Enhanced Mobile Cloud Computing Platform

Iraky Khalifa
Department of Computer Science, Faculty of Computers and Information, Helwan University, Egypt

Hala El-Sayed
Department of Computer Science, Faculty of Computers and Information, Helwan University, Egypt

Islam Abd El-Gaber
Department of Computer Science, Faculty of Computers and Information, Helwan University, Egypt

ABSTRACT
Mobile Cloud Computing is a technology that influences integrated resources of varied clouds and network technologies toward limitless mobility, functionality and storage. It serves a mass of mobile devices anywhere, anytime through different channels, being either Ethernet or Internet regardless of diverse environments and platforms; it also means to provide mobile users with data storage and processing power on a cloud computing platform. Still there are some problems integrating both Cloud Computing and Mobile Computing as Cloud Computing is not initially designed to be consumed from mobile devices with its limited resources; however the mobile devices could interact with cloud services containing drawbacks such as resources limitation, bandwidth, and unstable connection.

In this paper we present Enhanced Cloud Computing Platform, which enhances the interaction between the mobile devices and web services through cloud platform for optimized mobile web service communication by providing an alternative architecture for the XML based SOAP services.

Keywords

Mobile computing, cloud computing and mobile cloud computing.

1. INTRODUCTION
According to a new study from ABI Research [3] has revealed that Cloud computing will completely transform future of mobile applications development, and their use. Cloud Computing will dramatically reduce the requirement of advanced handsets for running mobile applications, according to the study. According to the latest study from Juniper Research, the market for cloud-based mobile applications will grow 88% from 2009 to 2014. The market was just over $400 million this past year, says Juniper, but by 2014 it will reach $9.5 billion.

Currently the number of mobile devices is 4 Billion 1.08 Billion devices are smart phones with internet enabled expecting to reach 5.8 Billion by 2013 however there exists only 1.1 Billion PC willing to grow up to 2 Billion by 2015. Given the fact that the mobile devices have limited resources including memory, storage and processing; their internet consumption mainly depend on web services. Today the most popular websites such as Google, YouTube, Facebook and Twitter are exposed as web services to be consumed through mobile devices with different manufactures such as Apple, Samsung, HTC, Nokia and BlackBerry are using large number of operating systems for example IOS, Android, Symbian, QNX OS and Windows phone.

Web services are Web-based applications composed of business functions accessed through the Internet. From a technical perspective, Web services are a standardized way of integrating Web-based applications using open standards including XML, the simple object access protocol known as SOAP, the Web Services Description Language WSDL [5], and the universal description, discovery, and integration specification. XML structures the message, SOAP transfers the message, WSDL describes the available services, and UDDI lists them. XML describes both the nature of each self-contained function and the data that flows to and from it.

Considering mobile devices' tenuous capabilities, Web services are one of the best ways to enable lightweight mobile devices to share the computing capability of workstations [1]. Though, a direct integration of mobile computing and web services imposes performance limitations because of XML's verbose nature and physical limitations of mobile computing.

Even a half a decade ago, there are a few users who access remote information from their mobile device. The information access from mobile devices, however, has become easier than ever recently with the help from advanced mobile devices and availability of internet phone networks. There are many projects that try to adopt mobile devices with data connections as major elements in Web Services.

However, the verbose nature of current XML based SOAP approach shows performance limitations in integrating mobile computing applications and conventional Web Services directly. SOAP achieves ubiquity by using highly universal XML as a form of data exchanging between distributed computing resources. Though, XML based SOAP possesses three major characteristics that may affect SOAP performance. First, the in-memory data model must be converted to textual format to build a SOAP message object and to extract information from it. Secondly, because of inevitable mobile computing characteristics – high latency, narrow bandwidth, limited computation, and small memory space, SOAP message processing consumes valuable resources. Finally, Mobile networks have limited bandwidth and are often billed based on the amount of data transferred which is usually not a problem on the powerful wired networks.

In this paper, we introduce a cloud based architecture designed to overcome some of mobile cloud computing problems including bandwidth, mobile limited resources, loss of connection and mobile devices multiple operating systems through adapting the cloud services to commensurate with mobile devices..

