Peer-to-Peer Communication Development of Twitter Social Networks

Taraka Praveen. Ede
Department of Electronics and Communication
Sri Chundi Ranganayakulu Engineering College
Chilakaluripet, India

S. Naga Raju[1], N. Prasad[2] and M. Suman[3]
Department of Electronics and Communication
Vignan's Lara Institute of Technology and Science
Vadlamudi, India

Abstract—*This Today different types of Social Networks available.Out of these, "Facebook" and "Orkut" are the most popular Social networks. These Social network applications are developed by using the Application Programming Interface (API) technique. Due to the existence of API, the privacy and security for the users is limited. These applications are working with both on PC based and Embedded platform. In order to add any new module, it becomes pay oriented. According to the Literature Survey it is noticed that most of the Social Networks published till today is mainly focused on Facebook applications only developed on Windows based Operating Systems.*

The above mentioned drawbacks in traditional Social Networks can be overcome by using Peer-to-Peer (P2P) Twitter Social Networks using Linux. In P2P networks P2P-Virtual Social Network (VSN) technique is used, which provides security to the users. This application is going to implement on Linux platform using x86 Processor which provides various services for the users at free of cost. In P2P grouping of users is done by Clique Technique. In this technique cliques are nested and can overlap. It is predicted that Linux Based Peer-to-Peer (P2P) Twitter Social Networks is going to facilitate security, sharing of knowledge, cost free oriented services like provides the facility of offline messages to its users via net to mobile and creation of communities related to their branches in a college can be added by anyone.

Decentralized / Peer-to-Peer Social Networking appears to be a very promising answer to the problem of many co-existing virtual social network platforms and the resulting problems of having to keep multiple identities and not being able to access the network overlapping the platform boundaries in a coherent manner. Both argumentations together imply that a suitable approach for modeling and demarcating sub-structures (e.g. sub-communities) in decentralized P2P social networking is necessary. We conclude by discussing candidate approaches for the problem. Final output is verified on Android OS as Twitter application is the main concern.

Keywords— Twitter; Qt; Android; Clique; Tweet

I. INTRODUCTION

The application Twitter client is implemented, in android. Social media and social networking have become very popular in recent years, especially with the emergence of websites such as Facebook, MySpace, and Twitter. In our project we are going to implement Twitter client application. Social media is a newer concept that has yet to adopt a standardized definition. However, social media can be best defined as various avenues through which people share information, public or private, with a select group of people. Social media and networking have become a diverse tool and can be accessed by anyone. One person can post or share information and share it with hundreds, even thousands, of people across the world in a matter of seconds. Now, with the emergence of social media, the same information can be shared in a matter of seconds.

Social media and networking has become such a large part of technology that companies are asking for social media skills from new applicants and are creating positions specifically in charge of social media. Just as public speaking and organizational skills have been important in the past, social media will probably become a basic job skill as the concept continues to evolve.

Under this aspect so many parameters like Twitter, Development relating to twitter networking are to be considered.

II. TWITTER & ITS CONSIDERATIONS

Twitter should probably be at number two on this list — they average 628,750,806 inbound links and weigh in with 23,000,000 monthly visitors, but their social impact is bigger than their numbers. With fewer monthly visitors than MySpace, they're relegated to third. Twitter is the "it boy" in social networking now, with some people leaving Facebook (way too gauche) for the more refined Twitter.

The cool part about Twitter is the ease with which you can find your favorite celebs and sports figures to follow. Unlike Facebook (just try "friending" one of your favorite musicians on Facebook) you can easily read star's and smart people's updates. Twitter could easily take over the number one spot on this list as more and more people join. Twitter Connections are shown in Fig1.

Navigating through the social media maze can be overwhelming at first. If you're just starting out, remember you can start small. In fact, I'd recommend taking smaller steps first, before you tackle writing your own blog or creating an online community. Here are a couple of easy ways to get started.Monitor relevant online conversations in social media.

