Cloud Systems and Their Applications for Mobile Devices

Jin-Hwan Jeong
Cloud Computing Division
Electronics and Telecommunication Research Institute
Daejeon, South Korea
jhjeong@etri.re.kr

Hak-Young Kim
Cloud Computing Division
Electronics and Telecommunication Research Institute
Daejeon, South Korea
h0kim@etri.re.kr

***Abstract*— In this paper, we discuss what mobile cloud is and what the differences between traditional cloud systems and mobile cloud systems are. At first, we point out mobile peculiarities such as battery constrains and computation weakness, and then we envision cloud system architecture for mobile devices that addresses these two issues. Specifically, we introduce mobile sensor virtualization and remote execution as essential components of mobile cloud system. At last, we show canonical services benefiting from our mobile cloud system.**

Keywords-Cloud computing; Mobile; Android; Sensor; Remote execution

I. INTRODUCTION

Today, computer science communities face very big two tides called cloud computing and mobile computing.

Since a few years, major software companies such as Amazon, Google, Microsoft have built large-scaled cloud systems and most SP (Service Provider) have transferred their services to the cloud systems. As cloud systems provide various software platforms from IaaS (Infrastructure as a Service) to SaaS (Service as a Service) and guarantees endless computing resources with scale up/out flexibility, SP can implement scalable services easily at the low cost. It is no doubt that SNS (Social Network Service) is one of live evidences.

Another big tide is mobile computing lead by smartphones. Smartphones, as a representative of mobile devices are very popular, and their hardware configuration is outstanding, for example, 1.2 GHz CPU, 1024 MB ram, 802.11g/n WIFI, 3G, and various sensor devices. These hardware components make recent smartphones possible to do everything that desktop can do. Especially, high-end smartphones (e.g., Apple, Android, and Window Phone 7) that have their unique application ECO system can do beyond desktop's capabilities on the behalf of various sensor devices.

In a sense, it might be natural to collaborating mobile devices with cloud system. As the performance of smartphone is superior, users want to use their smartphone as a client of new SNS as well as legacy services. Also, cloud system administrators begin to enhance the cloud systems with additional servers that help mobile devices. Typical examples of this efforts are PUSH (e.g., C2DM (Cloud To Device Messaging) from Google) servers and SYNC servers. Specifically, C2DM is a wonderful tool for SNS developers and it is good for network operators, because it prevents from wasting bandwidth for Keep Alive messages.

Nevertheless, both service developers and users found some limitations on services running in cloud system with smartphones. At first, users always keep attention for one resource, battery power, which is a most crucial resource. High performance smartphone can process complicated services, but it drains battery power much faster. Therefore, users cannot help recharging the battery, and then, this degrades the portability. Secondly, service developers have different difficulties. Most services including SNS require various sensor data in real-time such as geographical location or motion speed. This means that software part running on cloud system needs to query user's sensor data in real time. However, there is no uniform platform/library to aggregate and to deliver sensor data from various sensor devices of individual mobile devices for the server side system, so this is a big hurdle for SNS developers.

To address the first issue, Chun [7] proposed an augmented execution model. This approach has strength in alleviating computation loads of smartphones, but it has some constraints. In augmented execution, when application runs on server side, it is hard to synchronize intermediate data with mobile side. As mentioned in the previous paragraph, mobile services need various sensor data ceaselessly. However, augmented execution model does not provide a well-defined platform how server module can read mobile sensor data.

As a result, a cloud system for mobile devices should provide a tool that help work together easily and provide a way how to send mobile sensor data for rich mobile services. Therefore, we adopt augmented execution model as a cloud execution model and we propose sensor virtualization layer that gives transparent sensor data delivery. The latter can enhance the former's usability.

In this paper, we propose a virtual machine based-remote execution module with sensor device virtualization. With these tools, server module can co-work client modules with easy and utilizes client's sensor devices seamlessly. We start from building a virtual machine for Android, and implement remote execution mechanism for Java and also implement virtual sensor devices into Linux kernel as a form of kernel

 ISBN: 978-1-61208-164-9

modules. As virtual sensor devices are implemented in Linux, the devices can be used transparently by Android system.

II. BACKGROUND AND RELATED WORK

As illustrated in Fig. 1, the current mobile cloud system is much similar with typical cloud system except PUSH and SYNC servers. PUSH and SYNC servers are of "sugar" servers, because the servers enhance just mobile user experience much better and help other core servers. In a sense, two servers can be just regarded as a mobile gateway for better data exchange with cloud servers.

