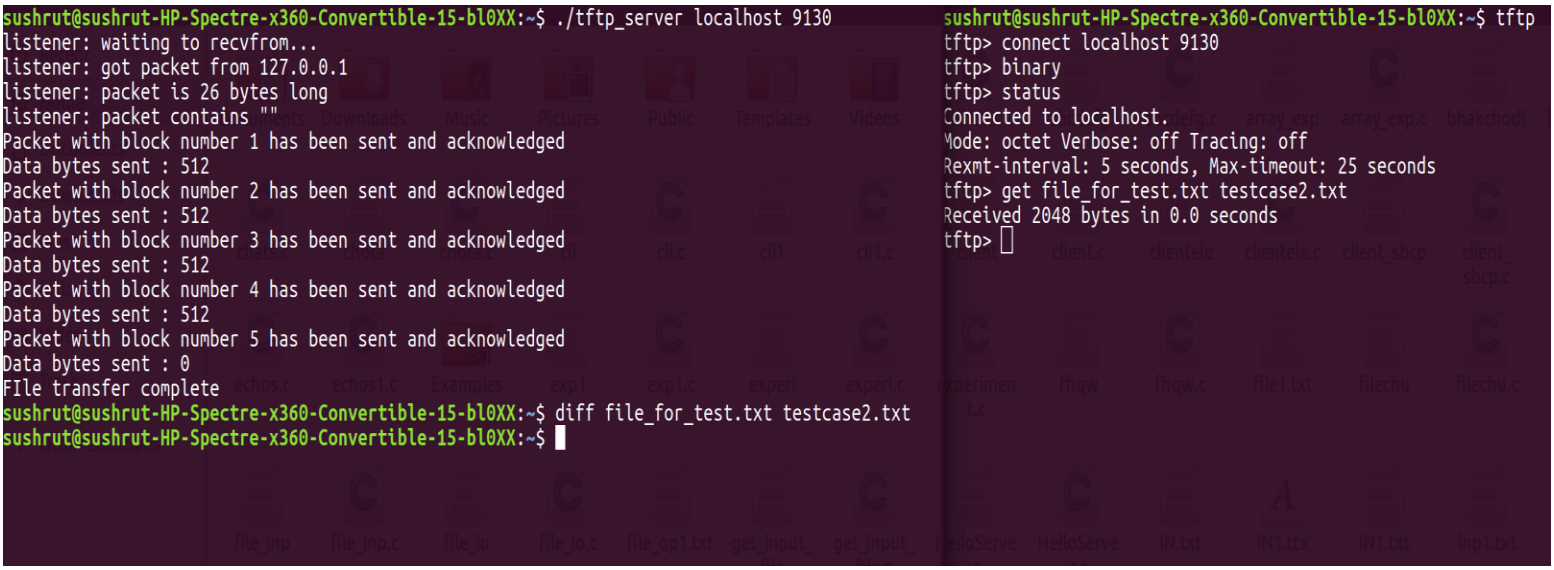
Course Number-602

Team Number-01 - Ishan Tyagi, Sushrut Kaul

TEST CASES

TEST CASE-1

A). transfer a binary file of 2048 bytes and check that it matches the source file

