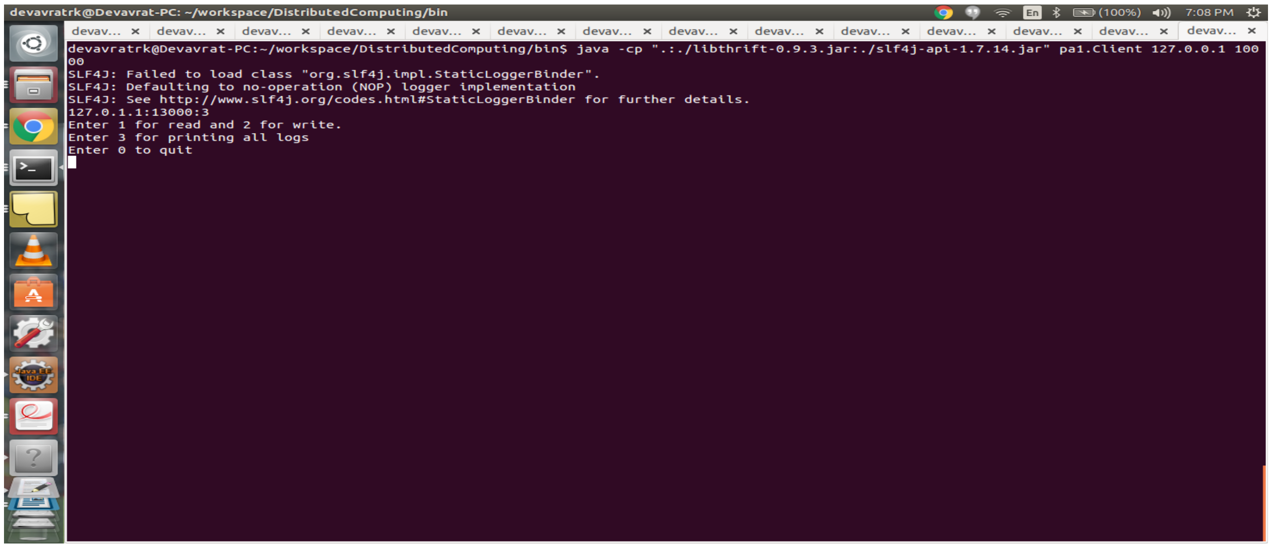
Test cases

Please have a look at the user document which details how to start all components of the system and how to use the interface. Before running these test cases, PLEASE ENSURE THAT YOU HAVE PLACED THE CODE IN A FOLDER WHERE YOU HAVE WRITE ACCESS, since our file system is persistent.

For the case that we ran these three test cases, we had node IDs 0, 1, 3, 8, 12 and 15 in the system. The random entry point that we got from the super node was node ID 3.