Now, the question is what the "mobile" does mean. Some cloud systems can be regarded as mobile cloud systems because their clients are mobile devices or because their deploying services are targeting for mobile devices or because they contains PUSH and SYNC servers as mentioned above.

Before defining the meanings of mobile, we point out the differences between mobile devices and desktop systems in the view point of clients (service target). About ten years ago, mobile devices were quite different from desktop. They had low performance processor, low memory, low quality displayer, so they was not able to run normal operating systems which meant that mobile applications were somewhat dedicated for mobile platforms. Therefore, the key philosophy for mobile applications was that the less mobile devices do, the better services are. Ten years change everything, and then, today, there is no quite difference between desktop and mobile devices such as iPhone or Android, and mobile devices can process most jobs that desktop can do in terms of performance.

However, there is one thing left, portability. The faster mobile processor is, the faster battery drains. In spite of brilliant material technology, the development speed of rechargeable battery technology is very slow, so everyone who has an iPhone or Android phone now is commonly worried about the shortage of battery power.

Also, a new issue comes out. Unlike the past, stock mobile devices have many sensor devices, and mobile services read values in real time. Most mobile platforms (e.g., iOS, Android, or Windows Phone) provide well defined API to handle sensor devices. However, the method for transferring sensor data from mobile devices to the cloud systems still is nothing changed. Developers just use TCP/IP communication with their own socket libraries.

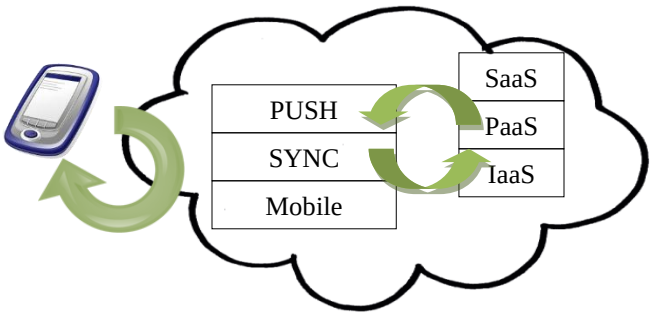

Figure 1. Typical mobile cloud system

There are some researches [6][7] for mobile issues. In [6], authors addressed battery lifetime with cloud computing and proposed off-loading computation for energy-saving. In [7], authors introduced clone computing to enhance smartphone capabilities. They [6][7] tried to catch two goals, computation limitation and energy limitation by Dalvik VM cloning (Augmented Execution) for Android. This work is remarkable, because augmented execution can leverage the degree of distribution between mobile device and cloud system on the same programming environment. However, they do not refer methods how to synchronize runtime execution environments (including run time data) with cloned Dalvik VM and real VM. Since sensor data have a tendency to change ceaselessly in mobile services, streaming sensor data is crucial. Nevertheless, augmented execution is worthy to evolve and we adopt it as a starting point of our remote execution.

III. MOBILE CLOUD SYSTEM ARCHITECTURE

Our system targets at two tools, remote execution and sensor device virtualization. Remote execution takes a responsibility for helping mobile device execution, which means that some sections of code are run on cloned VM instead of on mobile device, and sensor device virtualization that is used as sensor data sources for remote execution.

To simply remote execution model, we also restrict remote execution scope as following: 1) it has the same programming environment as mobile devices; 2) it does not share data between local modules and remote modules – stateless model; 3) only mobile side can initiates application. 4) only sensor data read from virtualized sensor devices are allowed for server. Like augmented execution in [7], our model is based on virtual machine.

