Distributed Peer to Peer Mobile File system

Project Report

Balamurugan Singaravadivelu
Department of Computer Science and Engineering
University at Buffalo
Buffalo, NY, USA
balamuru@buffalo.edu

Barry Lance Leo Wilfred
Department of Computer Science and Engineering
University at Buffalo
Buffalo, NY, USA
barrylan@buffalo.edu

I. Abstract

Every person uses multiple devices, there is no effective mechanism to securely and cheaply share files between these devices. Personal clouds are one solution to this problem, but there is an additional cost of hosting, this means users have to be selective about what files they want on the cloud. In today's world where all devices are connected to the internet, the files stored in each device should also be accessible across all other devices of that user. A peer to peer distributed file system that runs on mobile devices can solve this problem. Such a system should enable resource access in a secure manner with minimum overhead, also it must ensure that it does not hog the resources such as network and battery on mobile devices.

II. Literature Survey

We studied the following systems which lets the user share his files across his devices:

Segank, Dropbox and Bayou.

Segank is a distributed mobile storage system that relies on replication of files on devices with higher storage capacity and network bandwidth. Segank takes non uniform network into consideration and handles file requests by fetching from the closest source with the best network connectivity. This involves replication of files across multiple devices. Deploying Segank across multiple mobile devices has its own set of challenges. Replicating the files on such devices will strain the battery and consumes extra space on devices which have limited storage capacity. One other factor that Segank does not consider is impact on battery life, active replication of files in background can drain the battery of mobile devices. A better approach for mobile devices would be to provide access to the files on-demand by directly requesting the device holding the file.

The Bayou system replicates all the data on the querying devices before providing access to the data. Each Bayou device houses a complete replica of a database, and alternates between two distinct states of operation: ``disconnected" and ``merging." In the disconnected state, the user of the device only ``sees" local state stored on this device. In the merging state, the device communicates with a peer device, and new updates made on each are played onto the other. To access even one file, the entire database needs to be replicated first. In today's world of fast mobile internet replicating the files does not provide significant improvements in performance and also will be a significant drain on the battery.

Cloud storage services like Dropbox provide a centralized repository to store the user's files. This repository can be accessed from several devices, thus providing a mechanism to share files across devices. But cloud storage comes at a cost, so using it as a repository to store the entire file system of all the user's devices would be expensive. The user needs to be selective about the files that need to be shared. Another issue is that the files selected by the user need to be synced pro-actively, which hogs the network and drains the battery (in case of wireless devices).

III. Design

Our design for the distributed peer to peer mobile file-system has the following key features:

- No centralized directory
- File level locking
- Save network bandwidth and battery life by avoiding proactive replication of files across multiple devices.
- File permissions managed by file owners.

.
The proposed file system is made up of several mobile nodes. A mobile node is a smartphone device with internet access and a file system on top of a secondary storage.

Each mobile node has the following components:

A. Server

The mobile node receives messages from other nodes using the server component. New nodes joining the server also send the message to the server

B. Permissions Manager

Keeps track of files in the current mobile node, their access permissions and remote locking information.

C. *Remote nodes Manager*

Responsible for tracking other mobile nodes in the distributed system by tracking information such as the network address information for all nodes. Keeps track of when nodes become unreachable or when new nodes join the network.

D. *Remote file browser*

Responsible for browsing and accessing files across the remote system. It is responsible for making requests to remote clients for files.

IV. IMPLEMENTATION

A. *Communication*

Mobile nodes communicate with each other over a TCP connection.

B. *Group Creation*

During Group creation, the user provides their name and selects the files to be shared to the group. The permission manager keeps track of these files. Once the group is created the application provides a group link which needs to be used to add other nodes/members to the group.

C. *Member addition*

A node can join a group only if it is invited by an existing member of the group. To a member to the group an existing member should provide the group link to a candidate node. This candidate node can use this link to join the group. The following steps take place during a member addition:

1) Candidate node sends a join request to existing node

2) Existing node accepts connection, sends candidate information about other nodes in the network.

3) Existing node advertises to other nodes about new node being added.

D. *Fetch File System Metadata*

To get the list of files shared by a node, the requesting node sends a file system metadata request to the corresponding node. The receiving node contacts its permission manager to get the list of shared file and sends the list to the requesting node.

E. *File Open*

When a node needs a file, it sends a file request containing the absolute path of the file to node which owns the file. On receiving the request, the owning contacts in permission manager to check if the file is locked by any other node. If the file is locked already, it sends an error response to the requesting node else it locks the file and sends the file.

F. *File caching*

When a node opens a remote file, it stores the file in its permanent storage and this file is used for modification.

G. *File commit and close*

On commit, a commit request is sent to the node which owns the file. On receiving the commit request, the owning node contacts its permission manager to check if the requestor holds the lock on the file and then stores the committed file.