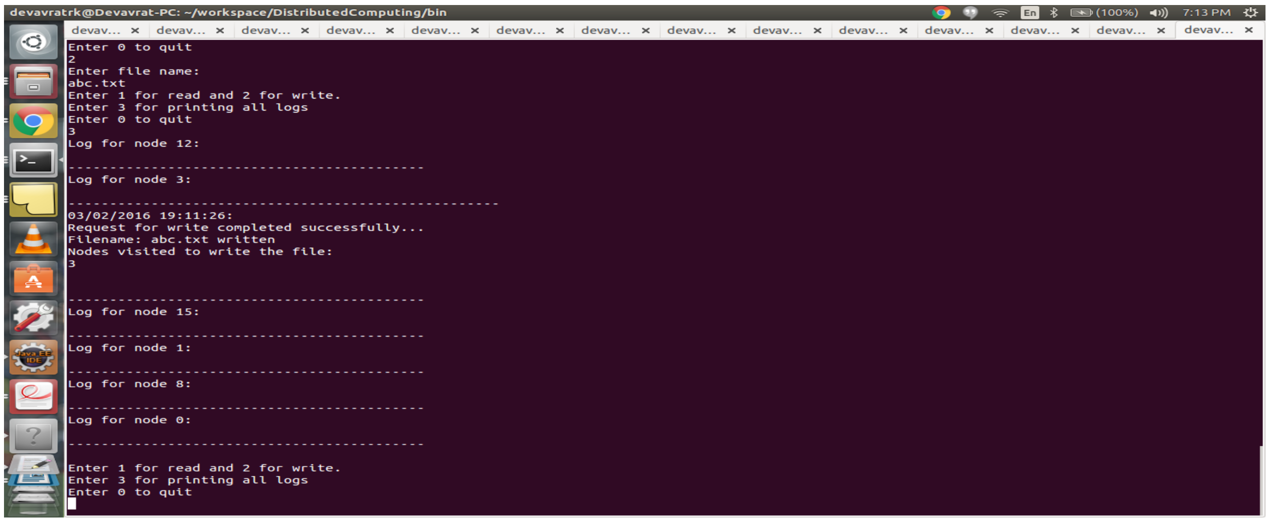
1) **Positive test case:**

Here is a screenshot of the client window just after starting the client.



```
devavratrk@Devavrat-PC:~/workspace/DistributedComputing/bin$ java -cp ".:./libthrift-0.9.3.jar:./slf4j-api-1.7.14.jar" pa1.Client 127.0.0.1 100
00
SLF4J: Failed to load class "org.slf4j.impl.StaticLoggerBinder".
SLF4J: Defaulting to no-operation (NOP) logger implementation
SLF4J: See http://www.slf4j.org/codes.html#StaticLoggerBinder for further details.
127.0.1.1:13000:3
Enter 1 for read and 2 for write.
Enter 3 for printing all logs
Enter 0 to quit
```

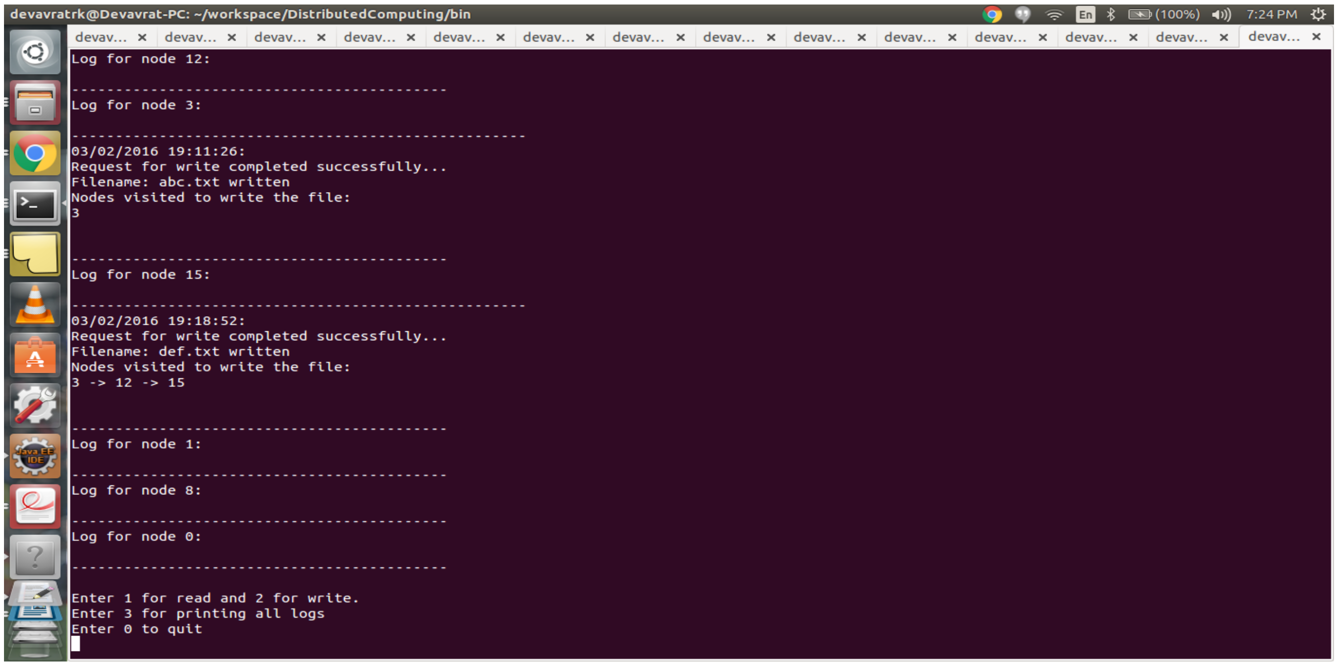
We now give an input of 2 and write the file abc.txt (the key obtained after hashing this filename was 3) to the DHT. We then use option 3 to print the logs of the DHT. Here is a screenshot after doing these two operations:



```
Enter 0 to quit
2
Enter file name:
abc.txt
Enter 1 for read and 2 for write.
Enter 3 for printing all logs
Enter 0 to quit
3
Log for node 12:

-------------------------------------------
Log for node 3:

-------------------------------------------------------
03/02/2016 19:11:26:
Request for write completed successfully...
Filename: abc.txt written
Nodes visited to write the file:
3


-------------------------------------------
Log for node 15:

-------------------------------------------
Log for node 1:

-------------------------------------------
Log for node 8:

-------------------------------------------
Log for node 0:

-------------------------------------------

Enter 1 for read and 2 for write.
Enter 3 for printing all logs
Enter 0 to quit
```

