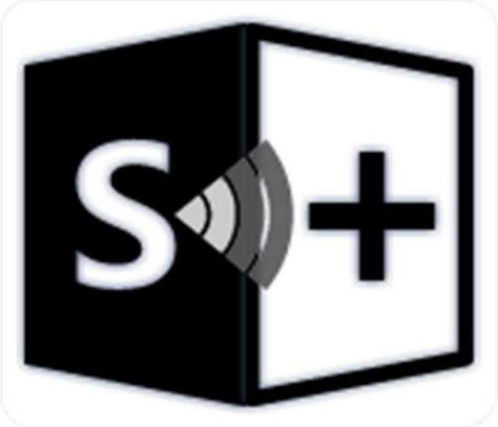
SHARE+



FINAL DESIGN DOCUMENT

CS 9033: Mobile Application Development

By
Ashish Tyagi
Nivedita S. Sonker
Prachi S. Marathe

Contents

REVISION HISTORY

Date	Version	Description
10/18/2014	1.0	Initial Design Document
11/04/2014	1.5	First prototype updates
12/16/2014	2.0	Final Design Document

PROBLEM OVERVIEW INTRODUCTION

Information and technology has rapidly grown in our world and are very much a inherit part of our life now. With this huge growth of digital information comes the desire to share and further expand it to others. As a matter of fact we all love to share information with others.

Another brilliant innovation of this fast and rapidly growing technology is the smartphone almost everyone have these days. With smartphones we can have the freedom to take all that digital information with us all the time. So its natural we would like to share things from these devices with other more than from any other way. And that's where things get a bit not so good.

No doubt these devices are smart and have some cool features to allow us share and enjoy thing with others, but the question is do they always provide a way to share things like we wanted them to be. The answer is "NO".

Well think about it you are hanging out with your friends and want them to see a cool video that you have taken. So what will you do you start on sending out the video to each of them one by one and then they all go to their phone and see that. Do you really call that an awesome experience to share that video. Or let's say you wanted to see a movie together with your friends so is it really possible for all of you to see that movie on a single smartphone screen at the same time, may be you can all see that one your own screens but will that be together anymore?

Same way if you and your friends are at someplace where you want to hear some music and dance to that but don't have big speakers and wished if your smartphone would have had a bigger speakers. Well smartphones cannot have bigger speakers but you have enough of smartphone around to have a loudspeaker impact.

These problems of not being able to share things the way we want motivated us. So to make things easy and your smartphone smarter we have taken the initiative to design an android application that can solve these issues and provide you much more pleasurable experience while sharing.

Its not that you simply can't share files or videos absolutely. No! There are many ways like sending an email, MMS, sharing using Whatsapp, Facebook or just transferring the file using Bluetooth. For sending an email you need to connect to some network or have a data plan. Same goes with Whatsapp and Facebook. Also these apps have a limit of size while sharing. In a similar way for MMS, your mobile carrier is going to charge you. While

transferring using Bluetooth if the distance between the devices increases the connection is lost and at the same time file transfer with Bluetooth is a lot time consuming. Besides the latency is too low. But the power of Share + is that it does not require you to connect to any network. It just uses the built-in feature of Wi-Fi direct service. That's why we propose Share + to be a better option than the current sharing mechanisms.

Share+ application offers you the flexibility to share things the way you want them and try to resolve the issues presented above.

Share+ Application provides a bunch of cool sharing methods like:

- Sharing single or multiple files at the same time from same or different folders
- Sharing same or multiple type files at the same time
- Sharing with single or multiple users
- Playing same audio across multiple devices

Unlike other data hogging online streaming services, or expensive Chromecast hardware Share+ offers all these cool features for you with just a built-in Wi-Fi, means there is no other requirement to have that awesome sharing experience other than installation of Share+.

CURRENT SOLUTION & RELATED APPS

BubbleUpnP

This app uses the services to connect to Chromecast and streams all the screen contents to Chromecast that is a hardware that monitors devices over its Wi-Fi network and when plugged in the HDMI port of some projector or TV set will display the contents being casted by other apps to the TV. But again this requires a hardware Chromecast and then it will work moreover it will just replicate the screen not any other functions.

Bluetooth Music Player

This app is designed to tie the Bluetooth connect event to playing the music player of your choice. Once the subject device is paired via Bluetooth to your phone you may play the music on that particular device. But this app is limited as it can play only the music files and only to particular device it is connected to.

Share Contacts

Share contact is an app by FlashPanel. Its contacts functionality allows admin to import and maintain a list of users who are to be emailed regularly. It creates contacts manually, via spreadsheet, or by importing a folder from Google Contacts. Selects which users, groups, or organizational units to share contacts with, designated by tags. View tagged contacts organized within folders in Google Contacts and on mobile devices. Allow end users to create, edit, and tag contacts from FlashPanel.

Ciacs Contact Sender

This app allows you to share contacts, business cards with multiple recipients via email, text or Bluetooth. The contacts may be exported to the sdcard in either txt, vcf, csv format. But again this app is limited only for sharing contacts.

