Remote Mobile Test System: A Mobile Phone Cloud for Application Testing

Jun-fei Huang, Yun-zhan Gong
State Key Laboratory of Networking and Switching Technology
Beijing University of Posts and Telecommunications, Beijing, China, 100876
huangjunfei@bupt.edu.cn, gongyz@bupt.edu.cn

Abstract – **It is difficult to enable a mobile application to work as expected on so many mobile phones currently on the market. Most of development teams may not have so many testing phones because of limited budgets, but the applications hasve to be launched on time under market pressure. In this paper, we present Remote Mobile Test System (RMTS), which is aimed at reducing the cost of buying many mobile phones, improving mobile application testing efficiency and helping ensure mobile application quality and reliability. Users could access a real mobile phone remotely and operate them through a web browser, including uploading, startup, and testing mobile applications by clicking and swiping actions. The mobile phones could be geographically distributed. The operation process could be recorded as automated testing scripts and to be run on other mobile phones to improve testing efficiency. The proposed system is soft-link based that causes lower cost and to deploy a new phone quickly.**

Keywords - RMTS; mobile phone cloud; testing cloud service;

I. INTRODUCTION

The market for mobile applications increases every day and is just going to get more demanding as the technology and market of mobile internet grow. It is well known that mobile software must be tested to make sure it can run on various phones correctly, that is called mobile application compatibility testing. However, there are so many mobile models that have been used or produced in the world that most development teams cannot own most of them. The existing variety of mobile technologies, platforms and devices causes additional difficulties in developing and testing mobile applications. Mobile application vendors have no alternative but to test their software based on several most popular mobiles, and then launch applications under market pressure. Users may find and report more and more incompatibility issues when the application cannot run it on their mobiles as announced, or even give it up.

This difficulty can be attributed to several causes. First, most development teams cannot own all kinds of mobiles. Second, there are many firmware and OS (operating system) versions for a mobile, and behaves slightly different. Third, different mobiles cannot provide unified operation interfaces. That means mobile application testing depends on manual testing and the automated testing scripts must be modified when testing on different mobiles. The cost to test each version of an application on tens or hundreds of mobiles may be inconceivable.

To make sure a mobile application could be as correct as expected, developers or testers typically have to validate the software before announcing. There are several approaches to validate the software.

One approach is to use a mobile emulator to emulate a mobile. Almost all manufacturers provide corresponding emulator for popular mobiles. The first drawback of an emulator is that the emulation model cannot capture all aspects of a real mobile, for example, emulators cannot have all kinds of screen size, have GPS or other sensors, or even initiate a real call. In addition, mobile vendors or mobile OS customization developers frequently release special versions of their firmware and OS. The emulation model would not be able to capture all those subtle details and updates. The second drawback is that one can only create a limited number of emulation models, yet, there are a large number of mobile phones. The third drawback is emulators themselves may have bugs that cause suspicious defects reported.

Another approach is to try your best to own more real mobile phones. But the cost to buy and maintain these phones may be unbearable for most of mobile application development teams.

Because of these limitations on emulators, most application developers take a compromised approach. They buy several popular mobile phones. When they make a software change, they first test out the changes on emulators and these mobile phones to make sure that everything works properly, and then they announce the software formally.

In this paper, we present Remote Mobile Test System (RMTS), which is designed to solve the problems associated with maintaining real mobile phones and inefficient compatibility testing. It consists of a set of mobile phones that are geographically distributed. In addition, it presents a web user interface and web services APIs which allow end users to access real mobiles remotely. RMTS is essentially a mobile testing cloud, and end users can request devices on-demand to test their software with automated testing features. RMTS is a cloud testing service as well.

II. THE DESIGN AND IMPLEMENTATION OF RMTS

RMTS architecture is shown in Figure. 1. It consists of a collection of mobiles that are scattered across the world. Even though some may be co-located in the same physical lab space, there is no physical constraint on where the mobiles are.

There are many interfaces could be used between mobile servers and mobiles, yet, the most stable and high-speed interface for RMTS is the USB interface. The mobile server is responsible for receiving control commands from end users and transferring to destined mobiles, and capturing screens in mobiles and transferring it to corresponding to end users. The server is the key role and also responsible for

978-1-4673-4510-1/12/$31.00 ©2012 IEEE

communicating with the web server. The communicated information includes reporting on what mobiles are available, keeping the transaction between end users and selected mobiles. When a mobile is locked by an end user, the mobile server keeps the mobile is excluded by others. When a mobile locked by a user expires, the mobile server will release the connection and the mobile is available again.