We see here that the log for node 3 shows that a request for write was completed successfully and since the start node was 3, it wrote the file there itself and thus the tracking line shows that only node 3 was visited.

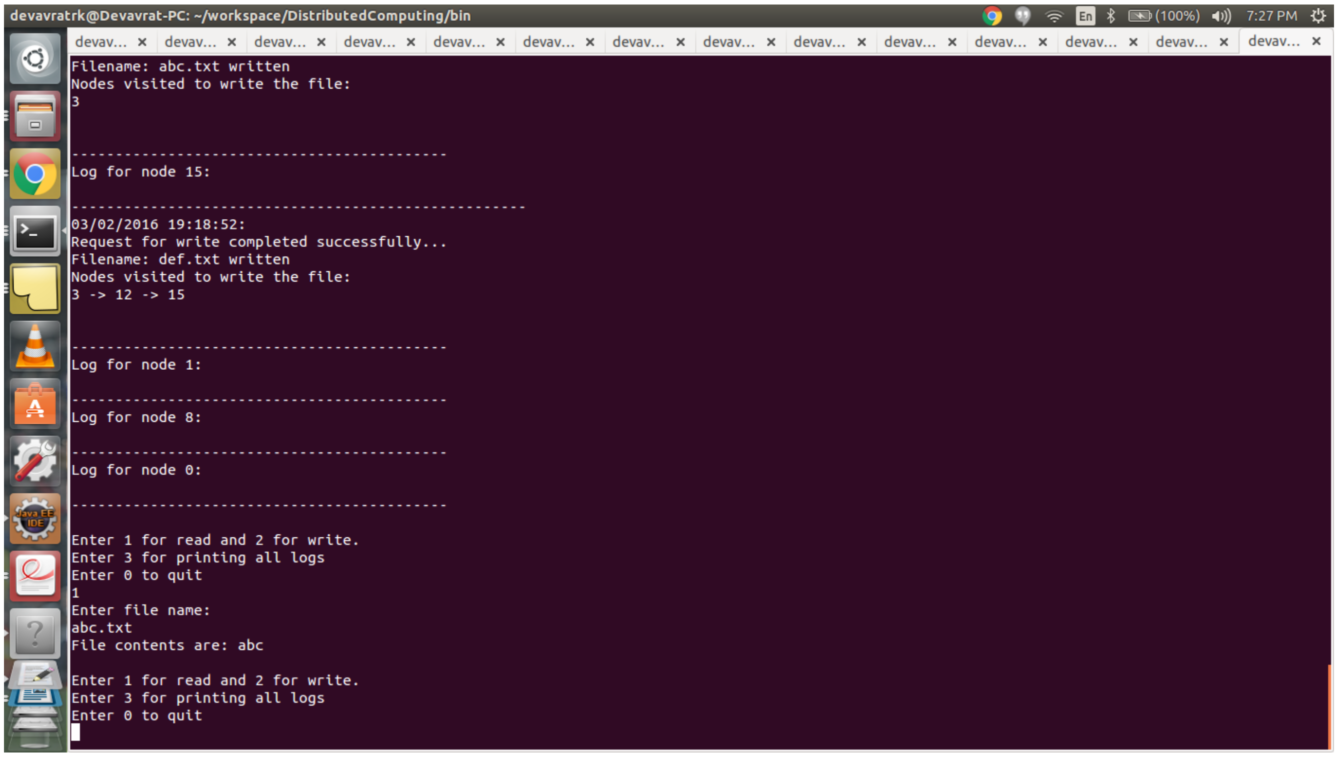
We now do another write of filename def.txt (the key obtained after hashing this file name is 15) to the DHT. We then again use option 3 to print all logs of the DHT. Here is the screenshot:



```
Log for node 12:

--------------------------------------------
Log for node 3:

----------------------------------------------------
03/02/2016 19:11:26:
Request for write completed successfully...
Filename: abc.txt written
Nodes visited to write the file:
3


--------------------------------------------
Log for node 15:

----------------------------------------------------
03/02/2016 19:18:52:
Request for write completed successfully...
Filename: def.txt written
Nodes visited to write the file:
3 -> 12 -> 15


--------------------------------------------
Log for node 1:

--------------------------------------------
Log for node 8:

--------------------------------------------
Log for node 0:

--------------------------------------------

Enter 1 for read and 2 for write.
Enter 3 for printing all logs
Enter 0 to quit
```

