INTERNET PROTOCOLS

PROJECT 1

Peer to Peer with Distributed Index System for Downloading RFCs

BHARATH VENKATESH
Student ID: 001037220
SRIDEVI JANTLI
Student ID: 200019504

Objective:

- Creating a Peer to Peer distribution system using Distributed Index for the purpose of downloading Request for Comments.[RFC].
- Creating a Registration Server that listens to all clients and provides them with list of active peers.
- Creating a Peer Server that listens to all other peer requests and responds with the RFC index and RFC files.
- Creating a Peer Client that carries out requests for RS and other Peer servers to obtain Active Peer lists, exchange RFC Indexes and download RFCs.
- Maintaining a distributed index that assists in handling multiple peer status and RFC file status.
- Using application protocol to communicate between server and peers using TCP sockets.

Design:

- Registration Server: The RS is a multi-threaded application that runs on a well-defined port [65423]. It handles requests from multiple clients and performs the four functions.
    - Register: Registering peer information.
    - PQuery: Providing the calling peer with a list of active peers that includes peer name, port number and IP address.
    - Leave: Indication that the peer is no longer active.
    - KeepAlive: Indication that peer wishes to remain active.
- Peer Server: A muti-threaded application that serves to requests from other clients. It handles sending of RFC Indexes that it maintains and the requested RFC files
- Peer Client: This modules concerns with three functionalities.
    - Requesting list of active peers from RS.
    - Requesting RFCIndex from active peers and merging them with its own index.
    - Downloading RFC files from the target peer servers.

Implementation:

Programming Language: Java 1.7 [Using open-jdk]

- The Registration Server is implemented as a multi-threaded socket program which persistently listens to request from peers. Each peer request is handled by a different thread.
- The Peer is of type *PeerObject* class and holds { peer name, cookie, active flag, time-to-live, port number, active count, date and ipaddress }
- The time to live field is implemented as a TimerTask that decrements every second for all peer objects.
- The RS maintains a list of PeerObjects and dynamically creates an ActivePeerList which it shares with the requested Peer.

- The ActivePeerObject is a minimal object definition and holds {peer name, port number and ip address.}
- The Peer Server is implemented as a multi-threaded socket program which persistently listens to request from other peers. Each request is handled by a different thread.
- It handles sending of RFC files and RFC index over socket connection to the requesting peer.
- The peer initially populates its RFCIndex list with all the files it contains in its directory.
- The RFCIndex is a class and holds { peer name, rfc title, time-to-live, rfc number}
- The time to live field is implemented as a TimerTask. It is constant for the RFC files that the peer holds and decrements every second for the indexes it obtains from other peers.
- Peer Client implements the task of requesting active peer list from the RS, sending out requests to each active peer for their index list, merging it with its local list and sending download request to the target peer to obtain the RFC files.
- Each of these communications is done through sockets and following standard application protocols.

Message Formats:

Peer to RS Communication:

- Registration Request:
  Syntax:

  ```
  GET REGISTER <peername> <portnumber> <ip address> P2P-DI/1.0
  Host: <hostname>
  OS: <Platform>
  ```

  Example:

  ```
  GET REGISTER PeerA 65411 152.1.42.150 P2P-DI/1.0
  Host: dan200-150-l.eos.ncsu.edu
  OS: Linux
  ```

- Registration Response: On Success
  Syntax:

  ```
  P2P-DI/1.0 <status code> <phrase> <cookie>
  Host: <hostname>
  OS: <Platform>
  ```

  Example:

  ```
  P2P-DI/1.0 200 OK 2345678
  Host: dan200-171-l.eos.ncsu.edu
  ```

```
OS: Linux
```

- Registration Response: On Failure [Peer name already exists]
  Syntax:

  ```
  P2P-DI/1.0 <status code> <phrase>
  Host: <hostname>
  OS: <Platform>
  ```

  Example:

  ```
  P2P-DI/1.0 409 CONFLICT
  Host: dan200-171-l.eos.ncsu.edu
  OS: Linux
  ```

- Registration Response: On Request for re-register
  Syntax:

  ```
  P2P-DI/1.0 <status code> <phrase> <cookie>
  Host: <hostname>
  OS: <Platform>
  ```

  Example:

  ```
  P2P-DI/1.0 409 CONFLICT 2345678
  Host: dan200-171-l.eos.ncsu.edu
  OS: Linux
  ```

- Leave Request:
  Syntax:

  ```
  GET LEAVE <cookie> <portnumber> P2P-DI/1.0
  Host: <hostname>
  OS: <Platform>
  ```

  Example:

  ```
  GET LEAVE 2345778 65413 P2P-DI/1.0
  Host: dan200-163-l.eos.ncsu.edu
  OS: Linux
  ```

- Leave Response - On Success
  Syntax:

  ```
  P2P-DI/1.0 <status code> <phrase>
  Host: <hostname>
  ```