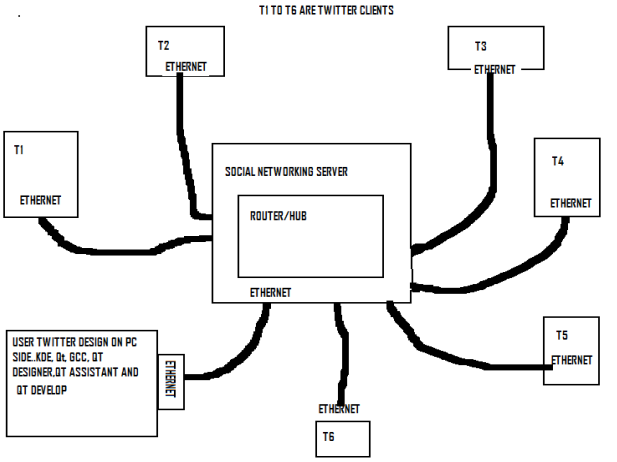

Fig. 1. Connection between Twitter Users

At first, Twitter can appear to be confusing, especially for those who may not be used to social networking or Internet lingo in general. However, with a couple of quick tips, Twitter can be a useful networking tool to use casually, professionally, or obsessively.

However, it should be cautioned that Twitter users should take the same steps to protect themselves as they would on other social networks, such as Facebook and LinkedIn

Each user accumulates a list of followers, or people who the user may know in real life, celebrities, companies, or organizations that the user has a genuine interest in. Users can follow many other users, creating many small social networks that can share information in a matter of seconds.

For the more advanced Twitter users, independent developers have created programs such as **HootSuite** and **TweetDeck**, which allow users to manage multiple lists of users, schedule tweets ahead of time, and further filter and sort tweets, trending topics, and other information coming across the Twitter channel of communication. These software platforms, which are usually free, also allow users to sync all of their social media profiles (Twitter, Facebook, LinkedIn, etc.) and manage them from one place.

Spreading the News on Twitter: **Retweeting** and **Retweets**

Join conversations. You don't have to write you own blog—you can comment and respond or answer questions in other blog posts, or on Twitter. Follow the same rules of etiquette you'd use in the physical world—make your comments relevant, behave ethically and be authentic—and remember to identify yourself and your company. Use relevant communities for market research. On LinkedIn, for instance, you can join relevant professional communities to discuss what's going on in your industry and ask questions. Or try Facebook Polls to poll targeted Facebook users, based on demographic data. With this tool, you can field a single-question poll in a few minutes, and get responses from hundreds of people in less than hour.

III. BASICS OF TWITTER & ITS WORKING

Twitter is quickly becoming one of the most popular forms of social networking. However, Twitter has some special characteristics that make it unique. Twitter, introduced as a form of quick social networking, is quickly becoming a popular media form for people to connect with others and to quickly share information.

Twitter works when users submit short messages of 140 characters (including spaces and punctuation) or less via cell phones, the Internet, or mobile applications to Twitter's website. The website will then send a user's update to his or her list of friends, also known as followers. These updates are called tweets.

Twitter had 400,000 tweets posted per quarter in 2007. This grew to 100 million tweets posted per quarter in 2008. By the end of 2009, 2 billion tweets per quarter were being posted. By March 2010, Twitter recorded over 70,000 registered applications, according to the company. In February 2010 Twitter users were sending 50 million tweets per day. In the first quarter of 2010, 4 billion tweets were posted. As of June 2010, about 65 million tweets are posted each day, equaling about 750 tweets sent each second, according to Twitter.

A. Additional Features

Some social networks have additional features, such as the ability to create groups that share common interests or affiliations, upload or stream live videos, and hold discussions in forums. Geo-social networking co-opts internet mapping services to organize user participation around geographic features and their attributes.

There is also a trend for more interoperability between social networks led by technologies such as OpenID and OpenSocial. Lately, mobile social networking has become popular. In most mobile communities, mobile phone users can now create their own profiles, make friends, participate in chat rooms, create chat rooms, hold private conversations, share photos and videos, and share blogs by using their mobile phone. Some companies provide wireless services which allow their customers to build their own mobile community and brand it, but one of the most popular wireless services for social networking in North America is Facebook Mobile.