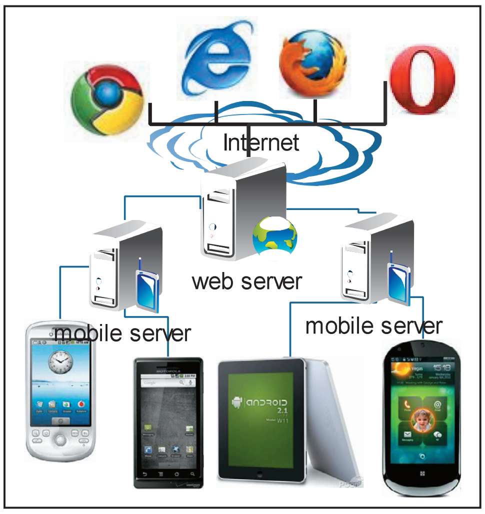

Figure 1. RMTS Architecture

There is a web server to maintain a testing session between users' browser and mobile server.

There are several key features of RMTS which set it apart.

1) Real Mobile phones: RMTS uses real mobile phones so that the users could perform realistic software testing. The current supported mobiles must use Android [1]as OS, and others, such as Window Phone and iOS would be supported in the future. The operations include installing, running and removing software packages, clicking keys, tapping/swiping/capturing screens, long press screens and keys, inputing texts, automated testing, and so on. Using real mobile phones, instead of emulators, allow users to accurately reproducing the behavior of an application that runs on an exact mobile.

2) Distributed Mobile phone: RMTS is aimed at testing software in real mobile phones remotely. Because there are many types of mobiles, the cost of purchasing an exhaustive list of mobiles is prohibitive. In addition, mobiles evolve quickly, thus it is also costly to keep the lab up to date. In order to make the system be cost effective and still be useful, we have designed a distributed architecture. Also the mobiles are located in several central data centers, the users can access them at their office, or even home.

3) Soft-link Based: RMTS connects mobile phones with standard USB interface. A hard-link solution as [6] needs to develop a hardware board to connect mobiles' screen and keys seperately, and that is not universal and higher costly.

4) Testing Script Language: The testing operations can be recorded by RMTS in the testing script language that is based on XML. The XML schema will be presented which will enable end users to edit the testing script and fully automate software testing on various mobile types. End users cloud automatically test software changes nightly and read the testing result in the morning to determine whether the change cloud be run on all these target mobiles correctly.

The following sections describe each component of the architecture in more detail.

A. Web User Interface and Web Server

When a mobile is locked, a web user interface will be shown in the web browser. The top part of left hand column is the function area to support the operations such as uploading file, unlocking mobile, capturing screen, inputting text, automation testing, and so on. The uploaded files will be stored in the user's space and can be pushed to the locked mobile. If it is .apk package, a button will appear to install it in the locked mobile. Other functions would be stated later.

The bottom part of left hand column is the operation history area. Any operation on the locked mobile will be recorded as an item and a script will be generated when it is released. The script could be running on other mobile phones and that is called automated regression testing. Script files are stored in the web server, and could be exported to users' local disks if desired.

The right hand pane shows the locked mobile phone. The screen panel is synchronized with the real mobile almost in real time and the touch and sliding screen operations can be done using mouse like operating a real device using fingers. Each key of the panel can be pressed in browsers. For each mobile model, a set of skin files should be created before those mobile phones are in service. The mobile server will choose the correct skin file set to show in the web browser.

Before testing, users first have to lock a mobile since there is only one instance of each mobile shown in the inventory and it is a shared facility. We are developing the reservation feature and a reserve button would bring up a calendar similar to that in Microsoft Outlook, which lists all mobiles and its current schedule information. The users could select the next free period for available mobiles and make a reservation.

The action to swipe screen should be simulated because end users can use only mouse and keys to operate web browsers. If the swiping track is almost straight line, the action can be modeled as sliding from x to y point with speed v. If the swiping track is curve, it should be split to several straight lines. If a mobile phone has the physical track ball, RMTS uses "up", "down", "left", "right" and "center" keys to replace it in the web GUI.

To show the mobile screen in web browsers, the picture updating technology with AJAX is adopted by RMTS. We use ImageSwitch which is easy to use jQuery plug-in for image switching. Test results show that to update the mobile screen for each one second in public Internet environment can satisfy almost all testing demands. For video or gaming

978-1-4673-4510-1/12/$31.00 ©2012 IEEE

applications, the bottleneck is located at the connection between RMTS and mobile devices.

