P2P SYSTEM WITH DISTRIBUTED INDEX

Shaurya Garg (sgarg7)

Shashank Jha (sjha5)

MESSAGE FORMATS BETWEEN SERVER/PEERS

Our message format is based on HTTP protocol and the sample format provided in project description. We have used pair of <cr> <lf> in between headers and data to differentiate between them. Also, the last set of <cr><lf> marks the end of the message. This is useful when we are extracting valid parameters from the messages.

Sample request message:

```
Method <sp>  Protocol  <cr> <lf>
Header <sp>  Value <cr> <lf>
Header <sp>  Value <cr> <lf>
<cr><lf>
<cr><lf>
```

Sample response message:

```
Protocol <sp> Status <cr> <lf>
Header <sp>  Value <cr> <lf>
Header <sp>  Value <cr> <lf>
<cr><lf>
Data
<cr><lf>
```

Below are the all the messages being exchanged:

RS-PEER TO RS-SERVER

Register:

```
Register  <sp>  P2P-DI/1.0  <cr> <lf>
COOKIE  <sp>  1  <cr> <lf>
RFCPORT  <sp>  65587  <cr> <lf>
<cr><lf>
<cr><lf>
```

We are not sending IP address explicitly as it can easily be extracted from socket. Hence, only RFCServer Port Number is sent in the message.

Also, if cookie number sent is 0, it means that new cookie number is to be assigned to the peer. Else, only same cookie number will be sent back in response.

PeerQuery:

```
PQuery  <sp>  P2P-DI/1.0  <cr> <lf>
COOKIE  <sp>  1  <cr> <lf>
<cr> <lf>
<cr> <lf>
```

Leave:

```
Leave  <sp>  P2P-DI/1.0  <cr> <lf>
COOKIE  <sp>  1 <cr> <lf>
<cr> <lf>
<cr> <lf>
```

KeepAlive

```
KeepAlive  <sp>  P2P-DI/1.0  <cr> <lf>
COOKIE  <sp>  1 <cr> <lf>
```

<cr> <lf>

<cr> <lf>

RS-SERVER TO RS-PEER

Reply - Register

This is sent in response to the register sent by the peer to RS-Server.

P2P-DI/1.0 <sp> 200 <cr> <lf>

COOKIE <sp> 1 <cr> <lf>

<cr> <lf>

<cr> <lf>

Reply – Pquery

This is sent in response to the pquery by peer. Response will be based on whether there are active peers present in the network (other than requesting peer). If present, status code is set to 200. Else, 400 is set.

If active peers are present:

P2P-DI/1.0 <sp> **200** <cr> <lf>

COOKIE <sp> 1 <cr> <lf>

<cr> <lf>

hostname_portNumber_cookie_hostname_portNumber_cookie_

<cr> <lf>

<cr> <lf>

Here, the data is list of peers present in the network. For simplicity, only two peers has been shown above.

If active peers are not present:

P2P-DI/1.0 <sp> **400** <cr> <lf>

COOKIE <sp> 1 <cr> <lf>

<cr> <lf>

<cr> <lf>

Reply – Leave

This one is sent in response to the leave request sent by peer to leave the network.

```
P2P-DI/1.0  <sp>  200  <cr> <lf>
COOKIE  <sp>  1 <cr> <lf>
<cr> <lf>
<cr> <lf>
```

Reply – KeepAlive

This message is response to the keep alive request sent by the peer to the server.

```
P2P-DI/1.0  <sp>  200  <cr> <lf>
COOKIE  <sp>  1 <cr> <lf>
<cr> <lf>
<cr> <lf>
```

RFC-CLIENT TO RFC-SERVER

This is the series of messages exchanged between RFCClient to RFCServer for quering about RFC and downloading particular file.

RFCQuery

This message is sent to query about RFCs present.

```
RFCQuery  <sp>  P2P-DI/1.0  <cr> <lf>
<cr> <lf>
<cr> <lf>
```

GetRFC

This message is sent when RFC has been located and request is sent to download the file.

```
GetRFC  <sp>  P2P-DI/1.0  <sp>  <cr> <lf>
RFCNUMBER  <sp>  10 <sp>  <cr> <lf>
<cr> <lf>
<cr> <lf>
```

RFC-SERVER TO RFC-CLIENT

Reply - RFCQuery

This message is sent by RFCServer to RFCClient in response to RFC Query request. The list present with it will be shared to the requestor in data segment of the message as follows:

```
P2P-DI/1.0  <sp>  200  <cr> <lf>
<cr> <lf>
rfcNumber_hostName_rfcNumber_hostName_
<cr> <lf>
<cr> <lf>
```

For simplicity, only two entries are shown above.