Other built-in applications in Android

Android devices come with some of their built-in features for sharing while some day-to-day usable apps support the file transfer. Built-in features include Bluetooth sharing while day-to-day apps include Whatsapp, Facebook, Emails, MMS, etc.

SOLUTION

There are many instances when we wish to share some kind of data with our fellow colleagues, friends and there are several applications available for sharing files. The same can be done from android devices as well. But there are a few who give facility to share the files and use them in congregation to give a more grand effect and impact of the file. For these we propose 'Share+'.

Share+ is simply put a file sharing application. It allows sharing of many file types like audio, video, pdf, jpeg, mpeg, text files, etc. The user can not only share the files but also play audio files in unison to get a loudspeaker effect. This can be achieved when user shares the files with other devices in specific mode. The files can be shared in three different methods, Play, Share, and Play & Share.

If user selects multiple files to share at a time, share method will be used by default and it will be a simple ordinary file sharing principle. The shared files will sit at the receiving devices as downloads. If an audio or video file is selected to be shared, it can be shared either by Play or 'Play & Share' method. In case of audio files shared by play method, the participating devices will receive the file and will start playing the file in union to give effect of loudspeaker. In case of an image file being shared in Play method, the participating devices' display configurations will be taken into consideration and each device will project specific part of the original image. In this way with appropriate alignment, which will also be given to users, the original image can be viewed on a wide screen. Play & Share method is similar to play but it will store the file in participating devices as well.

The files can be shared with single user, multiple users or can be broadcasted to all available devices nearby. Also includes the facility of excluding some users from the broadcast mode. When a file sharing is initiated by the user, the receivers will be prompted to accept the file transfer. Once the receivers accept the file transfer, the file sharing will begin with the filtered and verified receivers.

So how do we plan to transfer the data? We are transferring the data using the Android incorporated Wi-Fi direct service using our own protocol. The devices participating in sharing do not need to be connected to any network connections. There will be a Master device which will initiate the file sharing process to all the slave devices participating. Basically we are initiating a client-server communication. The control will throughout remain with the Master device.

