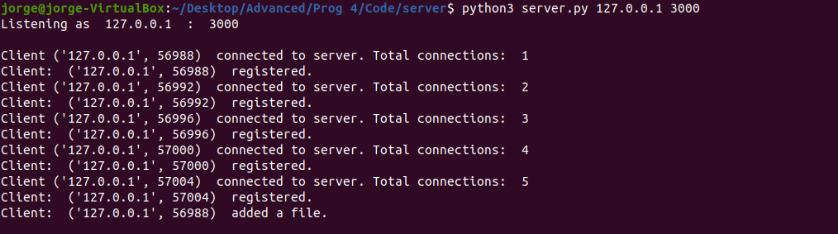
**Output files and Performance evaluation results:**

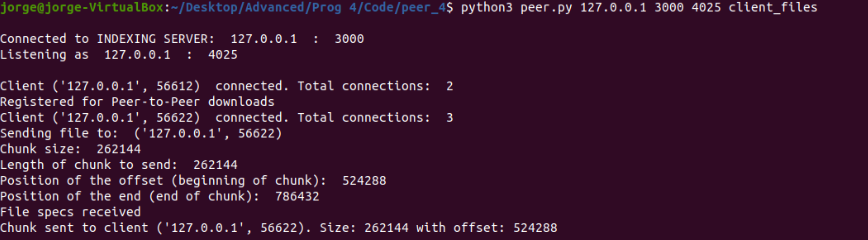
1. **Deploying at least 5 peers and 1 indexing server in the same virtual machine:**
2. Ensure you can transfer one file properly: No logs are going to be stored neither a bash script is going to be created. Just some screen captures are going to be shown.

Indexing server:



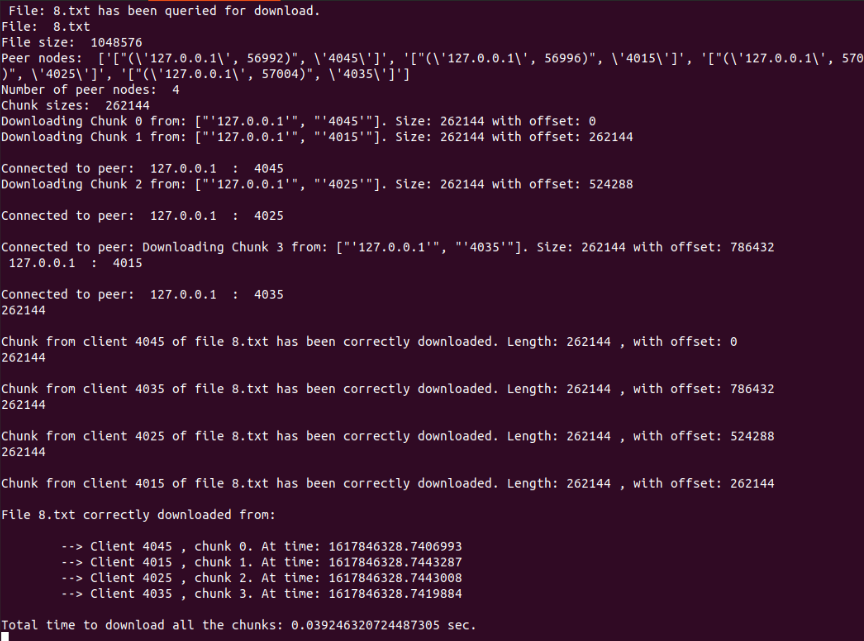
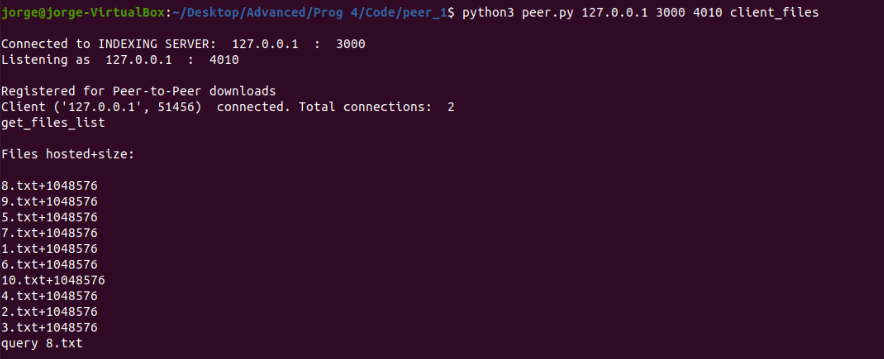
Sender Peer nodes:





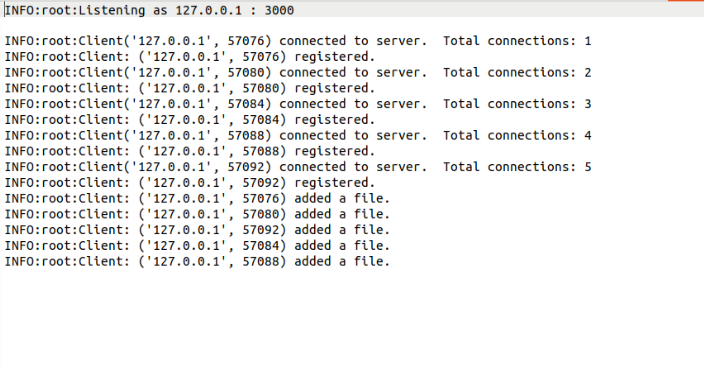


Receiver peer node:

