International Journal of Multimedia and Ubiquitous Engineering
Vol. 10, No. 3 (2015), pp. 1-12
http://dx.doi.org/10.14257/ijmue.2015.10.3.01

Experimental Comparison of Hybrid and Native Applications for Mobile Systems

Seung-Ho Lim

*Department of Digital Information Engineering
Hankuk University of Foreign Studies
slim@hufs.ac.kr*

Abstract

Currently, mobile applications can be developed with two ways, native or hybrid applications. In this paper, we develop representative mobile application, social network service application, in both of native method and hybrid method to evaluate with the application at the aspect of development cost and performance cost. There are two key factors in developing mobile applications; the one is user interface design, and the other is efficient utilization of device capabilities such as various sensors, cameras, network interfaces. The UI composition time and networking performance are experimented with developed applications. From the results, we identify that the UI for hybrid applications had better to develop with dynamic composition nature, and mobile device capabilities had better to be done aggregately when these are used in hybrid applications.

Keywords: *Mobile device, Native Applications, Hybrid Application, User Interface, Device Capabilities*

1. Introduction

During last five years, mobile devices, such as smartphones, and tablets have been much more popular than traditional desk-based devices such as personal computers and lap-tops. Since iOS has released with iPhone smartphone in 2007, and Android has released with various smartphones and tablets in 2008, the most popular Operating Systems running on the computing devices has become mobile operating systems. Accordingly, the more applications are running on the mobile Operating System than desk-based Operating Systems.

The development of mobile Operating System-based application has become popular. The application program that runs on smartphones and tablets is called mobile application. The mobile applications are typically developed and operated according to their Operating Systems, and are usually available from the application distribution platforms, such as Google Play, Apple App Store. The popularity of mobile applications has continued to rise, as their usage has become increasingly prevalent across mobile phone users [1]. A May 2012 comScore study reported that during the previous quarter, more mobile subscribers used apps than browsed the web on their devices: 51.1% vs. 49.8% respectively [2].

Different from the developers of desk-based applications, developers of mobile applications should consider the constraints of mobile devices, such as a wide variety of screen size and hardware specifications, due to the intense competition in mobile systems within each of the mobile platform. There are two key factors in developing mobile applications; the one is user interface design, and the other is efficient utilization of device capabilities such as various sensors, cameras, network interfaces. The mobile user interface design (UI Design) is one of the essential processes among the development issues. Mobile user considers constraints and contexts, screen, input and mobility as outlines for design, and interacts with their device with UI components and some actions. Thus, the objective of the mobile UI design is primarily for understandable and user-friendly interface design.

Current development mobile application is dependent on running Operating Systems and corresponding framework, and accordingly mobile application development [3] requires use of specialized integrated development environments. The developed mobile application can be run only the platform that it was developed, not on the different platform. The applications built for a specific platform with the platform Software Development Kit (SDK), tools, and language provided by the platform vendor is called Native Application. For example, a mobile application developed with Android framework is not run in iOS framework. The development of mobile application as a native method is good for performance since it can fully utilize the availability of development kit and resources of mobile system with the help of kit and API. However, the development time and cast may increase, since we have to develop and make differently for different platform with same application.

On the contrary, there is application that can run on different platforms with single development with the help of web technologies. A Hybrid Applications run on the mobile devices with the development of web technologies such as HTML5, CSS and JavaScript, with inside a native container. It leverages the device's browser engine to render the HTML and process the JavaScript locally. Therefore, the UI design and implementation can be relatively easy with HTML script and CSS technology. However, the mobile web has limitation to access to low-level capabilities of devices. A web-to-native abstraction layer enables access to device capabilities that are not accessible in web engine, such as the sensor devices, camera and local storage [4]. The representative web-to-native application frameworks are Sencha Touch[5] and jQuery Mobile[6].

Although the one-source-multi-platform capabilities is possible with Hybrid Application development, it has performance degradation, in comparison with Native Application, since the user interface developed with HTML5, CSS, and JavaScript is relatively heavier than native development, and web-to-native abstraction layer of hybrid application gives additional overhead to utilize device capabilities. The key two factors of developing mobile application are affected by Hybrid Application.

To compare the strength and weakness of Hybrid Application and Native Application in mobile systems, in this paper, we develop representative mobile application in both of Native method and Hybrid method, and evaluate with these applications at the aspect of development cost and performance cost. A Social Networking Service is an online service, platform, or site that focuses on facilitating the building of social networks or social relations among people who, for example, share interests, activities, backgrounds, or real-life connections [13]. In developing Social Network Service application, especially, we focus on user interface design and network interface operations to evaluate performance cost. Since UI responsiveness and interface is one of the key factors, we consider the UI design factor, and the network capabilities of mobile device is one of the most essential factor among many of device capabilities, especially for mobile applications such as SNS.

By evaluating with these two developing method, we can identify the strength of mobile application development methodologies, and the feasibility of development approaches of mobile applications, whether native application is better or not than hybrid application in some cases, or vice versa.

