Available Online at www.ijcsmc.com

International Journal of Computer Science and Mobile Computing

A Monthly Journal of Computer Science and Information Technology

ISSN 2320–088X

IJCSMC, Vol. 4, Issue. 11, November 2015, pg.77 – 83

RESEARCH ARTICLE

Mobile Computing on Android using Cloud Infrastructure

Alex Chirayath[1], Ruben Monteiro[2], Prof. Mahendra Mehra[3]

[1]B.E. Student, Computer Department, Fr.CRCE, Mumbai University, India

[2]B.E. Student, Computer Department, Fr.CRCE, Mumbai University, India

[3]Assistant Professor, Computer Department, Fr.CRCE, Mumbai University, India

Abstract- The Linux based Android OS offers a wide range of features, functionality and an open architecture that has become the most popular mobile and tablet OS in the world. However, Android devices even after being so widely used have many application processes that can cause the battery to drain very quickly. Along with this the new innumerable new applications getting added on the Android PlayStore everyday require more and more use of the important resources of the mobile device such as Processor, RAM, and memory .Hence, the users of Android devices must intelligently and proactively manage their applications and services on their phones. 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 the idea of using a more general and more useful method of solving this problem using cloud computing

Keywords: Android, Service, Cloud, Cloudlet, Smartphone.

1. Introduction

Android operating system offers features of connectivity such as Bluetooth for file transfer, WiFi and also other features such as GPS. Along with this, we have 3rd party applications that can be installed for different functionalities on the phone. However, as some applications need higher end devices, users of low end Android devices cannot make the full use of the wide range of applications available. Also, all these processes consume the battery of the device. 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, these lower your phone's performance. These processes run as a service on the android device and is not visible

© 2015, IJCSMC All Rights Reserved

to the user. But these processes can lead to unnecessary RAM usage and hence can lower the performance and speed of the device.

2. Cloudlets

A cloudlet is a mobility-enhanced small-scale cloud datacenter that is located at the edge of the Internet. The main purpose of the cloudlet is supporting resource-intensive and interactive mobile applications by providing powerful computing resources to mobile devices with lower latency. It is a new architectural element that extends today's cloud computing infrastructure. It represents the middle tier of a 3-tier hierarchy: mobile *device -- cloudlet -- cloud*. A cloudlet can be viewed as a *data center in a box* whose goal is to *bring the cloud closer*

3. Media Projection

Android 5.0 lets you add screen capturing and screen sharing capabilities to your app with the new **android.media.projection** APIs. This functionality is useful, for example, if you want to enable screen sharing in a video conferencing app.

The new**createVirtualDisplay()** method allows your app to capture the contents of the main screen (the default display) into a **Surface** object, which your app can then send across the network. The API only allows capturing non-secure screen content, and not system audio. To begin screen capturing, your app must first request the user's permission by launching a screen capture dialog using an **Intent** obtained through the **createScreenCaptureIntent()** method.

3.1 Capturing Screen Content

```
mSurfaceView= (SurfaceView)
findViewById(R.id.surface);

mSurface=
mSurfaceView.getHolder().getSurface();
// object which will contain screen
information
```

Once the variables are initialized the user will be asked for permissions using

```
startActivityForResult(mMediaProjection
Manager.
createScreenCaptureIntent(),
REQUEST_MEDIA_PROJECTION);
```

Thereafter the screen contents will be captured and be sent to the onActivityResult() the Intent passed on by this method will then have to be transferred onto a Surface Object for it to be transferred over a network.

© 2015, IJCSMC All Rights Reserved

```
mMediaProjection= mMediaProjectionManager.getMediaProjection(mResultCode, mResultData); // re initializing media projection


mVirtualDisplay = mMediaProjection.createVirtualDisplay("ScreenCapture", mSurfaceView.getWidth(), mSurfaceView.getHeight(), mScreenDensity, DisplayManager.VIRTUAL_DISPLAY_FLAG_AUTO_MIRROR,mSurface, null, null); // transferring contents onto Surface object
```

The Surface object thus obtained can thus be transferred over a network.

3.2 Obtaining Touch Co- ordinates

```
touchView.setOnTouchListener(newView.
OnTouchListener()
{
@Override
publicbooleanonTouch(View v,
MotionEventevent) {

textView.setText("Touch coordinates : "+
String.valueOf(event.getX()) + "x" +
String.valueOf(event.getY()));
returntrue;
}
});
```

These co-ordinates can be collected using a separate thread and be transferred over a socket.

3.3 Triggering a touch event

Motion events describe movements in terms of an action code and a set of axis values. The action code specifies the state change that occurred such as a pointer going down or up. The axis values describe the position and other movement properties.

© 2015, IJCSMC All Rights Reserved

```
MotionEventmotionEvent =
MotionEvent.obtain(
downTime,
eventTime,
MotionEvent.ACTION_UP,
x,
y,
metaState
);

// Dispatch touch event to view
view.dispatchTouchEvent(motionEvent);
```

For example, when the user first touches the screen, the system delivers a touch event to the appropriate **View** with the action code **ACTION_DOWN** and a set of axis values that include the X and Y coordinates of the touch and information about the pressure, size and orientation of the contact area.

Some devices can report multiple movement traces at the same time. Multi-touch screens emit one movement trace for each finger. The individual fingers or other objects that generate movement traces are referred to as *pointers*. Motion events contain information about all of the pointers that are currently active even if some of them have not moved since the last event was delivered.

The number of pointers only ever changes by one as individual pointers go up and down, except when the gesture is canceled.

