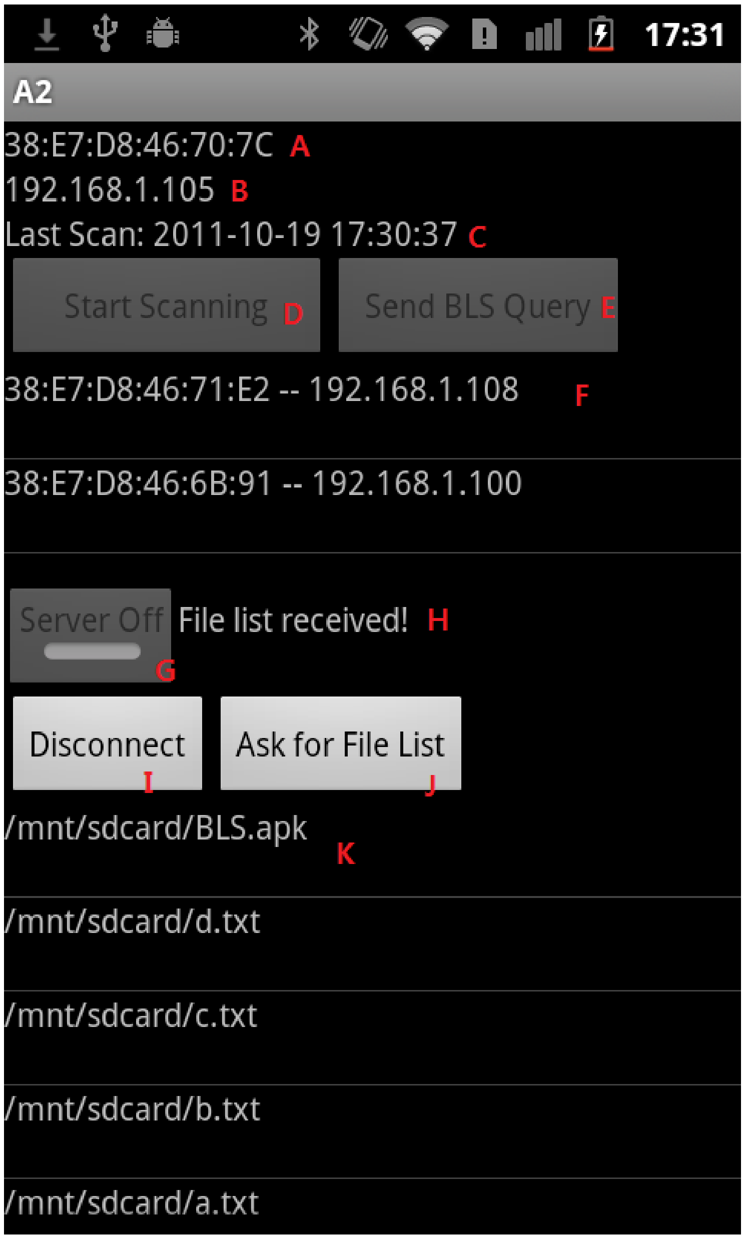
Design Document

CS 456 Assignment 2

Device-to-device Communication

Name: Mingqian Ye, UWID: 20281333, NexusID: m4ye
Name: Devin Howard, UWID: 20293980, NexusID: dmhoward

1. UI design:



38:E7:D8:46:70:7C

192.168.1.105

Last Scan: 2011-10-19 17:30:37

38:E7:D8:46:71:E2 -- 192.168.1.108

38:E7:D8:46:6B:91 -- 192.168.1.100

File list received!