B. Overview

As a social network, Twitter revolves around the principle of followers. When you choose to follow another Twitter user, that user's tweets appear in

reverse chronological order on your main Twitter page. If you follow 20 people, you'll see a mix of tweets scrolling down the page: breakfast-cereal updates, interesting new links, music recommendations, even musings on the future of education.

C. Messages

Users can group posts together by topic or type by use of *hashtags* — words or phrases prefixed with a #. Similarly, the letter d followed by a username allows users to send messages privately. Finally, the @ sign followed by a username is used for mentioning or replying to other users.

In late 2009, the "Twitter Lists" feature was added, making it possible for users to follow (as well as mention and reply to) lists of authors instead of individual authors.

Through SMS, users can communicate with Twitter through five gateway numbers: short codes for the United States, Canada, India, New Zealand, and an Isle of Man-based number for international use. There is also a short code in the United Kingdom which is only accessible to those on the Vodafone, O2 and Orange networks. In India, since Twitter only supports tweets from Bharti Airtel, an alternative platform called smsTweet was set up by a user to work on all networks. A similar platform called GladlyCast exists for mobile phone users in Singapore, Malaysia and the Philippines.

The messages were initially set to 140-character limit for compatibility with SMS messaging, introducing the shorthand notation and slang commonly used in SMS messages. The 140 character limit has also increased the usage of URL shortening services such as bit.ly, goo.gl, and tr.im, and content hosting services, such as Twitpic, memozu.com and NotePub to accommodate multimedia content and text longer than 140 characters. Twitter uses bit.ly for automatic shortening of all URLs posted on its website.

D. Trending Topics and Hashtags on Twitter

Sometimes, there are certain trends that Twitter picks up on through thousands of tweets that pass through the website. Once a large portion of tweets regarding one topic have been posted in a short amount of time, the topic of discussion becomes a trending topic.

Hashtags, or the # symbol, are used by Twitter users to identify topics that are trending at the time or that they wish to see trending. The hashtag symbol is placed in front of a word or a series of words to bring attention to them and group them with other users' tweets regarding the same topic.

When on Twitter's website, users can view tweets regarding trending topics by clicking on words or phrases immediately following the hashtag symbol. After understanding a few basic terms and concepts, Twitter can be a useful and fun form of social networking that can connect friends and strangers across the globe.

IV. TWITTER APPLICATION ON ANDROID

A. After Android introduction & its building blocks

Android is a comprehensive open source platform designed for mobile devices. It is championed by Google and owned by Open Handset Alliance. The goal of the alliance is to "accelerate innovation in mobile and offer consumers a richer, less expensive, and better mobile experience." Android is the vehicle to do so. As such, Android is revolutionizing the mobile space. For the first time, it is a truly open platform that separates the hardware from the software that runs on it. This allows for a much larger number of devices to run the same applications and creates a much richer ecosystem for developers and consumers. Android is a comprehensive platform, which means it is a complete software stack for a mobile device.

Android is a software stack for mobile devices that includes an operating system, middleware and key applications. Google Inc. purchased the initial developer of the software, Android Inc., in 2005. Android's mobile operating system is based on the Linux kernel. Google and other members of the Open Handset Alliance collaborated on Android's development and release. The Android operating system is the world's best-selling Smartphone platform.

Android has a large community of developers writing applications ("apps") that extend the functionality of the devices. The unveiling of the Android distribution on 5 November 2007 was announced with the founding of the Open Handset Alliance, a consortium of 80 hardware, software, and telecom companies devoted to advancing open standards for mobile devices. The Android open-source software stack consists of Java applications running on a Java-based, object-oriented application framework on top of Java core libraries running on a Dalvik virtual machine featuring JIT compilation.

Android is an open source platform. The entire stack, from low-level Linux modules all the way to native libraries, and from the application framework to complete applications, is totally open. More so, Android is licensed under business-friendly licenses (Apache/MIT) so that others can freely extend it and use it for variety of purposes. Even some third-party open source libraries that were brought into the Android stack were rewritten under new license terms.

