ISSN:0975-9646
Alex Chirayath et al, / (IJCSIT) International Journal of Computer Science and Information Technologies, Vol. 6 (5) , 2015, 4361-4363

# Battery Optimization of Android OS

Alex Chirayath, Alfred Gonsalves, Chinmayi Kulkarni, Mahendra Mehra
*Department of Computer Science, Fr.CRCE, Mumbai University Mumbai, India*

**Abstract- The Android OS, based on the Linux OS offers features, functionality and an open architecture that has become the most widely mobile and tablet OS in the world. However, Android devices even after being so widely used have many processes that can cause the battery to drain very quickly even when the functioning of these processes is not necessary. Hence, the users of Android devices must intelligently and proactively manage the energy in their batteries. Optimizing Android is always a challenge, because the Android stack is spread across tools, domain specific frameworks from community projects, frameworks developed by Google, Linux Operating System, protocol stacks, etc. This paper aims at showcasing all such activities that are eating into the device battery without the user having any idea about it. It presents a mechanism in the form of an application that will intelligently and efficiently manage different components that affects the battery life.**

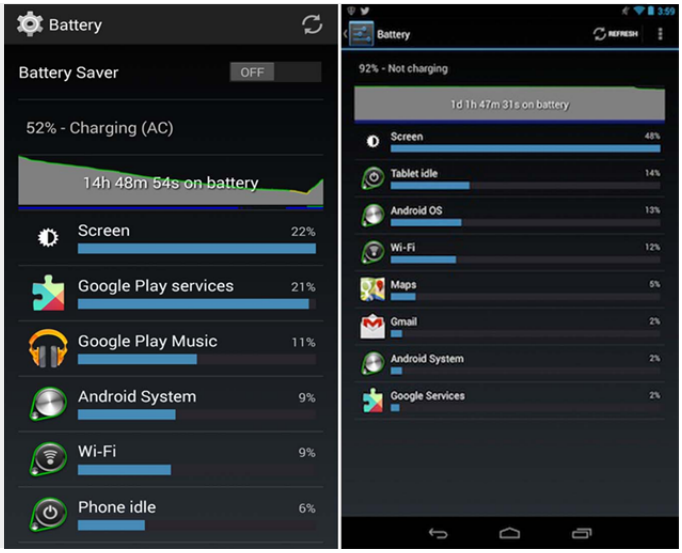
**Keywords: Android, Service, Broadcast, Intent, Wi-Fi, Bluetooth, GPS, Location, Smartphone.**

## 1. INTRODUCTION

Android operating system offers features of connectivity such as Bluetooth for file transfer, WiFi and also other features such as GPS. All these processes consume the battery of the device. The other factors that affect battery are the constantly running background services, screen brightness, constantly updating widgets on the home screen, NFC remaining ON even when not used. The smartphone is called a 'smart' phone because it runs multiple processes simultaneously. Every app you install in your phone take some storage space and runs some background processes. The more storage space occupied or the more background processes running on your phone, the slower your phone's performance. These processes run as a service on the android device and is not visible to the user. But these processes can lead to unnecessary RAM usage and hence can lower the performance and speed of the device. The other main factor in battery drainage is Wi-Fi and Bluetooth. If the device is not connected to any Wi-Fi network, the Wi-Fi switch should be toggled to off. If it is ON, the device continuously checks if a Wi-Fi network is available for it to connect and therefore eats up a lot of battery.

Here are some Bluetooth data samples, based on a Motorola Droid:

| *Mode* | *Energy* |
|---|---|
| *Bluetooth receive* | *751 mW* |
| *Bluetooth send* | *487 mW* |
| *Bluetooth standby* | *2.8 mW* |




Similarly Wifi sums upto 8-20% of the battery usage of a normal Android cellphone device depending on the actual use

## 2. SYSTEM SERVICES FOR BATTERY OPTIMIZATION

The Android platform includes various system services like background services, Bluetooth, GPS, Wi-Fi etc. which can be managed in a more efficient way to improve the battery life. Taking steps like disabling certain services when the connectivity is lost, or managing when to toggle the Wi-Fi, using the system location services for smart Wi-Fi toggling, the battery life can be improved.

### 2.1 Services

The Android platform provides and runs predefined system services and every Android application can use them, given the right permissions. These system services are usually exposed via a specific Manager class. Access to them can be gained via the *getSystemService()* method [4].