Architecture Design

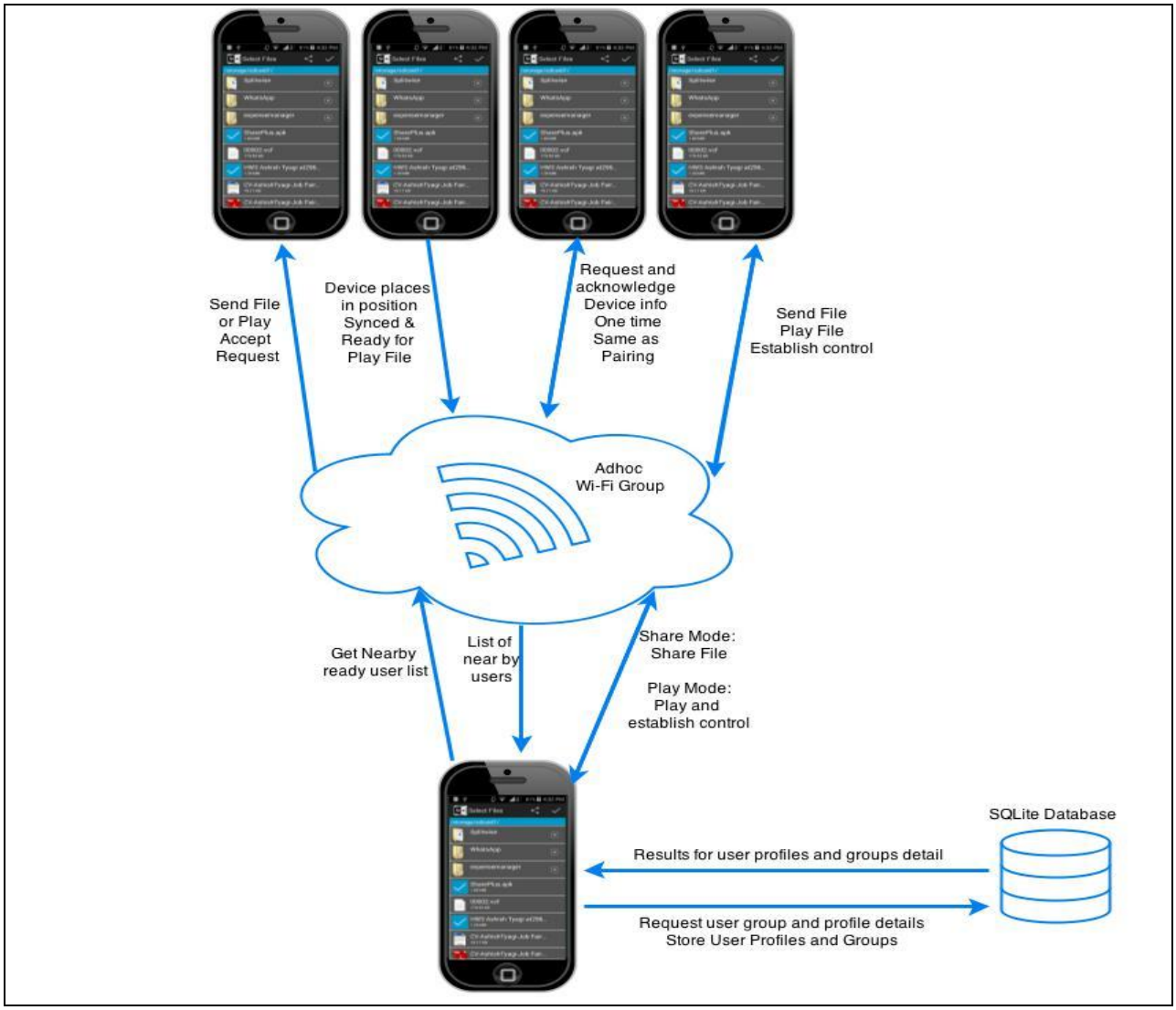


Figure 1: Architecture Design

USE CASE DIAGRAM & MVC FRAMEWORK

Following is a use case representation of the entire system how users will interact with the different part of the applications and what next options and menus he will be provided.

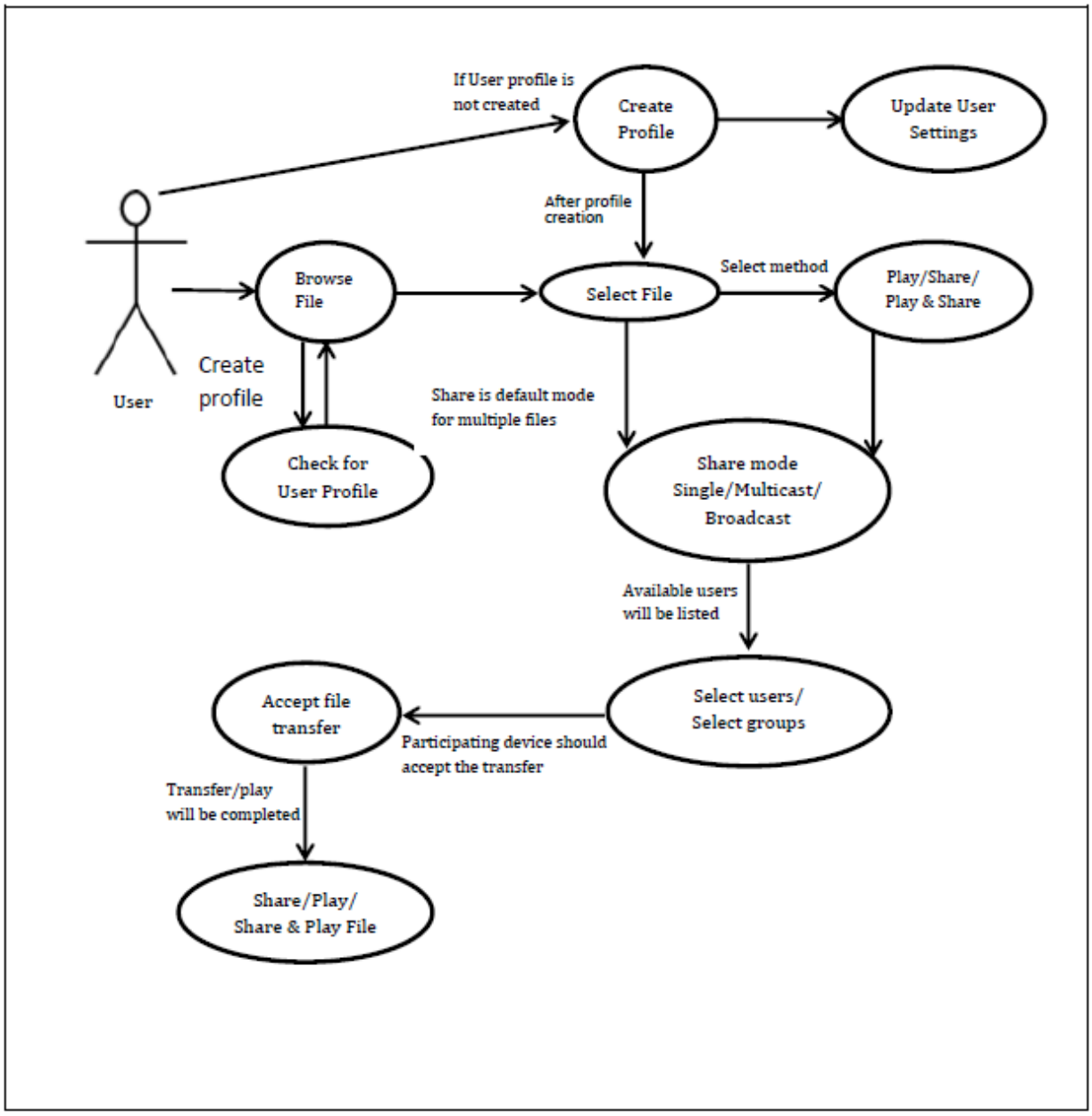


Figure 2: Use Case Diagram

Use case Description:

Firstly the user will be asked to create his profile. This will be a one-time activity. After profile creation, user will be directed to Browse File Activity. Here user can select type of file to be shared
File can be shared with users where it will simply be downloaded as ordinary file transfer.
There are file type specific ways in which play mode will behave.