B. *Mobile Interface*

There is a piece of software between the mobile server and the mobile drivers. We refer to it as the Mobile Interface Sub-system (MIS). It has two jobs: sending control commands to the mobile ports and capturing responses from the mobile ports.

The current RMTS supports Android mobile platform. Android provide an adb (Android Debug Bridge) tool which is a versatile command line tool that lets you communicate with an emulator instance or connected Android-powered device. It is a client-server program that includes three components: client, server and daemon. The daemon runs as a background process on each emulator or device instance. As a adb client, RMTS mobile server communicates to adb server that is part of Android SDK. The adb server connects to the adb daemon running on phones.

When starting up, RMTS sends a ‘wake’ command to all emulators and devices after setting up connections to all running emulator/device instances. It locates emulator/device instances by scanning odd-numbered ports in the range 5555 to 5585, the range used by emulators/devices. Where the RMTS finds an adb daemon, it sets up the connection to that port.

The supported commands by RMTS are shown in Table I.

TABLE I. SUPPORTED COMMANDS

Command	Description
touch	Touch screen at the specified location
long press	Long press key or screen at the specified location
swipe	Swipe on screen from one to another location
press	Press a physical button on the device
type	Type the characters to the device
wake	Wake the device
install_package	Install a package to the device
remove_package	Remove a package from the device
run_package	Run a package in the device
sync	Push a file to the device
capture_screen	Capture the screen of the device

How to treat the diversity of mobile devices? The document of Android and the compatibility testing of RMTS proves this soft-link USB interface works on all versions of Android, from 1.6 to 4.0.

C. *Mobile Server*

RMTS mobile servers are responsible for keeping tracking of all available mobiles, some of which may be plugged in and pulled out, or out of service at any time. When a mobile comes or goes, the users’ web browsers could get the dynamic update information.

When a mobile is locked by a user, it cannot be used by others till it is released or expired. Mobile server maintains the mapping relationship between users and mobiles.

Besides controlling mobile phones, RMTS mobile servers provide the information of attached devices to help user make a choice, such as the model type, manufacturer, and screen resolution. The captured screens of mobiles are saved in a temporary folder of RMTS mobiles. The size of a captured picture of the typical 320x480 resolution screen is about 20 ~ 40KB. In the case of refreshing pictures per second, each user needs about 160-320 kbit/s network bandwidth. If there is no difference between a captured screen picture and the previous, the RMTS server will respond the previous picture’s URL, and the web browser could show the picture from the cache directly to reduce network bandwidth.

The testing operations on a mobile phone could be recorded in XML. The core content of XML schema is shown as Figure 2.

```
<xsd:element name="process" type="tOperationList">
   <xsd:annotation>
      <xsd:documentation>
         This is the root element.
      </xsd:documentation>
   </xsd:annotation>
</xsd:element>
<xsd:complexType name="tOperationList">
   <xsd:sequence>
     <xsd:sequence minOccurs="0" maxOccurs= "unbounded">
      <xsd:element name="operation" type="tOperation"/>
    </xsd:sequence>
    <xsd:attribute  name="model" type="xsd:string">
   <xsd:attribute  name="screen-height" type="xsd:integer"/>
   <xsd:attribute  name="screen-width" type="xsd:integer"/>
   </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="tOperation">
   <xsd:sequence>
    <xsd:element name="time" type="xsd:integer"/>
    <xsd:element name="command" type="xsd:string"/>
   </xsd:sequence>
</xsd:complexType>
```

Figure 2. XML Schema of Testing Operations

End users could export the operation list recorded by mobile servers to local disks and edit the XML file. The file also could be accepted by RMTS and repeated nightly against other model mobile phones which have the same screen resolution.

III. USE CASES

A. *Mobile Applicationg Testing*

It is assumed that the tested application is tapp.apk.

1) The user opens a browser and locks a testing mobile (TM) with 320x480 pixels screen resolution.

2) The user uploads tapp.apk to RMTS and installs it in TM. The upload and installation successful messages are shown in the web pages.

3) The user finds the icon of tapp.apk and click it.

4) The user clicks or swipes the keys or screen to test the applications.

978-1-4673-4510-1/12/$31.00 ©2012 IEEE

B. Automated Testing

During the testing of tapp.apk above, the operations are recorded by RMTS.

1) The records are exported into a toper.xml script file.

2) The file is edited and uploaded to RMTS.

3) The user chooses several other mobiles and sets the updated toper.xml as the testing script.