So, as a developer, you have access to the entire platform source code. This allows you to see how the guts of the Android operating system work. As manufacturer, you can easily port Android OS to your specific hardware. You can also add your own proprietary secret sauce, and you do not have to push it back to the development community if you don't want to. There's no need to license Android. You can start using it and modifying it today, and there are no strings attached. More so, Android has many hooks at

various levels of the platform, allowing anyone to extend it in unforeseen ways.

B. Designed for Mobile Devices

Android is a purpose-built platform for mobile devices. When designing Android, the team looked at which mobile device constraints likely were not going to change for the foreseeable future. For one, mobile devices are battery powered, and battery performance likely is not going to get much better any time soon. Second, the small size of mobile devices means that they will always be limited in terms of memory and speed.

These constraints were taken into consideration from the get-go and were addressed throughout the platform. The result is an overall better user experience. Android was designed to run on all sorts of physical devices. Android doesn't make any assumptions about a device's screen size, resolution, chipset, and so on. Its core is designed to be portable.

The main building blocks are components that you use as an application developer to build Android apps. They are the conceptual items that you put together to create a bigger whole. When you start thinking about your application, it is good to take a topdown approach. Libraries written in C include the surface manager, OpenCore media framework, SQLite relational database management system, OpenGL ES 2.0 3D graphics API, WebKit layout engine, SGL graphics engine, SSL, and Bionic libc. The Android operating system, including the Linux kernel, consists of roughly 12 million lines of code including 3 million lines of XML, 2.8 million lines of C, 2.1 million lines of Java, and 1.75 million lines of C++.

C. Stack Overview

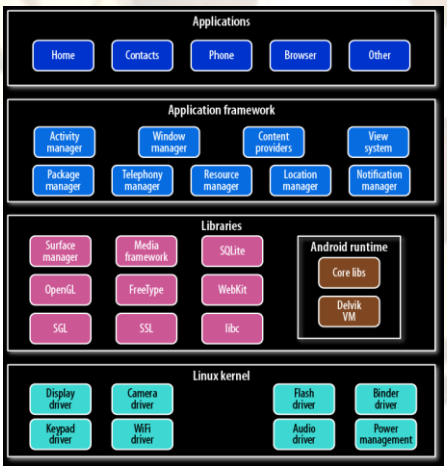

Fig. 2. Android Stack

The Android operating system is like a cake consisting of various layers shown in Fig 2. Each layer has its own characteristics and purpose. The layers are not cleanly separated but often seep into each other. When you read through this chapter, keep in mind that I am concerned only with the big picture of the entire system and will get into the nitty-gritty details later.

D. Linux

Android is built on top of Linux. Linux is a great operating system and the poster child of open source. There are many good reasons for choosing Linux as the base of the Android stack. Some of the main ones are its portability, security, and features.

1) Portability

Linux is a portable platform that is relatively easy to compile on various hardware architectures. What Linux brings to Android is a level of hardware abstractions. By basing Android on Linux, we don't have to worry too much about underlying hardware features.

2) Security

Most low-level parts of Linux have been written in fairly portable C code, which allows for third parties to port Android to a variety of devices.

3) Features

Linux comes with a lot of very useful features. Android leverages many of them, such as support for memory management, power management, and networking.

4) Dalvik

Dalvik is a purpose-built virtual machine designed specifically for Android, developed by Dan Bornstein and his team at Google.

The Java virtual machine (VM) was designed to be a one-size-fits-all solution, and the Dalvik team felt they could do a better job by focusing strictly on mobile devices. They looked at which constraints specific to a mobile environment are least likely to change in the near future. One of these is battery life, and the other is processing power. Dalvik was built from the ground up to address those constraints.

E. Android and Java

In Java, you write your Java source file, compile it into a Java byte code using the Java compiler, and then run this byte code on the Java VM. In Android, things are different.

You still write the Java source file, and you still compile it to Java byte code using the same Java compiler. But at that point, you recompile it once again using the Dalvik compiler to Dalvik byte code. It is this Dalvik byte code that is then executed on the Dalvik VM. Figure illustrates this comparison between standard Java (on the left) in Android using Dalvik (on the right).Shown in figure3.