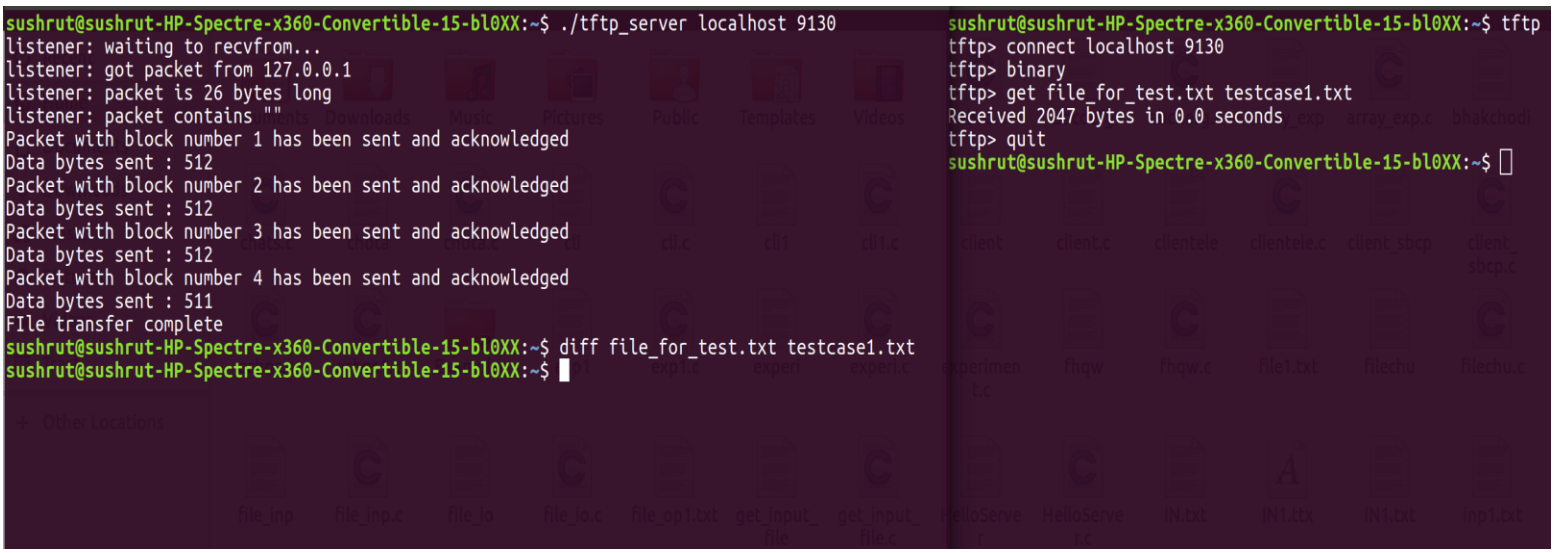


For this test case we generated a binary file (file_for_test.txt) of size 2048 bytes. The server then implements reading that file using the RRQ. Afterwards, we copied the file and transferred the content onto a new file (testcase1.txt). Finally we have compared both the files for the content. The result showed that there was no difference between the files.

Also, worth noticing is that the file was sent in 4 packets (integral multiple of 512 bytes), which can be seen in the above screenshot.

TEST CASE-2

A). transfer a binary file of 2047 bytes and check that it matches the source file

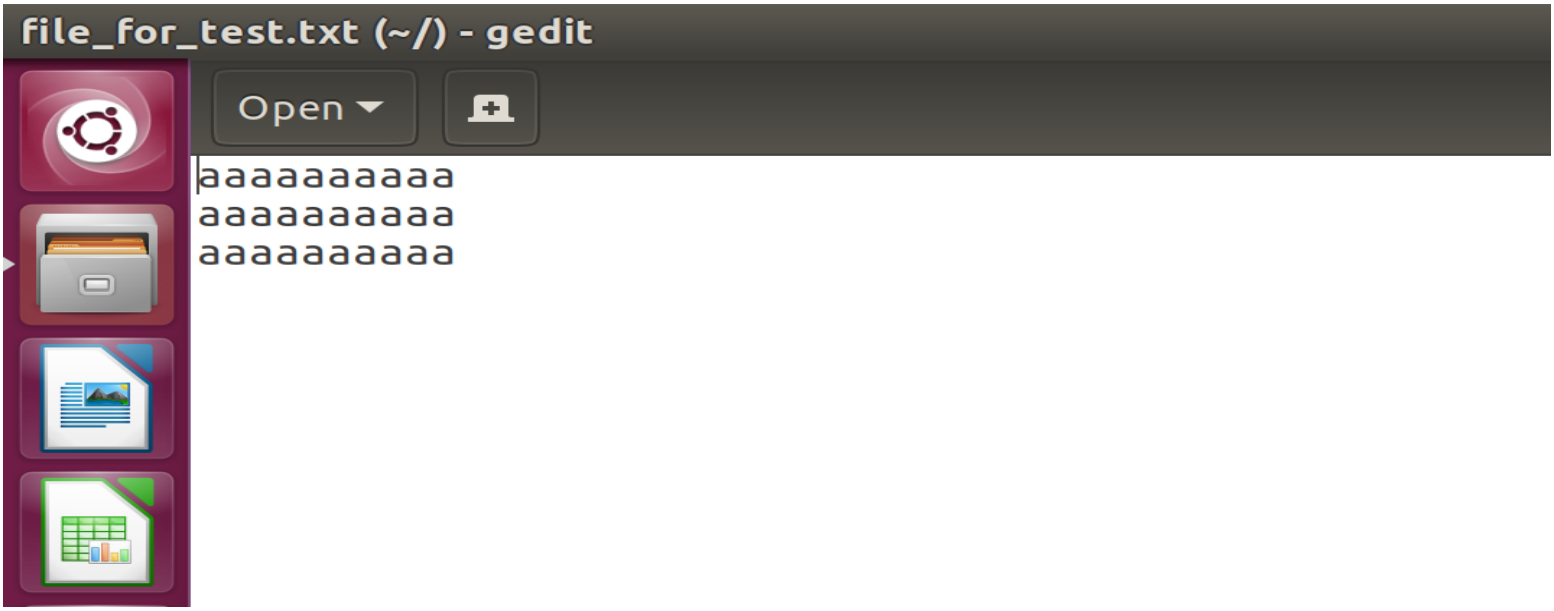


In this test case, we did everything quite similar to the first test case. The only difference is that the file size is not an integral multiple of 512 bytes (2047 bytes) this time. The bytes sent can be clearly seen in the screenshot of the command window on the left side (511 bytes).

