DESIGN DOCUMENT

Created By:

Anuj Rastogi

anujrast@buffalo.edu

The document is an attempt to meet the question on **how we can prevent the other peers from filling our disk spaces and the design for SYNC Command.**

APPROACH FOR SYNC

***DESIGN for SYNC:** I am maintaining a struct containing the information on what all peers client is connected to. This struct is also containing the socket descriptors of the connections. As soon as the sync command is given each socket descriptors in the client struct is transferred a series of file from him.

FOR EG:

If A has three socket descriptors and 4 files in the directory the logic goes like:

```
Foreach Socket_fd in strcut

{

Foreach File in directory{

Send (Socket_fd, File);

}

}
```

NOTE*: Along with this, A also transmits SYNC command prefixing the message.

On the other side, say B, if there is an activity on the socket of A then B first analyses the header of the message. If it is SYNC then along with receiving the files B also runs a loop to transfer files to the A's Socket_fd.

FOR EG:

```
ForEach File in Directory
{
  Send(Socket_fd-A, SendFile);
}
```

HOW WE CAN PREVENT PEERS FROM FILLING YOUR DISK SPACES:

There are three ways in which we can do it.

1. As soon as a file is transferred the other per will know that which client is willing to transfer a file. Also, since I am attaching a header to the message specifying that the message or file is a result of SYNC so in this way the peer will know that the file transfer through SYNC is happening. Then at that instance we can prompt the client asking if he wants to accept the file contents, as they are through SYNC.
2. On the client side we can fix a buffer that specifies the size of the file it can receive. Depending upon the size of message this buffer can deny or accept the message.
3. Also, we can initially transmit the total size of all the files to the peer before every transmit and hence, can have receivers approval if he wants to participate in SYNC.

HOW WE CAN REVENT PEERS FROM ACCESSING THE FILE OUTSIDE OUR DIRECTORY:

Whenever a peer gives a file name it should be tokenized on delimiters such as /. Post tokenization this should take only the last term of token which is the NAME OF FILE. Hence in this way accessing outside the directory can be restricted.