The remainder of this paper is organized as follows. In Section 2, background and related work is described. The developed native and hybrid mobile SNS systems are described in detail in Section 3, and its performance evaluation and comparison are described in Section 4. Section 5 concludes this paper.

2. Background and Related Work

In this Section, the backgrounds and related work are described. At first, we explain the difference of native application and hybrid application in the aspect of development environment and impact for mobile application, especially for UI design and usage methodologies for device capabilities. Then the background technical issues of developed representative mobile application for experimental comparison between native application and hybrid application is introduced.

Native application is developed by native software development tools and languages provided by the platform provider. Among many platform providers, we focus on Android platform since Android is dominant platform in the world currently. Android uses the Dalvik virtual machine (VM) to execute and contain programs in Java. Dalvik is modified version of Java Virtual Machine for mobile embedded systems [11]. Based on the Dalvik, Android provides a comprehensive set of development tools and software frameworks including UI, network interfaces, device capabilities, libraries, and so on. The Application Program Interfaces (API) provided by Android framework gives sufficient repository for developer to implement their idea to application with full utilization of device capabilities.

One of the most popular development environment for hybrid application is PhoneGap [9,12] development framework, which is purchased by Adobe Systems in 2011 [7, 8]. The PhoneGap enables developers to build mobile applications with HTML5, CSS, and JavaScript, not to use platform specific APIs provided by platform provider. The languages are oriented by web technologies to describe web contents and UI interfaces easily with simple script expressions. In the PhoneGap development framework, making mobile application with these is possible by wrapping up of HTML and JavaScript source codes from the platform framework of the mobile devices. The results for wrapping these codes from platform are hybrid application, which means that they are neither purely native application, nor purely web application. In addition to PhoneGap framework, the standard JavaScript library is required to describe precisely of mobile interfaces. There are two main libraries for mobile application, jQuery Mobile and Sench Touch. jQuery Mobile is a touch-optimized mobile framework and JavaScript library, currently being developed by the jQuery project team. The development focuses on creating a framework compatible with a wide variety of smartphones and tablet [6]. Sencha Touch is another user interface JavaScript library or framework, specifically built for the mobile application. It is fully based on web standards such as HTML5, CSS3 and JavaScript.

As part of the development process, mobile user interface (UI) design is an essential in the creation of mobile apps. Mobile UI considers constraints & contexts, screen, input and mobility as outlines for design. For Android native application development, each UI components can be generated by Java source code provided by Android libraries. In addition to Java code, Android has the ability to utilize XML to perform UI components for many standard tasks. The Android XML schema is highly flexible and may be used in combination with code, exclusively, or not at all, without loss of performance overhead. On the contrary, HTML5 tags and JavaScript descriptions of UI components are embedded in web engine that is integrated in device platform, so conversion is required to be displayed into the screen from Script expression to UI components by the wrapper of web engine, which might generated overhead of latency and user interaction. The other essential consideration part of choosing development environment is how to utilize device capabilities such as network interface, device storage media, and several sensor devices including location sensor, camera sensor, and accelerometer sensor. Since current mobile applications usually dedicated focus on mobile use like communication, location-based services, requiring mobile device capabilities. Moreover, mobile applications likely utilize cloud computing systems for higher networking operations like message pushing, data backup, and storage. Thus, the ability to use device capabilities well is one of the key issues for developers to implement their idea to mobile applications.

In this paper, we chose representative mobile application to compare native and hybrid application is mobile social network services. Mobile social networking service is the networking that individuals converse and connect with each other through their mobile device. A current trend for social networking service is to give their users instant and real-time access from their device with very UI components. The advances of mobile devices are able to use social networking activities with advanced UI components and advanced access from networks. However, a lot of social network service applications suffer from developing and maintaining of their applications for user to use with high quality. This is not only the platform related developing cost, but also performance issues for user latency and IO operations with device components including network and storage media.

From next section, the development issues for social network service application are described in detail for both of native application and hybrid application. For native application, Android platform and framework-based development is uses, and PhoneGap and Sencha Touch libraries are used for hybrid application development.

3. Design and Implementation of SNS Mobile Application

In this section, the detailed design and implementation issues for developing native application and hybrid application is explained. At first, we describe overall architecture of SNS mobile application designed in this paper, and then the comparison points for native and hybrid application are dealt. Among many implementation issues, we focus on implementation methodologies for UI components and request transmission and receive methodologies via networking interfaces.

3.1 Overall SNS Applications

Usually, a mobile SNS application gives personal bulletin boards to each user to update their personal activities and share these with friends. The updated contents of bulletin board are broadcasts to other users who connect with each other to notice the updating of activities. In addition to that, private communication channels between each users or groups of users are provided to communicate each other. The implemented Mobile SNS applications are composed of the described features of SNS application, which has personal user profile, friends list, timeline newsfeeds space called blog, and messaging with text, voices, or video. The user profile manages private user information such as affiliation, phone number, and so on. The personal timeline space shared user timeline space for updating personal activities and sharing these with friends as a blog manner. Chatting service gives messing and communication channel between friends of group of friends. The overall architecture of designed SNS application is described in Figure 1.

