**PROJECT REPORT**

**Title: BlueDeath Streamer**

**Subject: Design and Analysis of Algorithm**

**Code: CSE 5311**

**Semester: Fall 2015**

**Problem Statement:**

Establish a communication link between Android devices via Bluetooth that are out of range using intermediate devices that can act as relay.

Implementation of algorithms that assist in the efficient mechanism of transmission. Algorithms such as the Shortest Path Algorithm, Traversal algorithms are used in this project.

**Abstract:**

The App namely “**BlueDeath Streamer**” provides the facility to send or receive media files via Bluetooth between Android phones that are *not in range.*The app uses relay devices that are available android phones that are in range between the source and destination phones.

**App Functionality:**

1. Show Common Song list.
2. Play songs off the Song list (if owner).
3. Stream songs off Song list (if not owner).
4. Turn On/Off Bluetooth.
5. Show paired devices.
6. Search new devices.

**UI Flow:**

1. Bluetooth Streaming: This *activity* populates a radio button list of all the songs from the Common Song List.
2. Request: Plays or streams song selected from list.
3. Settings: Used to turn on/off Bluetooth, show pair devices and show connectable devices.
4. Stream via Bluetooth: It uses Bluetooth API to achieve streaming of the data from a selected choices of android devices.

**Implementation details:**

Bluetooth Access: We use *Bluetooth Adaptor* to access the setting of the Bluetooth such as to switch Bluetooth on or off are desired by the user. The adaptor requires the user permission to allow app to access Bluetooth configurations. The app also shows the previously paired devices as well as the devices that are available in the range. The Settings activity provides these functionalities from the Settings Button on the first activity.

Song List: The app usesa database with two tables namely *Songs* and *Node* to access the information that can be shared and also the routing table by which the relay can be chosen in case the file needs to be shared to device that is not in range. The *Songs* table has names and owners of the songs that are stored in the phone memory. The table is accessed to populate the Radio Button list in the first activity.

Request: The song selected from first activity is sent as an intent to the request activity. If the songs’ owner is itself then the songs is accessed from the devices’ phone and played. Otherwise, the songs owner is referred from the routing table *Node* and request is sent to the owner as the Destination of the packet.

The other scenario is that the owner can send the send to others. In this case the user himself sends the selected song to one of the desired paired devices.

**Database implementation:**

The application has a database requisite for the establishment of the communication channel. The database is used for the implementation of the routing table and songs that the device can access. The later part can be improves to contain a dynamic list as to the songs that are currently available in the neighboring network. The database is implemented in **SQLITE** technology. The java files namely DatabaseOperation.java and TableData.java have the database implementation.

**Shortest Path Algorithm:**

After the app receives a request for song, the app needs a path for the destination nodes via intermediate nodes. This is implemented by finding *Shortest Path* by checks all the possible relays.

This phase has successful implementation of **Dijkstra’s Algorithm** in which the numbers of nodes, edges with weights are hard coded. The code starts the path from the *Egde\_0* that is the owner of the app to the node that is received from the request packet. The algorithm returns the shortest path in the form of *list of nodes* to the destination.

Implementation of algorithm: The package has various models such as Vertex, Edge, Graph and Dijkstra’s Algorithm. The Vertex class is the data structure of the nodes of each of the node of the graph. Similarly, Edge and Graph is also data structure of the weighted edges and structure of graph respectively.