2. Background and Related Work
There are several notable researches from industry and academia that try to overcome performance limitations of current Web Services approach. The report of the W3C Workshop [4] on Binary Interchange of XML Information

Item Sets (Infoset) is the result of the increasing demand of binary form of XML based communication.

W3C XML Protocol Working Group released the draft of Message Transmission Optimization Mechanism (MTOM) and XML-binary Optimized Packaging (XOP). The XML encoding would damage the data integrity. XOP is an alternate serialization that looks like a MIME package. It avoids data binding overhead, though still preserves XML structure - tags. Thus, XOP and MTOM, which describes how XOP is layered into SOAP HTTP transport, still possess a parsing issue inherited from SOAP/XML. Cross Format Schema Protocol (XFSP) is another project that serializes XML document based on schema. Initially it is motivated by the flexible definition of network protocols. It is written in Java and uses DOM4J model to parse the schema. With XML Schema-based Compression (XSBC), XFSP provides binary serialization and parsing framework.

All previous remarkable efforts tried to modify the XML based SOAP. There is another approach called RESTful web service that needs to be investigated as traditional SOAP web services are memory and processor intensive, which does not naturally lend itself to the limited memory and processing of smart mobile device environment. on the other hand RESTful web services lend themselves perfectly to smart mobile device environments as they are easy to invoke, produce a discretely formatted response and can usually be easily parsed using event-driven XML parsing which is less memory intensive than tree based parsing.

2.1 Cloud Computing

Cloud computing [8] is the delivery of computing services over the Internet. Cloud services allow individuals and businesses to use software and hardware that are managed by third parties at remote locations. Examples of cloud services include online file storage, social networking sites, webmail, and online business applications. The cloud computing model allows access to information and computer resources from anywhere that a network connection is available. Cloud computing provides a shared pool of resources, including data storage space, networks, computer processing power, and specialized corporate and user applications.

The main characteristics of cloud computing include on-demand self-service, broad network access, resource pooling, rapid elasticity and measured service. On demand self-service means that customers (usually organizations) can request and manage their own computing resources. Broad network access allows services to be offered over the Internet or private networks. Pooled resources means that customers draw from a pool of computing resources, usually in remote data centres. Services can be scaled larger or smaller; and use of a service is measured and customers are billed accordingly.

The cloud computing service models are Software as a Service (SaaS), Platform as a Service (PaaS) and Infrastructure as a Service (IaaS). In Software as a Service model, a pre-made application, along with any required software, operating system, hardware, and network are provided. In PaaS, an operating system, hardware, and network are provided, and the customer installs or develops its own software and applications. The IaaS model provides just the hardware and network; the customer installs or develops its own operating systems, software and applications.

Cloud Computing offers advantages by allowing users to use infrastructure, platforms, and software (e.g., application programs) provided by cloud providers like Google, Amazon, Microsoft and Salesforce at low cost. In addition Cloud Computing enables users to elastically utilize resources in an on-demand fashion. As a result, mobile applications can be rapidly provisioned and released with the minimal management efforts or service provider's interactions [8]. With the explosion of mobile applications and the support of Cloud Computing for a variety of services for mobile users, Mobile Cloud Computing is introduced as an integration of cloud computing into the mobile environment. Mobile cloud computing brings new types of services and facilities for mobile users to take full advantages of cloud computing.

2.2 Mobile Computing

Mobile devices allow users to run powerful applications that take advantage of the growing availability of built-in sensing and better data exchange capabilities of mobile devices. As a result, mobile applications seamlessly integrate with realtime data streams and Web 2.0 applications, such as mashups, open collaboration, social networking and mobile commerce [1], [2]. The mobile execution platform is being used for more and more tasks, e.g., for playing games; capturing, editing, annotating and uploading video; handling finances; managing personal health, micro payments, ticket purchase, interacting with ubiquitous computing infrastructures. Even mobile device hardware and mobile networks continue to evolve and to improve, mobile devices will always be resource-poor, less secure, with unstable connectivity, and with less energy since they are powered by battery. Resource poverty is major obstacle for many applications [3]. Therefore, computation on mobile devices will always involve a compromise.