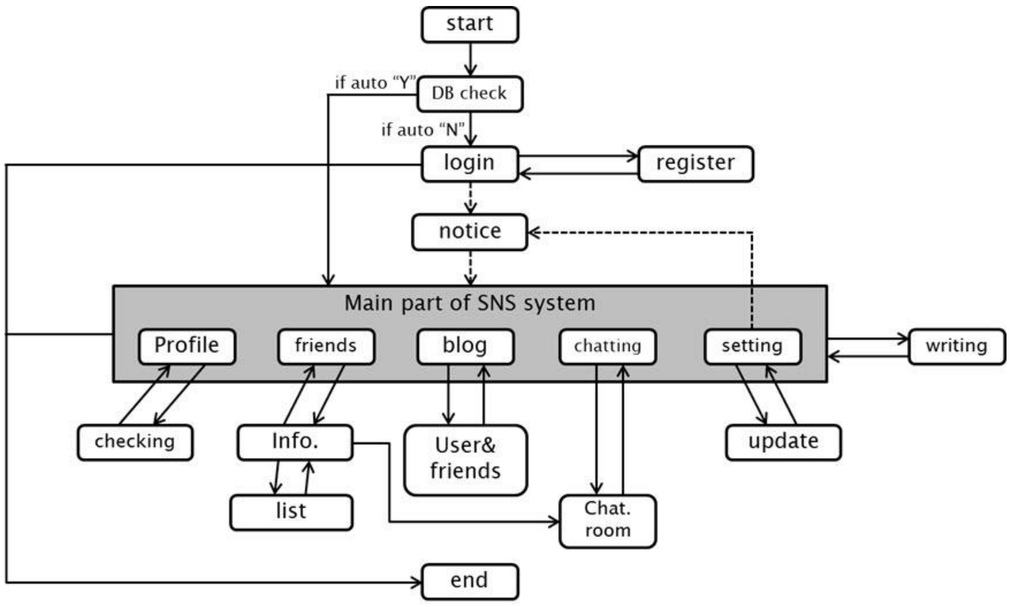


Figure 1. Overall Architecture of Mobile SNS Application

In the SNS service system, server is required to network operations. Since network operation is HTTP request-based in our SNS application, we have set up web server with Apache, and PHP is used to implement client service, and the back-end, database server is also present. All the information of user in SNS system is stored with a database table. The overview of client and server system of SNS application, and database table for maintaining user information is described in Figure 4. Database table consists of members, chat_list, board, and userID table. Members table manages personal information of each user, chat_list table collects all the chatting list of each user and group of users, board table manages the personal blog's information for each user, sharing of friend's information. Lastly, userID table is used for relationship between users. For each user, it has chatting list having chatting partners, and friend list which connect to, so with this information, Server can interconnect users for blogs and chat histories.

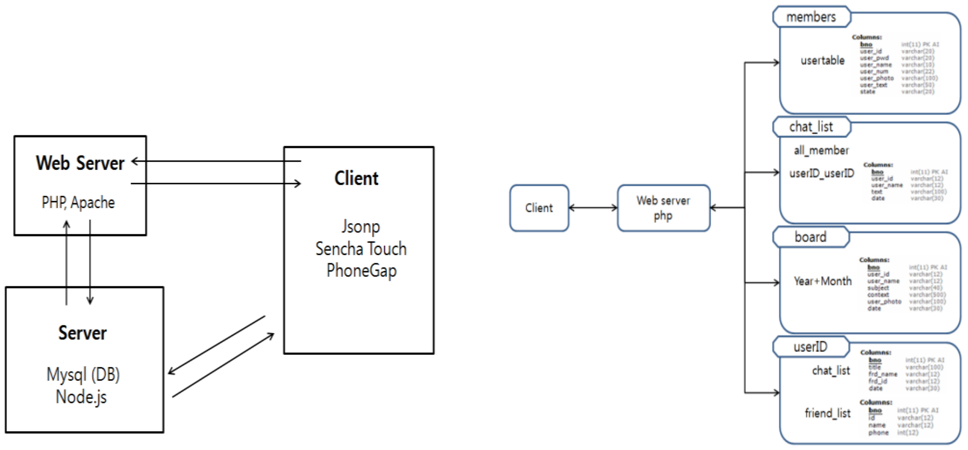

Web Server
PHP, Apache
Client
Jsonp
Sencha Touch
PhoneGap
Server
Mysql (DB)
Node.js
Client
Web server php
members
usertable
Columns: bno int(11) PK AI, user_id varchar(20), user_pwd varchar(20), user_name varchar(10), user_num varchar(22), user_photo varchar(100), user_text varchar(50), state varchar(20)
chat_list
all_member
userID_userID
Columns: bno int(11) PK AI, user_id varchar(12), user_name varchar(12), text varchar(100), date varchar(30)
board
Year+Month
Columns: bno int(11) PK AI, user_id varchar(12), user_name varchar(12), subject varchar(40), context varchar(500), user_photo varchar(100), date varchar(30)
userID
chat_list
Columns: bno int(11) PK AI, title varchar(100), frd_name varchar(12), frd_id varchar(12), date varchar(30)
friend_list
Columns: bno int(11) PK AI, id varchar(12), name varchar(12), phone int(12)