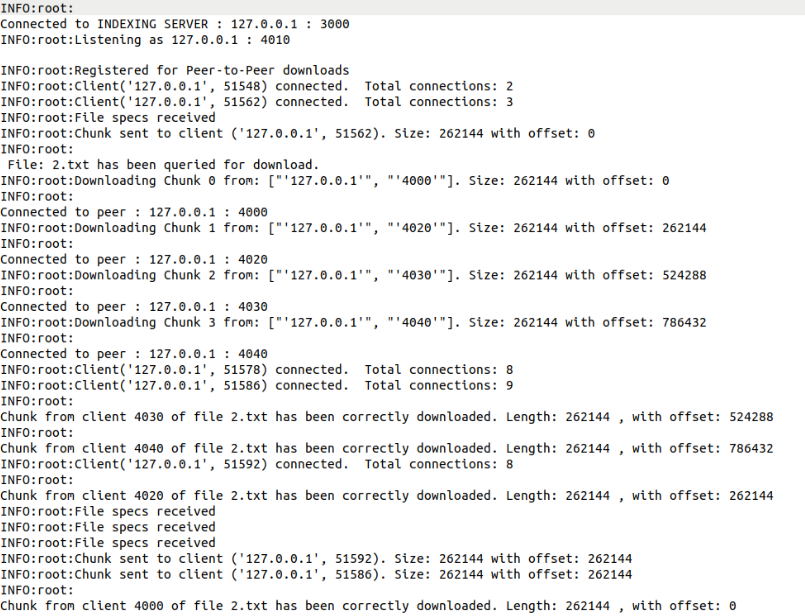


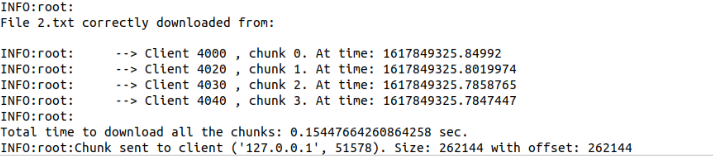
1. Ensure multiple peer nodes could simultaneously upload and download files: The logs of this evaluation are stored in the folder Out/ev 1. In this evaluation, 5 peers were connected to the Indexing server and each one of them requested a different file at the same time. The screenshots of the server and 2 peers logs are shown next:

Indexing server:

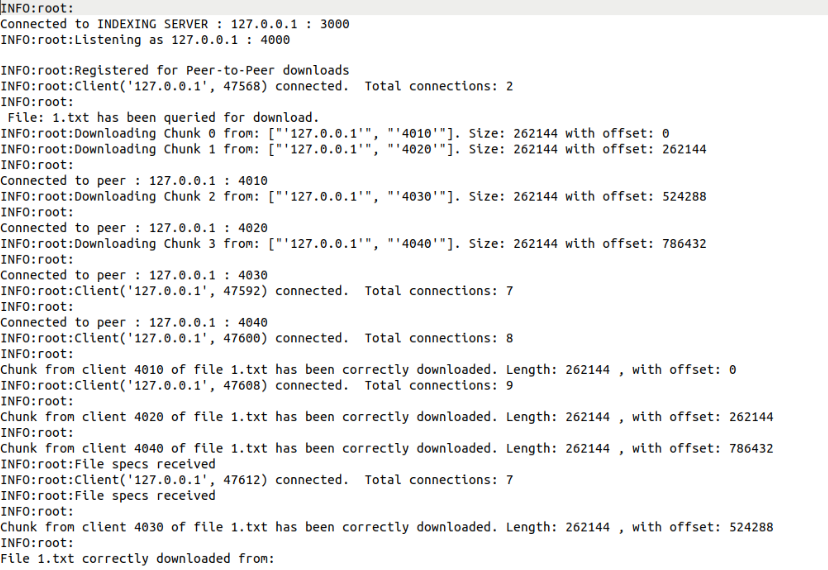


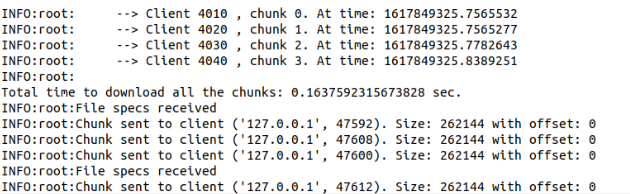
One peer:





Another peer:





As it can be seen in the previous figures, multiple nodes can download and send files between them without any problems.

1. **Measuring how the transfer speed changes when varying the number of nodes that holds the same requested file.**

To perform this evaluation, a bash script has been created to automate the process by creating different screens and run the server and all the peers simultaneously:

q2.sh <num\_peers>

The bash script deploys the indexing server, <num\_peers> (2, 4, 8 or 16) peers that will send the files and an additional peer that will request 10 files of 1 MB each (1.txt … 10.txt) to download from the rest of the peers.

Finally, the results that have been obtained are the following:

As it can be seen in the Figure, the average response time with just 2 peers is the worst of them all and increasing the number of peers improves the transfer time. However, when the number of peers is too large (16) the transfer of the files starts getting slower. It is important to note, that the files transferred are split into the same number of chunks as peers registered for P2P downloads in the Indexing server, and all the chunks are downloaded in parallel.

These results show that using a parallel download with a moderate number of chunks improves the speed and performance of the file transfer. However, if the file is divided into too many small chunks, the transfer gets slower as the actual transfer time of the chunks does not improve as much as the time it takes to handle all those simultaneous connections.