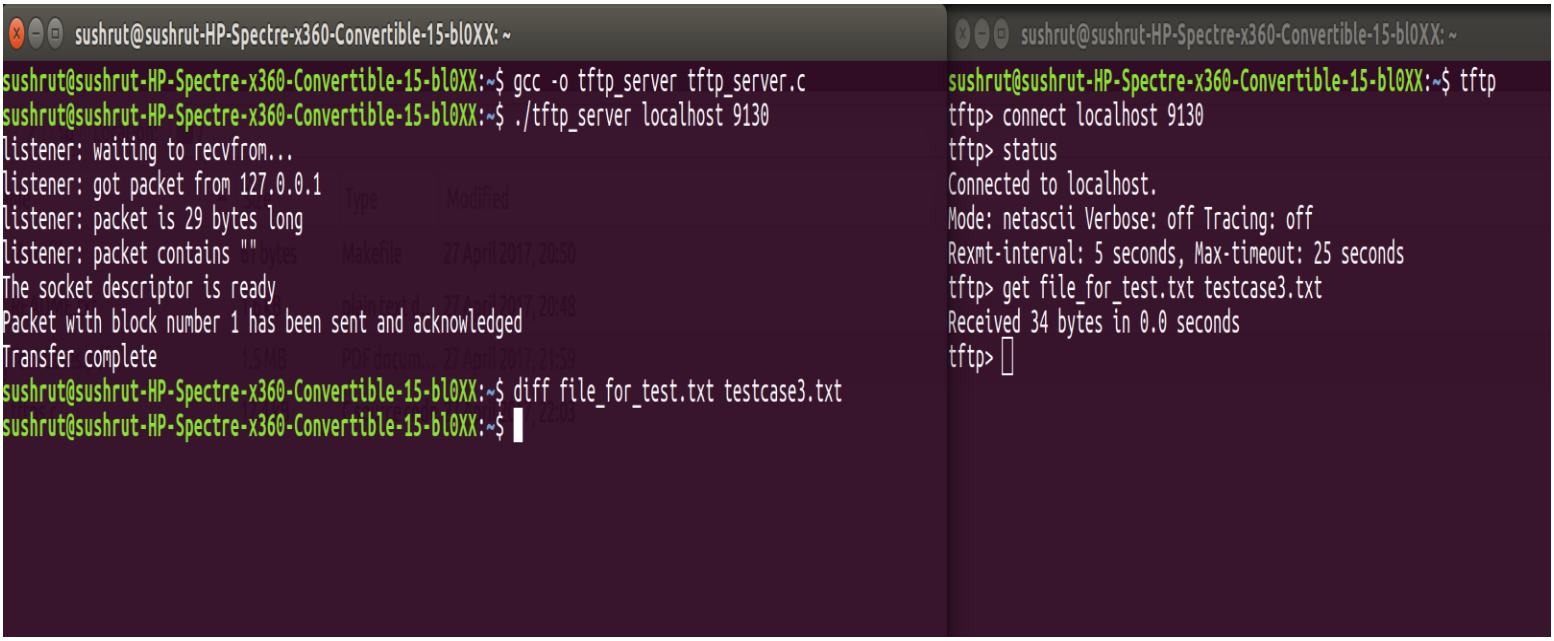
TEST CASE-3

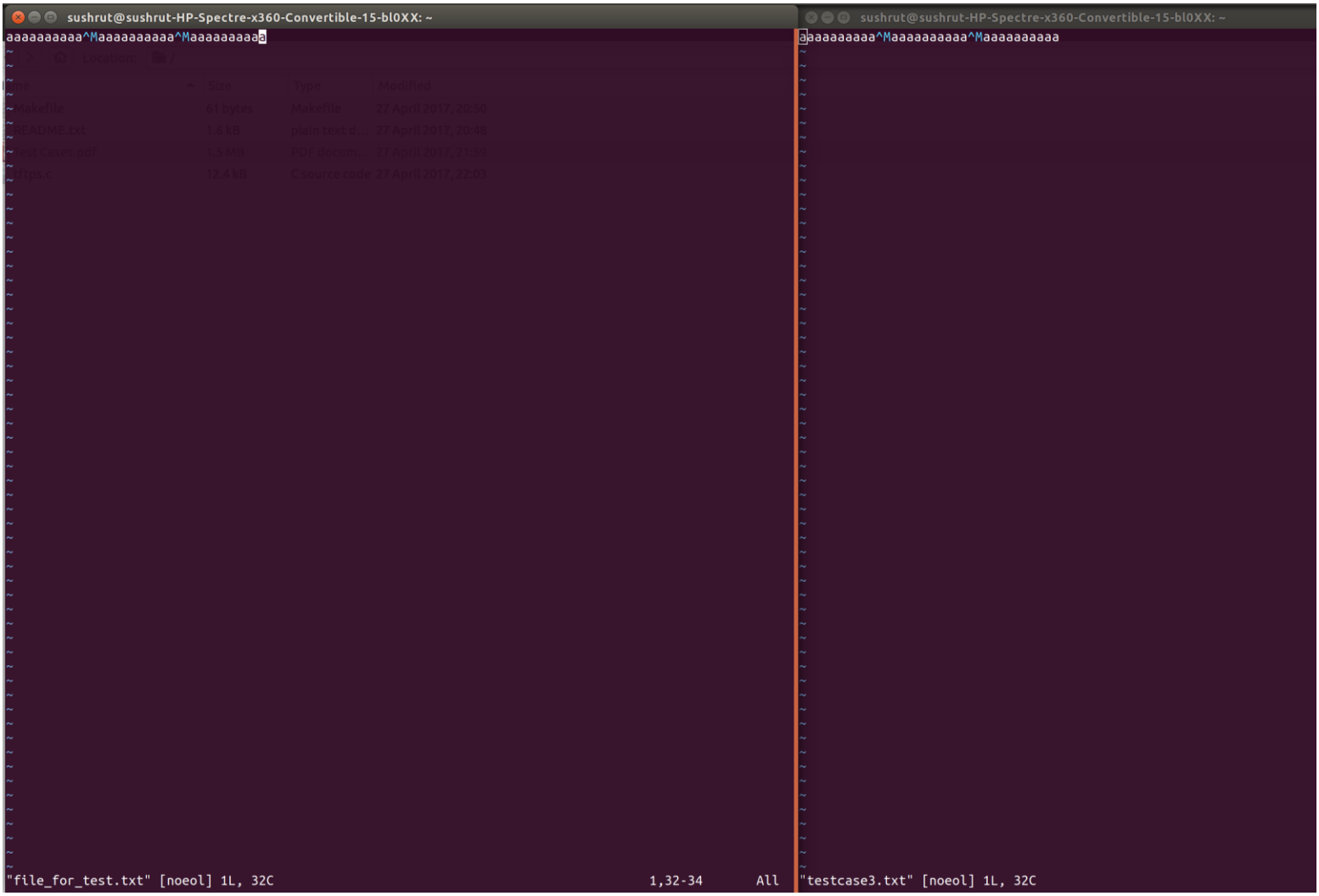
A). transfer a netascii file that includes two CR's and check that the resulting file matches the input file

We created a file named file_for_test.txt with 2CR's and the server then implements reading that file using the RRQ and transferred the contents onto a file (testcase3.txt). For