- The video files can be played in union with other devices where every device will share parts of original video projection; in order to give a wide screen effect
- The audio file can be played in unison on all participating devices in order to give effect of loud speaker

Play and share mode is the combination of both the modes specified above.
A file can be shared in three ways

- Share with single user: The mode will be share by default if the file is shared with a single user
- Multicast: Multicast can be among individuals or groups if any members from the groups are available
- Broadcast: Broadcast will be among all available participating devices/users. But user can exclude any devices/users

Once users are selected, a notification will be sent out to accept the file and once the participating device/user accepts the file, the file will be played/shared depending on the mode selected.

MVC Framework

Share+ Application will use following broad Models, Views and Controllers:

Models:

- **User_Profile:** This model stores information about the users of the device. While selecting a user to send file there will be a list of available devices which shows device as a snapshot of User Profile to make it much more recognizable who the person on other end is. User Profile will include details like First and last name of the user, Profile picture, Contact number etc
- **Device_Info:** While share+ installs and run it stores the information of the important device in this model like device name, device make and model, device screen size, screen rotation enabled or not, volume levels, brightness level, Wi-Fi info. These info will be used within play activity to support the seamless video play functionality and selecting the order and orientation of devices
- **Sharable:** This model stores the information about the selected files to be shared and file locations, file types and if single file then is that play compatible or not. Used to first select the appropriate activity to be invoked share or play. After that it will

be used by Wi-Fi service module and Play activity to actually stream file or send files to other devices

- **Browse List_View_Item_Data:** Data flow between adapters to browse activity
- **Client:** Initiates client socket and passes the data to communicate activity
- **Server:** Initiates server socket and passes the data to communicate activity
- **Communicate:** Implements the Wi-Fi Share Plus (SP) protocol
- **External_Storages:** Provides external storage depending on various devices
- **User_group_Select_list_view_item_data:** Data flow between adapters to user group activity

Views:

- **Browse Screen:** This is the main screen that user sees when application is launched from launcher. This screen allows user to select multiple files and folders to be sent to another device
- **Play Share Screen:** This screen provide option to simply share the file or to share and play the file. If there are multiple files selected then only share option is allowed and this screen is skipped
- **Share Mode Selection Screen:** This screen ask user how to send the file i.e. just to a single user, or to multiple users or to broadcast the files
- **User Selection Screen:** This screen shows the available users to send the file. If single user was selected then users will be listed and only one of them can be selected. If multiple user's option was selected then this screen shows the users in addition it also shows groups that can be selected to send files to entire group of people
- **Share ready Screen:** this screen shows all the users who have accepted to incoming file. User can now hit send to send the files or play to play the file on other devices. This is mapped to After_request screen in our code
- **Play Screen:** If in video play mode then this screen first shows a number that tells where to place device relative to other devices
- **Settings Screen:** This screen shows various settings available for this application
- **User Profile Screen:** This screen shows and enable user to edit the user profile
- **About Screen:** This screen shows details of application, current version, and device details

Controllers:

- **Browse Activity:** This activity supports the Browse screen and provides means for selecting files and folders. Also this activity handles the incoming file shares from other applications using share menu
- **Share Mode Activity:** This activity handles all the relate work behind the Share mode screen

- **Play Activity:** This activity handles all the related work for wide screen experience screen splitting as well as music streaming functionality
- **Profile Group activity:** This activity handles profile creation and group manipulation
- **User group Select:** This activity handles the Peer selection part
- **Share Ready Activity:** This activity handles on screen incoming file request acceptance and in application verified user list display
- **Wi-Fi Service:** This service handles all the work related to Wi-Fi connectivity and handling transfer and streaming of files
- **List Populate Helper:** This helper handles population of customized lists in application by overriding getView method
- **Database Helper:** This helper handles all the database related requests and updates
- **Notification Helper:** This helper handles all the notification related work
- **About Share Plus:** Gives the detail about the Share+ app

Broadcast Receivers

- **Connectivity Receiver:** Handles the turning on & off of Wi-Fi service
- **SharePlus Wi-Fi Broadcast:** Handles on the events related to Wi-Fi

Adapter

- **Browse List Adapter:** Handles the browsing of files in the Browse Model
- **User group List view:** Handles the browsing of files in the User Group Model

