By: Matthew Patience

Vicinity SDK Tutorial

Current Version: 2.0

What is Vicinity?

Vicinity is an SDK for Android devices that allows them to connect devices to each other over Wireless LAN for communication purposes. The original intention of the library was to facilitate connecting Mobile Android Devices to Google TV, both of which run Android, but can also work with Smartphone to Smartphone and Smartphone to Tablet.

- You have 2 options when deciding how to build your application. You may either choose to build 2 separate applications, one for the remote and one for the server, or you can build 1 for both. The following describes 2 separate applications.
- For the SDK to work properly both devices must be on the same local network since a direct IP connection is made and no firewalls can be allowed to block the connection.
- This SDK supports Android 2.1+
- This library is licensed under Apache License 2.0

Important Classes

- **Vicinit**y - This class provides important static methods for getting the IP address of the server device to the remote device. These methods include scan() and getIpQrCode().
- **ServerActivity** - This class should be extended by Activities that handle the server logic.
- **RemoteActivit**y - This class should be extended by Activities that handle the remote logic.
- **VicinityApplicatio**n - This class should be extended and set in your Android Manifest as your Application.

Tutorial

Server Implementation

1. Start a new project and open up your AndroidManifest.xml. Insert the following permissions:
    a. **android.permission.INTERNET**
    b. **android.permission.ACCESS_NETWORK_STATE**

2. Create a new class in your root package called "MyApplication" that extends VicinityApplication. If you have anything specific for your application to do here, then do so, otherwise you can leave it as an empty class.
3. In your AndroidManifest add the following to your application tag:
    a. **android:name=".MyApplication"**
4. Create an Activity and extend ServerActivity. Implement all the methods that are required to compile the class, these include **connected()**, **disconnected()**, **messageReceived()**, and **connecting()**. In this activity you will have to display an ImageView for the next step. Where you choose to put it is up to you.
5. On your ImageView from the last step call **ImageView.setImageBitmap(Vicinity.getIpQrCode())** in your activity's **onCreate()** method. You can resize this bitmap before applying it the image if you wish.
6. Immediately after setting the QR Code image, call **getServerManager().startListening()**. **getServerManager()** is a method from the ServerActivity that you extended.
7. Ensure that you call **stopListening()** on the ServerManager after you are finished connecting devices. This could be done in **onDestroy()** for example.
8. Please note that the methods **connected()**, **disconnected()**, and **messageReceived()** will be called off the UI Thread. If you plan to modify UI during these methods you can post your changes to the UI Thread via **View.post(Runnable)**.

Remote Implementation

1. Complete steps 1 - 3 of the server application setup for a new application.
2. Create an Activity and extend RemoteActivity. Implement all the methods that are required to compile the class, these include **connected()**, **disconnected()**, **messageReceived()**, and **connecting()**.
3. You may make this UI interaction however you like but you will essentially require a way for users to start the scanning process. For example, in **onClick()** for a button you will need to call **Vicinity.scan(Activity)**.
4. When the users returns from the scanning process the connection to the server will automatically be made. When the device successfuly connects the **connected()** method will be called.

Interaction Options

From both applications simply call **getServerManager()** or **getRemoteManager()** and call **sendMessage()** , **sendBroadcast()**, or **sendDirectMessage()** to send messages.