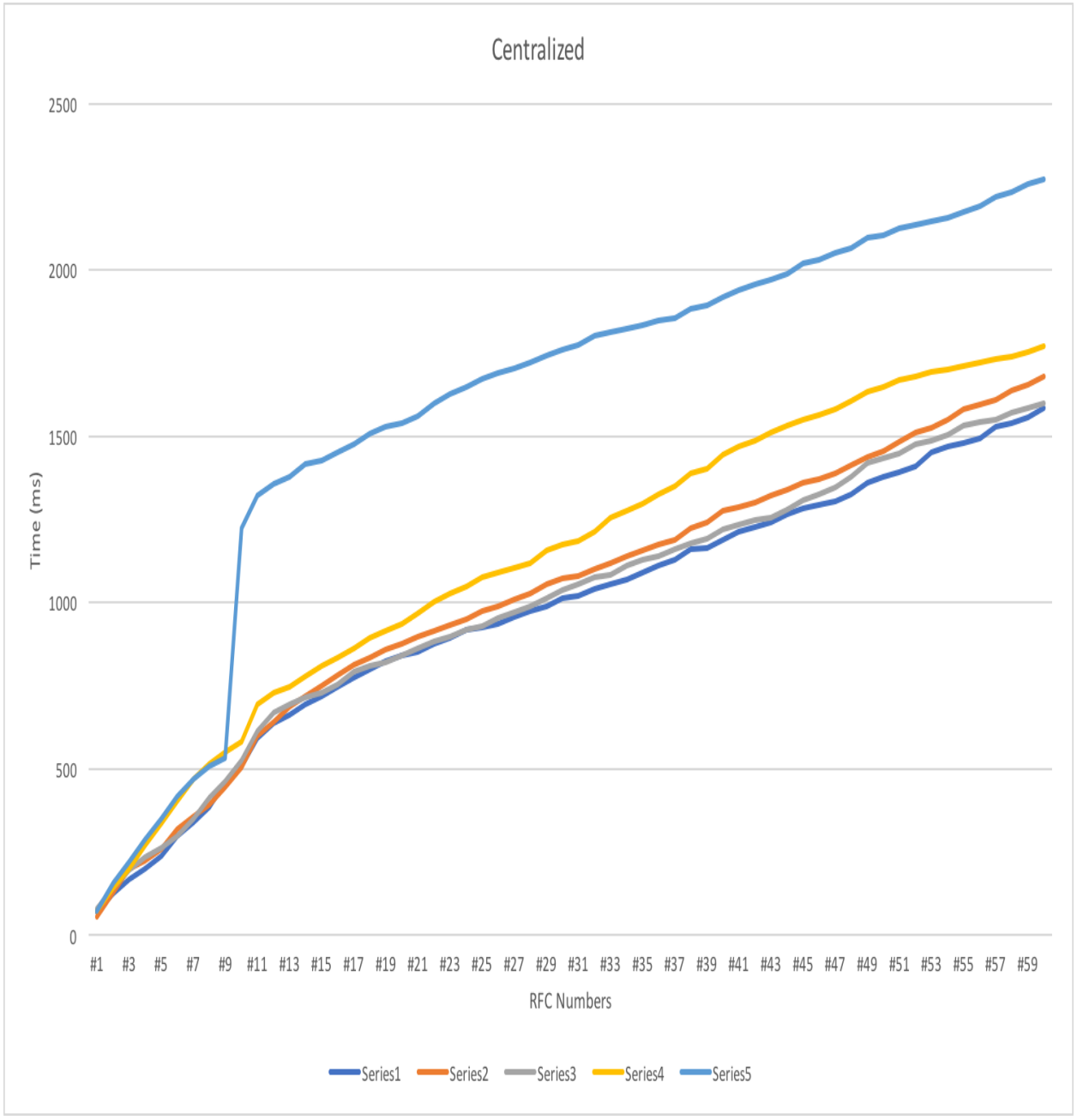
Reply - GetRFC

This message is sent by the RFCServer to requestor RFCClient along with the data from RFC file.

```
P2P-DI/1.0  <sp>  200  <cr> <lf>
<cr> <lf>
data
<cr> <lf>
<cr> <lf>
```