OS: <Platform>

Example:

```
P2P-DI/1.0 200 OK
Host: dan200-171-l.eos.ncsu.edu
OS: Linux
```

- Leave Response - On Failure [Peer Not Found]
  Syntax:

  ```
  P2P-DI/1.0 <status code> <phrase>
  Host: <hostname>
  OS: <Platform>
  ```

  Example:

  ```
  P2P-DI/1.0 404 NOT FOUND
  Host: dan200-171-l.eos.ncsu.edu
  OS: Linux
  ```

- KeepAlive Request:
  Syntax:

  ```
  GET KEEPALIVE <cookie> <portnumber> P2P-DI/1.0
  Host: <hostname>
  OS: <Platform>
  ```

  Example:

  ```
  GET KEEPALIVE 2345778 65413 P2P-DI/1.0
  Host: dan200-163-l.eos.ncsu.edu
  OS: Linux
  ```

- KeepAlive Response - On Success
  Syntax:

  ```
  P2P-DI/1.0 <status code> <phrase>
  Host: <hostname>
  OS: <Platform>
  ```

  Example:

  ```
  P2P-DI/1.0 200 OK
  Host: dan200-171-l.eos.ncsu.edu
  ```

```
OS: Linux
```

- KeepAlive Response - On Failure [Peer Not Found]

  Syntax:

  ```
  P2P-DI/1.0 <status code> <phrase>
  Host: <hostname>
  OS: <Platform>
  ```

  Example:

  ```
  P2P-DI/1.0 404 NOT FOUND
  Host: dan200-171-l.eos.ncsu.edu
  OS: Linux
  ```

- PQUERY Request:

  Syntax:

  ```
  GET PQUERY <cookie> <portnumber> P2P-DI/1.0
  Host: <hostname>
  OS: <Platform>
  ```

  Example:

  ```
  GET PQUERY 2345778 65413 P2P-DI/1.0
  Host: dan200-163-l.eos.ncsu.edu
  OS: Linux
  ```

- PQUERY Response - On Success

  Syntax:

  ```
  P2P-DI/1.0 <status code> <phrase>
  Host: <hostname>
  OS: <Platform>

  <data> <data> <data> …..
  ```

  Example:

  ```
  P2P-DI/1.0 200 OK
  Host: dan200-171-l.eos.ncsu.edu
  OS: Linux
  Peername : PeerB       PortNumber : 65413     IPAddress : 152.1.42.163
  ```

- PQUERY Response - On Failure [No Active Peer found]
  Syntax:

  ```
  P2P-DI/1.0 <status code> <phrase>
  Host: <hostname>
  OS: <Platform>
  ```

  Example:

  ```
  P2P-DI/1.0 404 NOT FOUND
  Host: dan200-171-l.eos.ncsu.edu
  OS: Linux
  ```

Peer to Peer Communication:

- RFCIndex Request:
  Syntax:

  ```
  GET RFCINDEX P2P-DI/1.0
  Host: <hostname>
  OS: <Platform>
  ```

  Example:

  ```
  GET RFCINDEX P2P-DI/1.0
  Host: dan200-163-l.eos.ncsu.edu
  OS: Linux
  ```

- RFCIndex Response: [On Success]
  Syntax:

  ```
  P2P-DI/1.0 <status code> <phrase>
  Host: <hostname>
  OS: <Platform>
  <data> <data> <data> …..
  ```

  Example:

  ```
  P2P-DI/1.0 200 OK
  Host: dan200-171-l.eos.ncsu.edu
  OS: Linux
  Peername : PeerB      RfcName : rfc6501    RfcNumber : 6501
  Peername : PeerB      RfcName : rfc6502    RfcNumber : 6502
  ```

- RFCIndex Response - On Failure [List is empty]
  Syntax:

  ```
  P2P-DI/1.0 <status code> <phrase>
  Host: <hostname>
  OS: <Platform>
  ```

  Example:

  ```
  P2P-DI/1.0 404 NOT FOUND
  Host: dan200-171-l.eos.ncsu.edu
  OS: Linux
  ```

- Request for RFC Download
  Syntax:

  ```
  GET RFC <rfcnumber> P2P-DI/1.0
  Host: <hostname>
  OS: <Platform>
  ```

  Example:

  ```
  GET RFC 6501 P2P-DI/1.0
  Host: dan200-163-l.eos.ncsu.edu
  OS: Linux
  ```

- RFC Download Response: [On Success]
  Syntax:

  ```
  P2P-DI/1.0 <status code> <phrase>
  Host: <hostname>
  OS: <Platform>
  Content Type: <content-type>
  Content Length: <bytes>
  <data> <data> <data> …..
  ```

  Example:

  ```
  P2P-DI/1.0 200 OK
  Host: dan200-163-l.eos.ncsu.edu
  OS: Linux
  Content Type : application/pdf
  Content Length : 20562
  <RFC File Content comes here>
  ```