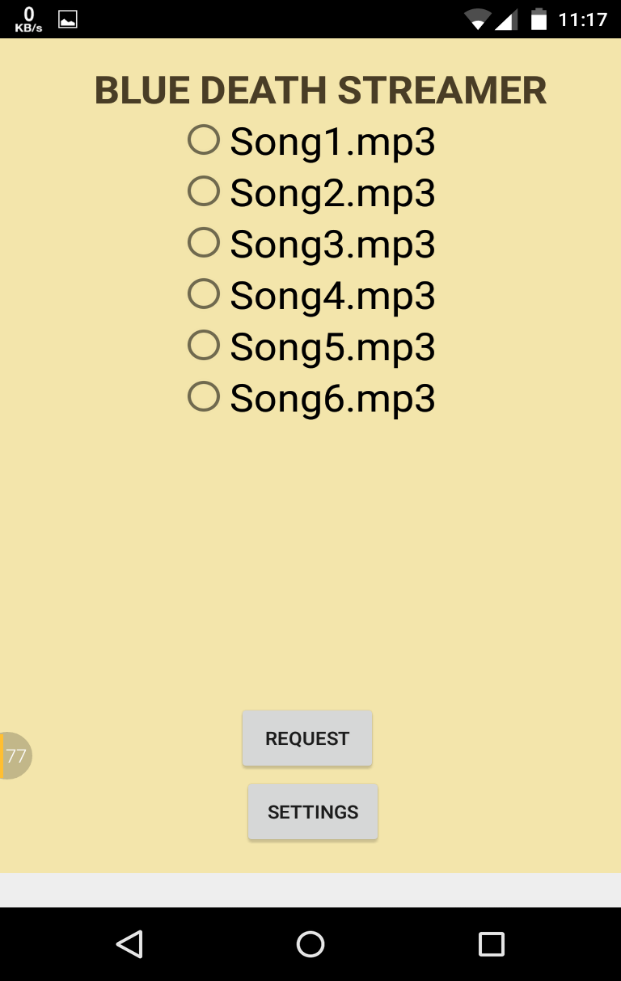
The *TestDijkstraAlgorithm* is a Junit class that runs in the same package as the above mentioned models. The values are hard coded here and then called for the Dijkstra’s Algorithm.

**Dynamic routing tables:**

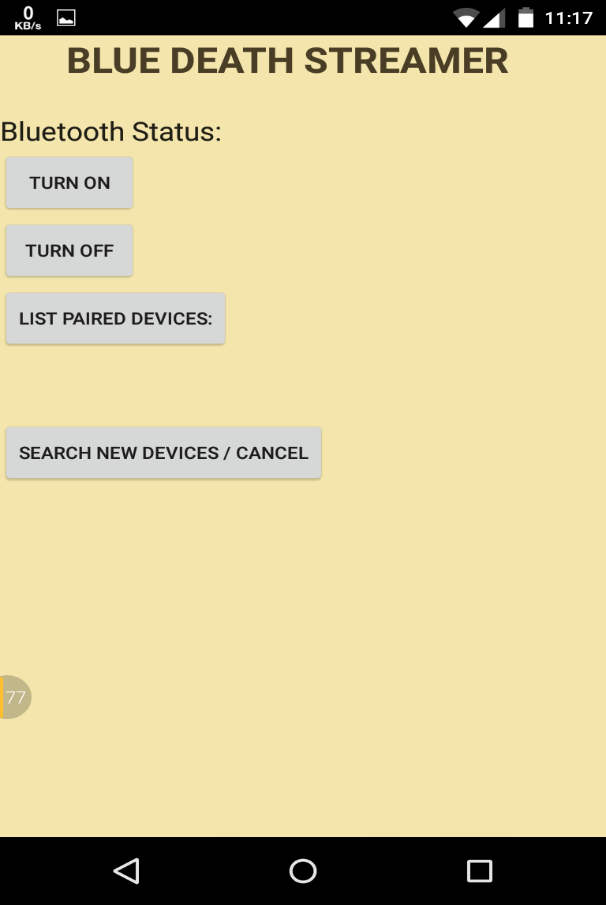
The routing tables are generated using **Traversal algorithm**. The applications should run the traversal for its node in the graph to every other node in the graph. The traversal shall populate the routing table for each device individually. The routing table is a database entity that is created in the DatabaseOperation.java file. The TableData.java has the attributes of the name of the table and the table information.