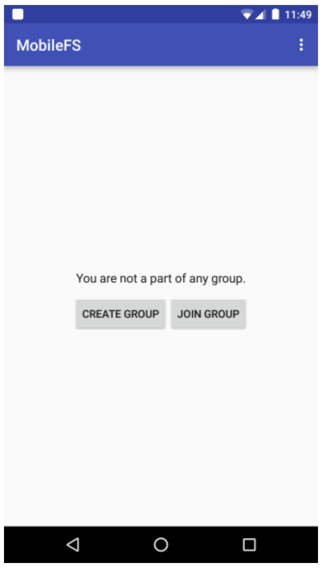
On close, a close request is send to the node which owns the file. On receiving the close request, the owning node releases a file lock if it's held by the sender.

H. *Handling change in network connectivity*

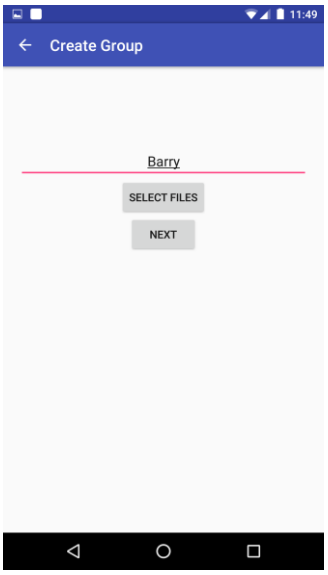
If a mobile node goes offline and come back online, it sends re-sync request to the members it is aware of. These members respond with the information about the current members in the group.

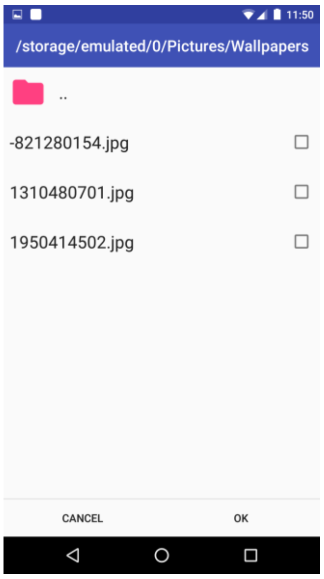
V. APPLICATION WORKFLOW

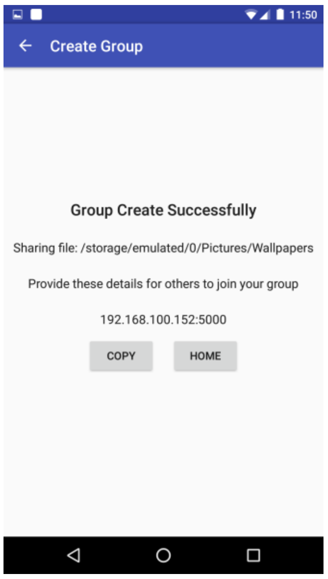
A. *Group Creation*



You are not a part of any group.