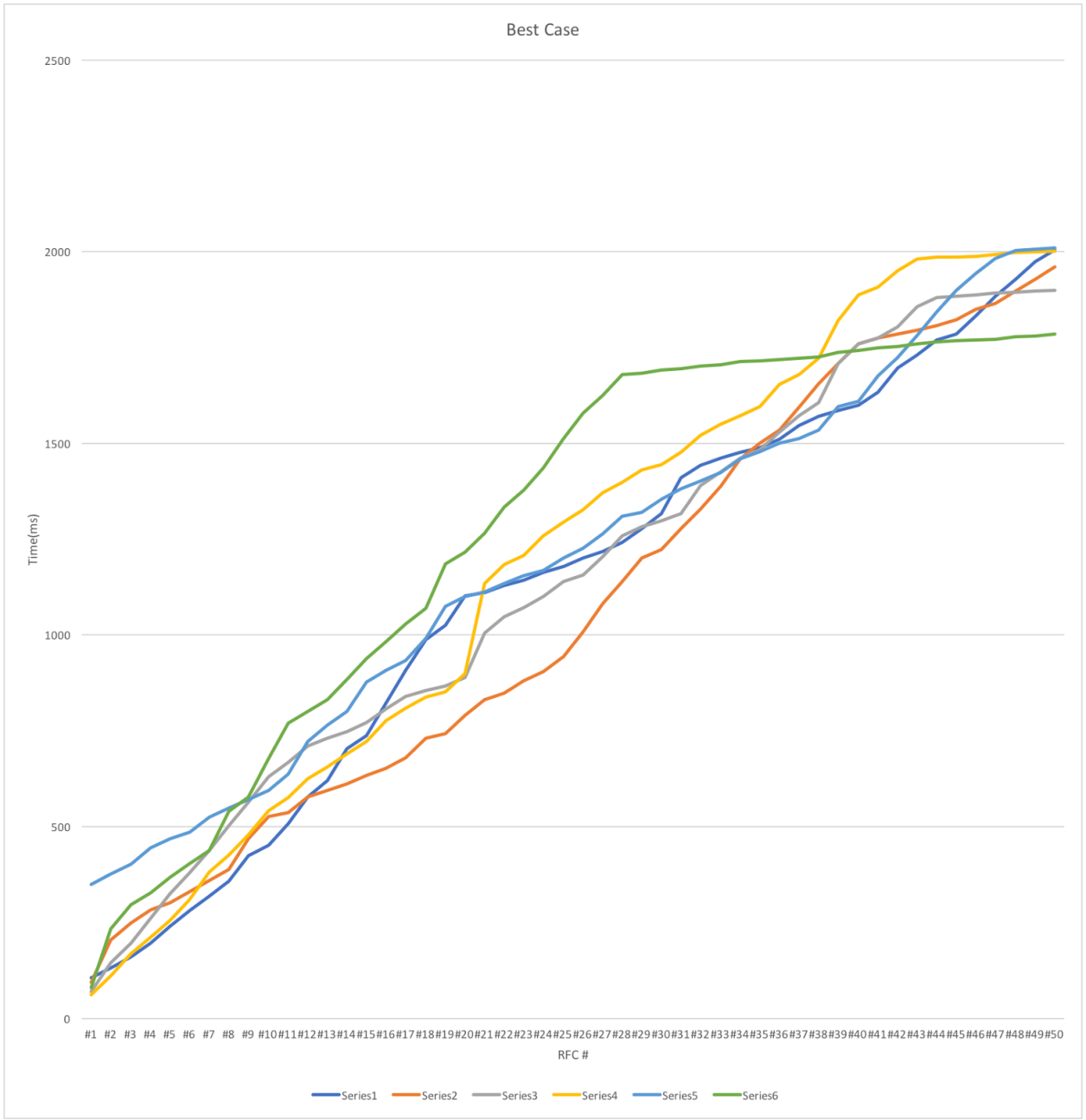
TEST1 (CENTRALIZED) RESULTS

We can see from the image that time keeps on increasing linearly. Series1, Series2 refer to different nodes.

