MCSC: Mobile Collaborative Service Cloud using Instant Adaptive Orchestratio

Chol-Hong Im and Chang-Sung Jeong*

Distributed Supercomputing Laboratory, Korea University
imich@korea.ac.kr, csjeong@korea.ac.kr

Abstract

Although there are various kinds of mobile application technologies, users spend time with their smartphone mostly surfing the Internet, playing games, and communicating with their friends because the computing power of smartphone still lags behind other devices in terms of resources such as CPU, memory, etc. Furthermore, in many cases, users can’t easily find suitable mobile applications without implementing them by themselves, because it’s very difficult and takes much effort to implement a mobile application. So in this paper, we propose MCSC, a mobile collaborative service cloud for Android and iOS which requires less computing power while providing rich functionalities. While implementing an application using native language takes significant time and requires a specialized skill set, using our proposed service cloud, users can implement new mobile applications easily and quickly using instant adaptive orchestration and Mashup. MCSC facilitates the use of other devices’ resource and context information such as contact lists, location, etc. We evaluated MCSC in several scenarios, and confirmed that it performs as fast as native mobile applications and works well in heterogeneous environments incorporating both Android and iOS devices. In the future, MCSC will create new opportunities for mobile cloud computing and service oriented architecture.

Keywords: mobile, collaboration, cloud, android, iPhone, Mashup, orchestration, SOA

1. Introduction

Most people expect their smartphone to provide the same level of service as a high performance PC. But, mobile device has limitations such as computing power, memory, etc. Furthermore, users can’t find suitable mobile applications easily. MCSC can solve these problems. It can be used to implement new applications easily using instant orchestration and Mashup. The process of arranging for multiple web services to act in a predefined sequence is therefore known as “orchestration.” [4]. Mashup is an application that combines data from multiple public sources in the browser and organizes it through a simple browser user interface [1]. We can implement a new Mashup component by orchestrating some components such as device, REST and SOAP components. MCSC can retrieve various kinds of context information, for example, contact list, location, and others from other devices and use other devices’ resources using remote invocation. Using MSCR, mobile devices can interact with each other in a peer-to-peer way. Under this paradigm, mobile devices cease being merely clients and become both severs and providers. MCSC can also collaborate with other devices such as PCs and workstations. MCSC can realize a mobile cloud computing environment using collaboration, instant orchestration, and Mashup. In this paper, we propose implementing and combining instant orchestration and Mashup to create a mobile cloud environment first time. As a result of our research, we believe that MCSC can have far-
reaching effects by allowing mobile devices, PCs and workstations to work like one machine and thus creating a truly mobile cloud computing environment.

2. Related Work

To overcome a limit of mobile device’s computing power, researchers have developed offloading technology which can efficiently utilize not only mobile device resource but also server resource. In initial offloading model, all work should be done in a server and mobile devices just receive results from the server [6]. Recently, researchers developed efficient partitioning methods which can divide and operate tasks in both server and mobile device concurrently [7]. After finishing the tasks, mobile device receives result from server and synchronizes them. Additionally, Android platform can make its own clone on server, and mobile device and clone partition work and execute tasks concurrently [8]. Also, there is mobile cloud research using DSL (Domain Definition Language) which can describe interface information like IDL (Interface Definition Language) [9]. DSL can generate server skeleton and client stub automatically. The codes can integrate mobile device and cloud server using REST method.

3. Mobile Collaboration Service Cloud

3.1. Mobile ESB

MCSC can communicate with others as both provider and consumer. It can make new Mashup services composing its own device services, external open API like the programmable web, and even other MCSC’s Mashup services. Orchestration and Mashup can make a composite service which works as single service. Additionally, it can change or replace certain functions easily, because the service interacts as efficient interaction type, loosely coupling. MCSC is similar to ESB (Enterprise Service Bus) which orchestrates various kinds of services [5]. However, there are some commonalities and differentiations between ESB and MSCS. MCSC compared with ESB is explained in Figure 1.

Figure 1. MCSC Concept Comparing with ESB
MCSC is designed to be mobile ESB which can make Mashup services and provide them not only to PCs, but also to other devices. ESB is an important middleware in enterprise SOA environment. So we mainly focused on the mobile device working as a server providing its own service directly, not being a consumer or client anymore.