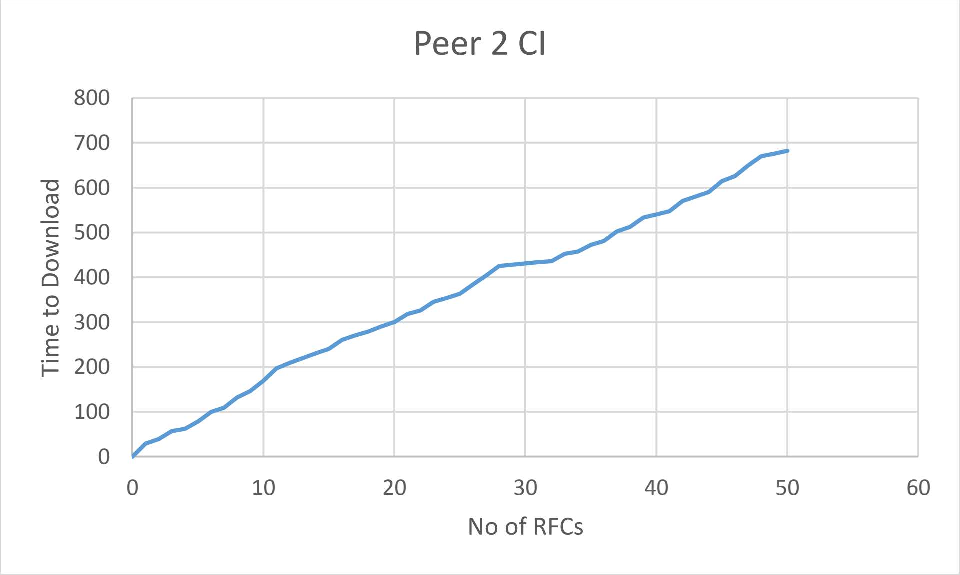
- Response for RFC Download- On Failure [File not found]
  Syntax:

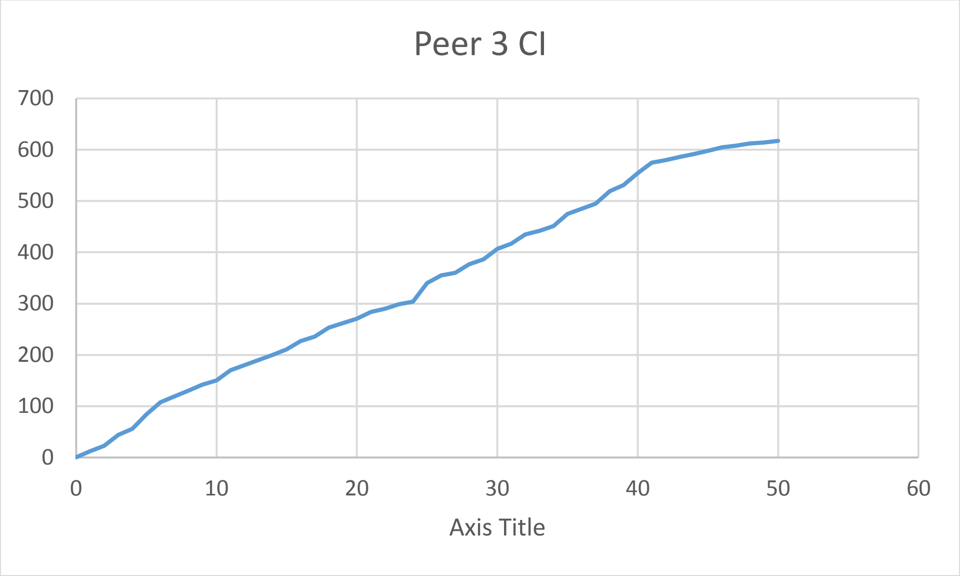
  P2P-DI/1.0 <status code> <phrase>
  Host: <hostname>
  OS: <Platform>

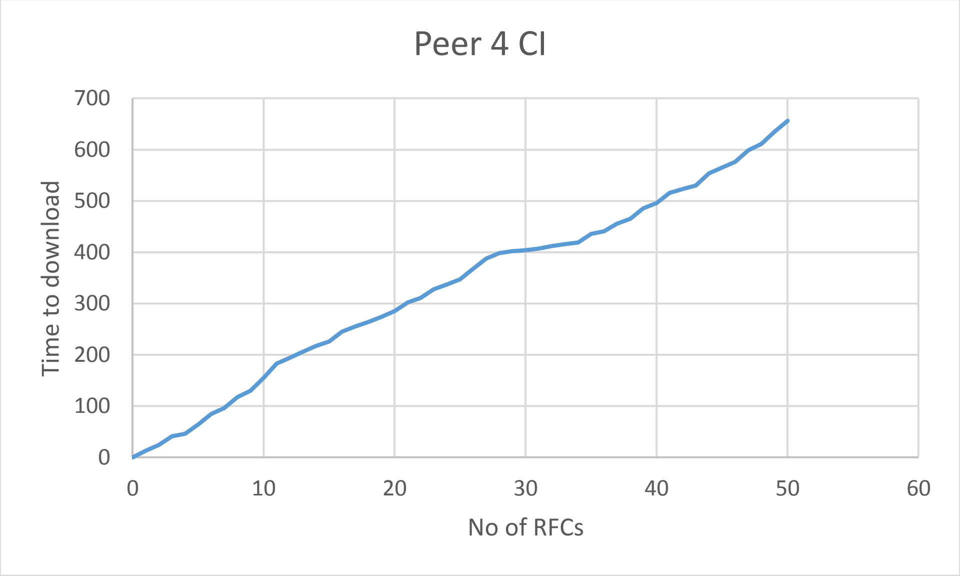
  Example:

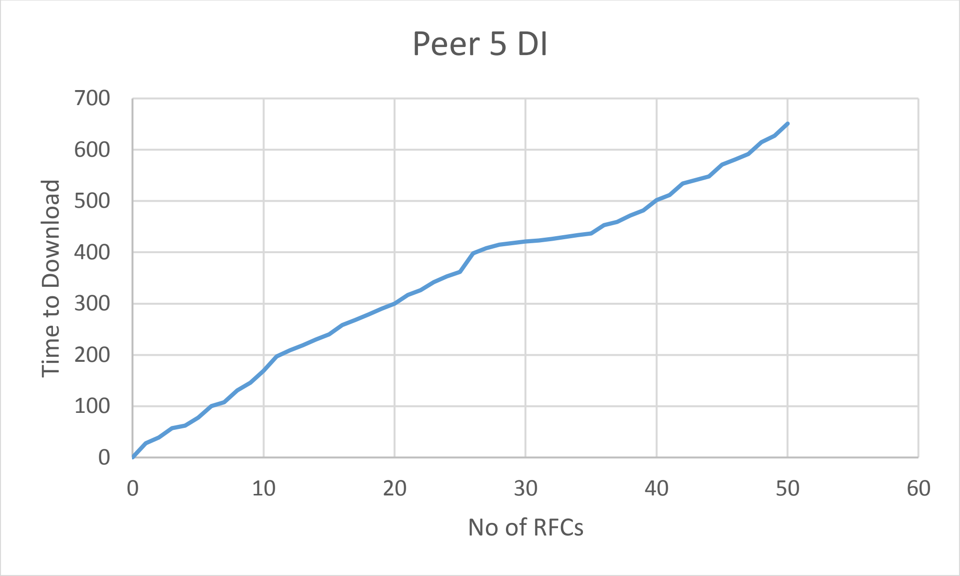
  P2P-DI/1.0 404 NOT FOUND
  Host: dan200-171-l.eos.ncsu.edu
  OS: Linux