Figure 2. Developed Mobile Client and Server System, and Database Table in SNS Application System

3.2 UI Components

In native application, especially Android native application, design and implementation of user interface uses similar UI components and concepts of Java user interfaces, however there are several differences between native Java UI and Android UI. All user interface components in Android applications are built using View and ViewGroup objects and each component is defined as a hierarchy of View and ViewGroup. The ViewGroup can be defined with layout making visual structure of each component. For easy making up UI elements, Android framework provides two ways to define layout and UI elements; implementing java native code to instantiate layout and elements at runtime and declaring elements with XML format.

Android UI elements can be implemented with programmatically with java code, or XML description. One of the problems of UI implementation for native application is downing the readability and increasing the complexity for source code, developing of GUI application is more difficult than developing web-based UI application, as well. For Android, making up UI with XML can give easy understanding of UI hierarchy of ViewGroup by separating the UI description from programming source code. Also, XML description is very readable in comparison with native source code and developer can modify UI without modification of source code. The basic UI elements and widget are provided by Android framework. Based on the basic Android framework, custom design of UI elements is possible to show more fancy UI description.

Some UI developments of native SNS application are described in the Figure 2, which includes login, and registration, personal newsfeed, friends and write new activities. In our native application, almost all UI elements are described with XML and some UIs are customized.

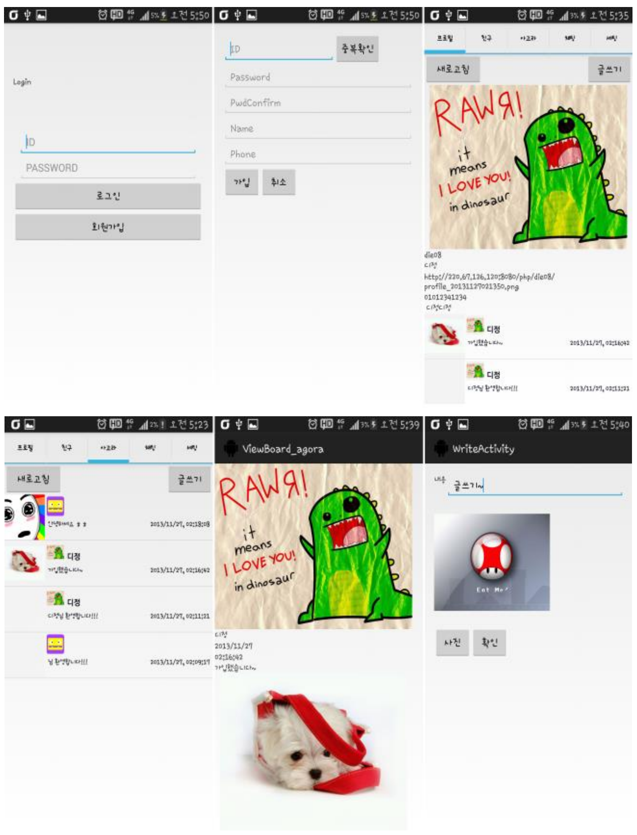
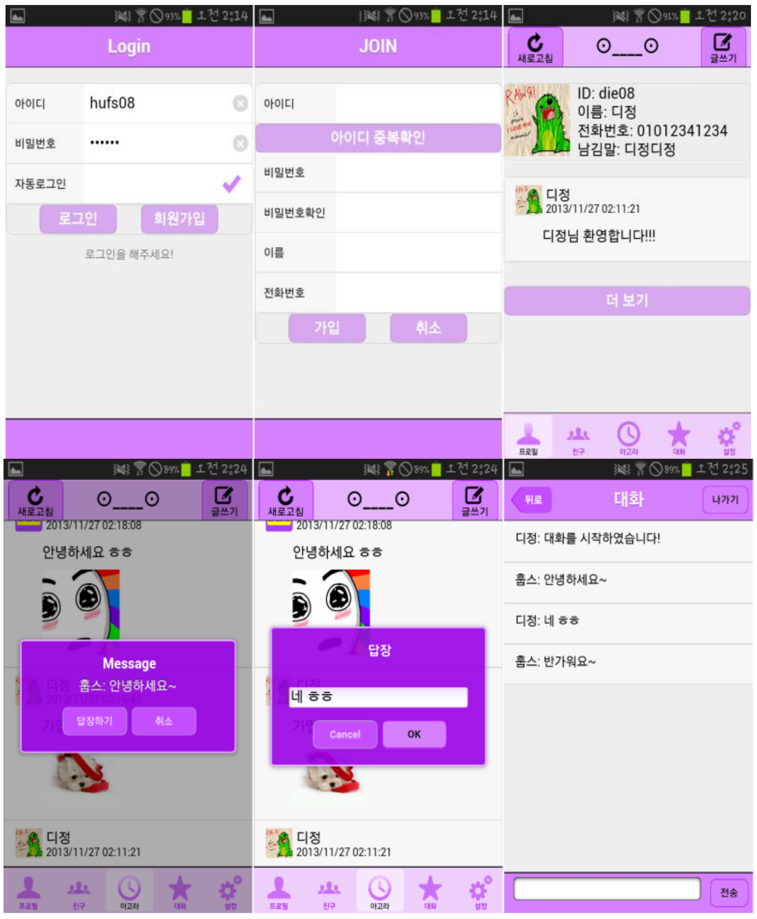