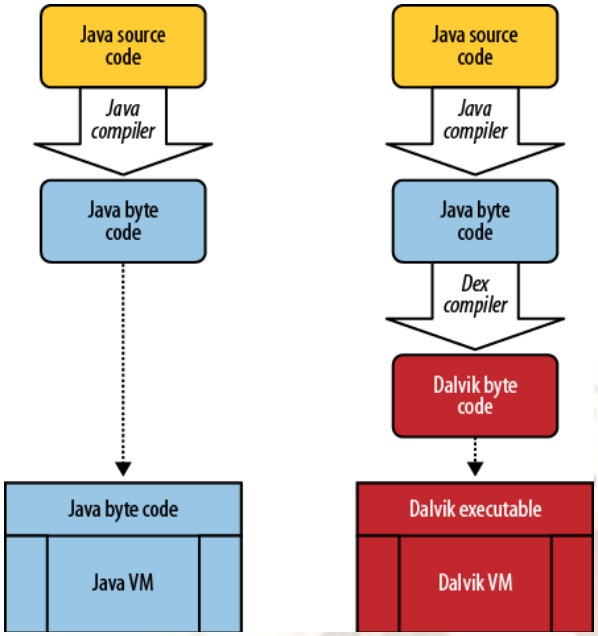

Fig. 3. Comparison between standard Java (on the left) in Android using Dalvik (on the right)

F. Activities

An activity is usually a single screen that the user sees on the device at one time. An application typically has multiple activities, and the user flips back and forth among them. As such, activities are the most visible part of your application.

I usually use a website as an analogy for activities. Just like a website consists of multiple pages, so does an Android application consist of multiple activities. Just like a website has a "home page," an Android app has a "main" activity, usually the one that is shown first when you launch the application. And just like a website has to provide some sort of navigation among various pages, an Android app should do the same.

On the Web, you can jump from a page on one website to a page on another. Similarly, in Android, you could be looking at an activity of one application, but shortly after you could start another activity in a completely separate application.

For example, if you are in your Contacts app and you choose to text a friend, you'd be launching the activity to compose a text message in the Messaging application.

G. Services

Services run in the background and don't have any user interface components. They can perform the same actions as activities, but without any user interface. Services are useful for actions that we want to perform for a while, regardless of what is on the screen. For example, you might want your music player to play music even as you are flipping between other applications.

Services have a much simpler life cycle than activities shown in Fig3.5. You either start a service or stop it. Also, the service life cycle is more or less controlled by the developer, and not so much by the system. Consequently, we as developers have to be mindful to run our services so that they don't consume shared resources unnecessarily, such as the CPU and battery.

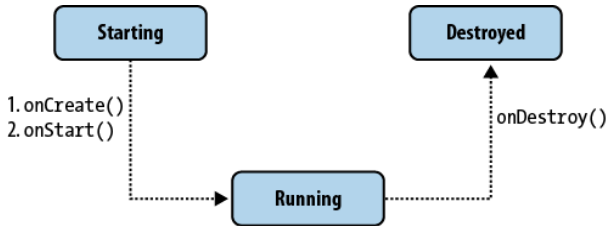

Fig. 4. Services

H. Twitter Application & its Activity

We know that the user should be able to post status updates. We also know the user should be able to see what her friends are up to. Those are basic features. Beyond that, the user should also be able to set her username and password in order to log into her Twitter account. So, now we know we should have these three screens.

Next, we would like this app to work quickly regardless of the network connection or lack thereof. To achieve that, the app has to pull the data from Twitter when it's online and cache the data locally. That will require a service that runs in the background as well as a database.

We also know that we'd like this background service to be started when the device is initially turned on, so by the time the user first uses the app, there's already up-to-date information on her friends. So, these are some straightforward requirements. Android building blocks make it easy to break them down into conceptual units so that you can work on them independently and then easily put them together into a complete package.

