

Implementation of a simple peer-to-peer (P2P) system with a distributed index(DI)

Submitted by:

Rajshree Jain - rjain27

Prashanthi Kannappan Murthy - pkannia

The exact format of the messages exchanged between the peers and the RS and between peers:

&* is the separator used

REGISTER - command sent

P2P-DI/1.0 - version of the protocol used

Rajshrees-MacBook-Pro.local - host of the machine running

Darwin - Host OS

65452 - PORT at which the client is running

For peer-to-RS communication:

1) REGISTER:

1) A client sends the following message to the register to register for the first time :

REGISTER&*P2P-DI/1.0&*Rajshrees-MacBook-Pro.local&*Darwin&*65453

2) Registration server replies back with the message :

P2P-DI/1.0&*200&*OK&*Rajshrees-MacBook-Pro.local&*Darwin&*1

version - status code - status phrase - host - os - cookie

3) When the client registers for the second time, it sends the cookie in the request it sends:

REGISTER&*P2P-DI/1.0&*Rajshrees-MacBook-Pro.local&*Darwin&*65453*
1
version - host - os - port - cookie

2) KEEPALIVE:

KEEPALIVE&*P2P-DI/1.0&*1&*Rajshrees-MacBook-Pro.local&*Darwin

command - version - cookie - host - os

3) LEAVENETWORK :

LEAVE&*P2P-DI/1.0&*1&*Rajshrees-MacBook-Pro.local&*Darwin

command - version - cookie - host - os

4) PQUERY :

1) Request :

PQUERY&*P2P-DI/1.0&*2&*Rajshrees-MacBook-Pro.local&*Darwin

command - version - cookie - host - os

2) Reply :

The server sends you the entire list of peer index which is active

For peer-to-peer communication:

1. RFCQuery :

1) Request :

GET&*RFC-Index&*P2P-DI/1.0&*Rajshrees-MacBook-Pro.local&*Darwin

2) Respond :

Server sends you the RFC-Index that it has

2. GetRFC:

1) Request :

GET&*RFC&*768&*P2P-DI/1.0&*Rajshrees-MacBook-Pro.local&*Darwin

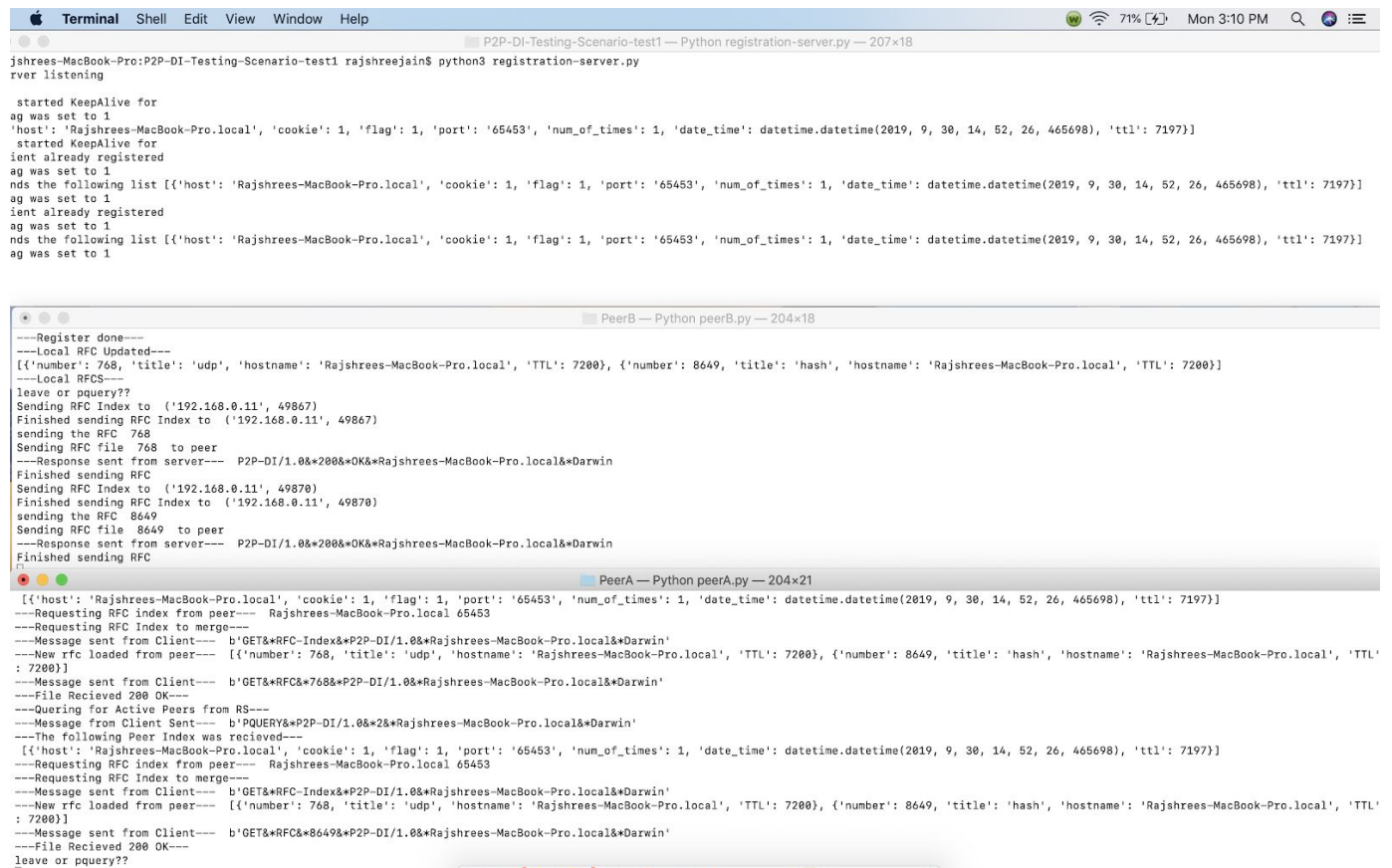
method - RFC - Number - version - host - os