**Screenshots:**

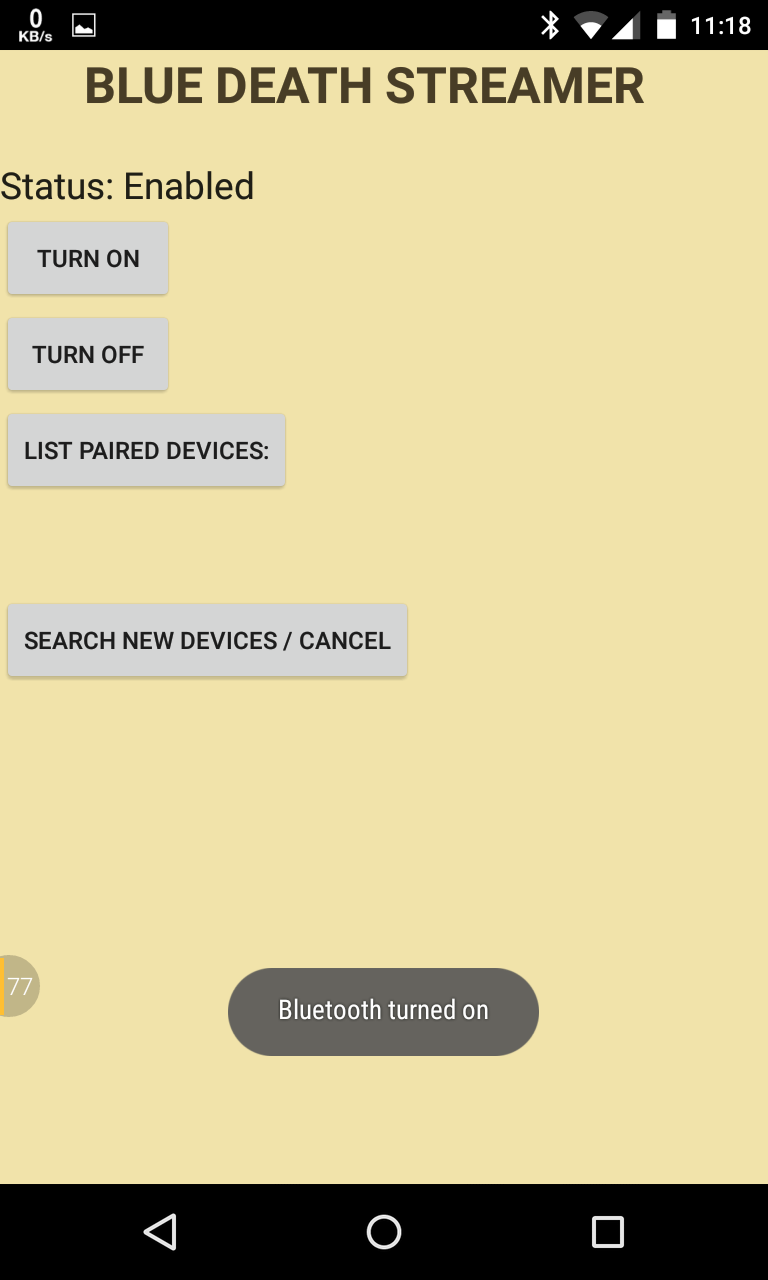
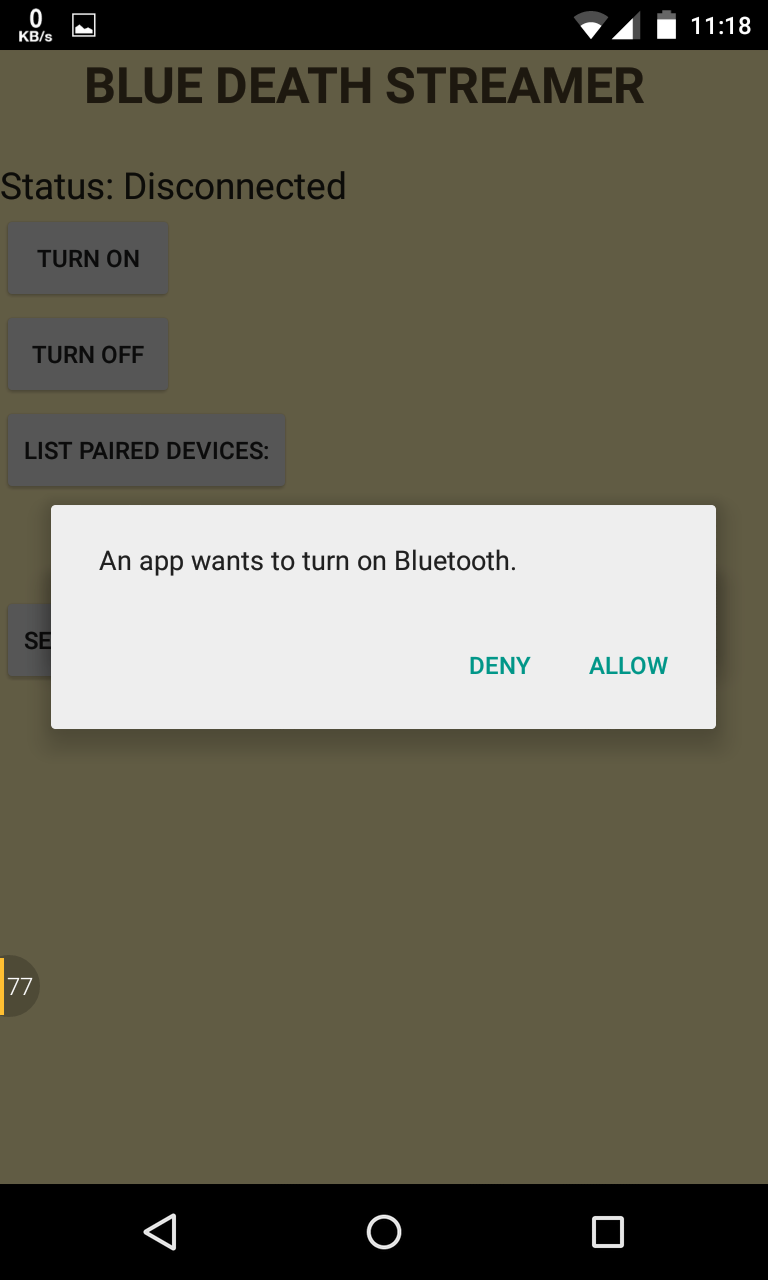
1. **Main Application Home:**