2.3 Web Services

The Web Services architecture [9] has been based on a number of open and freely available internet standards such as XML, SOAP and WSDL. Its main purpose is to provide a basic architecture for interoperability between applications.

The Web Services architecture consists of several components. One of them is an interface description language that defines the methods and bindings of the Web Service, which constitute the API of the Web Service. An XML-based interface description language, called the Web Service Description Language (WSDL), is used for this purpose.

To invoke a Web Service, the request and response are sent using the Simple Object Access Protocol (SOAP). The transmitted parameter names, values and possible error conditions are encoded through XML. HTTP is often used as an underlying transport protocol, but other protocols can also be employed. The discovery of the service and its interface description can be achieved by querying the UDDI (Universal Description, Discovery and Integration) registry. A response from the UDDI contains the description of a service and a URL pointing to its WSDL file.

In the current Web Services architecture there exist a small number of well-known global UDDIs that maintain information about available Web Services and replicate data among themselves. Since the code for a UDDI registry is freely available, it is possible to deploy UDDI registries locally in order to reduce internet traffic. This is especially interesting for services that are to be hidden from anyone outside the local domain.

Representational State Transfer (REST) was originally introduced as an architectural style for building large-scale distributed hypermedia systems. This architectural style is a rather abstract entity, whose principals have been used to

explain the excellent scalability of the HTTP 1.0 protocol and have also constrained the design of its following version, HTTP 1.1. Thus, the term REST very often is used in conjunction with HTTP.

RESTful Web services are perceived to be simple because REST leverages existing well-known W3C/IETF standards (HTTP, XML, URI, and MIME) and the necessary infrastructure has already become pervasive. HTTP clients and servers are available for all major programming languages and operating system/hardware platforms, and the default HTTP port 80 is commonly left open by default in most firewall configurations.

Such lightweight infrastructure, where services can be built with minimal tooling, is inexpensive to acquire and thus has a very low barrier for adoption. The effort required to build a client to a RESTful service is very small as developers can begin testing such services from an ordinary Web browser, without having to develop custom client-side software. Deploying a RESTful Web service is very similar to building a dynamic Web site.

2.4 Mobile Applications

Mobile Application is a type of application software designed to run on a mobile device such as a smartphone or tablet computer. Mobile applications frequently serve to provide users with similar services to those they access on their PCs. There are three types of mobile applications:

1. Native Application: An application specifically designed to run on a device's operating system and machine firmware; it typically needs to be adapted and adjusted for different devices.
2. Web Application: An application in which all or some parts of the software are downloaded from the Web each time it is run; it can usually be accessed from all web-capable mobile devices.
3. Hybrid Application: Hybrid Application combines mobile web development with native development; it enables the use of HTML, CSS (Cascading Style Sheets) and JavaScript in a mobile application and it also extends native device capability into the mobile web browser.

Table 1: Native vs. Web vs. Hybrid Mobile Applications

Feature	Native	Web	Hybrid
Development Language	Native Language	Web Language	Native and Web
Code Portability	Low	High	High
Access to Device APIs	High	Low	Moderate
User Interface	High	Low	Moderate
Upgrade Flexibility	Low	High	Moderate
Installation Experience	High	Low	High
Development Time	High	Low	Moderate

As shown in table 1 hybrid application contains most of the advantages of the two types because it is implemented by developing a native application containing a customized web browser.

3. Proposed System Architecture

This section presents system architecture of mobile cloud computing platform that is designed to provide enhanced communication between the cloud and mobile devices. First the overview system architecture is introduced, and then both of the bridge server and mobile client designs will be discussed.

3.1 Architecture Overview

The main goal of the mobile cloud computing architecture is to provide a proxy for mobile clients connecting to cloud services. Figure 1 shows an overview of the mobile cloud computing and its main features. The architecture consists of three parts, the mobile clients, the bridge server and the cloud services. Since cloud services are usually controlled by service providers, the bridge server performs all the necessary adaptation to the mobile clients.