2) Response :

P2P-DI/1.0&*200&*OK&*Rajshrees-MacBook-Pro.local&*Darwin&*<file>

version - status code - status phrase - host - os - file data

The entire conversation of the transaction can be seen as shown in the screenshot :



```
Terminal Shell Edit View Window Help
P2P-DI-Testing-Scenario-test1 — Python registration-server.py — 207x18
jshrees-MacBook-Pro:~$ python3 registration-server.py
rver listening

started KeepAlive for
ag was set to 1
'host': 'Rajshrees-MacBook-Pro.local', 'cookie': 1, 'flag': 1, 'port': '65453', 'num_of_times': 1, 'date_time': datetime.datetime(2019, 9, 30, 14, 52, 26, 465698), 'ttl': 7197}}
started KeepAlive for
ient already registered
ag was set to 1
nds the following list [{'host': 'Rajshrees-MacBook-Pro.local', 'cookie': 1, 'flag': 1, 'port': '65453', 'num_of_times': 1, 'date_time': datetime.datetime(2019, 9, 30, 14, 52, 26, 465698), 'ttl': 7197}}]
ag was set to 1
ient already registered
ag was set to 1
nds the following list [{'host': 'Rajshrees-MacBook-Pro.local', 'cookie': 1, 'flag': 1, 'port': '65453', 'num_of_times': 1, 'date_time': datetime.datetime(2019, 9, 30, 14, 52, 26, 465698), 'ttl': 7197}}]
ag was set to 1

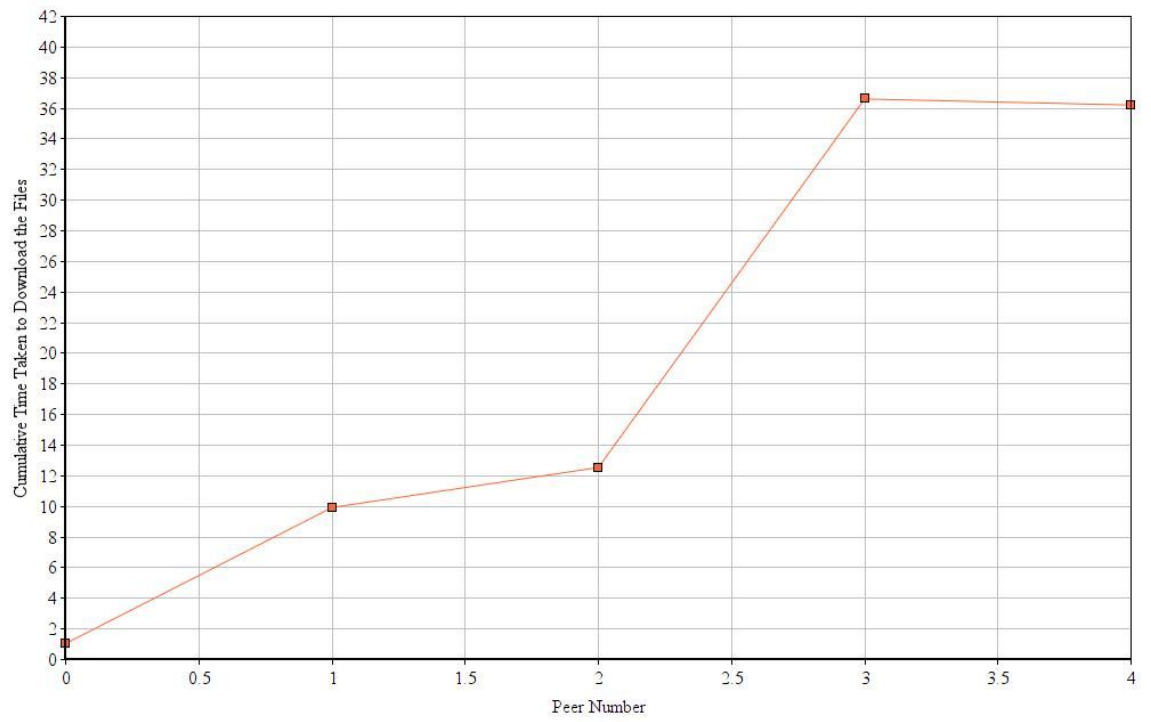
PeerB — Python peerB.py — 204x18
---Register done---
---Local RFC Updated---
[{'number': 768, 'title': 'udp', 'hostname': 'Rajshrees-MacBook-Pro.local', 'TTL': 7200}, {'number': 8649, 'title': 'hash', 'hostname': 'Rajshrees-MacBook-Pro.local', 'TTL': 7200}]
---Local RFCs---
leave or pquery??
Sending RFC Index to ('192.168.0.11', 49867)
Finished sending RFC Index to ('192.168.0.11', 49867)
sending the RFC 768
Sending RFC file 768 to peer
---Response sent from server--- P2P-DI/1.0&*200&*OK&*Rajshrees-MacBook-Pro.local&*Darwin
Finished sending RFC
Sending RFC Index to ('192.168.0.11', 49870)
Finished sending RFC Index to ('192.168.0.11', 49870)
