Effective Development of Andriod Applications using Andriod Services

M Hariprasad Reddy[1], Rekha B S[2]

[1]*Department of Information Science & Engg, R V College of Engineering, Bangalore*
[2]*Department of Information Science & Engg, R V College of Engineering, Bangalore*

[1]moole.hari@gmail.com
[2]rekhakumar26@gmail.com

***Abstract*— Android is the most popular operating system for mobile devices like smart phones and tablets. Android is developed on top of Linux with some modifications suitable for mobile devices. Gartner survey [1] stated that android shares 43% of smart phone market, while apple is only 18%. There are 400,000+ apps available in android market and 556,750+ apps are there in Apple app store These figures shows the importance of developing apps for android. Here the question is how to make your app selective/popular/effective for the users among thousands of Apps. Android provides a lot of flexibility to the developers. SERVICES are the most important application components in Android. This paper will discuss about services that can be used to make your apps effective. How to use services where to use services, and where they are not helpful, and how services differ with Threads and Async task with the help of three example Apps.**

***Keywords*— Services in Android make apps effective, how to use services, Services, Effective Android Apps, Advantages of services.**

I. Introduction

Today, smart phones are the replacement of Computers/Laptops. Smart phones demands apps with high Responsiveness, Rich UI. Using Services in apps can make them High responsive. Android provide four types of application components [2]. They are Activities, Services, Content Providers and Broadcast receivers. A service can perform long running operations in background. Service does not provide UI.

Services might use for playing Music, performing File I/O operations, Network transactions all from the background. Here for these operations developers can also use Async task or threads .But why we have to use services? The answer is whenever resources required the system kills some processes. Async task and threads are having more priority than services to get killed by System. And another advantage is service can be specified with what to do after recreation of service. There are three types of services [3] 1) Started service 2) Bounded service 3) Started cum bounded services. Basically there are only two types. Type 3 is the ways services can also be exist. In this paper i will discuss on the implementation of three types of services and use of services for three kinds of applications, comparison of usage of service than Async task and thread for those apps. Using this one can understand the importance of Services and how to use services in apps to make them effective.

II. Implementation of Services

As mentioned in section 1 there are three types of services 1) started service 2) bounded service 3) started cum bounded service

A. *Started Service*

Started services [4] are Services that are started by an application component by calling startService() method. Started service will hold the resource even if the components which started it are not destroyed. Started services do not allow the activities to interact with them. Once started they performs a operation and does not return any result to the caller. A started service runs until someone stops the service by stopService() or service stopped by itself using stopSelf(), after completion of operation. If no stopService() or stopSelf () is called then the service will be exist even after completion of operation. The service is destroyed if system running out of resources.

B. *Bounded Service*

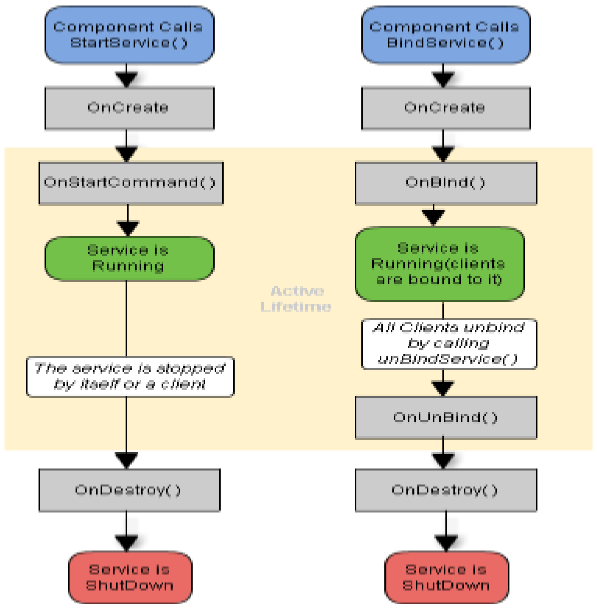
A service is said to be "Bound" [5] when an application component is bound to it by calling bindService(). Advantage of the bounded services is they offer a client server interaction. It means the clients bounded to a service can send a request and get the results from the service. Multiple components can get bound to a service. Here important thing to note is among the android's four application components only 3 types of components can bound to a service. They are Activities, Services and Content providers. We cannot bind a service from broadcast receiver. A bounded service runs until the components that are bounded to the service and will destroyed if all the components bounded to it are destroyed.