An activity is usually a single screen that the user sees on the device at one time. An application typically has multiple activities, and the user flips back and forth among them. As such, activities are the most visible part of your application.

I. API web service calls

Web service is a method of communication between two electronic devices over a network. The W3C defines a "Web service" as "a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically Web Services Description Language WSDL). Other systems interact with the Web service in a manner prescribed by its description using SOAP messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards."

The W3C also states, "We can identify two major classes of Web services, REST-compliant Web services, in which the primary purpose of the service is to manipulate XML representations of Web resources using a uniform set of "stateless" operations;

and arbitrary Web services, in which the service may expose an arbitrary set of operations.

While making the network calls, you'll notice that the UI starts behaving sluggishly, due to the unpredictable nature of the network. The network latency might even cause our application to stop responding. At that point, we will introduce multithreading in Android and explain how to develop an app that works well regardless of external circumstances.

1) Web API

Web API is a development in Web services (in a movement called Web 2.0) where emphasis has been moving away from SOAP based services towards Representational State Transfer (REST) based communications. REST services do not require XML, SOAP, or WSDL service-API definitions. Web APIs allow the combination of multiple Web services into new applications known as mashups.

When used in the context of Web development, Web API is typically a defined set of Hypertext Transfer Protocol (HTTP) request messages along with a definition of the structure of response messages, usually expressed in an Extensible Markup Language (XML) or JavaScript Object Notation (JSON) format.

When running composite Web services, each sub service can be considered autonomous. The user has no control over these services. Also the Web services themselves are not reliable; the service provider may remove, change or update their services without giving notice to users. The reliability and fault tolerance is not well supported; faults may happen during the execution. Exception handling in the context of Web services is still an open research issue. Still it can be handled by responding with an error object to the client.

J. Android Interface & its implementation

Android is an open-source software stack for mobile devices that includes an operating system, middleware and key applications.

Google Inc. purchased the initial developer of the software, Android Inc., in 2005. Android's mobile operating system is based upon a modified version of the Linux kernel.

Google and other members of the Open Handset Alliance collaborated on Android's development and release. The Android Open Source Project (AOSP) is tasked with the maintenance and further development of Android. Android is an operating system based on Linux with a Java programming interface. It provides tools, e.g. a compiler, debugger and a device emulator as well as its own Java Virtual machine (Dalvik Virtual Machine - DVM). Android is created by the Open Handset Alliance which is lead by Google.

An Android application consists out of the following parts: ***Activity, Views, Services, Content Providers, Intents, Broadcast Receiver***. During the installation of an Android application the user get a screen in which he needs to confirm the required permissions of the application.

Activity - Represents the presentation layer of an Android application, e.g. a screen which the user sees. An Android application can have several activities and it can be switched between them during runtime of the application.

Services - perform background tasks without providing an UI. They can notify the user via the notification framework in Android.

Content Provider - provides data to applications, via a content provider your application can share data with other applications. Android contains a SQLite DB which can serve as data provider.

Intents - are asynchronous messages which allow the application to request functionality from other services or activities. An application can call directly a service or activity (explicit intent) or asked the Android system for registered services and applications for an intent (implicit intents). For example the application could ask via an intent for a contact application. Application register themself to an intent via an IntentFilter. Intents are a powerful concept as they allow to create loosely coupled applications.

Broadcast Receiver - receives system messages and implicit intents, can be used to react to changed conditions in the system. An application can register as a broadcast receiver for certain events and can be started if such an event occurs.

Android defines certain permissions for certain tasks. For example if the application want to access the Internet it must define in its configuration file that it would like to use the related permission. During the installation of an Android application the user get a screen in which he needs to confirm the required permissions of the application.

Components required for Android Application development: Environment setup on host machine. Eclipse, Android SDK, according to the requirement API level is set, and the emulator setup. Here the emulator will be on Qemu as shown in Fig 5, Fig 6, and Fig 7.