Figure 3. Subset UI Appearances Implemented for Android Native Application

On the contrary, HTML5 markup language and JavaScript uses for developing UI elements in hybrid application. HTML5 is a core technology markup language of the Internet used for structuring and presenting content for the World Wide Web. The strength of user interface design with HTML5 and JavaScript is that many UI developers are already familiar with these languages and easy to develop many UI elements with these. It can be argued that there is less of a learning curve when developing hybrid mobile applications compared to native applications [4]. However, the weakness of UI with hybrid applications is that the UI logics is build and run in the web browser using emulated web engine framework, resulting in extra layers of abstraction to display these UI element in display device of mobile systems.

We develop UI elements with HTML5 and Sench Touch JavaScript libraries with Phonegap framework environment. The developed storyboard of UI is described in Figure 3.



아이디 hufs08
비밀번호 ••••••
자동로그인 ✓

아이디
비밀번호
비밀번호확인
이름
전화번호

ID: die08
이름: 디정
전화번호: 01012341234
남김말: 디정디정

디정
2013/11/27 02:11:21
디정님 환영합니다!!!

2013/11/27 02:18:08
안녕하세요 ㅎㅎ

훕스: 안녕하세요~

네 ㅎㅎ

디정
2013/11/27 02:11:21

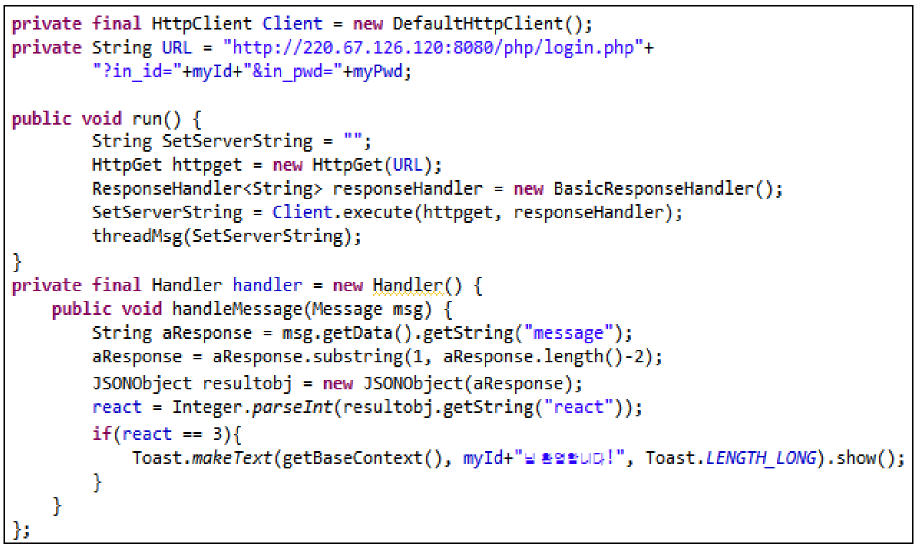
디정: 대화를 시작하였습니다!
훕스: 안녕하세요~
디정: 네 ㅎㅎ
훕스: 반가워요~

Figure 4. Subset of UI Appearances Implemented for Hybrid Application with Sench Touch