C. Started Cum Bounded Service

As the name suggests, Service first started by any application component by startService() and then some other components may get bound to this service. Here the service is remains even if all the components get unbounded from service. This service is stopped in a way similar to started service. Simply we can say this as a started service with interaction. This type is not mentioned anywhere but this is possible.

Like all Application components are declared in manifest file Services also should be mentioned in Android manifest file. You can also define intent filters for service. The intent filters allow the components to call your service even from any other applications installed on user's device. For now I am not discussing more about intent filters here is the way to declare services in manifest file [6]

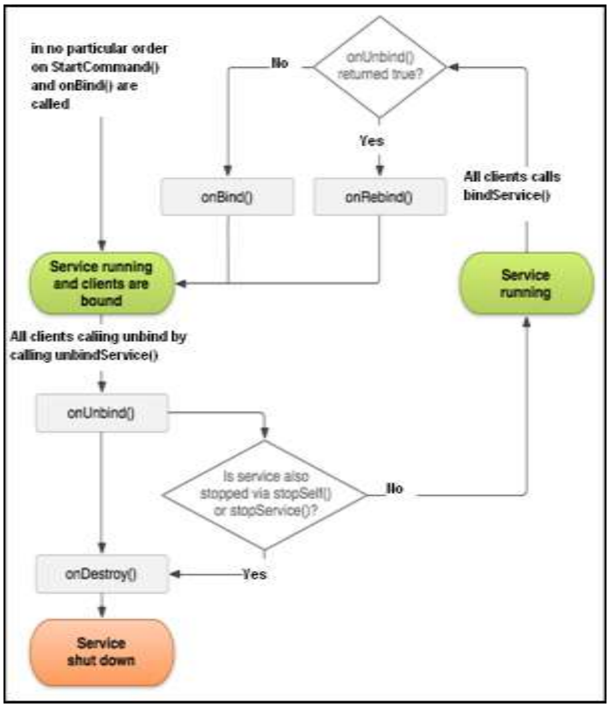
```
<ssManifest ... >
    <application ... >
      <service android:
name=".Example Service" />
   </application>
</manifest>
```



Component Calls StartService()
OnCreate
OnStartCommand()
Service is Running
The service is stopped by itself or a client
OnDestroy()
Service is ShutDown
Component Calls BindService()
OnCreate
OnBind()
Service is Running(clients are bound to it)
All Clients unbind by calling unBindService()
OnUnBind()
OnDestroy()
Service is ShutDown
Active Lifetime

Figure 1 Lifecycle of Started and Bound Services

The figure1 clarifies the states of both started and boundServices. The active life time of started service is until onDestroy() method is called, if no stopSelf() of stopService() is called. But for Bound services if the application components bounded to it are unbinded, the service gets destroyed. figure2 [7] sows the started cum bound service. Where a bound service even run until stopself() or stopService() is called.



in no particular order on StartCommand() and onBind() are called
onUnbind() returned true?
No
Yes
onBind()
onRebind()
All clients calls bindService()
Service running and clients are bound
Service running
All clients calling unbind by calling unbindService()
onUnbind()
Is service also stopped via stopSelf() or stopService()?
No
Yes
onDestroy()
Service shut down

Figure 2 started-cum-bounded service lifecycle [7]

Generally a service runs in application's main thread. You can make a service to create a separate thread and do the operations in that thread to avoid application not responding error (ANR)

D. How To Define Service's Thread

Generally a service runs in application's main thread. You can make a service to create a separate thread and do the operations in that thread to avoid application not responding error (ANR).