### 3.2. Instant Orchestration and Mashup

We cannot make Mashup services easily with ESB, because we have only few services covering our requirements perfectly. Though there are a lot of open APIs, we can’t orchestrate them by ourselves. If we can orchestrate existing services in mobile device directly, we can make new Mashup services easily. We call it “Instant Orchestration and Mashup” in a mobile device. In case of Instant Orchestration and Mashup, we should consider target service and input-output values only. We call it IMDL (Instant Mashup Description Language). It enables orchestrating various kinds of components and making as a Mashup component. Figure 2 shows the structure of IMDL and its example. This example describes two components: “GETLOCATION” and “VIEWMAP”. “GETLOCATION” retrieves latitude and longitude of device while “VIEWMAP” displays map using obtained latitude and longitude. Moreover, we can use two components separately in any situations. Developing more adopting automatic dynamic reconfiguration capabilities are remained as future works [2].

![Figure 2. IMDL (Instant Mashup Description Language) Structure](image)

### 3.3. Context Information Server

Mobile device has important personal context information such as contact list, phone state, calendar, location, etc. [3]. Furthermore, mobile device has various kinds of sensors like accelerometer, light, proximity, gyroscope, and others. It can give us useful context information of environment. However if we need other’s context information, there are no significant methods to invoke other’s context information. So in this case, MCSC can get this context information from other mobile devices, PCs, and workstations. Consumer can request available services and get their service list and can invoke a service among them and get context information like other device’s location, contact, calendar, etc. Figure 3 explains how consumer can get context information from provider.
4. MCSC Capabilities

4.1. Service Component

MCSC has eleven device service components containing services such as location service, contact service, phone state service, and so on. These services are implemented by native android SDK which can be invoked while sending request XML message. Additionally, remote device or PC can invoke them if we publish service as REST component. Hence, users can know other’s location, phone number on contact list, and phone state like calling or waiting via invoking other’s device component. Mobile devices can be connected to service cloud using SOA technology such as SOAP and REST [10]. There are REST and SOAP components in MCSC. REST components send HTTP request to other device having registered URL. This component can make the HTTP message using input parameter and can send message using GET or POST method. If the component receives a response from target, it can interpret the response and display to user. SOAP component can communicate with SOAP provider. It can interpret SOAP interface following the WSDL standard, and generate client side stub code. With the stub code generated in real-time, we can execute remote method quick and easy.

4.2. Mashup Component

Mashup component can be composed as joining device components, REST components, and SOAP components. Due to instant Mashup and orchestration, Mashup component can work like a single component. Interesting thing is that we can invoke other device’s Mashup component, and make another Mashup component. So, a Mashup component can be composed as a part of other Mashup component. MCSC can work as both server and client. We can access other device’s Mashup components, and invoke them. Finally, we can register their information, and make them as a part of our Mashup component. There are three kinds of Mashup Component in Figure 3. One is Local type, another is remote type, and the other is mixed type. Local Mashup component is invoked, and receives results by its own device. Remote Mashup component is invoked by remote device, and returns results to the device. Mixed Mashup component is complicated one, since one device’s Mashup component invokes other device’s Mashup component.

![Figure 3. The Types of Components](image-url)
5. Implementation and Evaluation

Android MCSC has full functions including Provider Service and Client Service. Moreover, Android provides many device services like location, device control, file control, and others via Android application framework in Android SDK Platform. Android MCSC is composed by activities, services, and configurations such as AndroidManifest.xml. It can configure Mashup component using instant Mashup, and can invoke remote Mashup component. Figure 4 shows execution of Android MCSC on Samsung galaxy S and iOS MCSC on iPhone.

![Figure 4. Snapshot of Android and iOS Implementation](image)

We evaluated MCSC with 3 scenarios. Table 1 explains the scenarios. Components with square brackets like” [GETLOCATION]” means other device’s component. We invoked other device’s component remotely using REST and got the results from the device.

<table>
<thead>
<tr>
<th>Type</th>
<th>No</th>
<th>Evaluation Scenario</th>
</tr>
</thead>
<tbody>
<tr>
<td>Local</td>
<td>1</td>
<td>GETLOCATION - VIEWMAP</td>
</tr>
<tr>
<td>Mixed</td>
<td>3</td>
<td>[GETLOCATION] - VIEWMAP</td>
</tr>
</tbody>
</table>

We evaluated 3 cases with the above 3 scenarios. Table 2 shows test cases for evaluation. We compared local component with remote component in test case 1. We compared Mashup Component with pure native application in test case 2, and evaluated MCSC in heterogeneous environment both Android and iOS in test case 3.