3.3. Network Request Transmission and Receive Methodologies

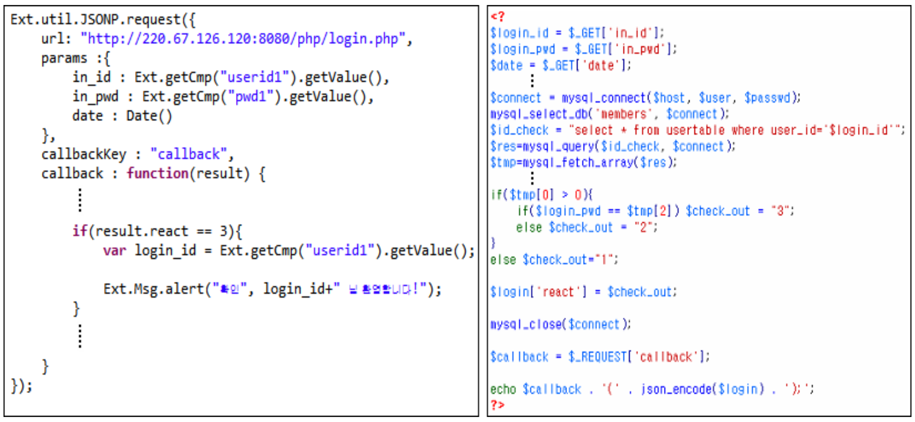
In our development, network operation is HTTP request-based in both of native application and hybrid application, and the request type of transmitting and receiving JSONP [14], where JSONP is usually used in JavaScript programs running in web browsers. Thus, there is no problem to use JSONP in hybrid application, and Android provides JSONP libraries to convert JSON data to and from Java string format with library APIs. In the server system, we deploy Node.js to provide client's request, where Node.js is an event-driven architecture and non-blocking I/O API to enhance throughput and scalability. The both of developed SNS applications do network operations based on this Node.js and JSONP data request type, via HTTP requests.

In the SNS application, network operation is one of the most frequently used processing types, where updating of personal activities to blog, viewing other users activities, chatting with others are all communication operations. The figure 5 and figure 6 represents an example of implementation of networking operations in native application and hybrid application, respectively. In native application, HTTP request APIs are used to communicate, and JSONobject library is used to parse JSON object data. The JSON and HTTP request is used with more nature in hybrid application since it is oriented from web system and JavaScript. The request format is transferred to server via web-engine directly, so which is a little overhead of web packing of framework around the web-engine.



```
private final HttpClient Client = new DefaultHttpClient();
private String URL = "http://220.67.126.120:8080/php/login.php"+
        "?in_id="+myId+"&in_pwd="+myPwd;

public void run() {
        String SetServerString = "";
        HttpGet httpget = new HttpGet(URL);
        ResponseHandler<String> responseHandler = new BasicResponseHandler();
        SetServerString = Client.execute(httpget, responseHandler);
        threadMsg(SetServerString);
}
private final Handler handler = new Handler() {
    public void handleMessage(Message msg) {
        String aResponse = msg.getData().getString("message");
        aResponse = aResponse.substring(1, aResponse.length()-2);
        JSONObject resultobj = new JSONObject(aResponse);
        react = Integer.parseInt(resultobj.getString("react"));
        if(react == 3){
            Toast.makeText(getBaseContext(), myId+"님 환영합니다!", Toast.LENGTH_LONG).show();
        }
    }
};
```

Figure 5. An Example of HTTP request in Android Native Application



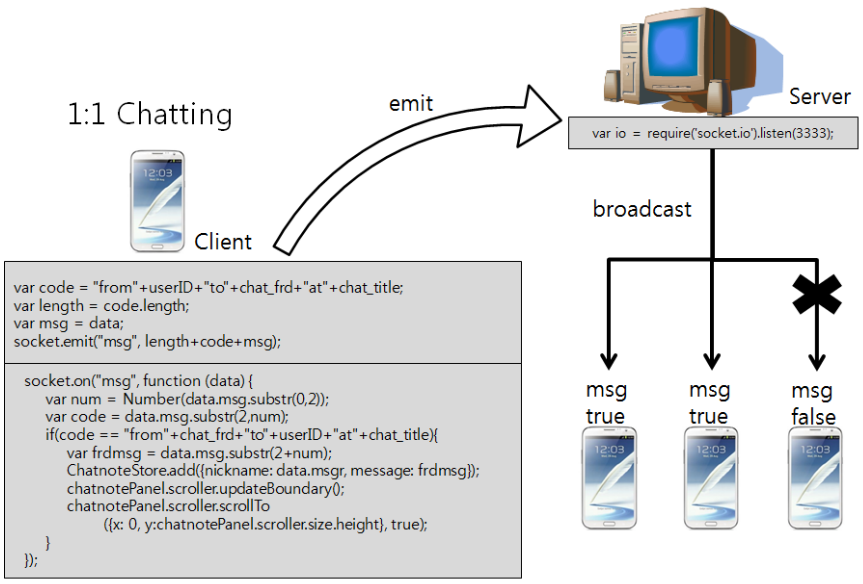
```
Ext.util.JSONP.request({
    url: "http://220.67.126.120:8080/php/login.php",
    params :{
        in_id : Ext.getCmp("userid1").getValue(),
        in_pwd : Ext.getCmp("pwd1").getValue(),
        date : Date()
    },
    callbackKey : "callback",
    callback : function(result) {
        ⋮
        if(result.react == 3){
            var login_id = Ext.getCmp("userid1").getValue();

            Ext.Msg.alert("확인", login_id+" 님 환영합니다!");
        }
        ⋮
    }
});
```

```
<?
$login_id = $_GET['in_id'];
$login_pwd = $_GET['in_pwd'];
$date = $_GET['date'];
      ⋮
$connect = mysql_connect($host, $user, $passwd);
mysql_select_db('members', $connect);
$id_check = "select * from usertable where user_id='$login_id'";
$res=mysql_query($id_check, $connect);
$tmp=mysql_fetch_array($res);
      ⋮
if($tmp[0] > 0){
    if($login_pwd == $tmp[2]) $check_out = "3";
    else $check_out = "2";
}
else $check_out="1";

$login['react'] = $check_out;

mysql_close($connect);

$callback = $_REQUEST['callback'];

echo $callback . '(' . json_encode($login) . ');';
?>
```