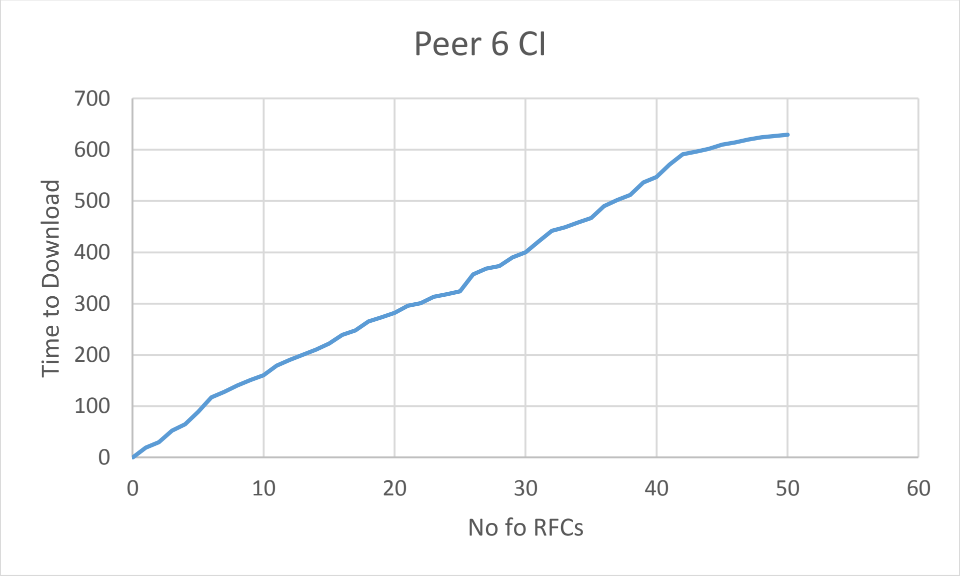
Cumulative Plots for Centralized Index

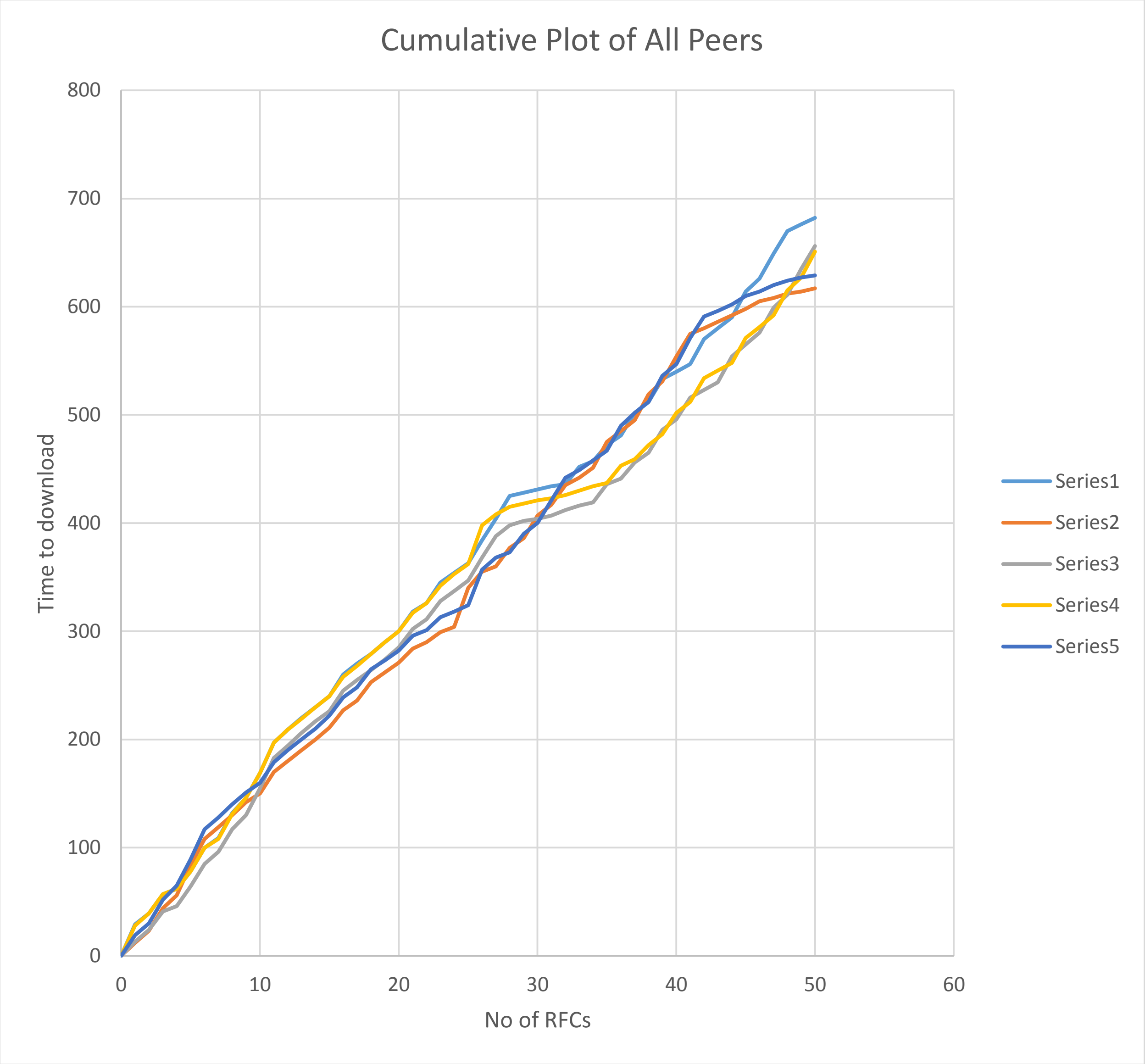



Peer 3 CI
700
600
500
400
300
200
100
0
0
10
20
30
40
50
60
Axis Title

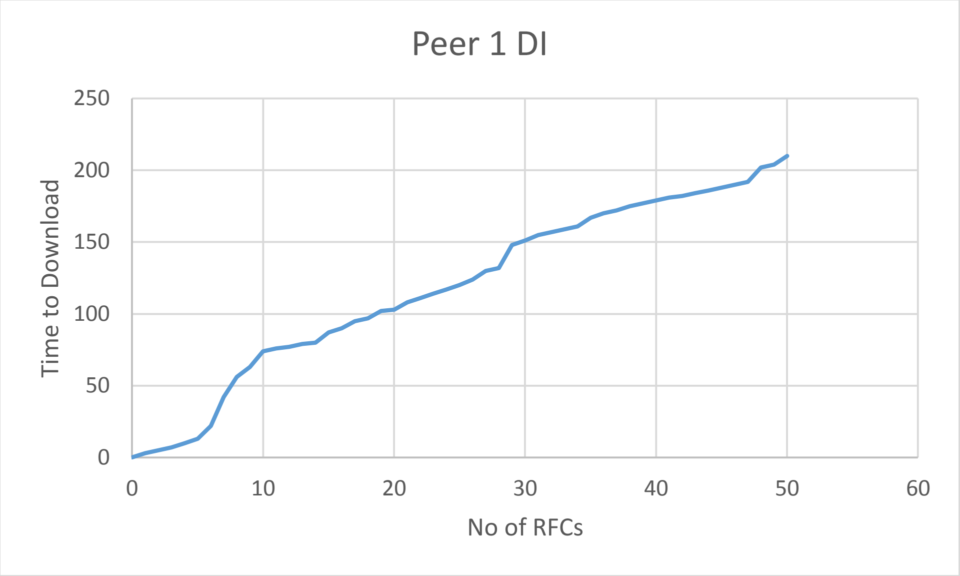

Peer 4 CI
Time to download
700
600
500
400
300
200
100
0
0
10
20
30
40
50
60
No of RFCs