the creation of a file with 2CR's we wrote a code to introduce the required carriage returns. Below is a screenshot of the file that was read:



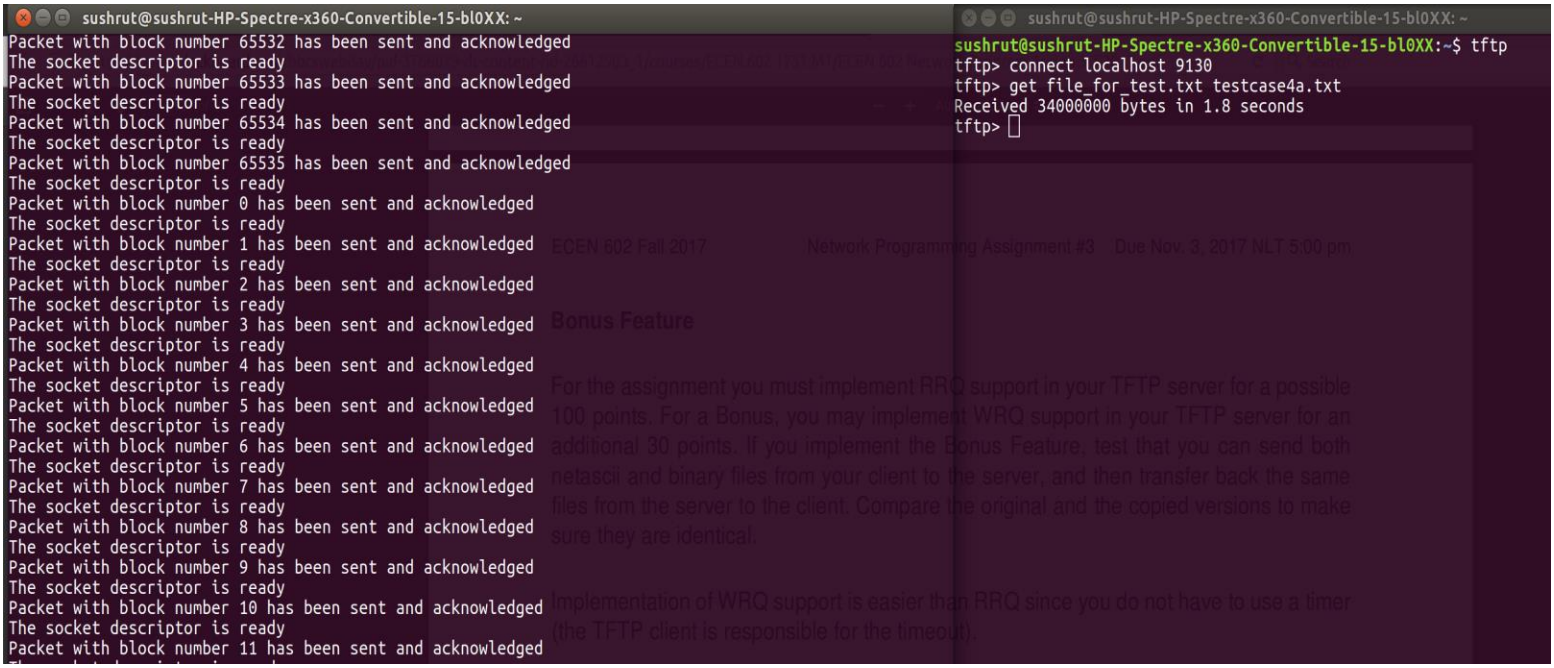
From the screenshots below it can be observed that the difference between the two files is turning out to be zero which proves that the read operation as executed successfully.