****

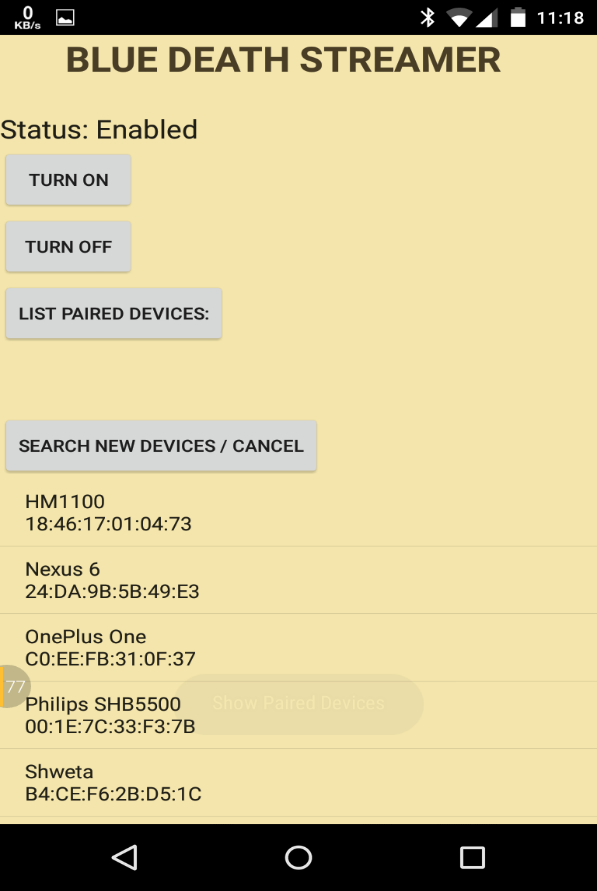
1. **Settings Page:**

****

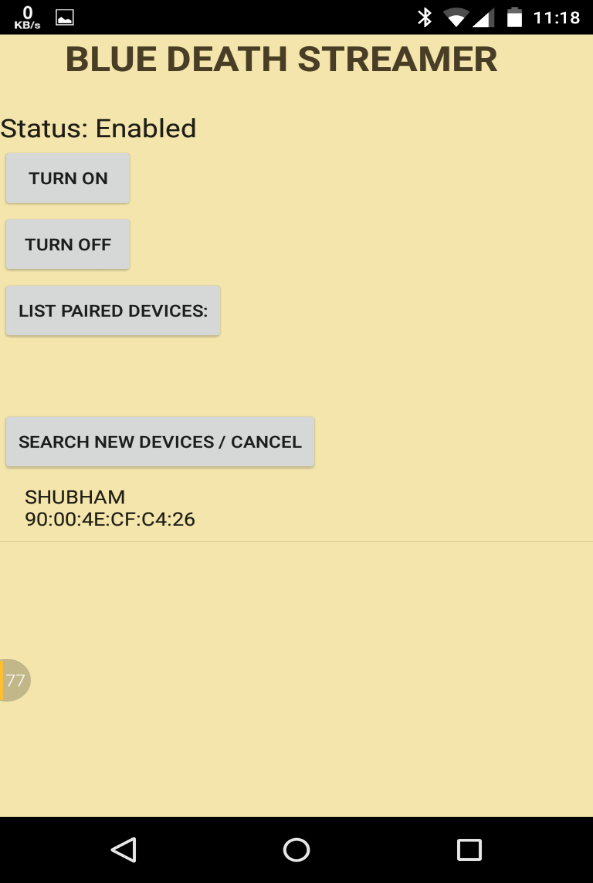
1. **Bluetooth Enable feature:**

****

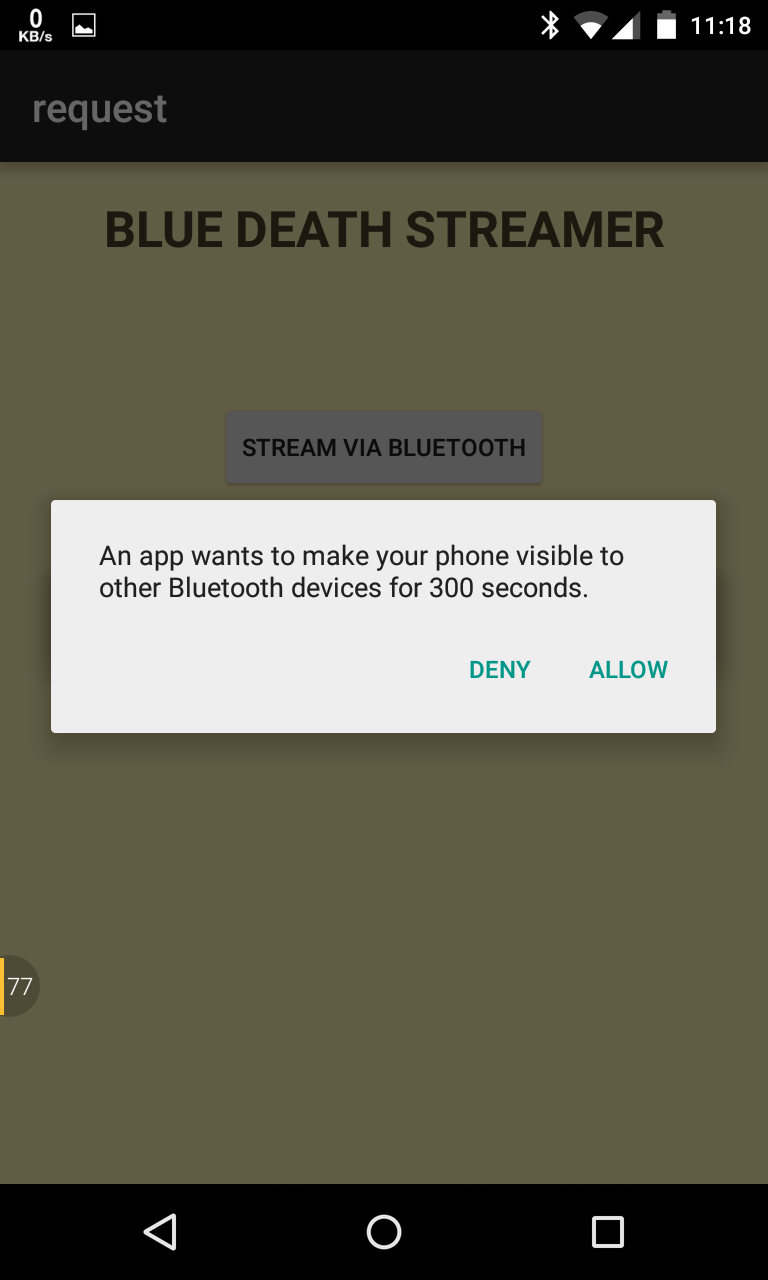
1. **Show Paired Devices:**

****

1. **Show Available Devices:**

****

1. **Sending file by streaming:**

****

1. **Sending file to selected device:**

