Project Deliverable #A

Title: NetBin

Members: Anshul Jain, Marc Lindsay

Project Description: NetBin will be a service that will provide a scalable, reliable, and seamless peer to peer (P2P) file hosting service for local area networks (LAN). Users would interface with NetBin solely from a unix terminal window. On a computer where our small terminal application is installed, users would be able to execute a series of commands. There would be five primary commands which describes the core functionality of NetBin: upload to NetBin, download from NetBin, list files hosted on NetBin, remove a hosted file, and favorite a hosted file. The upload and download commands would use the syntax: NetBin < local_file, NetBin > hosted_file (respectively). The list files command would executed simply with: NetBin list. Just as the 'ls,' which is shorthand for list, command on an unix terminal would output the files in the user's current directory, "NetBin list" would return the files hosted on NetBin in that local network. The remove a host file would attempt to remove the data on every node in the network that is associated with the specific file the NetBin remove file is being called upon. The favorite command would allow a user to favorite a file hosted on NetBin. All files hosted on the NetBin service will have an expiration time; however, favorited files will have an infinite expiration time and will not expire from the network unless removed manually. As a note expiration times will be incremented after each access, allowing for popular files to remain on the network longer. Additionally, If the network data capacity limit has been reached files will have to be removed from the network before others can be added. Favorited items will never be removed; if an user is attempting to upload a file that would exceed the network storage capacity and if only favorited files are hosted the upload will be denied. But, that same user could remove all the files on the NetBin network, even favorite files and files that she or she did not upload.

In order to implement a reliable and scaleable LAN file server for NetBin, our team has decided to tackle the challenge of transferring the files peer to peer and storing fragments of the files on multiple computers connected to the network. Every computer that is connected a LAN and that is running NetBin will be hosting chunks of file data. This will be implemented through a distributed hash table (DHT). A distributed hash table will provide us with a lookup service quite similar to a standard hash table (key, value), but it also allows every device (or node) connected to our distributed

network to retrieve the values associated with any given key. Every computer actively using the NetBin network will have a local hash table that will be maintained by the NetBin application. Using a DHT data will not only be able to be transmitting between nodes but actually store them as well, as noted above. Our DHT will use a structured key-based routing in order to actually obtain the data from other nodes on the network. Every Time a user executes an upload, download, or remove command, or even when a file expires a broadcast will be sent out to every computer actively using the NetBin service to update their local hash table. When a computer installs NetBin a small chunk of hard drive space will be allocated to storing the actual NetBin file data. This chunk of hard drive can be simply implemented with the creation of a maintained directory on the computer. Duplicate chunks of fragmented data, or block of data, will be created and stored for redundancy. All chunks of fragmented data, or blocks of data, are handled independently by the NetBin application allowing for a single file's data to be distributed and stored on multiple devices that are connected to the LAN and are running NetBin.

Implementing NetBin file storing in this manner forces us to face the issue of what happens when a user who is hosting blocks of data leaves the network. To tackle this challenge, our team has decided duplicate blocks of data when they are first uploaded. This creates redundancy. NetBin would traverse through the nodes on the network to check that each file hosted on the service has at least two complete, but fragmented, copies of the file stored. When a node leaves the network the blocks of data stored on that computer will be found on other nodes then redistributed to other nodes on the network; therefore, ensuring that NetBin will be a no data-loss file hosting service.

Updates:

In implementing the basic NetBin client server architecture, we realized that it was impossible to maintain the application running in the background. As a result, we have made a UX change in that a user who wants to run NetBin, Now a user calls netbin init from their terminal. NetBin then starts running, and queries the local area network for all neighbors, and then looks to see if any of them are actively listening on the NetBin port, which is currently defaulted to port 7878.

If any are, that means there is an active host and the user's NetBin connection initializes as a TCP client. If there aren't, then the user's NetBin initializes as a TCP host.

The TCP host manages all the active clients and accepts all connections. It also identifies a next host, which it will let know to be the next host when it's connection

terminates. Thus, when the next host terminates, all the clients know where to open a new connection. We actually experienced a number of issues looking to discover all nodes on a subnet, and we ended up with a functional solution using pings and multithreading that did not take too long to run.

We identified a protocol rather late in the timeline for this deliverable -- SSDP. We plan on implementing over the next week to improve the speed of our initialization process. This way we do not have to query the local area network, and instead we can receive a message indicating who the host is and who the other clients are.

Our next steps are to reimplement the query system with SSDP, and debug the connection management. We also plan on implementing the general functionality of the basic netbin commands and their receipt by the host. We plan on using words and strings instead of files for test data. This should take approximately a week.

After that, we plan on first implementing file storage by storing files with their respective client and maintaining a map of filenames and their respective client IPs with the host. We will follow this up by implementing and solving the distributed hashing and peer to peer implementation for file storage.

Our final step will be to implement the command line architecture to make netbin an easily installable system through a utility like pip, and test the application thoroughly --we anticipate this taking the final week.

While the delays and misunderstandings in some of the server architecture were frustrating, we believe its still feasible to meet the deadline for the project in 3 weeks and present a fully functioning solution.

Deliverables: Source Code, Report (with charts and analysis of data-loss/scalability) Source code depicts our initialization process and basic host/client interaction.

Timeline: (3 weeks left in Quarter)

Week	To Do
5	Implement SSDP, Implement front end

6	Minimum Viable Product (MVP) by end of the week
7	Debugging, testing, polishing, filling out extra features (if time)