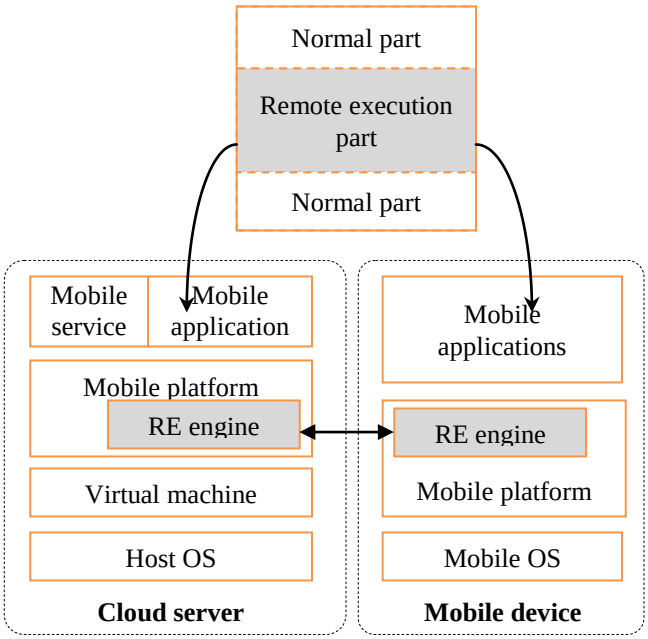

Figure 2. Mobile Cloud System

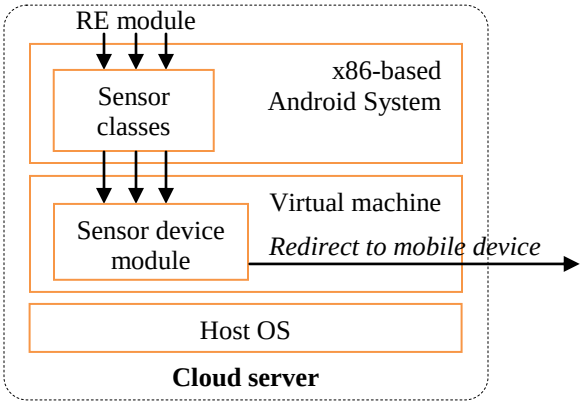

Figure 3. Sensor device virtualization

Fig. 2 illustrates overall architecture of our mobile cloud system. We adopted x86-based Android system for mobile virtual machine, and RE (Remote Execution) is implemented as Java library, which is similar with JAVA Remote Thread library. Like other remote thread libraries, RE is also based on a stateless model. Instead, our RE thread function takes one "Serializable" Java class as a function parameter.

In the cloud system side, RE modules are often in need of sensor data for mobile services. So far, developers should make a connection to read them, and they should endure various overheads such as checking device capabilities or network status. To alleviate such overheads, we introduce the sensor device virtualization that lets developers use client (remote) sensor devices locally. Fig. 3 shows the sensor device virtualization flow. As Android system is based on Linux, we hooked sensor devices I/O at kernel modules. This implies that applications (RE modules) can invoke sensor device I/O calls transparently. Specifically, when a request arrives from RE modules, virtual devices redirect a request to a peer mobile device. Although not presented in Fig. 3, mobile device also should have corresponding modules to process a redirected request. The corresponding modules receive a request and finally reply with live sensor value.

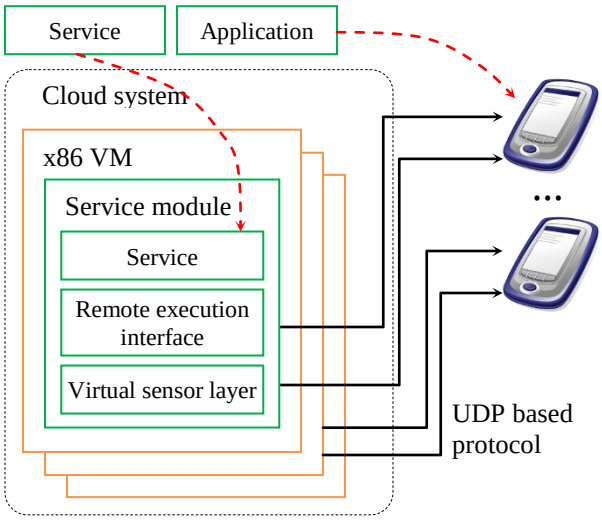

Figure 4. Overall cloud system for mobile devices

Our mobile cloud system with remote execution and sensor device virtualization is shown in Fig. 4. As one of the goals of our cloud system is to provide the same application environment as mobile devices to developers, our cloud supports mobile virtual machines. In case of Android phones, our cloud system allocates an Android x86 virtual machine per Android client for server part modules.

Specifically, for server module, service is linked with remote execution callee interface and sensor virtualization (requester) interface, and then it is initiated while VM creation implicitly (or explicitly by user's request). These interfaces eventually are connected on-the-fly with incoming mobile device using simple protocol. Similarly, mobile application has two modules, remote execution caller and sensor replier. Remote execution caller has various methods for instantiation / destruction / synchronization of remote thread object. Sensor replier simply replies sensor data requests from peer server.

Although our service execution model is based on the stateless model, it can process rich mobile services thanks to sensor device virtualization which provides live sensor data. In order to reduce response time and lightness, our model uses UDP based protocol with packet sequencing and limited ARQ features.

IV. APPLICATION

In this section, we show a typical example of services benefiting from our remote execution and sensor device virtualization. The example is about security. The scenarios are supposed that one security developer wants to verify both physical and logical security simultaneously. He/she needs sensor data for physical authentication and wants to run decoder for logical authentication on server. The main goal of this example is not to show a fact that security is enhanced due to our system. Instead, the goal is to show a fact that the developers can use mobile sensor data easily for physical authentication by sensor device virtualization just like they implement and run it on a local device.