Figure 6. An Example of JSONP request with JavaScript in Hybrid Applications

There is critical technique for transmitting and receiving data in mobile systems, called push technology. Push presents a request for a request transaction is initiated by the central server not client. The push technology is key for mobile system, since it can reduce networking overhead at although the communication systems. In the SNS applications, the push technology is required in many parts of application, which includes notification of news feed update, notification of getting chatting message, alert of request for friend registration, and so on.

In Android framework, the push can be performed with Google Cloud Messaging which enables developers to send data from servers to mobile application, so in our application we implemented push technology with Google Cloud Messing in the required area. In web-based system, HTTP server also provides server push technologies. The server push technology is applied to our hybrid application to push data from server to mobile client. Figure 7 depicts a method of pushing chat message from one client to other users via server. When a user transmits a data to server, the server pushes this data to other people participate the chatting room as a broadcast manner, than the users can get chat data as a push manner.



```
var io = require('socket.io').listen(3333);
```

```
var code = "from"+userID+"to"+chat_frd+"at"+chat_title;
var length = code.length;
var msg = data;
socket.emit("msg", length+code+msg);
```

```
  socket.on("msg", function (data) {
      var num = Number(data.msg.substr(0,2));
      var code = data.msg.substr(2,num);
      if(code == "from"+chat_frd+"to"+userID+"at"+chat_title){
          var frdmsg = data.msg.substr(2+num);
          ChatnoteStore.add({nickname: data.msgr, message: frdmsg});
          chatnotePanel.scroller.updateBoundary();
          chatnotePanel.scroller.scrollTo
                ({x: 0, y:chatnotePanel.scroller.size.height}, true);
      }
  });
```

Figure 7. Tramission and Receiving of Chatting Data in Group of Chatting Room

4. Evaluation

With the developed native and hybrid application, we have evaluated and compared the performance, specifically user response time for UI pages and latency of communications between mobile devices. The experimental devices we use are Android smartphone devices, Galaxy S3 and Galaxy Tab 10.1. The Galaxy S3 has Quad-Core 1.4GGz Cortex-A9 embedded processors with 1GB DRAM. With these devices, we had two experiments. The first is estimating UI pages composition. During experiments, we generates from 10 to 1000 UI pages which is related with personal news feeds and updates these to blog, in which one feed consists of image and text. While doing this, the UI composition time, which is the time from login to completion of UI pages with a fixed number of data, are measured. The experimental Results for UI page composition in both of Native and Hybrid Application are plotted in Figure. As shown in the figure, the UI composition times for native application are lower than that of hybrid application, throughout the experiments, almost half level. For hybrid application, although making up UI page is easier than native application, web engine and external framework layer for UI composition has noticeable overhead still in the current high-end mobile device. In our experiments, UI is made up all at once for each UI page, which is not good approach, since the UI composition time increases according to the number of UI elements, as shown in the figure. To reduce UI time, it is better to make UI dynamically not make all the UI statically at once, which can be done with dynamic HTML loading, for hybrid application.

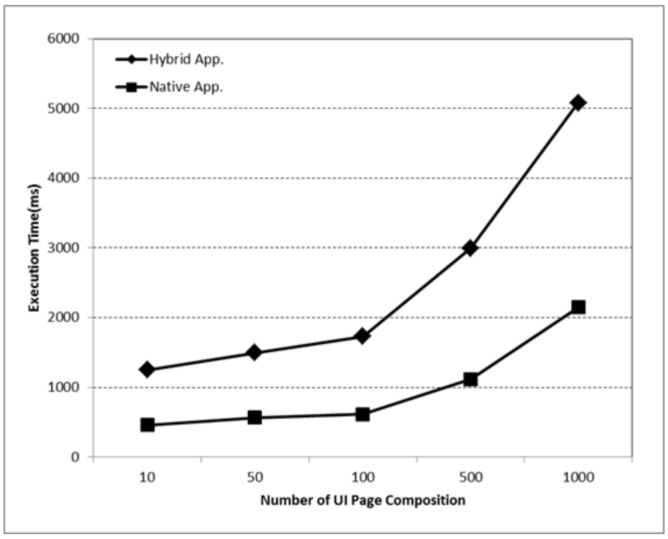


Figure 8. Experimental Results for UI Page Composition in both of Native and Hybrid Application