There are two ways to define a service Extend your service with "Service". Service is super class of all services. This makes your service to run in same application thread. If you extend your service with "IntentService" your service's task will run in separate thread. It means your service will be in main thread but when you implement a service with extend service you should implement onHandleIntent() method.

This method is similar to doinBackground() method of Asynctask class. The task mentioned in onHandleIntent () is executed in separate thread. There are other benefits with this IntentService, Like it will stops the service after all start requests of that service are finished. You no need to call stopself() method for service.

Android systems will force-stops any application component or threads when memory is low. It must recover system resources for foreground process. To know the possibilities of service get killed you should aware of types of processes in android system. There are 4 types of processes.

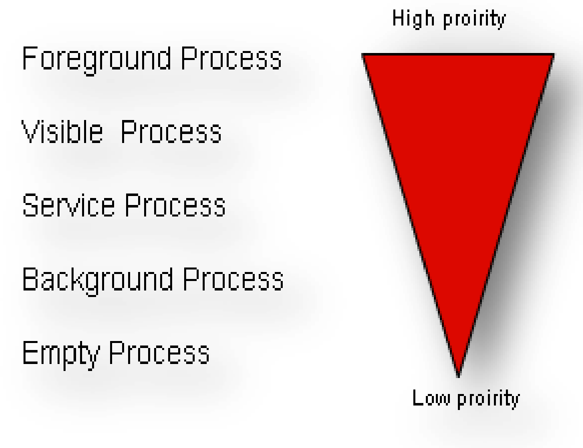


Figure 3 priority levels of android processes

The service has less priority than an activity. So system will kill a service when it needs resources for user focused activity, and system will recreates the service when resources are available. Services give flexibility to determine its nature at the time of recreation. For services you can specify how the system should continue the service in the event that the system kills it. There are 4 types how the system continue with service while restarting.

1) START_NOT_STICKY

If a service started with method onStartcommand() which returns this, then the system does not restart the service unless there are pending intents to deliver. This return type is useful to avoid running your service when not necessary and when your application can simply restart any unfinished jobs. Pending intent is a description of intent and target action performs with it. Here I am not covering the pending intents now.

2) START_STICKY

If onStartcommand() method of a service returns this then the system restarts the service but do not re deliver the last intent. Instead of this the system calls onStartcommand() with null intent, unless there were pending intents to start the service, in which case, those intents are delivered.

This return type is useful for services which are used to play media players that are not executing commands, but running indefinitely and waiting for a job.

3) START_REDELIVER_INTENT

If the onStartcommand() of a service returns this, then the system restarts the service and call the onStartcommand() method with last intent that was delivered to the service. And if there are any pending intents they are redelivered. This return type is suitable for services that are actively performing a job that should be immediately resumed, such as downloading a file.

Another advantage of services over threads and Async task is that they both usually run in background. As shown in figure 3 the back ground processes are more prone to get killed by System and the service run in service process having more priority than background process, and service can be made foreground.

III. Example Apps

Now we will consider three examples apps and see the requirement of service to these apps so that their usage can improve the performance.

Consider the apps one is espn-cricinfo and Photoviewer and Packagetracker App applications. These two needs stringent network operations like getting data from server and sending requests/uploading photos to server.

A. ESPN CRICINFO App

First consider the espn-cricinfo app which shows the cricket Scores. Here we need to refresh the score as frequently as possible. Optimally refresh for every minute is preferable with respect to the processing capacity of the variety of devices. And the major requirement of apps like these is the updates have to be shown even app is closed.

Now we will see how WorkerThreads /Async tasks help us to satisfy above requirement. First requirement is to refresh the score frequently to make this possible through threads/Asynctask it can be possible if app is open. But important thing is using Async task/ WorkerThreads[x] might leads to ANR (application not responding) errors.