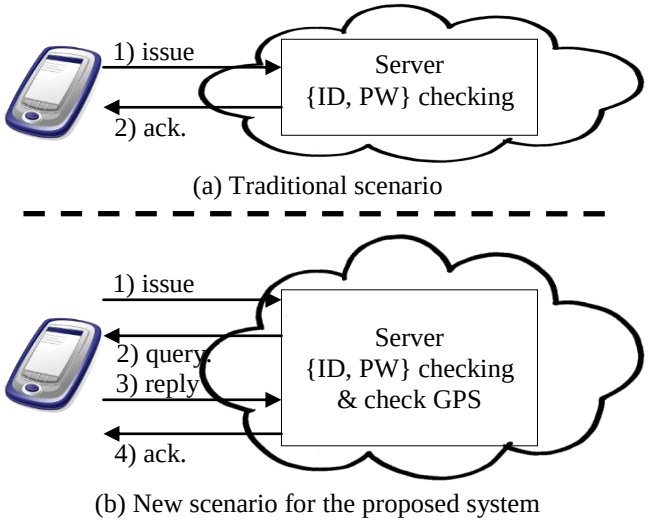

Figure 5. Two authentication scenarios

 ISBN: 978-1-61208-164-9

Mobile authentication with GPS: User authentication is one of eternal problems. Basically, if the tuple, {ID, Password} is correct, system regards an incoming user as being verified. However, suppose that a hacker got a password, and try to login. How can system detect this?

In the lower scenario (b) in Fig. 5, the developer of servers implemented verification module with additional features. Whenever the ID/Password arrives, server-side verification module queries GPS data without intervention of mobile user via local (virtualized) sensor devices and constructs location DB {user/location area}. One day, the user requests login, and verification module notices that his/her location is far from the usual location area. With only this fact, it is enough to suspect the user, and system can proceed to more robust verification step.

This scenario also can be implemented with traditional manner (a), but it is not simple. Developers should consider how to read GPS data and how to transfer to server. Also, developers should take care of upgrading of client part. However, if developers use our remote execution, they don't need TCP/IP to send ID/Password data, and if they use virtualized sensor devices, they don't need to know user's device model and how to access.

V. CONCLUSION

In this paper, we addressed two peculiarities called portability and sensor devices that make mobile cloud systems different from typical cloud systems, and we emphasize on the facts that mobile cloud system should take a care of two peculiarities. Especially, manipulating of various sensor data is considered as one of decision factors for whether the services are categorized as mobile services or not. For portability, we think the battery consumption. As a result, we propose remote execution with stateless model.

We are currently building Linux kernel supporting virtualized sensor devices and also implementing JAVA interfaces for remote execution. While we are still working on this project and are building prototypes, we take a look at the possibility that it can be indeed a good candidate of mobile cloud system.

REFERENCES

[1] http://www.android.com, April, 2011.

[2] http://developer.android.com/sdk/index.html, April, 2011.

[3] Bornstein, D. Dalvik virtual machine. http://www.dalvikvm.com, April, 2011.

[4] Kozuch, M. A., Ryan, M. P., Gass, R., Schlosser, S. W., O'Hallaron, D., Cipar, J., Krevat, E., López, J., Stroucken, M., and Ganger, G. R. "Tashi: location-aware cluster management," In Proceedings of the 1st Workshop on Automated Control For Datacenters and Clouds (Barcelona, Spain, June 19 - 19, 2009).

[5] Avetisyan, A.I., Campbell, R., Gupta, I., Heath, M.T., Ko, S.Y., Ganger, G.R., Kozuch, M.A., O'Hallaron, D., Kunze, M., Kwan, T.T., Lai, K., Lyons, M., Milojicic, D.S., Hing Yan Lee, Yeng Chai Soh, Ng Kwang Ming, Luke, J-Y., and Han Namgoong, "Open Cirrus: A Global Cloud Computing Testbed," in IEEE Computer, Vol. 43, Issue 2, pp. 35 ~ 43, April, 2010.

[6] Kumar, K., Y.-H. Lu, "Cloud Computing for Mobile Users: Can Offloading Computation Save Energy?," in Computer, Vol. 43, Issue 4, pp. 51~56, April 2010

[7] B.-G. Chun and P. Maniatis, "Augmented smartphone applications through clone cloud execution," in Proceedings of the 12th conference on Hot topics in operating systems (HotOS'09), pp. 8~12, 2009

 ISBN: 978-1-61208-164-9