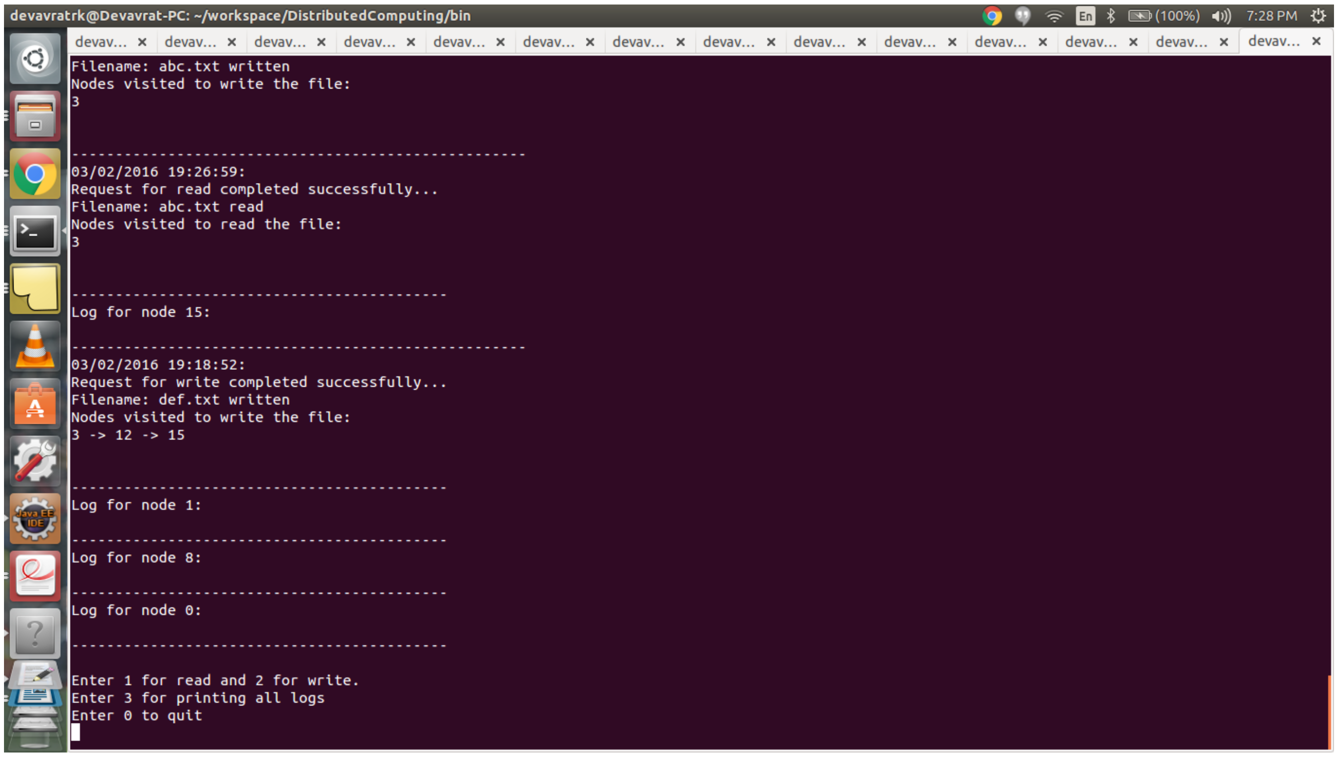
We know that since the hash value is 15, the file should be written to node 15 and sure enough, we see that in the logs. The path followed is 3 → 12 → 15 as seen in the logs of node 15.