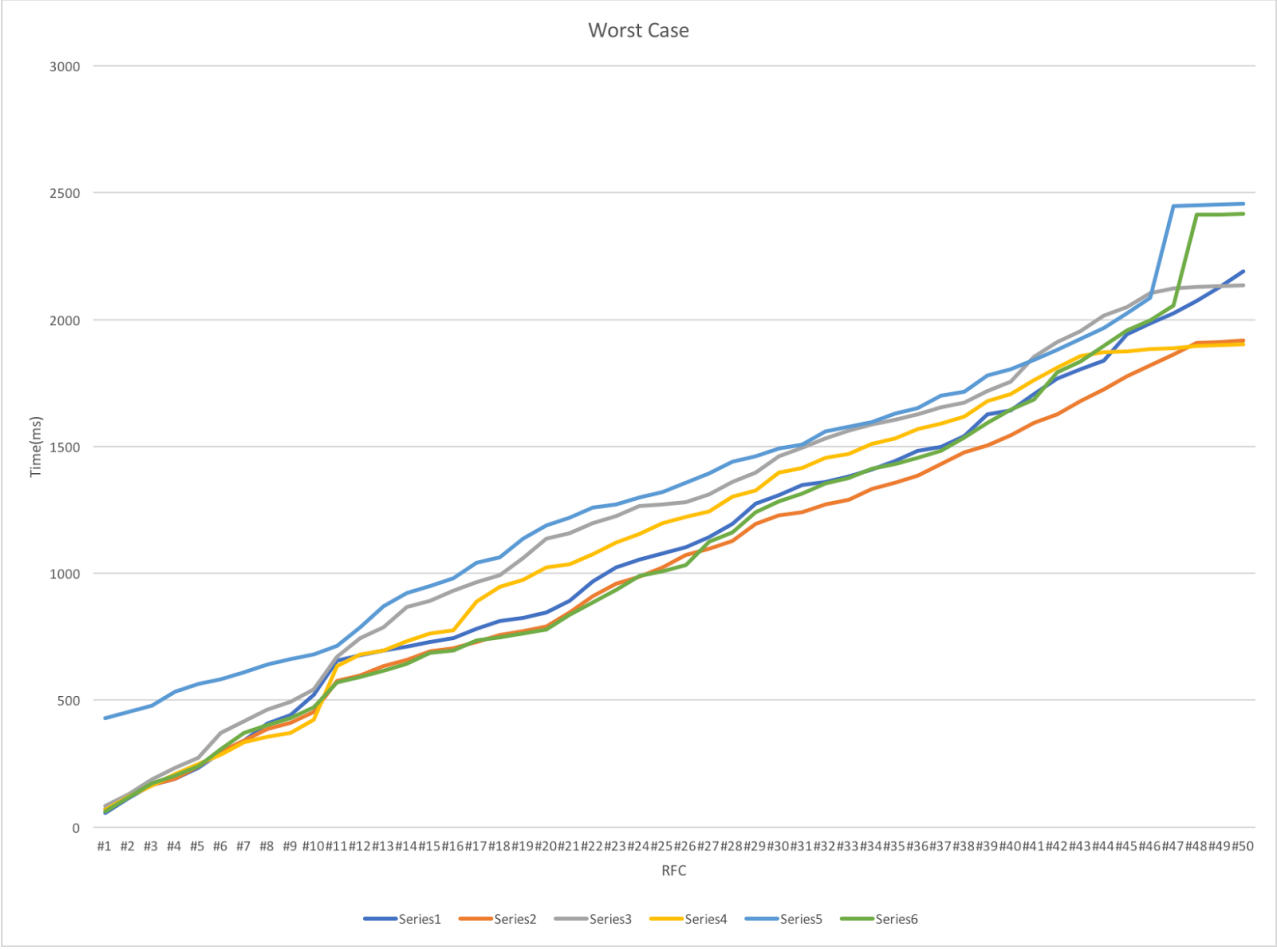


TEST2 RESULTS

Best Case



Worst Case:



Discussion:

As we can see from graphs above, worst case scenario of distributed file download is slightly higher than the best case. We created that scenario by making all the peers download RFC from 1 to 60. Hence, each peer will open connection to peer 1 first, then peer 2, etc. Hence, there is as much overlapping as possible.

For best case scenario, we changed the code a bit. That is, since peer 1 has already RFC 1 to 10, it will start download files from 11 to 60. Peer 2 already has files RFC 11 to 20, hence, it will download files from 21 to 60 and then 1 to 10. In this way, each peer will fetch from its next peer. Hence, we reduced overlapping as much as possible. The average download time in this case was better than the worst case.

We can easily see that the timings keep on growing linearly in case of centralized server where as in case of distributed files the curve towards the end becomes constant which is what we expected. Also in case of worst case of files being copied in distributed environment we still are able to achieve this curve its just that the initial timings are bad as most of the times all the hosts are requesting for the same copy of rfc from a single host.

Note: All the tests were performed on different hosts using DigitalOcean Platform.