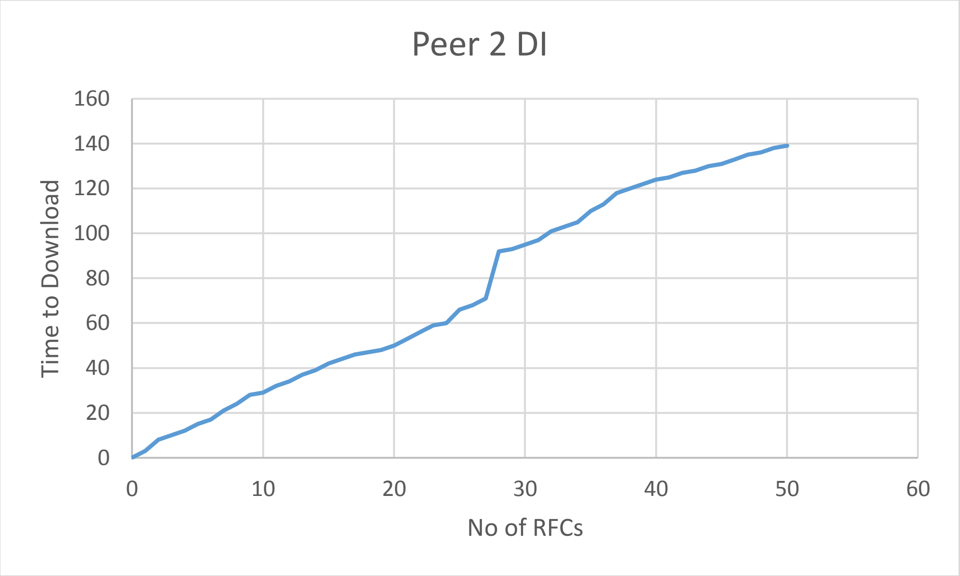

Peer 5 DI
700
600
500
400
300
200
100
0
0
10
20
30
40
50
60
Time to Download
No of RFCs

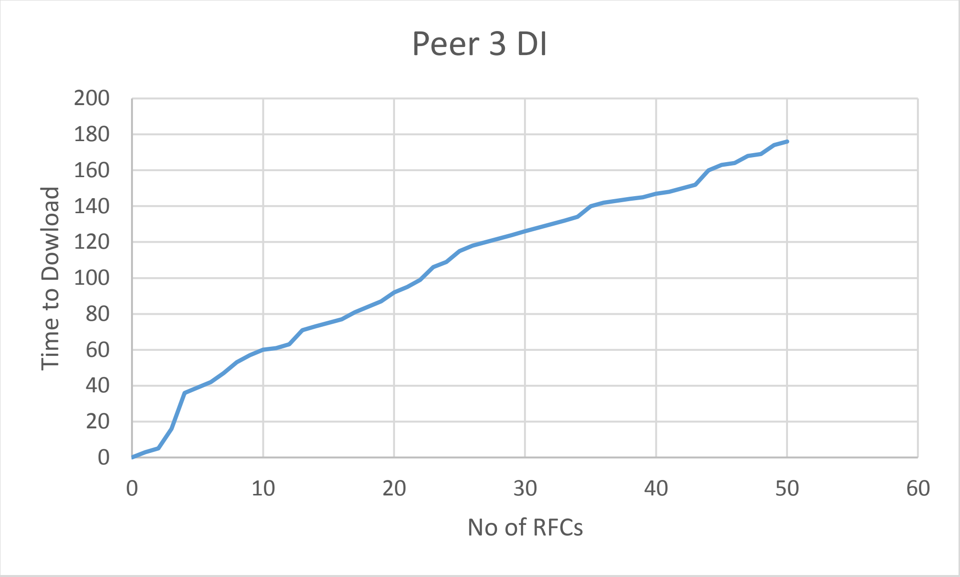

Peer 6 CI
700
600
500
400
300
200
100
0
0
10
20
30
40
50
60
Time to Download
No fo RFCs