Worker Thread [8] needs to interact with UI Thread to update UI (user interface). And the worker thread will be performing network operations that consume high amount of resources , so it may affect the performance of UI thread .So it is not preferable to use Asynctask for network operations even app is open. But services can run in separate process thus avoiding ANR errors. If you wants to see the updated score notification even app is closed it is not possible using WorkerThreads/Asynctask. It is done by using services.

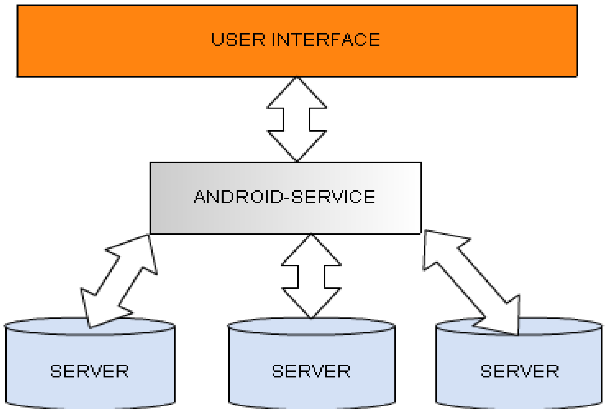


Figure 4 services interaction with UI and servers

B. *PHOTOVIEWER App*

Now consider the Photoviewer app. Photoviewer app allow the user basically to view/upload photos and comments. To upload a photo we can use Async task or services. using Async task requires app to be open until photo upload completes and even user cant switch from the activity .but using service it is possible, User can upload/download photos even app is closed. in Photoviewer apps authentication part plays important role because some permissions are given to particular users only.

Most of the operations need authentication credentials to be sent along with the request. For every request authentication credentials need to be sent increases LOC (lines of code) leads to increase in execution time. Generally if Async task is used for network calls then authentication credentials can be shared using sharedPreferences among all classes. But using service makes it in more trouble-free manner. Because a service can be used to make all type of server requests like view photos/upload photos/ upload comments. Saving the credentials in service have two advantages like network operations done using services and authentication credentials are stored at one place thus reducing LOC (lines of code).

C. *PACKAGE TRACKER App*

Now consider the package tracker application. Package tracker is an app that is used to track your parcel or order .Now a day's online shopping getting more demand there the shopped goods are delivered to the customer later. So to keep track the delivery status package tracker is most useful.

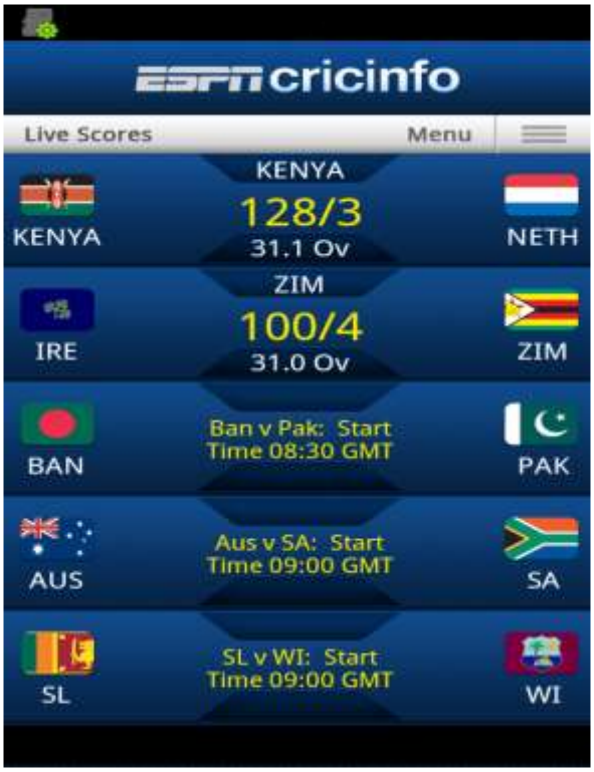
There is lot of scope for these kind Apps. The package tracker app needs to interact with tens of courier service servers. So here important thing is implementation of server calling mechanism. App should show the updated status instantly. It should not like the status is shown whenever user clicks on find status.