<table>
<thead>
<tr>
<th>No</th>
<th>Test Cases</th>
<th>Evaluation conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Remote vs. Local</td>
<td>Scenario 1 in Galaxy S2, Scenario 3 in Galaxy S2</td>
</tr>
<tr>
<td>2</td>
<td>Native vs. Mashup</td>
<td>Scenario 1 in Galaxy S2, Native App with same function with scenario 1 in Galaxy S2</td>
</tr>
<tr>
<td>3</td>
<td>Android vs. iOS</td>
<td>Scenario 2 in Galaxy S2, Client: Galaxy S2, Server: Galaxy S2, Scenario 2 in iPhone, Client: iPhone, Server: Galaxy S2</td>
</tr>
</tbody>
</table>
Table 3 shows evaluation result in each test case. We evaluated 5 times in each case and dropped out the highest and lowest values. We calculate final average value. In test case 2, there were no performance effects in point of computing power. However, if there are a lot of concurrent requests, performance of Galaxy S2 may be better. Native application was much faster than Mashup component in test case 3. But, 23ms is good performance to use. MCSC component works well and has similar performance in heterogeneous environment in test case 4. In case 2, "S" means server, and "C" means client.

<table>
<thead>
<tr>
<th>No</th>
<th>Evaluation conditions</th>
<th>Evaluation results</th>
<th>Avg.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>#1</td>
<td>#2</td>
</tr>
<tr>
<td>1</td>
<td>Scenario 1 in Galaxy S2</td>
<td>26ms</td>
<td>21ms</td>
</tr>
<tr>
<td></td>
<td>Scenario 3 in Galaxy S2</td>
<td>408ms</td>
<td>347ms</td>
</tr>
<tr>
<td>2</td>
<td>Scenario 2 in Galaxy S(S)</td>
<td>200ms</td>
<td>175ms</td>
</tr>
<tr>
<td></td>
<td>Scenario 2 in Galaxy S2(C)</td>
<td>489ms</td>
<td>373ms</td>
</tr>
<tr>
<td></td>
<td>Scenario 2 in Galaxy S2(S)</td>
<td>315ms</td>
<td>252ms</td>
</tr>
<tr>
<td></td>
<td>Scenario 2 in Galaxy S2(C)</td>
<td>526ms</td>
<td>511ms</td>
</tr>
<tr>
<td>3</td>
<td>Scenario 1 in Galaxy S2</td>
<td>1ms</td>
<td>1ms</td>
</tr>
<tr>
<td></td>
<td>Native App in Galaxy S2</td>
<td>25ms</td>
<td>29ms</td>
</tr>
<tr>
<td>4</td>
<td>Scenario 2 in Galaxy S2</td>
<td>526ms</td>
<td>511ms</td>
</tr>
<tr>
<td></td>
<td>Scenario 2 in iPhone</td>
<td>531ms</td>
<td>611ms</td>
</tr>
</tbody>
</table>

6. Conclusion

In this paper, we designed and implemented MCSC as a new paradigm in mobile computing. MCSC can transform mobile device into personal server machine. We can implement mobile applications in fast and easy way using instant Mashup in MCSC. The applications are new Mashup components which orchestrate prebuilt components in MCSC. We realized the aim of Service Oriented Computing which can compose new services easily by connecting some readymade components. MCSC makes us get other device’s context information like location, phone state, and so on. Applying the context information, there can be various kinds of context aware services. We composed the three prototype services and evaluated them. As a result, MCSC operates fully in heterogeneous environment while having competitive performance compared to native application. MCSC opened a new way to the advanced mobile context aware platform.

Acknowledgements

This research is supported by the MKE(Ministry of Knowledge Economy), Korea, under the ITRC (Information Technology Research Center) support program supervised by the NIPA (National IT Industry Promotion Agency) (NIPA-2012-H0301-12-3006), Next-Generation Information Computing Development Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science and Technology (2012-0006425) and Ministry of Culture, Sports and Tourism(MCST) and Korea Creative Content Agency(KCCA) in the Culture Technology(CT) Research & Development Program (R2012030096).
References


Authors

Chol-Hong Im
He finished his undergraduate studies in Materials science and Engineering from Korea University, in 1996. He received MS degree in Electronic Computer engineering from Korea University, in 2003. He is now studying in Ph.D. in distributed multimedia computing from Korea University. He got certified as professional engineer in information engineering. He has been working as SW developer and architect for SK C&C and several companies in Korea.

Chang-Sung Jeong
He is a professor at the Department of Electrical Engineering at Korea University. He received his M.S.(1985) and Ph.D(1987) from Northwestern University, and B.S.(1981) from Seoul National University. He was a professor at POSTECH during 1982-1992. He has been working as a chairmain of Collaborative Supercomputing Group in GFK(Global Forum in Korea) since 2001, and a chairmain of Computer Chapter at Seoul Section of IEEE region 10. His research interests include distributed concurrent computing, grid computing, and collaborative ubiquitous computing.