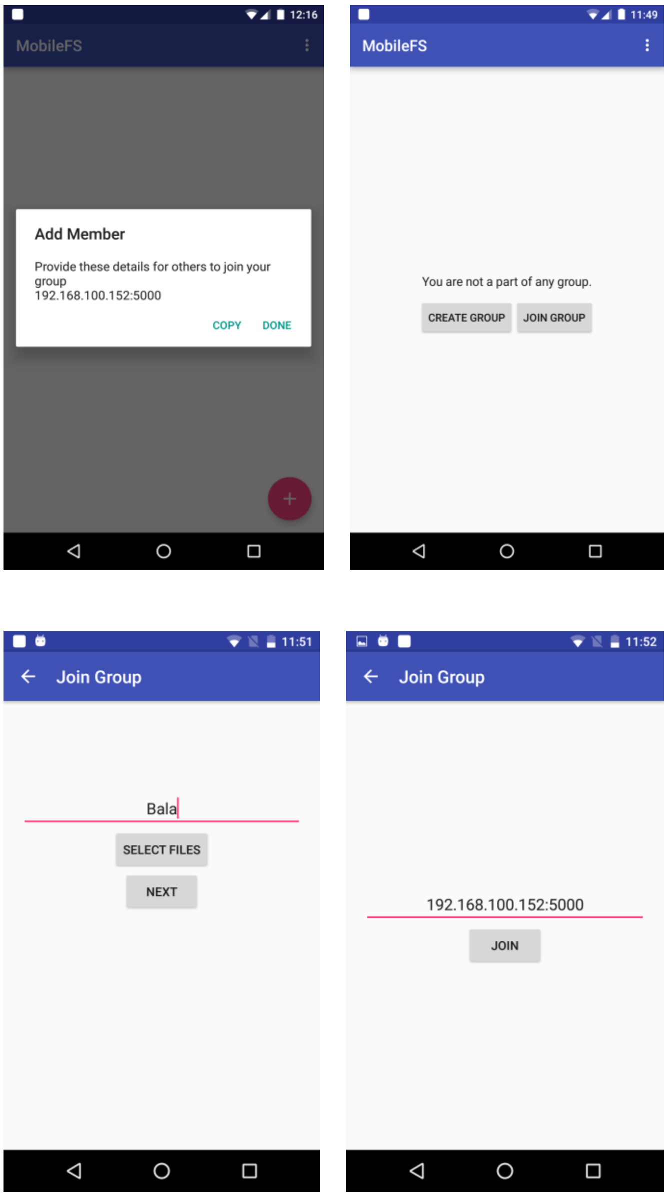
Group Create Successfully

Sharing file: /storage/emulated/0/Pictures/Wallpapers

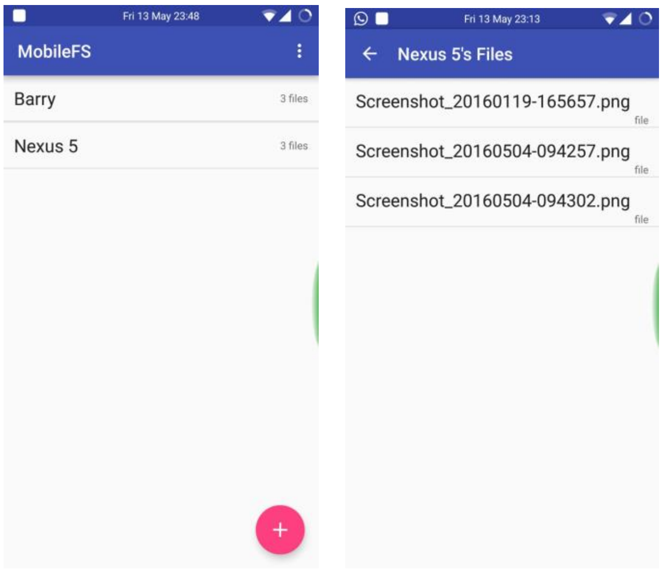
Provide these details for others to join your group

192.168.100.152:5000

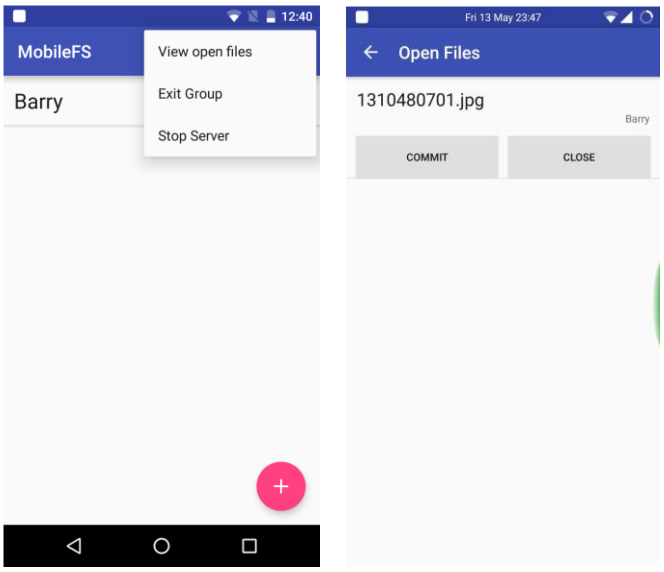
B. *Node Joining*



C. *Member List and Remote File Browsing*


MobileFS
Barry
3 files
Nexus 5
3 files
Nexus 5's Files
Screenshot_20160119-165657.png
file
Screenshot_20160504-094257.png
file
Screenshot_20160504-094302.png
file

D. *Committing and closing files*



VI. Proposed Future Improvements

A. Directory level remote file browsing

B. Conserve battery by using multicast trees

C. Transmit binary diffs to conserve bandwidth

D. Connect to nodes across different subnets

VII. References

[1] S. Sobti, N. Garg, F. Zheng, J. Lai, Y. Shao, C. Zhang, E. Ziskind, A. Krishnamurthy, and R. Y. Wang. Segank: A distributed mobile storage system. In Proceedings of the 3rd Annual USENIX Conference on File and Storage Technologies, San Francisco, CA, March/April 2004.

[2] D. B. Terry, M. M. Theimer, K. Petersen, A. J. Demers, M. J. Spreitzer, and C. H. Hauser. Managing update conflicts in Bayou, a weakly connected replicated storage system. In Proceedings of the 15th ACM Symposium on Operating Systems Principles, pages 172–182, Copper Mountain, CO, 1995.

[3] Dropbox wikipedia page, https://en.wikipedia.org/wiki/Dropbox_(service)