UI Drawings

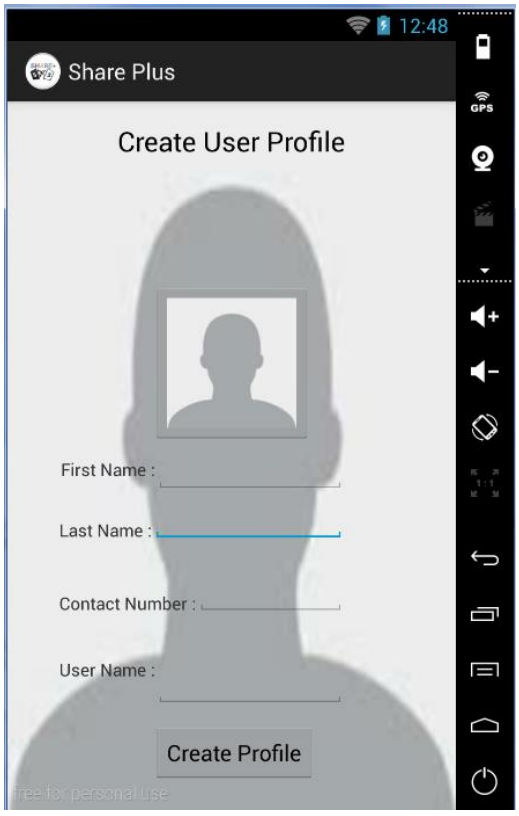


Figure 3: User Creates his profile

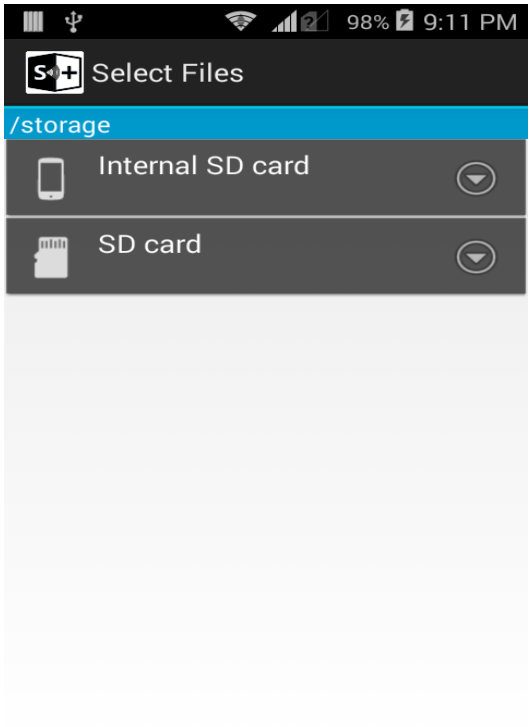


Figure 4: Select Memory

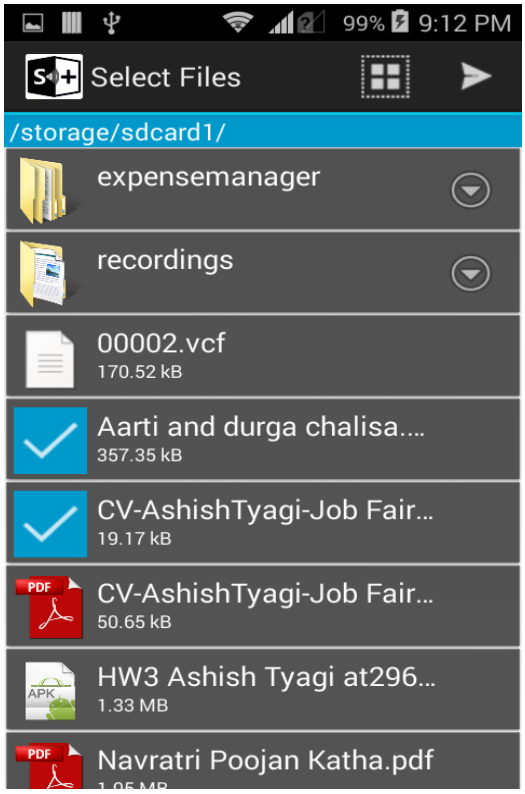


Figure 5: Files selected

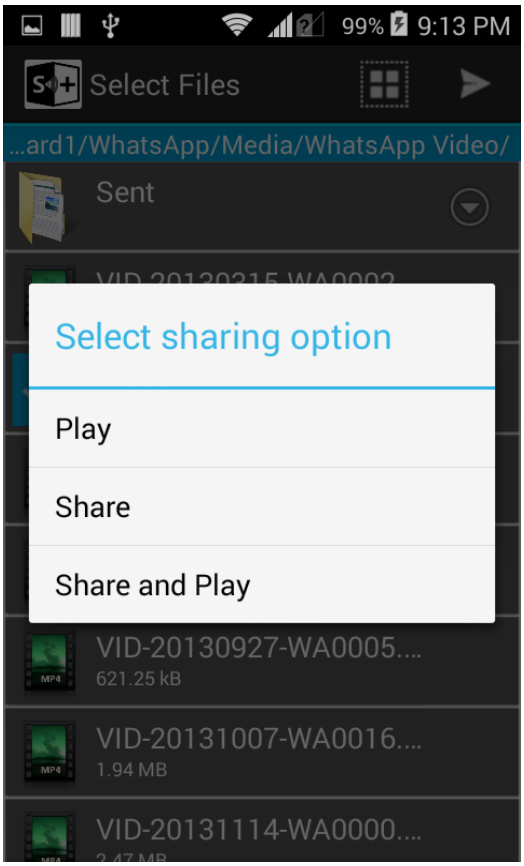


Figure 6: Select Method

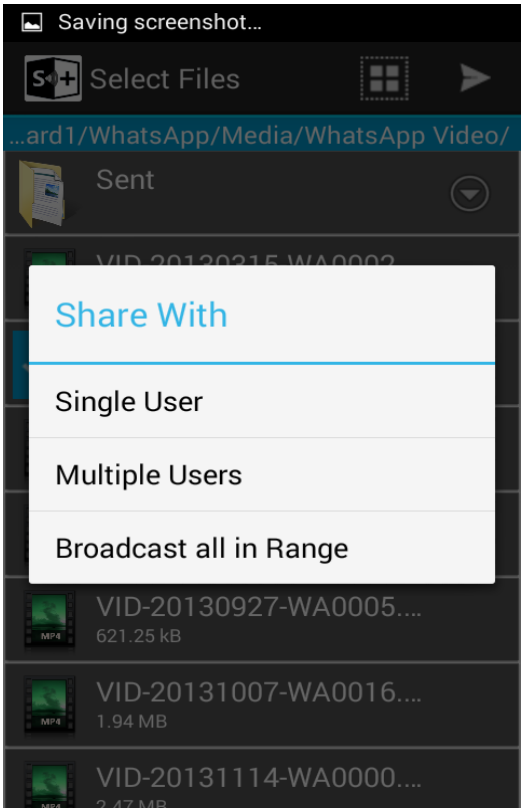


Figure 7: Share Mode

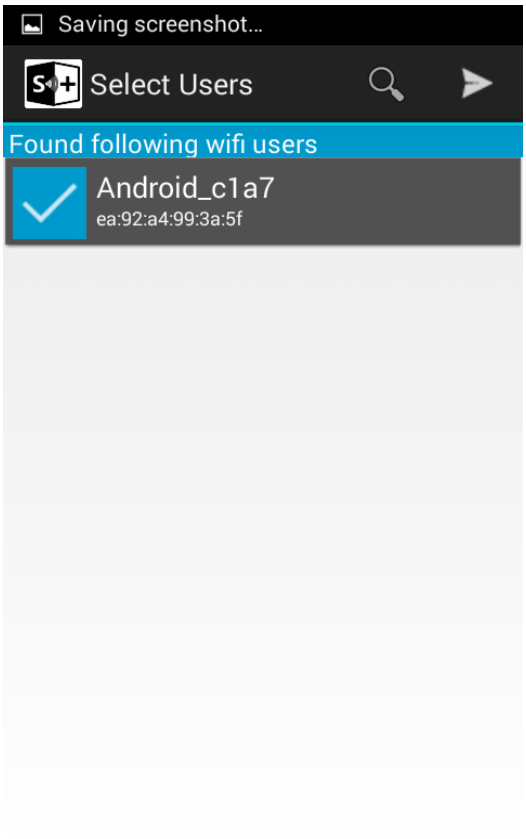


Figure 8: Peer Detected

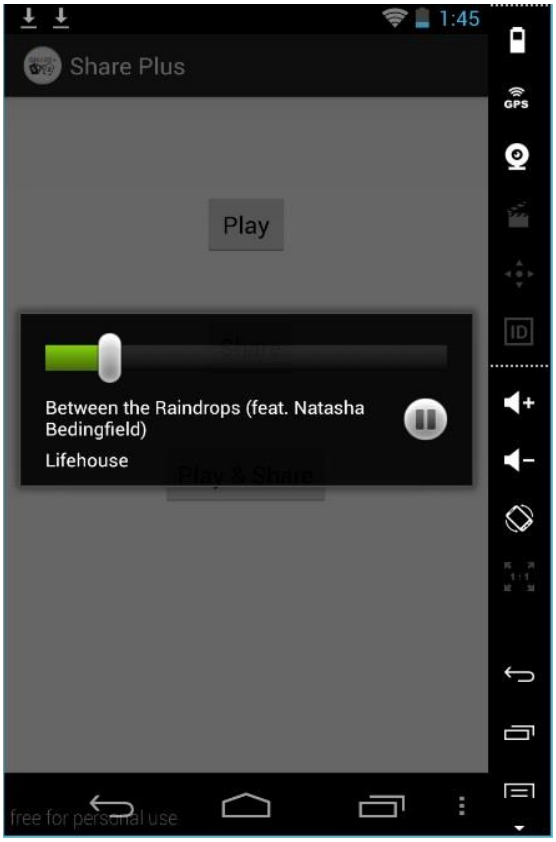


Figure 9: File transferred & playing

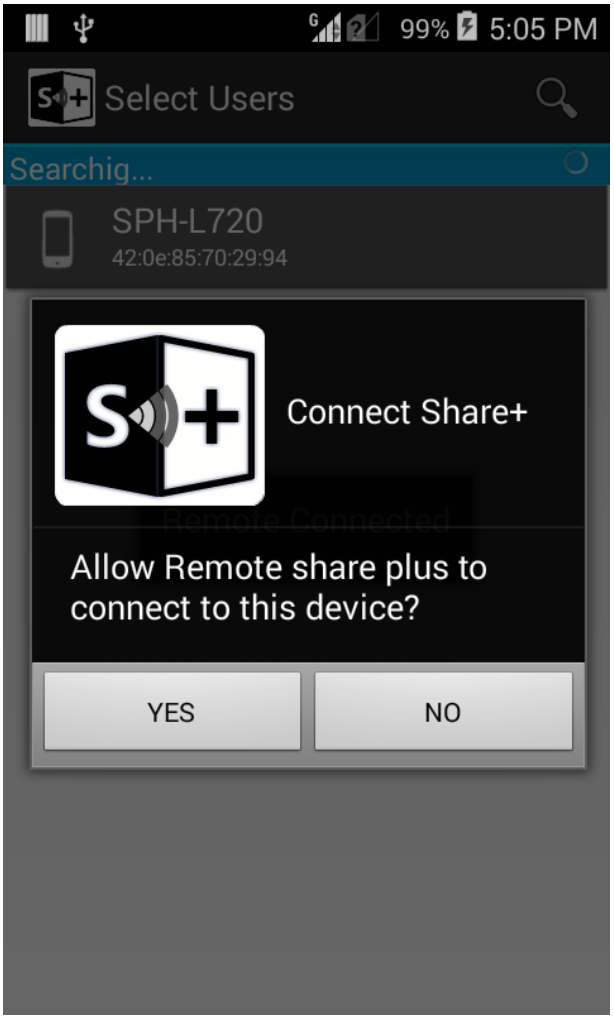


Figure 10: Connect popup

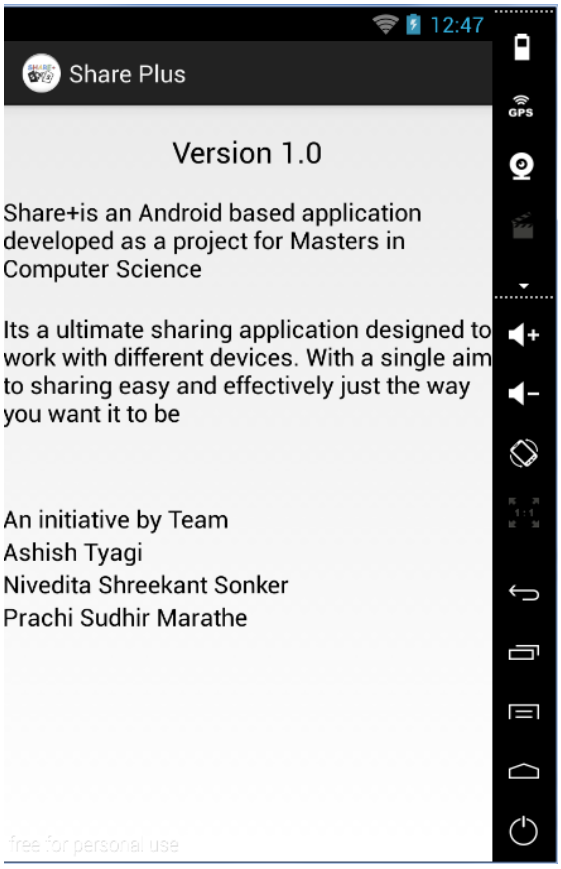


Figure 11: About page

FEEDBACK, CHALLENGES & TRIUMPHS

Feedback and Suggestions Received at AT&T:

- Resolutions are different of each devices, how do you pan to implement the screen mirroring and widescreen effect?
- While connecting or disconnecting the Wi-Fi direct service, make sure the transition is smooth
- Will you be able to share and watch the videos at the same time? Means sending continues in the background, while you watch the video
- Have you given it a thought to share some files or music not available locally on the device? Means if you are listening to radio and want to share that song with the nearby device. Will that be possible?
- Future scope for consideration- Once the Wi-Fi direct service is implemented, you can go ahead and plan on sharing the files with the nearby devices which are active on their data plans

Implementations:

- Added a "Play & Share Mode" to share and watch videos or images simultaneously
- Resolutions and configurations of different devices were taken into account while splitting the image and sharing the same

Problems Encountered and Solved:

- Socket programming
  Initially the client-server synchronization was not taking place. When Client used to send the data, server didn't accept it and the app crashed. So after the initial client-server request and handshake, we sent the data in chunks to the server. This way server could handle all of the data.

- Thread errors
  All the communication between the client-server couldn't be handled by a single thread. So moving them on separate threads and running them sequentially was a task which we implemented successfully.