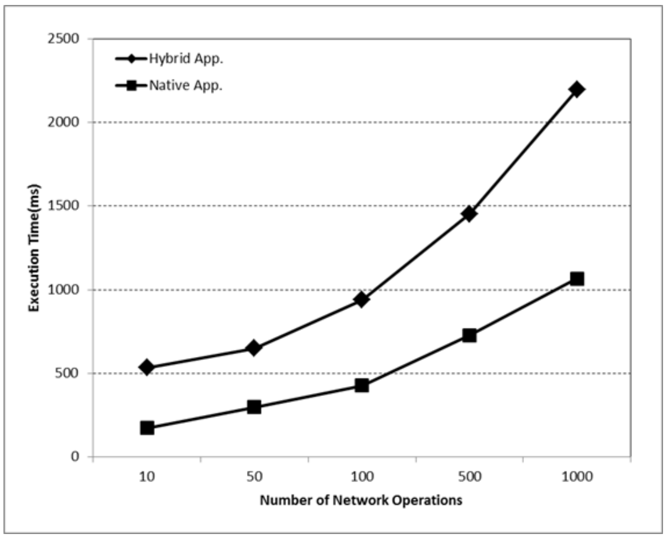


Figure 9. Experimental Results for Network Operations in both of Native and Hybrid Application

In the second experiments, the response time for data communication are measured. We generates transmission data at the unit of news feed contents from 10 to 1000, and the data are transferred from server to mobile device, and during the experiments, response time is measured. The experimental results are plotted in Figure 9 for both native and hybrid applications. Likewise previous experimental results, there is data communication overhead for hybrid application, which is from the web-engine framework layer for hybrid application. All the other device capabilities give similar overhead in hybrid application. Therefore, the overhead of making hybrid application exists in the current state of mobile systems, however, is can be better for the near future, where device capabilities are advanced, as well as hybrid framework is advanced.

5. Conclusion

There are two kinds of developing mobile applications, native and hybrid application. The applications built for a specific platform and language provided by the platform vendor is called Native Application, while, hybrid applications run on the mobile devices with the development of web technologies such as HTML5, CSS and JavaScript, with inside a native container. There are two key factors in developing mobile applications; the one is user interface design, and the other is efficient utilization of device capabilities such as various sensors, cameras, network interfaces.

In this paper, we develop representative mobile application, social network service application, in both of native method and hybrid method to evaluate with the application at the aspect of development cost and performance cost. In developing, we focus on user interface design and network interface operations. The UI composition time and networking performance are experimented with developed applications. From the results, we identify that currently native applications have almost double the UI composition times and networking overhead in certain conditions. The UI for hybrid applications had better to develop with dynamic composition nature, and mobile device capabilities had better to be done aggregately when these are used in hybrid applications. This gap will be reduced with the advanced web technologies as well as advances of device capabilities in the near future.

Acknowledgements

This work was supported by Hankuk University of Foreign Studies Research Fund.

This paper is a revised and expanded version of a paper entitled “Design and Implementation of HTML5 based Hybrid Application for Mobile Social Networking Service” presented at SIP 2014 on December 20-23, 2014 at HNA Grand Hotel Mingguang Haikou, Hainan China.

References

[1] S. Ludwig, "Mobile app usage grows 35%, TV & web not so much", in venturebeat.com, **(2012)**.
[2] S. Perez, "comScore: In U.S. Mobile Market, Samsung, Android Top the Charts; Apps Overtake Web Browsing", in techcrunch.com, **(2012)**.
[3] Hitech, "Mobile Application Development Guidelines", **(2011)**.
[4] AppBuilder, “What is Hybrid Mobile App?”, Telerik Blogs, **(2012)**.
[5] Sencha Touch, http://www.sencha.com/products/touch/
[6] JQuery Mobile, http://jquerymobile.com/
[7] Andre Charland, "Andre Charland's Answers on PhoneGap". Quora, **(2012)**.
[8] Adobe, "Adobe Announces Agreement to Acquire Nitobi, Creator of PhoneGap", Adobe.com, **(2011)**.
[9] J. Fermoso "PhoneGap Seeks to Bridge the Gap between Mobile App Platforms", Press of GigaOM, **(2009)**.
[10] S. Higginbotham, “The Global Rise of the Smartphone”, Press of GigaOM, **(2010)**.
[11] N. Gandhewar and R. Sheikh, “Google Android: an emerging software platform for mobile devices”, international Journal on Computer Science and Engineering, **(2010)**.
[12] Phonegap, http://phonegap.com.
[13] Social Network Service, http://en.wikipedia.org/wiki/Social_networking_service
[14] JSONP org. , "Safer cross-domain Ajax with JSON-P/JSONP", http://json-p.org/
[15] M. Choi and S.-H. Lim, “Development of HTML5 Photo Album Smart Application”, International Journal of Multimedia and Ubiquitous Engineering, vol. 8. no. 6, **(2013)**, pp 349-360.
[16] S.-H. Lim, “Design and Implementation of HTML5 based Hybrid Application for Mobile Social Networking Service”, 8th International Conference on Future Generation Communication and Networking, **(2014)**.