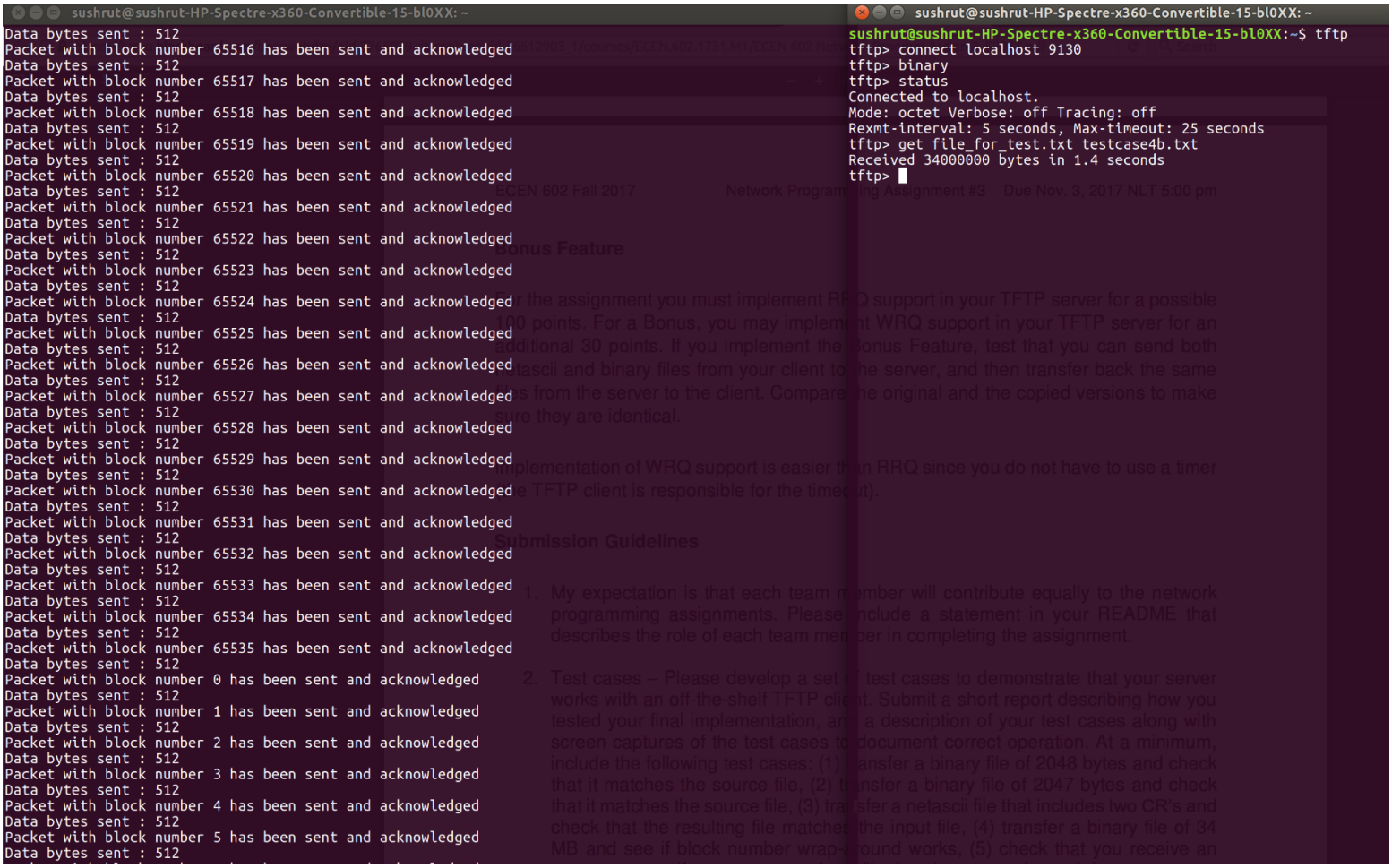


TEST CASE-4

A). transfer a binary file of 34 MB and see if block number wrap-around works- **NETASCII**