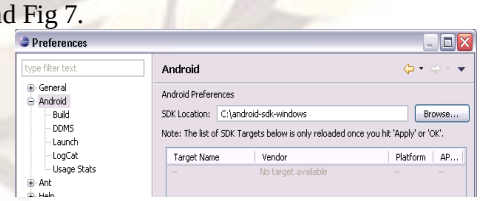

Fig. 5. Selection of location for SDK

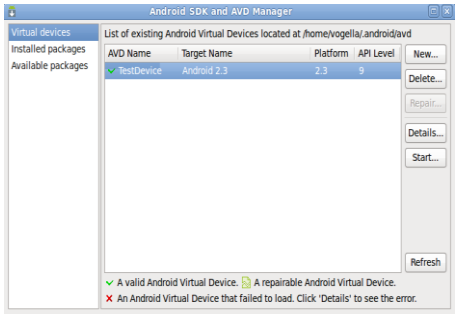

Fig. 6. AVD Emulator Creation

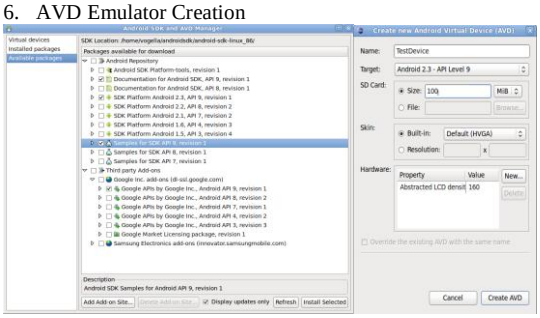

Fig. 7. Android AVD Selection

K. Android Interface Desktop

Fig. 8. Android Desktop

L. Twitter Application Final Output Window

Fig. 9. Android OS View

Fig. 10. Twitter Application

V. CONCLUSION & FUTURE SCOPE

Social networking models and their implementation compared with the earlier models and their drawbacks in the development have been reviewed. Here implementation is done on the Open Source Platform (Linux) and using Android OS as the key aspect in the Application (Twitter) development. This kind of development lags most of the drawbacks behind and lets you to get the best output.

There is also a provision for the deployment of the **.apk** file which is generated as part of the development of the main application can also be implemented on the real Android Mobile Phone.

The Project developed is only intended for the text based messaging service only.

This can be further modified for the audio based voice messaging service also. This requires some tools to be developed and implemented.

REFERENCES

[1] Georg Groh , Verena Rappel, Towards Demarcation and Modeling of Small Sub- Communities / Groups in P2P Social Networks 2009.IEEE, CSE, pp 304-311.
[2] Lextrait, Vincent (January 2010). "The Programming Languages Beacon, v10.0". Retrieved 2010-01-05.
[3] "Philosophy and Goals". *source.android.com*. Google Inc. 2011. Archived from the original on 2011-02-23. Retrieved 2011-02-23.
[4] "Open Core". Retrieved 2010-06-03.
[5] "Open Handset Alliance". Open Handset Alliance. Retrieved 2010-06-10.
[6] "About the Android Open Source Project". Retrieved 2010-11-15.
[7] "Google's Android becomes the world's leading smart phone platform (Canalys research release: r2011013)". *Canalys*. 31 January 2011. Retrieved 1 February 2011.
[8] Android-x86 - Porting Android to x86.
[9] Kirsner, Scott (2007-09-02). "Introducing the Google Phone". *The Boston Globe*. Retrieved 2008-10-24.
[10] D. J. Watts and et al. Identity search in social networks. Science, 269(5571), 2002.
[11] G. Groh Groups and Group Instantiations in Mobile Communities–Detection, Modeling and Applications Proc. ICWSM07, Boulder, Co, USA, Mar 2007.
[12] R. Steinmetz and K. Wehrle, editors. Peer-to-Peer Systems and Applications, volume 3485 of Lecture Notes in Computer Science. Spring
[13] Markoff, John (2007-11-04). "I, Robot: The Man Behind the Google Phone".\The New York Times. Retrieved 2008-10-14.
[14] Kirsner, Scott (2007-09-02). "Introducing the Google Phone". *The BostonGlobe*. Retrieved 2008-10-24.
[15] "Open Handset Alliance". Open Handset Alliance. Retrieved 2010-06-10.