A *service* is a component which runs in the background without direct interaction with the user. As the service has no user interface, it is not bound to the lifecycle of an activity. They are used for repetitive and potentially long running operations, i.e., Internet downloads, checking for new data, data processing, updating content providers and so on. Services run with a higher priority than inactive or invisible activities and therefore it is less likely that the Android system terminates them.

*BroadcastReceivers* [5] simply respond to broadcast messages from other applications or from the system itself. It is used as a base class and will receive intents sent by *sendBroadcast()*. These messages are sometime called events or intents.

There are two important steps to be carried out to make Broadcast Receivers work: 1) Creating the Broadcast Receiver
2) Registering the Broadcast Receiver A Broadcast Receiver can be created by overriding the *onReceive()* method of *BroadcastReceiver* class where each message is received as an *intent* object parameter.

Intents [6] can be created to call other activity from another activity.

```
Intent intent = new Intent(packageContext,
className.class);
intent.startActivity(className);
```

Android services are not meant to run all the time. They are more like a task solver, e.g. download a file. After finished with their task, they should call *Service.stopSelf()*.
If your app needs the service again, it should only restart it then for the appropriate task. We can use Broadcast notifier and receiver to restart only the services which are needed for a particular task.

There are many applications which run a single service or multiple services. Google services use services to manage the location, connection manager, etc. Messaging applications like Whatsapp, Facebook Messenger, Snapchat etc.
use the 'Push Message Service'. Other applications also start services depending on the task at hand. Some of these services do not need to be running permanently. For instance, if the device does not have access to internet, services which use the internet need not be in the running state. They are the unnecessary users of RAM. Such services should be switched off and only be restarted once the internet is available or when the application is started by the user. We can use broadcast receiver to notify when the internet access is available. Thus we can save on the RAM usage and in turn reduce the battery consumption. In such cases, custom broadcasts can be written to be implemented on a particular event change.

### 2.2 Bluetooth

The Android platform allows wireless exchange of data between two devices that support the Bluetooth network stack [7]. Every device that supports Bluetooth service has a Bluetooth Adapter associated with it.

```
BluetoothAdapter ba=
BluetoothAdapter.getDefaultAdapter();
```

The *BluetoothAdapter* provides functionalities such as device discovery, obtaining list of paired devices and listening to connection requests. Any Bluetooth enabled device can connect with any other Bluetooth enabled device located in proximity to one another.
The Bluetooth continues to be switched on even when no communication is taking place. As long as the Bluetooth is switched on, it keeps scanning for devices for communication. This drains the battery of the device. To avoid this situation the Bluetooth should be switched off when the device leaves the area or if the device is in the hold state for a considerable time span.

*2.2.1 Using Bluetooth Socket*

The actual exchange of data between the devices takes place through the Bluetooth Socket. This socket is created using the Bluetooth Device which is in turn created using the Adapter.

```
BluetoothDevice bd =
ba.getRemoteDevice (String address)
BluetoothSocket bs =
bd. createRfcommSocketToServiceRecord
(UUID uuid)
```

The socket provides functions which inform whether the device is connected or not. If the device isn't connected for a large time span, the socket should be closed and the Bluetooth must be turned off.

```
if (bs.isConnected) then
ba.disable();  //Disable Bluetoothadapter end
```

*2.2.2 Using BroadcastReceiver and Intent*

Broadcast Receiver keeps a track of all the occurring events in the system. Intents are used to inform the Android system that a certain event has occurred.
The next action to be performed by the system is based on this event.

```
IntentFilter  intent =new IntentFilter
(bd..ACTION_ACL_CONNECTED);

//Register the Broadcast Receiver
registerReceiver(mReceiver ,f); private final
BroadcastReceiver mReceiver=new
BroadcastReceiver(){ public void onReceive(Context
context ,Intent intent)}.

//Listen to current activity
String action = intent.getAction();
if(bd.ACTION_ACL_CONNECTED.equals
(action))
then //BT is connected
else
ba.disable          //Disconnect Bluetooth
}}
```

*****The time interval of the device connection can be checked using the system time and implementing functions of the Alarm Manager Class available in the Android OS and Threads.