- External Storage based devices
  Every device has its own device configurations which aren't supported by the built-in environmental variables. This was taken care by reading a file "/proc/mounts"

CORE FUNCTIONALITIES and WORKING

Step by Step Process:

- User can browse files through the browse activity of Share+ and select the files to be shared or can share the files via Share+ by selecting share+ option in share mode
- On the next screen user will be presented with the available list of devices that he can connect to and share data to.
- Master device will call "Wifi service module" for the peer discovery to enlist participating devices
- User then selects the devices that he needs to connect to and hit the send button
- On hitting the send button activity will pass the currently selected devices to the service which will try to connect the devices
- Service will open the server connection on the Master device and client connection on the receiving device and will start the communication process by passing these sockets to the communication class
- Communication class will send the initial handshake signals to the devices on the other end
- On the receiving device a connection pop up will appear to accept or decline the connection
- After user accepts the connection the communication class will send commands back and forth to match up the devices and provide a connected device list to the Master device user
- Once the User is satisfied with the connected devices he can then hit send again and start the screen sharing process
- Once the user hit the send button the "WiFi Service" will again send the commands and data to the remote device using the communication class and send the entire data files to the device on other end
- Once both the devices has the file they will send the "Startdisplay" commands to each other
- Then both the devices will start the Play activity according to the share mode and file type
- On file share completion or user hitting the back button a service call will be made to close all connections

Service Calls sequence:

On completion of boot there is a "WiFiConnectivity" broadcast receiver that will tell launch the "WiFiShareplusService" which handles all the "WiFi" related processes.

This service detects if Wi-Fi is on or off and display a list of all available devices. Once we select the peers to be taking part in the sharing, this service call thread opens the connection for all the receivers who have accepted the request from the master device. On the slave device a socket connection is opened to receive the data sent by the

server/master device. So now we have a successful socket connection established between the master and the slave.

Communication and data transfer as per the SharePlus protocol:

- Communication between the devices is initialized by a Handshake command
- If Handshake command is initialized from slave device, master will reply with a Startshare+ command. On the other hand if Handshake command was issued by the master device then client will send the MAC addresses to the master devices
- Master will give an Ask connection command to the slave devices and an alert box to accept or reject the connection will be shown on the slave devices
- If slave replies in decline then "ConnectNo" command will be sent to the server and then master will close all connections for that particular slave and client will destroy any connection ports created on its end
- If slave replies in accept then a "Connectypes" command is issued to the master device. In response to this the master device will launch the next activity and supplies the slave with three piece of information such as Device ID, Play Type, and Total number of devices
- After this master sends a DATA command to slave to inform slave that it will begin data transfer
- Slave acknowledges the connection and DATA command by SEND PACKET command indicating that slave is ready to accept data now
- Master will start sending data in chunks to all slaves
- When master sends last chunk of data and slave receives it, slave will acknowledge Master with a command "START DISPLAY"
- On sending "START DISPLAY" command slave will start the display activity whereas Master device starts its own display activity
- If any of the slave hits BACK button, master will be notified that the slave is backing out from the connection. Then master will notify all other slaves that a device is backing out and the connection will be shut down using the shutdown command
- Finally the master will close all connections for all the slave devices and so does the slaves close their own connections if are yet open

FUTURE DEVELOPMENT

Moving ahead with the project beyond the scope of the semester we would like to add more functionalities to it. Some of the features we would like to add are:

- Broadcasting the file to multiple devices
- Playing single video split over multiple device screens providing a wide screen experience.
- Sharing the video in the background while watching the video.
- Creating groups while sharing the file. This will speed up the sharing process, if there are previously created groups as each peer doesn't have to be selected individually.

PROJECT MEMBER BREAKDOWN

Ashish Tyagi

- Coined the Idea for this Project
- Created the base template of the project in initial design phase
- Created the File Browser part of the application
- Designed the External Architecture, screen layouts for application and for design document
- Designed the design of project
- Implemented the WiFi Module of application
- Combined the code from other team members to suit the screen layouts
- Unit testing and Integration Testing of the above modules
- Provided a detailed description of Models, Views and Controllers in the documentation

Nivedita Sonker

- Implemented Play method wherein the file will be played depending on its type
- Created the selection modules for selecting the method like (play the file, share it or play & share it) and mode of sharing like (unicast, multicast and broadcast)
- Handled the play activity to share the media files and play them
- Unit testing and Integration Testing of the above modules
- Poster presentation at AT&T
- Entire documentation and presentation

Prachi Marathe

- Created User Profile and the about the app page
- Poster presentation at AT&T
- Implemented the Image Splitting logic and code based on the number of devices participating in the share
- Helped in device connection with WIFI Services part and debugging of the same
- Worked on Solution & Use case diagram in Initial Design documentation
- Worked on Core Functionalities and Working part in Final Design Document
- Testing of app on other devices, bug fixing, unit and integration testing of above modules

PROJECT MILESTONES ACHIEVED

Date	Milestones
10/13/2014	Initial Design Document
11/4/2014	Initial Prototype Presentation
11/25/2014	Presentation at AT&T
12/16/2014	Final Demo and Final Design Document Hand In

BIBLOGRAPHY

Current Work:
https://play.google.com/store/apps/details?id=com.bubblesoft.android.bubbleupnp&hl=en

Wi-Fi Direct Services:
http://www.androidside.com/docs/resources/samples/WiFiDirectDemo/src/com/example/android/wifidirect/index.html
http://developer.android.com/guide/topics/connectivity/wifip2p.html

Image splitting:
Referred Android developers' page for bit mapping.