sending the RFC 8649
Sending RFC file 8649 to peer
---Response sent from server--- P2P-DI/1.0&*200&*OK&*Rajshrees-MacBook-Pro.local&*Darwin
Finished sending RFC

PeerA — Python peerA.py — 204x21
[{'host': 'Rajshrees-MacBook-Pro.local', 'cookie': 1, 'flag': 1, 'port': '65453', 'num_of_times': 1, 'date_time': datetime.datetime(2019, 9, 30, 14, 52, 26, 465698), 'ttl': 7197}}]
---Requesting RFC index from peer--- Rajshrees-MacBook-Pro.local 65453
---Requesting RFC index to merge---
---Message sent from Client--- b'GET&RFC-Index&*P2P-DI/1.0&*Rajshrees-MacBook-Pro.local&*Darwin'
---New rfc loaded from peer--- [({'number': 768, 'title': 'udp', 'hostname': 'Rajshrees-MacBook-Pro.local', 'TTL': 7200}, {'number': 8649, 'title': 'hash', 'hostname': 'Rajshrees-MacBook-Pro.local', 'TTL': 7200})]
---Message sent from Client--- b'GET&RFC&*768&*P2P-DI/1.0&*Rajshrees-MacBook-Pro.local&*Darwin'
---File Recieved 200 OK---
---Querying for Active Pears from RS---
---Message from Client Sent--- b'PQUERY&*P2P-DI/1.0&*28&*Rajshrees-MacBook-Pro.local&*Darwin'
---The following Peer Index was recieved---
[{'host': 'Rajshrees-MacBook-Pro.local', 'cookie': 1, 'flag': 1, 'port': '65453', 'num_of_times': 1, 'date_time': datetime.datetime(2019, 9, 30, 14, 52, 26, 465698), 'ttl': 7197}}]
---Requesting RFC index from peer--- Rajshrees-MacBook-Pro.local 65453
---Requesting RFC index to merge---
---Message sent from Client--- b'GET&RFC-Index&*P2P-DI/1.0&*Rajshrees-MacBook-Pro.local&*Darwin'
---New rfc loaded from peer--- [({'number': 768, 'title': 'udp', 'hostname': 'Rajshrees-MacBook-Pro.local', 'TTL': 7200}, {'number': 8649, 'title': 'hash', 'hostname': 'Rajshrees-MacBook-Pro.local', 'TTL': 7200})]
---Message sent from Client--- b'GET&RFC&*8649&*P2P-DI/1.0&*Rajshrees-MacBook-Pro.local&*Darwin'
---File Recieved 200 OK---
leave or pquery??
```