### 2.3 GPS-Wi-Fi

Battery power used by Wi-Fi while not connected can be significant, presumably since it keeps scanning to try to find networks to connect to. Whenever the Wi-Fi is ON but not connected to any Wi-Fi network for a considerable amount of time, it is disabled by using setWifiEnabled(false) provided by the WifiManager class [8]. This class provides the primary interface for managing all aspects of Wi-Fi connectivity.

*Disable Wi-Fi when not connected*

```
WifiManager wifi = (WifiManager)
getSystemService(Context.WIFI_SERVICE); if
((wifi.isWifiEnabled)&&!(wifi.isConnected)) then
setWifiEnabled (false); End.
```

The Wi-Fi status check can be achieved at regular time intervals by using the *onTick()* and *onFinish()* methods of the *CountDownTimer* class.

Applications access the location services supported by the device through classes in the *android.location* package. The central component of the location framework is the *LocationManager[9]* system service, which provides APIs to determine location of the underlying device. The following string constants can be used with certain classes for accessing the location services of device.

1. *ACCESS_FINE_LOCATION*

Allows an app to access precise location from location sources such as GPS, cell towers, and Wi-Fi.

2. *ACCESS_COARSE_LOCATION*

Allows an app to access approximate location derived from network location sources such as cell towers and Wi-Fi.

3. *NETWORK_PROVIDER*

This provider determines location based on availability of cell tower and WiFi access points. Results are retrieved by means of a network lookup

Whenever the device is connected to a Wi-Fi network, the latitude and longitude of the device can be maintained in an array list for future reference.
This is done by using the *LocationManager* class along with the *LocationListener*.

The *LocationManager* makes use of the Listener and gives notifications when the location changes. At a later stage, when the device Wi-Fi is OFF, and its location is already present in the array list, it indicates that the device has entered the range of a previously used network. The device is then notified [10] regarding the same and given an option to toggle the Wi-Fi.

*Retrieve current location of the device*

```
locManager = (LocationManager)getSystemService
(Context.LOCATION_SERVICE);
locManager.requestLocationUpdates (provider, 0,
0, locListener);
Location last = locManager.getLastKnownLocation
(provider); last.getLatitude; last.getLongitude;
```

*Notify user*

```
NotificationManager nm =
(NotificationManager)getSystemService
(Context.NOTIFICATION_SERVICE);
Notification n=new Notification
(android.R.drawable.alert_light_frame,
"Message",System.currentTimeMillis());
nm.notify(0,n);
```

Switching on the WiFi is areas where recognised WiFi networks are available also enables the device to consume lower battery amounts as receiving and sending data packets(connecting to the Internet) consumes a lot more of the battery resources of the Android device when using the cellular connection compared to that when the device is connected on Wi-Fi.

## 3 OTHER FACTORS

*Display Screen*

Most of the android devices have a high resolution display which displays different colors very distinctly. But this feature, though very attractive Chances are that screen brightness is somewhere near the top of the battery use list. Making a few easy tweaks like reducing the brightness or setting low screen timeout will help reduce battery drain from screen brightness.

If the device has an AMOLED screen one can enjoy big power savings just by using a black background for the home screen. That's because AMOLED screens use less power to display black.

Screen brightness can also be adjusted depending on the time of the day using the system clock of the device(for eg- reduce the brightness at night time)

## CONCLUSION

Though there have been many updates to the Android OS and with various other applications making the Android OS even smoother, a major concern is the battery life of the devices. Increasing the battery capacity seems to be a solution, however, it is not feasible as it increases the weight and cost of the phone and will also help only the future gadgets. Therefore, the ideas discussed focus on the modifications that can be made on the existing services provided by the platform and in turn enhance the battery usage by enabling certain smart features. If these features can be applied to the existing OS itself, the battery consumption of the phone can be reduced by a considerable amount giving a more efficient way to use the resources of one of the most essential devices of our day to day lives.

## REFERENCES


1. The New Boston android tutorials
2. Professional Android 2 Application Development by Reto Meier
3. http://www.howtogeek.com/howto/25319/ complete-guide-to-maximizing-your-androidphones-battery-life/
4. AndroidServices/article.html
5. http://developer.android.com/reference/android/ content/BroadcastReceiver.html
6. http://developer.android.com/reference/android/ content/Intent.html
7. http://developer.android.com/guide/topics/ connectivity/bluetooth.html