IV. Results

If we apply these fundamentals in implementation of internet based Android apps we will get effective results. The example screens for the cricinfo App are shown below. As shown in figure4 the service is running indefinitely and getting updates from server and also post them in UI.

Figure 5 shows the score screen. When app is opened, it readily displays the updated score without any delay. This is because service is updating the score, so whenever app is opened it displays the updated score. But without services user need to wait to see the score, because the request for updated score is fired when user opened the App.

Figure 6 shows the screen with notification message. If user closes the app and if there is any major update in the score to be shown, the service sends a notification toast and an animation for the message also possible



KENYA 128/3 31.1 Ov
KENYA NETH

ZIM 100/4 31.0 Ov
IRE ZIM

Ban v Pak: Start Time 08:30 GMT
BAN PAK

Aus v SA: Start Time 09:00 GMT
AUS SA

SL v WI: Start Time 09:00 GMT
SL WI

Figure 5 score displayed on the screen [9]

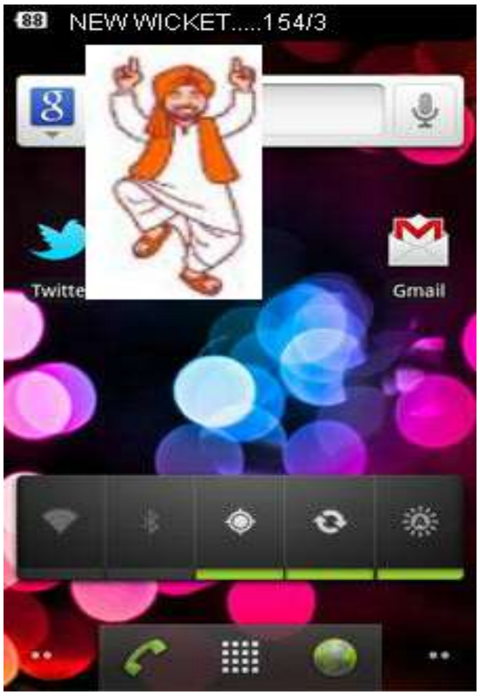


Figure 6 Menu Items on the App bar

V. CONCLUSION

The Android Phone platform has offered a lot of opportunities for developing new kinds of applications and games. This paper provides an insight to the developers to use services in developing internet based applications. This noble approach of services provides flexibility of implementation of developer's thoughts with fewer efforts .usage of services in makes apps effective and can implement the rich user interface methodologies.

The service mechanism allows developer to increase the multitasking ability of the android phone. Services play an important role in developing legacy applications like developing a generic Photoviewer application. This can upload photos to multiple photo hosting sites simultaneously.

REFERENCES

[1] Gartner survey on smart phones market: http://www.gartner.com/it/page.jsp?id=1689814
[2] Android application components http://developer.android.com /guide/topics/fundamentals.html
[3] Type of Serviceshttp://developer.android.com/guide/topics /fundamentals/services.html
[4] Started services http://developer.android.com/guide/topics/ fundamentals/services.html
[5] Bound services http://developer.android.com/guide/topics/ fundamentals/bound-services.html
[6] Declaration of service in manifest http://developer.android.com /guide/topics/fundamentals/services.html
[7] Managing the Lifecycle of a Bound Service http://developer.android.com/guide/topics/fundamentals/bound-services.html
[8] Worker threads http://developer.android.com/guide/topics/ fundamentals/processes-and-threads.html
[9] Espn-cricInfo App Scree http://thehandheldblog.com/2011/02/15/ espn-cricinfo-android-ios-app/