Link to a Screen Cast video which shows how the Testing Scenario works successfully:

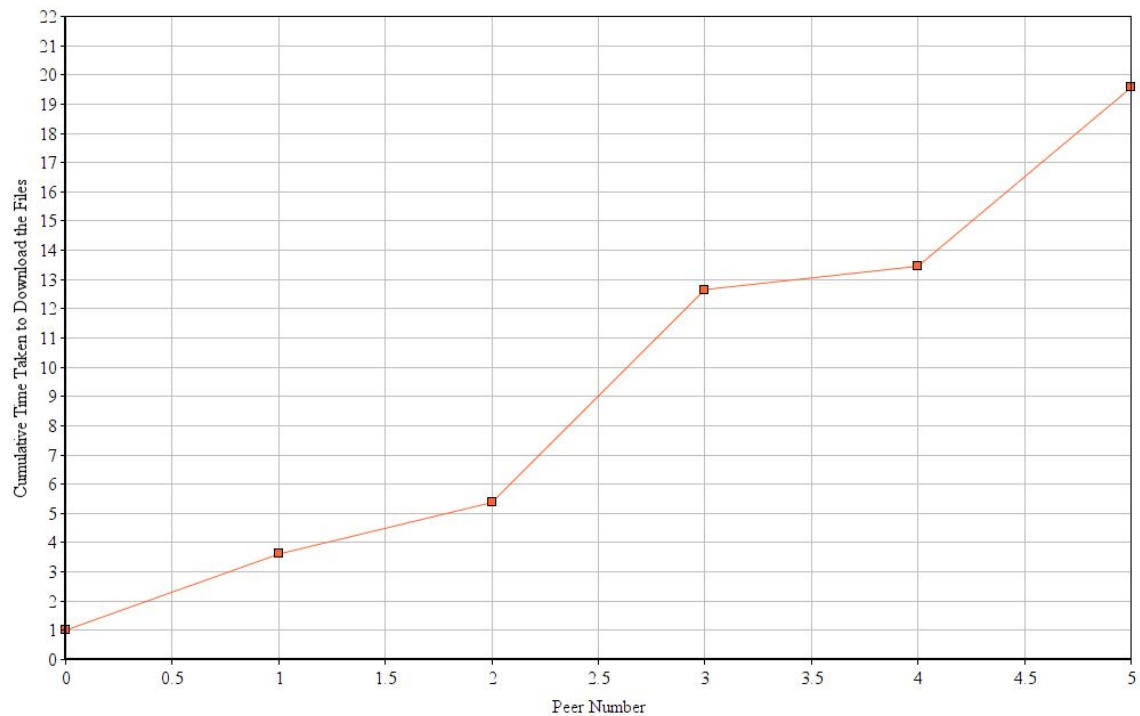
The download time curves for Task 1:

Task 1 Graph



the download time curves for Task 2:

Task 2 Graph



Discussion of the differences you observed in the above download time curves and any conclusions you may draw regarding the scalability of P2P versus centralized systems for file downloading:

1) Centralized systems can give way to a central point of failure. When we tried out our code first with centralized, we had a small syntax error in Peer0 and because of that, the entire network was stale and not receiving anything as the system (Peer0) that had every file went down.

Whereas, if it was P2P, at least the other active peers could have done partial downloading of files that are available with the other active peers.

Also, we could observe a drastic difference in the total times where the total times for distribution of files reduced a lot. This must-have happened because the files were distributed and there was no load on one peer for distributing files.

2) In a centralized system, we started the systems one by one. The first system downloaded relatively faster, as more requests were placed, the downloads were slower as compared to the P2P system.

3) Finding content in a P2P is difficult. For eg, when the test scenario was tested, we had to start PeerB first, so PeerA knows that there is an active peer with the RFC it wanted. For large scale systems, this may be difficult.