We now do a read of filename abc.txt, which is present in the DHT by giving an input of 1 to the client. Here are the contents of the file, printed at the bottom of the screen (abc):



```
Filename: abc.txt written
Nodes visited to write the file:
3


--------------------------------------------
Log for node 15:

----------------------------------------------------
03/02/2016 19:18:52:
Request for write completed successfully...
Filename: def.txt written
Nodes visited to write the file:
3 -> 12 -> 15


--------------------------------------------
Log for node 1:

--------------------------------------------
Log for node 8:

--------------------------------------------
Log for node 0:

--------------------------------------------

Enter 1 for read and 2 for write.
Enter 3 for printing all logs
Enter 0 to quit
1
Enter file name:
abc.txt
File contents are: abc

Enter 1 for read and 2 for write.
Enter 3 for printing all logs
Enter 0 to quit
```

Here are the logs:

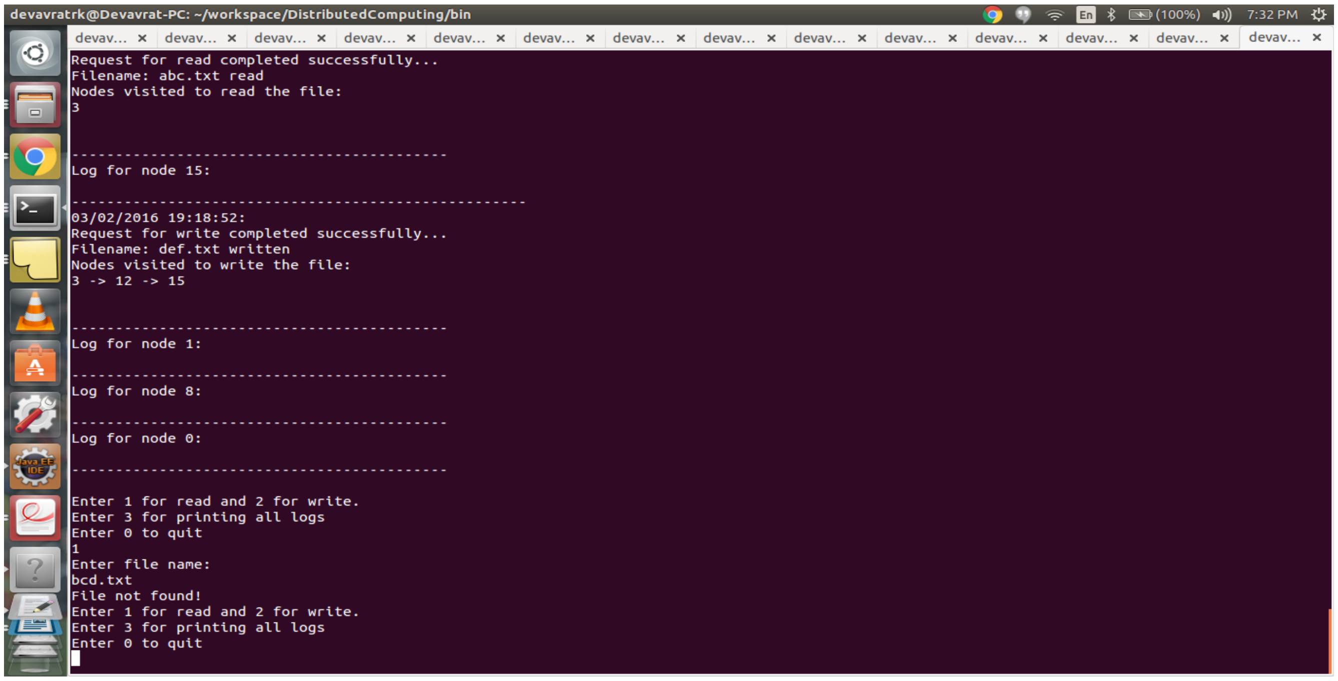


```
devavratrk@Devavrat-PC: ~/workspace/DistributedComputing/bin
Filename: abc.txt written
Nodes visited to write the file:
3


---------------------------------------------------
03/02/2016 19:26:59:
Request for read completed successfully...
Filename: abc.txt read
Nodes visited to read the file:
3


-------------------------------------------
Log for node 15:

---------------------------------------------------
03/02/2016 19:18:52:
Request for write completed successfully...
Filename: def.txt written
Nodes visited to write the file:
3 -> 12 -> 15


-------------------------------------------
Log for node 1:

-------------------------------------------
Log for node 8:

-------------------------------------------
Log for node 0:

-------------------------------------------

Enter 1 for read and 2 for write.
Enter 3 for printing all logs
Enter 0 to quit
```

We see from the logs of node 3 that the file was successfully read from node 3.