/mnt/sdcard/BLS.apk

/mnt/sdcard/d.txt

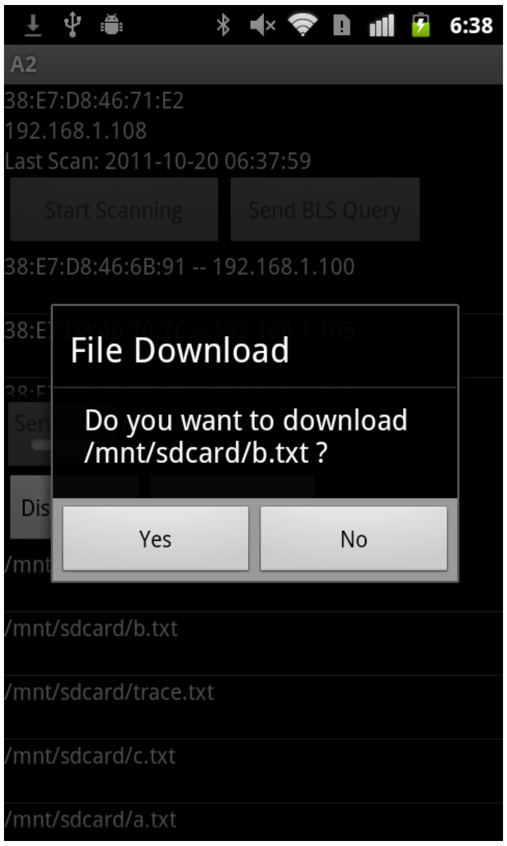
/mnt/sdcard/c.txt

/mnt/sdcard/b.txt

/mnt/sdcard/a.txt

A. Textfield that displays device MAC address
B. Textfield that displays device IP address
C. Textfield that displays time of last scan performed
D. Button to discover nearby Bluetooth devices
E. Button to query the BLS server for device IP address
F. Listview of discoved Bluetooth mac addresses; after sending queries to BLS server, registered devices will be displayed. Tap the available device will connect to it.
G. Button to turn server On/Off. The device is connectable only when server is On.
H. Textfield that displays the connection message
I. Button to disconnect from a device; only available when there is device connected.
J. Button to ask the connected device for the file list in mnt/sdcard/
K. Listview of received file list from connected device. Tap any file to send a request for downloading.
See the following screenshots for this process.

2. Screenshots for file transfer process


6:38

A2

38:E7:D8:46:71:E2

192.168.1.108

Last Scan: 2011-10-20 06:37:59

Start Scanning | Send BLS Query

38:E7:D8:46:6B:91 -- 192.168.1.100

File Download

Do you want to download /mnt/sdcard/b.txt ?