B). transfer a binary file of 34 MB and see if block number wrap-around works- **OCTET**

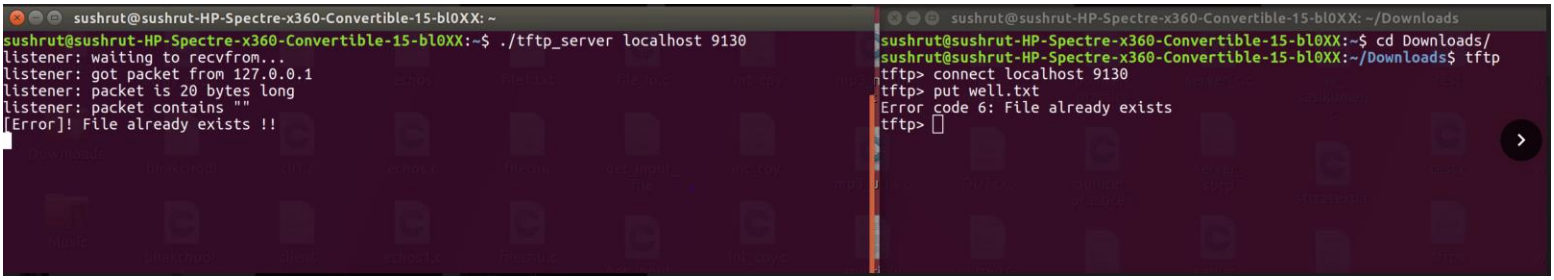


For test case 4, we created a file of size 34 MB (file_for_test.txt). The rest of the operations are pretty similar to first two test cases. The only difference being the wrap-around function. This can be clearly seen in that that after the block number 65535 the program again starts with the packet number 0.

Above screenshots show scenarios for the case of NETASCII and OCTET.

TEST CASE-5

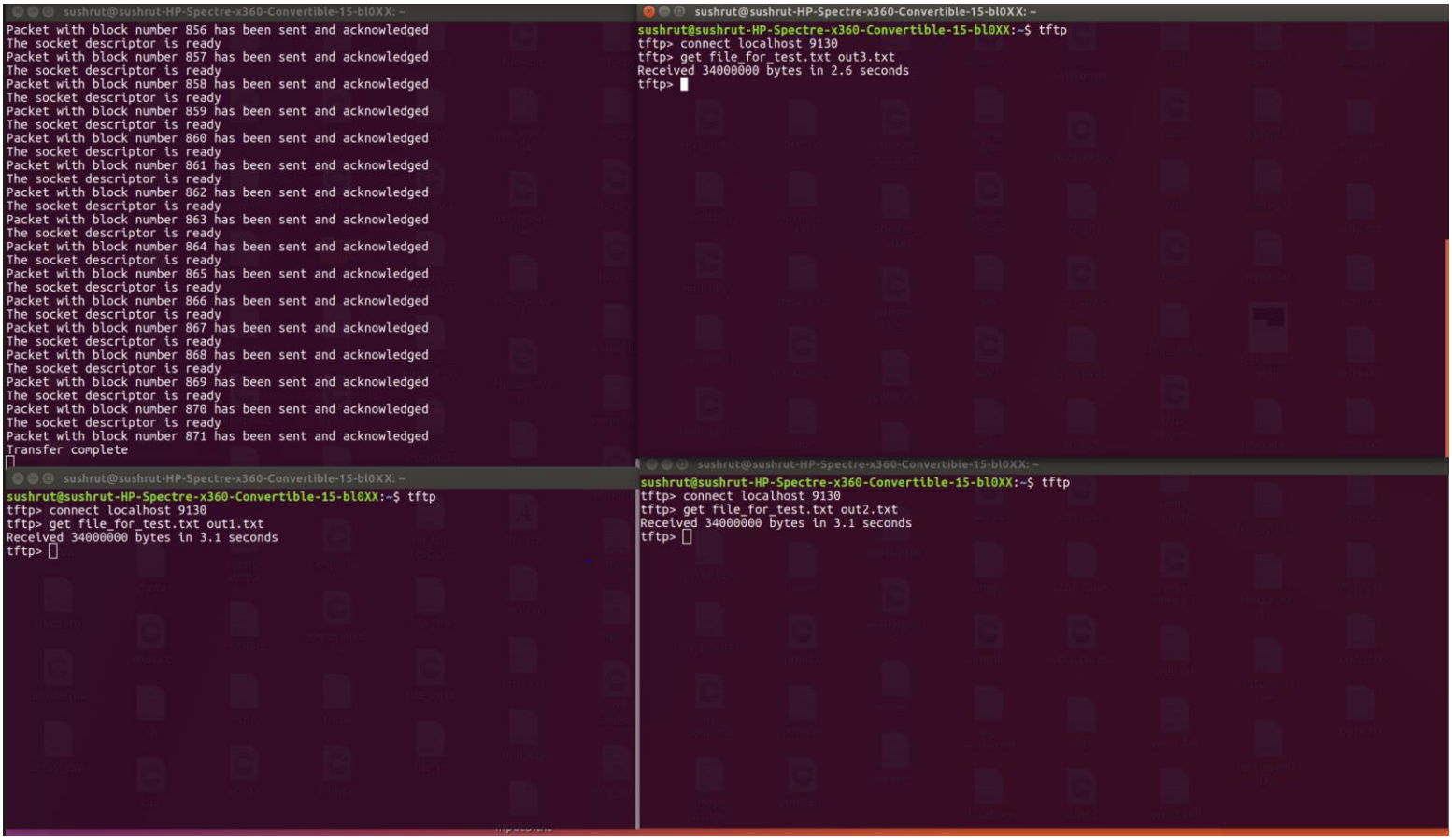
A). check that you receive an error message if you try to transfer a file that does not exist and that your server cleans up and the child process exits