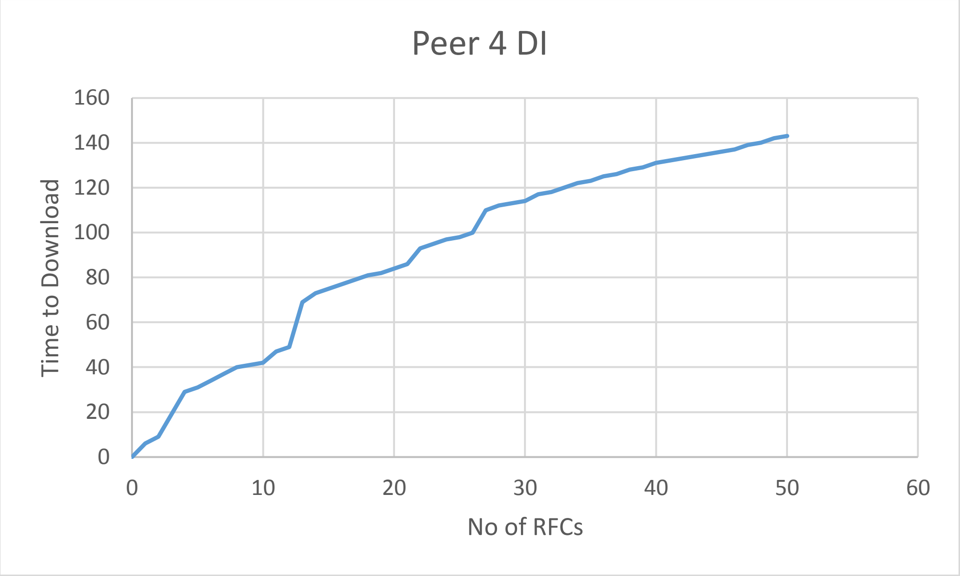


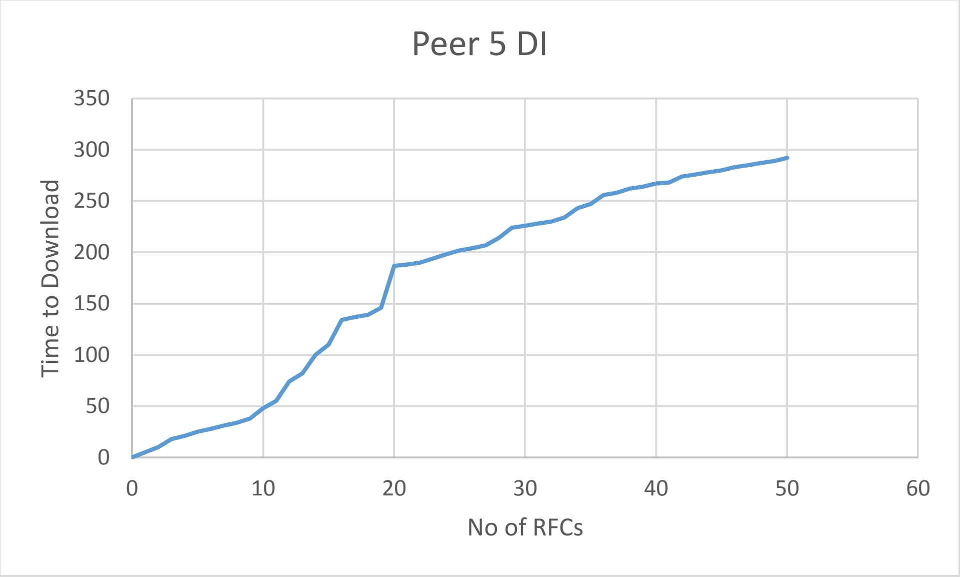
Cumulative Plots for Distributed Index

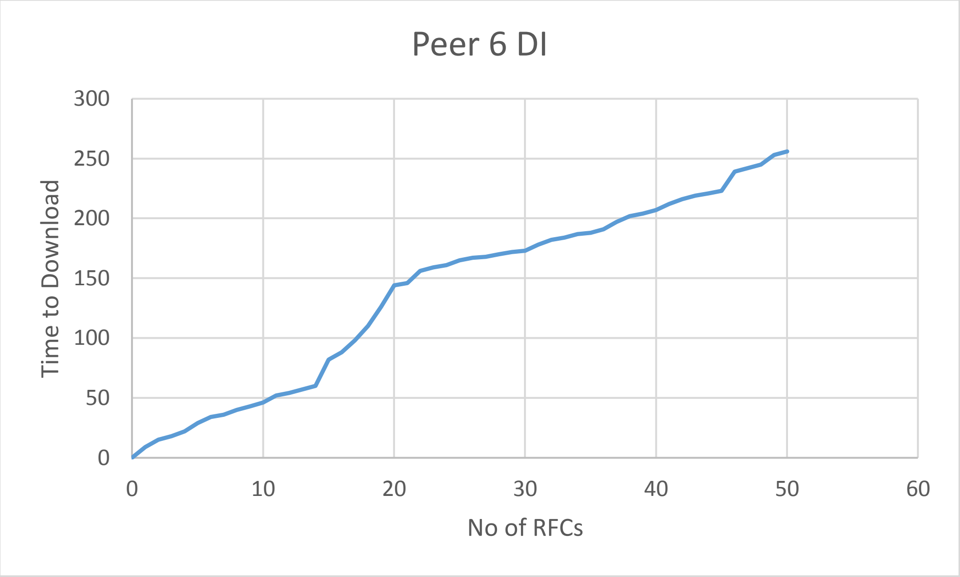

Peer 1 DI
Time to Download
250
200
150
100
50
0
0
10
20
30
40
50
60
No of RFCs

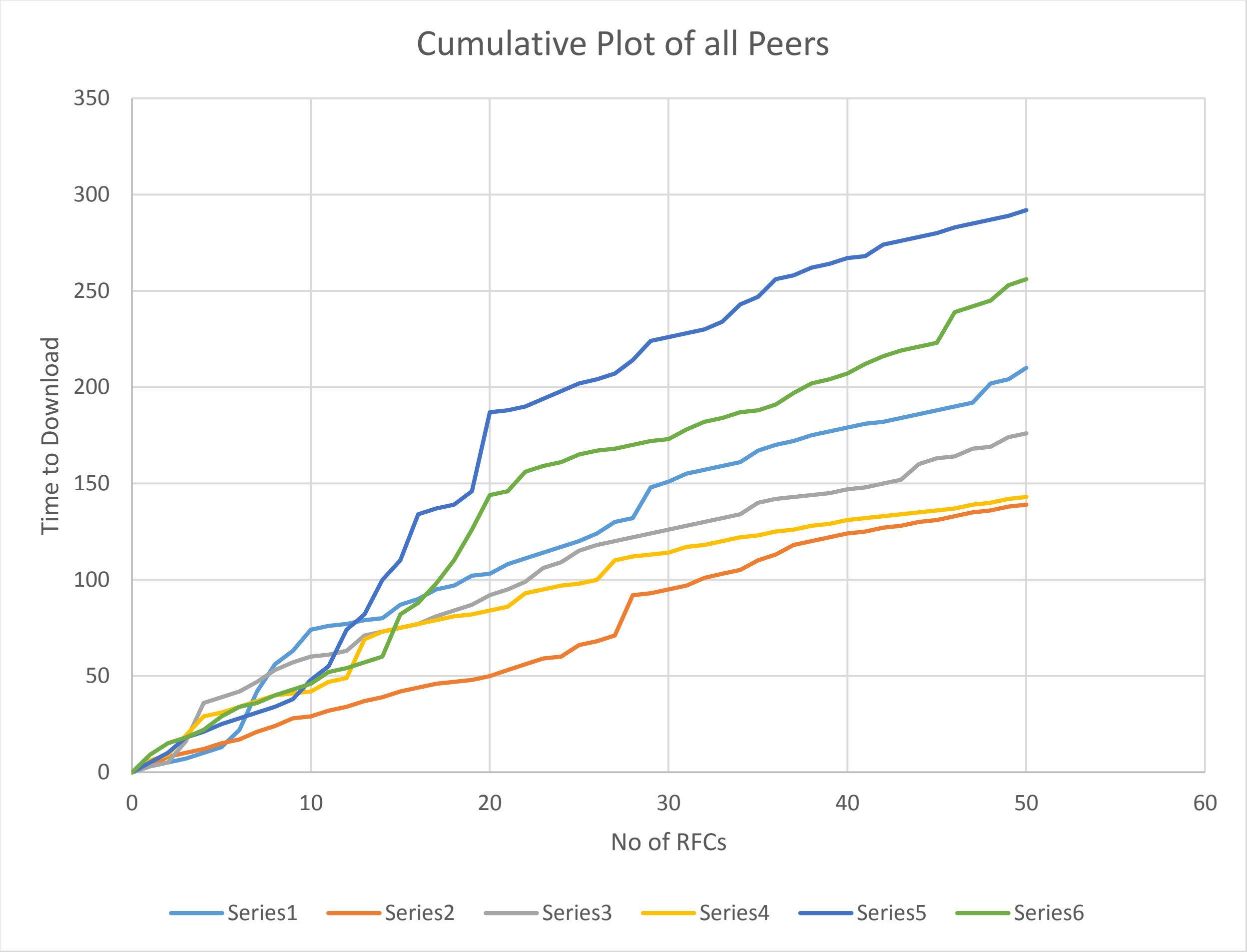

Peer 2 DI
Time to Download
160
140
120
100
80
60
40
20
0
0
10
20
30
40
50
60
No of RFCs


Peer 3 DI
Time to Dowload
200
180
160
140
120
100
80
60
40
20
0
0
10
20
30
40
50
60
No of RFCs


Peer 4 DI
Time to Download
160
140
120
100
80
60
40
20
0
0
10
20
30
40
50
60
No of RFCs


Peer 5 DI
Time to Download
0
50
100
150
200
250
300
350
0
10
20
30
40
50
60
No of RFCs


Peer 6 DI
Time to Download
0
50
100
150
200
250
300
0
10
20
30
40
50
60
No of RFCs


Cumulative Plot of all Peers
350
300
250
200
150
100
50
0
Time to Download
0
10
20
30
40
50
60
No of RFCs
Series1
Series2
Series3
Series4
Series5
Series6