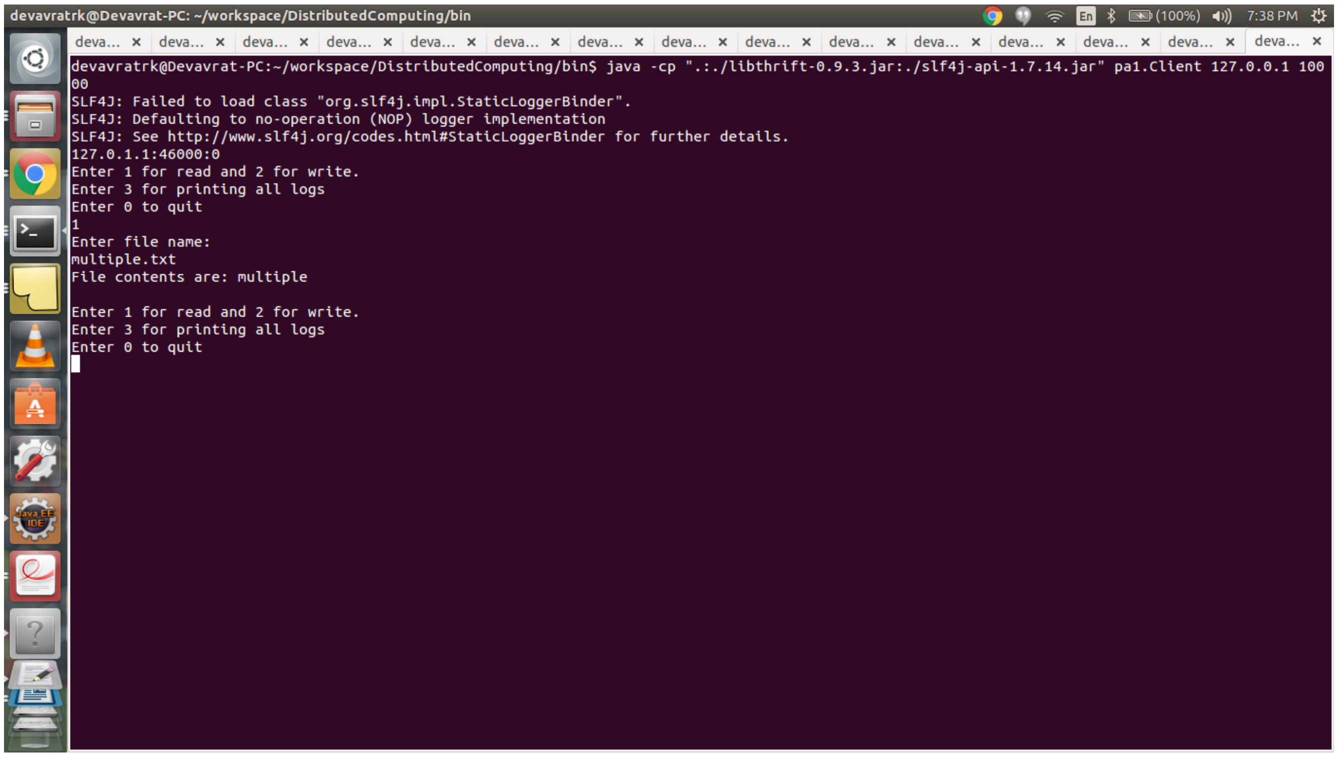
2) **Negative test case:**

We continue from the results of the previous test case. We now read a file name which we have not written before, which is bcd.txt. We see in the next screenshot that the client receives an error from the DHT which says that the file was not found.



```
devavratrk@Devavrat-PC: ~/workspace/DistributedComputing/bin
Request for read completed successfully...
Filename: abc.txt read
Nodes visited to read the file:
3


-------------------------------------------
Log for node 15:

---------------------------------------------------
03/02/2016 19:18:52:
Request for write completed successfully...
Filename: def.txt written
Nodes visited to write the file:
3 -> 12 -> 15


-------------------------------------------
Log for node 1:

-------------------------------------------
Log for node 8:

-------------------------------------------
Log for node 0:

-------------------------------------------

Enter 1 for read and 2 for write.
Enter 3 for printing all logs
Enter 0 to quit
1
Enter file name:
bcd.txt
File not found!
Enter 1 for read and 2 for write.
Enter 3 for printing all logs
Enter 0 to quit
```

3) **Fun test case:** This test case shows that our system supports multiple clients connected at the same time. We continue from the results of the previous test case. In a new terminal window, we now open a new client. From the first client, we then write a new file called multiple.txt as explained before. We then go to the second client and try to read this file as explained before. Here is the result:



```
devavratrk@Devavrat-PC:~/workspace/DistributedComputing/bin$ java -cp ".:./libthrift-0.9.3.jar:./slf4j-api-1.7.14.jar" pa1.Client 127.0.0.1 100
00
SLF4J: Failed to load class "org.slf4j.impl.StaticLoggerBinder".
SLF4J: Defaulting to no-operation (NOP) logger implementation
SLF4J: See http://www.slf4j.org/codes.html#StaticLoggerBinder for further details.
127.0.1.1:46000:0
Enter 1 for read and 2 for write.
Enter 3 for printing all logs
Enter 0 to quit
1
Enter file name:
multiple.txt
File contents are: multiple

Enter 1 for read and 2 for write.
Enter 3 for printing all logs
Enter 0 to quit
```

We see from the above screenshot that the second client was successfully able to read the contents of the file multiple.txt and print it to the console.

We finally close both the clients by entering a 0 in the console. PLEASE DO NOT CLOSE THE CLIENT using Ctrl + C, since we are deleting our persistent file system when the client exits. This is also a drawback for the test case mentioned above, since if the first client exits and then the second client tries to read a file, it will not find the file because the distributed file system would have been deleted.