Here we try to read a file (file_for_test.txt), which was not present in the server at that time. Which gives us an error message with error code1, which can be seen in the screenshots above.

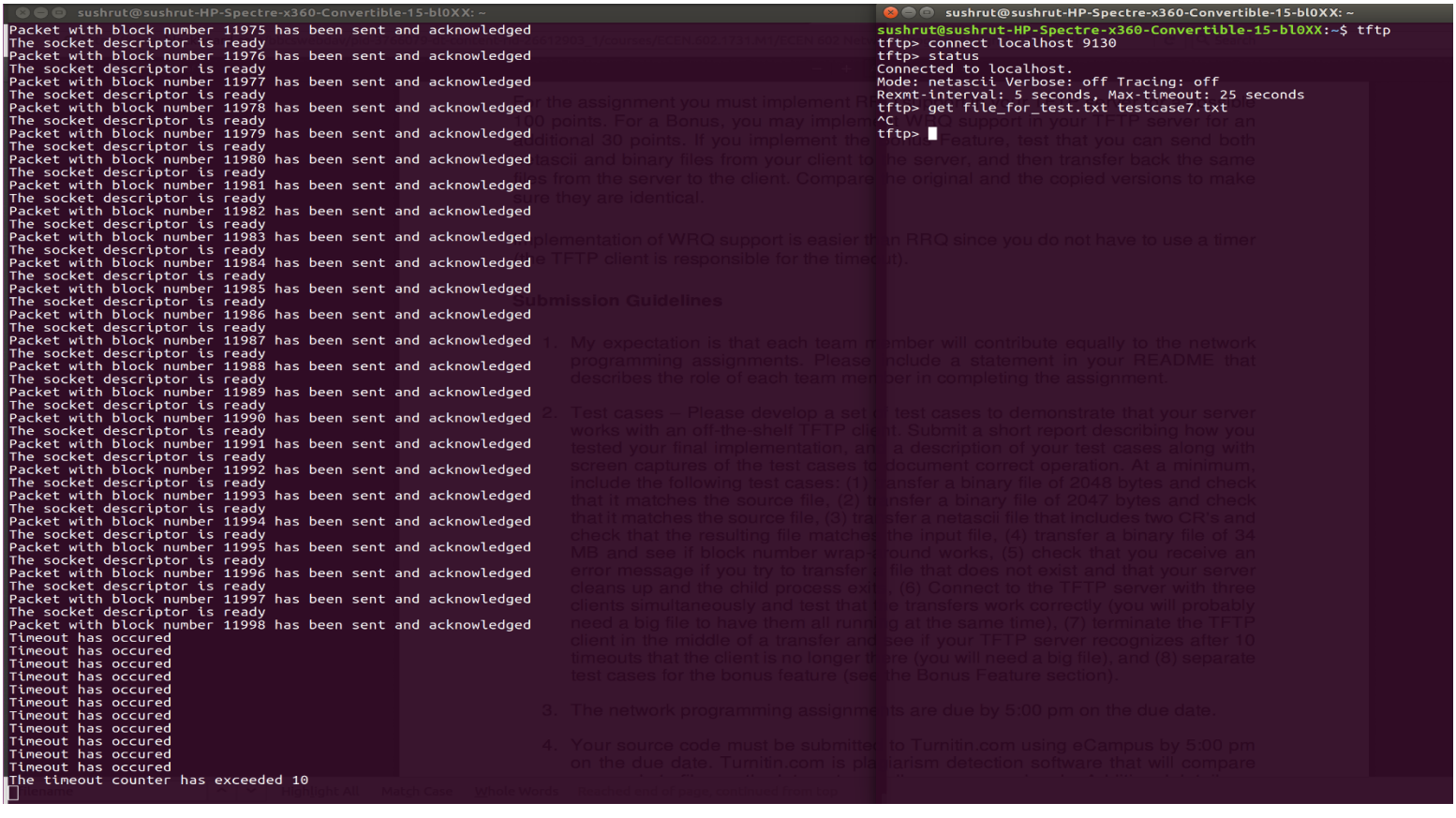
TEST CASE-6

A). Connect to the TFTP server with three clients simultaneously and test that the transfers work correctly (you will probably need a big file to have them all running at the same time)



TEST CASE-7

A). terminate the TFTP client in the middle of a transfer and see if your TFTP server recognizes after 10 timeouts that the client is no longer there (you will need a big file)



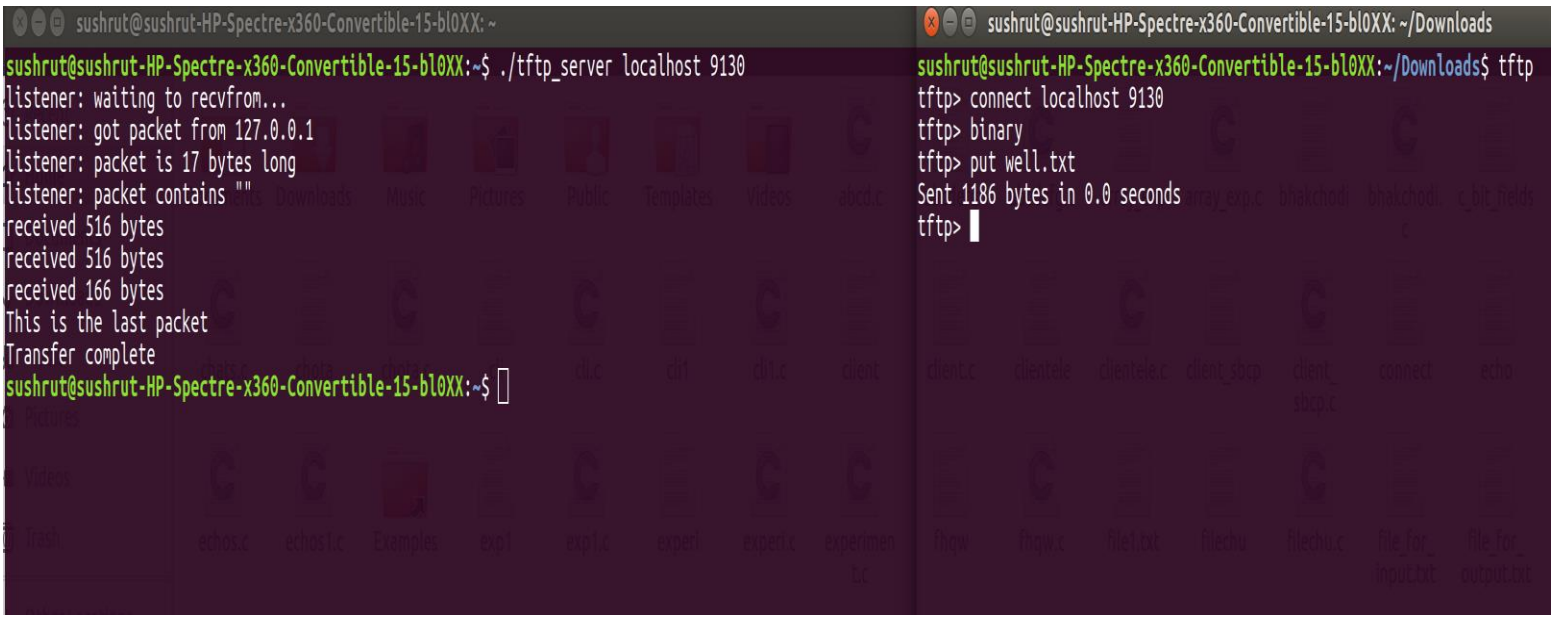
In this test case, we try to transfer a large file but abruptly kill our client process in the middle. Our server recognizes the absence of the client after 10 timeouts which can be clearly seen in the screenshot above.

TEST CASE-8

A). Bonus- test that you can send both netascii and binary files from your client to the server, and then transfer back the same files from the server to the client. Compare the original and the copied versions to make sure they are identical.

ASCII

OCTET



Here for this test case, we implemented the BONUS feature where server implements WRQ for both NETASCII & OCTET. It can be seen that the both the files have been transferred successfully in both the cases.