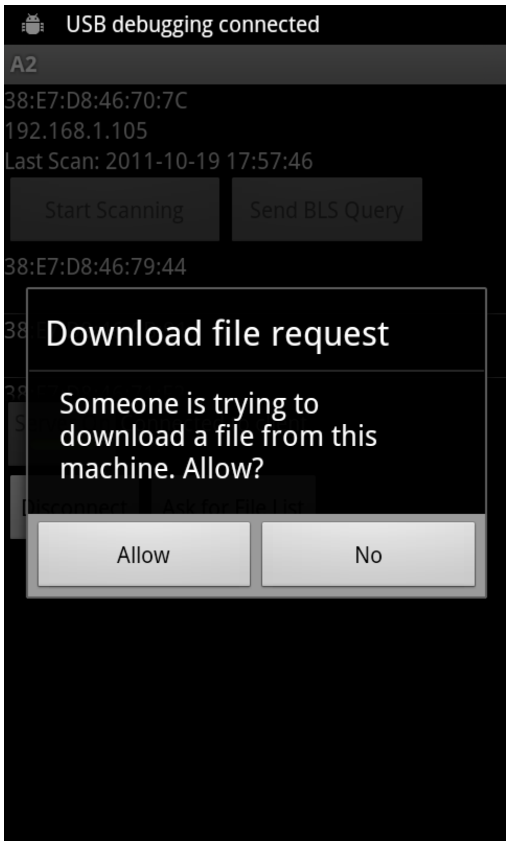
Yes | No

/mnt/sdcard/b.txt

/mnt/sdcard/trace.txt

/mnt/sdcard/c.txt

/mnt/sdcard/a.txt


USB debugging connected

A2

38:E7:D8:46:70:7C

192.168.1.105

Last Scan: 2011-10-19 17:57:46

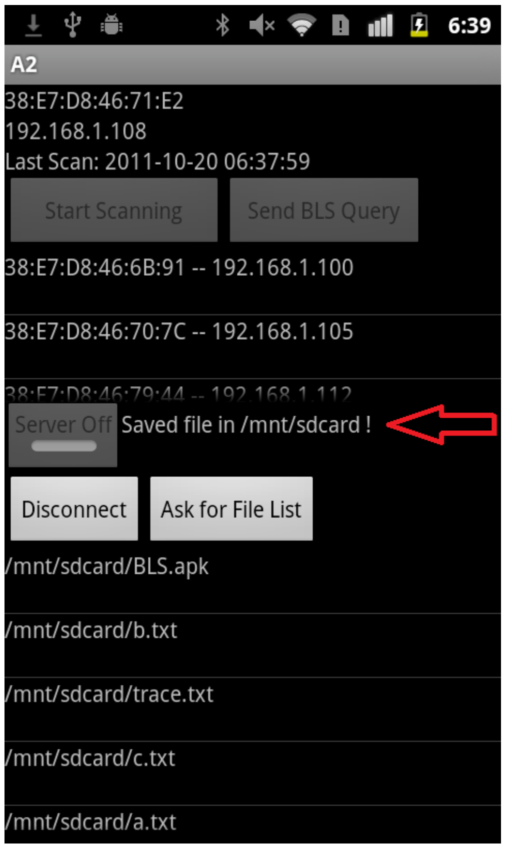
Start Scanning | Send BLS Query

38:E7:D8:46:79:44

Download file request

Someone is trying to download a file from this machine. Allow?

Allow | No


6:39

A2

38:E7:D8:46:71:E2

192.168.1.108

Last Scan: 2011-10-20 06:37:59

Start Scanning | Send BLS Query

38:E7:D8:46:6B:91 -- 192.168.1.100

38:E7:D8:46:70:7C -- 192.168.1.105

38:E7:D8:46:79:44 -- 192.168.1.112

Server Off Saved file in /mnt/sdcard !

Disconnect | Ask for File List

/mnt/sdcard/BLS.apk

/mnt/sdcard/b.txt

/mnt/sdcard/trace.txt

/mnt/sdcard/c.txt

/mnt/sdcard/a.txt

3. Assignment Completion
Scanning - Done.
BLS integration - Done.
Connection - Done.
Exchange file listing - Done.
Bonus file transferring - Done.

4. Problems Encountered and Design Decisions
Overall code structure

At the planning stage of this assignment, we proposed a code structure which divided the project into different modules by their funtionalities (ie. Bluetooth, BLS, Server, Client etc). However, after completing the BLS integration we found that this structure did not meet our design requirement; it was hard to maintain and it became overwhelmingly difficult to add server/client features. After research, we decided to restructure the code by applying the MVC(Model, View, Controller) standard Android design pattern. Currently, the Controller Class detects all inputs such as Button-click, Listview-click, and Server/Client requests, etc. The Model Class accepts command from the Controller Class, then takes care of all logical operations such as starting a Bluetooth scan, sending a query to the BLS server, or generating the file list in /mnt/sdcard/. The UI Class is notified by the Model Class if UI updates are needed, which could be anything from updating the ListViews to enabling or disabling Buttons.

Server-Client Model:

The server-client model required in this assignment is implemented by 3 classes: ConnectionManager, ServerThread, and ClientThread. The Model Class, which takes care of all logical operation as discussed above, only communicates with the ConnectionManager Class and is not aware of the ServerThread Class and the ClientThread Class. In this way, we greatly reduced the degree of coupling between the server-client model and the Android core model.

Server-client Communication:

When a message arrives at a server/client, it applies the following protocol to determine the message type and retrieve data from the message:
1. Read the first integer from the buffer, and use a switch statement on what that integer is. The integer could be anything from 1 to 999. For example, if the integer is 3, the socket knows that it is followed by a serialized object which contains a file-list array, so the socket simply reads a SerializedStringArray object.
2. After retrieving the data from the message, the ConnectionManager then sends a corresponding message to the UI Class. The message handler in the UI Class forwards this message to the Model Class and lets it handle the logical operations.

Possible improvements

It would be possible to create a base class that the ServerThread and ClientThread could inherit from; some of the code is duplicated so it could be inherited.

Thanks for reading.