Each pointer has a unique id that is assigned when it first goes down (indicated by **ACTION_DOWN** or **ACTION_POINTER_DOWN**). A pointer id remains valid until the pointer eventually goes up (indicated by **ACTION_UP** or **ACTION_POINTER_UP**) or when the gesture is canceled (indicated by **ACTION_CANCEL**).

The MotionEvent class provides many methods to query the position and other properties of pointers, such as getX(int), getY(int),getAxisValue(int), getPointerId(int), getToolType(int), and many others. Most of these methods accept the pointer index as a parameter rather than the pointer id. The pointer index of each pointer in the event ranges from 0 to one less than the value returned by **getPointerCount()**.

The order in which individual pointers appear within a motion event is undefined. Thus the pointer index of a pointer can change from one event to the next but the

© 2015, IJCSMC All Rights Reserved

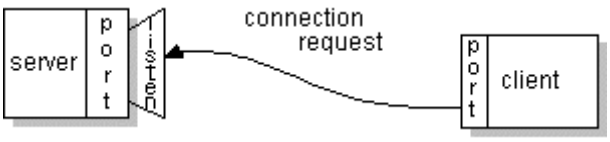
pointer id of a pointer is guaranteed to remain constant as long as the pointer remains active. Use the getPointerId(int) method to obtain the pointer id of a pointer to track it across all subsequent motion events in a gesture. Then for successive motion events, use the findPointerIndex(int)method to obtain the pointer index for a given pointer id in that motion event.

Mouse and stylus buttons can be retrieved using getButtonState(). It is a good idea to check the button state while handling ACTION_DOWN as part of a touch event. The application may choose to perform some different action if the touch event starts due to a secondary button click, such as presenting a context menu.

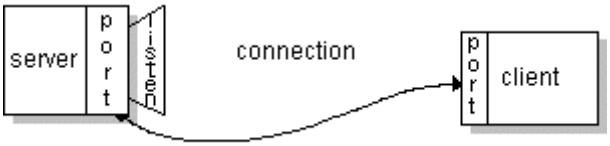
4 Socket Connection

Normally, a server runs on a specific computer and has a socket that is bound to a specific port number. The server just waits, listening to the socket for a client to make a connection request.

On the client-side: The client knows the hostname of the machine on which the server is running and the port number on which the server is listening. To make a connection request, the client tries to rendezvous with the server on the server's machine and port. The client also needs to identify itself to the server so it binds to a local port number that it will use during this connection. This is usually assigned by the system.



If everything goes well, the server accepts the connection. Upon acceptance, the server gets a new socket bound to the same local port and also has its remote endpoint set to the address and port of the client. It needs a new socket so that it can continue to listen to the original socket for connection requests while tending to the needs of the connected client.



On the client side, if the connection is accepted, a socket is successfully created and the client can use the socket to communicate with the server.

© 2015, IJCSMC All Rights Reserved

An endpoint is a combination of an IP address and a port number. Every TCP connection can be uniquely identified by its two endpoints. That way you can have multiple connections between your host and the server.

```
Thread socketServerThread = new
Thread(new SocketServerThread());
socketServerThread.start();

Socket socket = serverSocket.accept();

message = "connected";
LoginPage.this.runOnUiThread(new
Runnable() {

@Override
public void run() {
Log.d("MESSAGE", message);
    }
});

SocketServerReplyThreadsocketServerR
eplyThread = new
SocketServerReplyThread(socket);
socketServerReplyThread.run();
```

The java.net package in the Java platform provides a class, Socket, that implements one side of a two-way connection between your Java program and another program on the network. The Socket class sits on top of a platform-dependent implementation, hiding the details of any particular system from your Java program. By using the java.net.Socket class instead of relying on native code, your Java programs can communicate over the network in a platform-independent fashion.

Additionally, java.net includes the ServerSocket class, which implements a socket that servers can use to listen for and accept connections to clients. This lesson shows you how to use the Socket and ServerSocket classes.

If you are trying to connect to the Web, the URL class and related classes (URLConnection, URLEncoder) are probably more appropriate than the socket classes. In fact, URLs are a relatively high-level connection to the Web and use sockets as part of the underlying implementation.

5. Multi User Platform

The multi user feature allows more than one user on a single Android device by separating their accounts and application data. For instance, parents may let their children use the family tablet. Or a critical team might share a mobile device for on-call duty. The multi-user feature is disabled by default in the Android 5.0 release. Each user

© 2015, IJCSMC All Rights Reserved

is intended to be used by a different physical person. Each user has distinct application data and some unique settings, as well as a user interface to explicitly switch between users. A user can run in the background when another user is active; the system manages shutting down users to conserve resources when appropriate. Thus, a single Android OS on a cloud can be used by different users and can make use of the resources simultaneously.

6. Conclusion

There has always been many methods to optimize the device locally. Though there have been many updates to the Android OS and with various other applications making the AndroidOS even smoother, a major concern is the unnecessary use of important resources of the phone and battery life of the devices. Thus, using a cloud server to do all the work can reduce the work drastically for the local device. Therefore, the ideas discussed focus on little or no modifications on the existing services provided by the platform but to make use of a centralized cloud to run apps. Thus, the device would have to incur the cost of only the bandwidth for sharing data as all the processing cost would be at the server. With only the limitation of a good internet connection, cloudlets hold the key to the future of Mobile Computing

7. References

[1]http://developer.android.com/about/versions/android-5.0.html
[2]http://developer.android.com/reference/android/view/MotionEvent.html
[3]http://developer.android.com/training/gestures/multi.html
[4]https://en.wikipedia.org/wiki/Cloudlet
[5]https://docs.oracle.com/javase/tutorial/networking/sockets/definition.html
[6]https://source.android.com/devices/tech/admin/multi-user.html

© 2015, IJCSMC All Rights Reserved