4) RMTS schedules the testing to run toper.xml on the chosen mobiles and generates the testing report, including log information and captured screens.

5) The user reviews the report to make sure if tapp.apk could run on those mobiles well.

C. Remote Mobile Show

Apple Experience Centre (AEC)[2] could exhibit Apple hardware and other third-party products, and have seminar rooms for training sessions and other events. RMTS also allows phone manufacturers to show products remotely. Millions of dollars for hypostatic store may be saved. Users could experience the features of new mobile products just using web browsers at any time.

IV. ONGOING WORK

RMTS has not been deployed for production use and only a demo site is ready.

To use a real mobile means that a physical mobile could be used by only one end user at a time. Although we expect to increase the number of mobile resources constantly, it is still desirable to share the mobiles as efficient as possible. Mobile virtualization is a technology that enables multiple operating systems or virtual machines to run simultaneously on a mobile phone or connected wireless device[3][4]. We plan to investigate the maturity, reliability and performance of this technology to know if the mobile virtualization could meet demand.

The network delay and limited bandwidth restrict end users have a smooth feeling during operating virtual mobiles in web browsers. The size of a complete gif formatted screen picture is fixed, but normal operations on mobiles often lead to change in a small sector within one second. To transfer only the changed sector may decrease the network requirements and then we can increase the refreshing frequency to gain much better user experience.

The mobile server only support to communicate with Android mobile phones currently. It is a fairly common point of view that Android, iOS and Windows Phone may be the most popular smart phone platforms in these years. RMTS should support iOS and Windows Phone platforms besides Android. But they may not have the mobile interface as adb that in Android. We are considering a new solution based on Virtual Network Computing (VNC) that uses the RFB protocol to remotely control another compute.

V. RELATED WORK

To put hardware devices to networks as a cloud service could be feasible. Huan Liu et al. presented Remote Network Labs, which is aimed at leveraging the expensive network equipment more efficiently and reducing the cost of building a test lab. Users could request network equipment remotely and connect them through a GUI or web browsers [5].

DeviceAnywhere provides the enterprise-class cloud-based platform based on hard-link technology for testing and monitoring the functionality, usability, performance and availability of mobile apps and websites [6]. HP has the test automation for mobile phones products that could support Windows Mobile OS and not be a cloud service to use real mobile phones [7].

China Mobile deployed a hard-link based mobile remote testing service (RTS)[8] that requires a 2M bps network bandwidth for each concurrency user and flash plug-in in users' web browsers.

VI. CONCLUSION

We present Remote Mobile Test System (RMTS), an online mobile cloud facility from where end users could request a mobile phone to test their mobile applications. It is soft-link based and no special hardware is needed and a new mobile device could be deployed within one day. As a lower cost solution, it is designed to efficiently utilize mobile phones. Mobile developers don't need to buy too many devices any more. Beyond cost saving, it also has many features that were not possible before. It can reduce mobile phones maintenances, automate the regression test from one mobile model to others, and help training. RMTS is based on a flexible architecture that supports to plug in and pull out mobile phones on demand and add or reduce mobile servers at any time.

RMTS has its limitations. Currently, only Android mobile phones can be connected to RMTS mobile servers thought the development to support iOS and Windows Phone is in progress. Another limitation is that each mobile phone can only be used by one end user at a time. This may be addressed through mobile virtualization. We are also considering enhancing the automatic testing capabilities.

REFERENCES

[1] Google Projects for Android. http://code.google.com /intl/en/android/

[2] Apple Campus Experience Centre. http://www.apple. com/asia/education/how-to-buy/sg/aacs.html

[3] Mobile virtualization. http://en.wikipedia.org/wiki/Mobile_virtualization

[4] VMware Horizon Mobile and VMware Mobile Virtualization Platform. http://www.vmware.com /products/mobile/ overview.html

[5] Huan Liu, Dan Orban. Remote network labs: an on-demand network cloud for configuration testing. Proceedings of the 1st ACM workshop on Research on enterprise networking (2009), pp. 93-102.

[6] DeviceAnywhere Enterprise-Class Mobile App Testing & Monitoring. http://www.deviceanywhere.com /mobile-application-testing-overview.html

[7] Test Automation for Mobile phones - HP QuickTest Professional software and Jamo Solutions. http://www.powertest.com/files/datasheet-hp-quicktest-professional-and-jamo-solutions.pdf

[8] Remote test service (RTS). http://120.132.135.142 /rts/rts/rts-home.do?nnn=-591459640