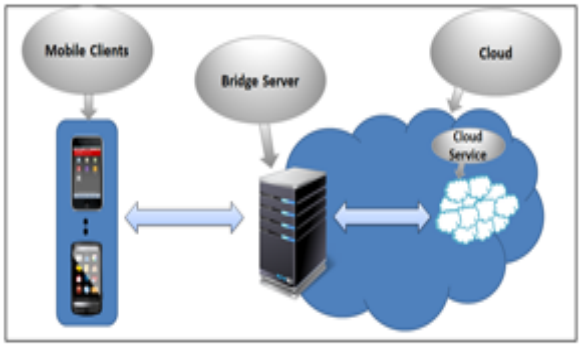

Figure 1: Proposed architecture overview

Some services require real-time updates, for example, RSS, Facebook, and Twitter service. The bridge server also pushes updates of service results to mobile clients via HTTP or email immediately after it receives the updates.

The main function of the bridge server is to consume raw could services whether they are SOAP or RESTful and deliver it to mobile clients; it also provides a JSON based RESTful Services which will be consumed through the mobile clients. Figure 2 shows how Bridge Server interacts with mobile clients; when a mobile client sends a request to the Bridge Server, the following phases are executed:

1. The mobile client sends a HTTP GET request to the Bridge Server.
2. The Bridge Server calls the cloud servers and generates the results.
3. The Bridge Server converts and optimizes the results and sends the optimized results back to the mobile client.

The bridge server is the main component in the architecture as it is responsible for all the business logic and adaptation phases that are based on web services as communication objects.

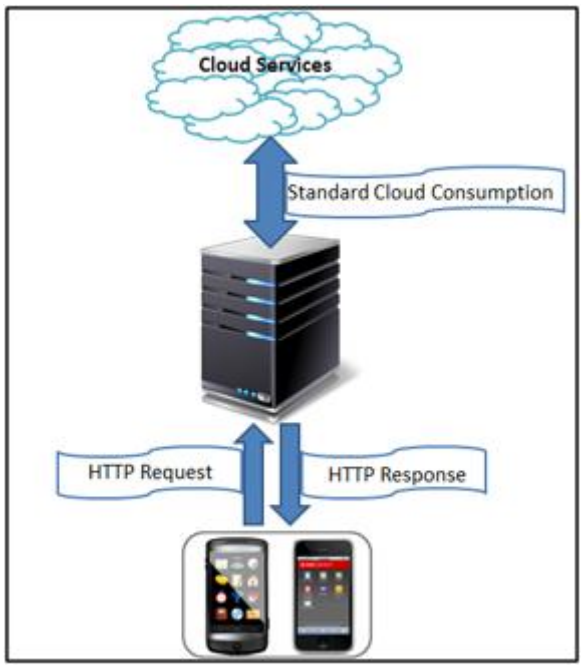

Figure 2: Consuming cloud services through mobile clients

3.2 Bridge Server Architecture

The bridge server architecture as shown in figure 3 has a RESTful service repository which is responsible for handling the mobile clients' requests through receiving those requests then processes it by passing it to the service handler and finally sends the result back to the mobile client. The service handler has two main components: the service executer and optimizer; the service executer creates a web requests and dispatch them to be executed at the cloud services; the executer receives the raw results and deliver them back to the handler which sends them to be processed and optimized at the result optimizer. The bridge server mainly performs the following processes:

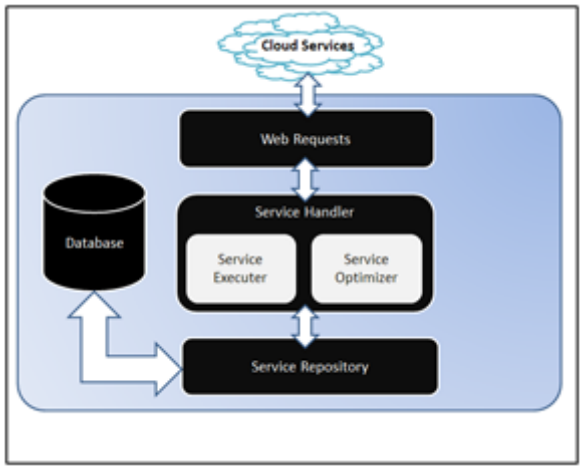

Figure 3: Bridge server architecture

1. Protocol Transformation: the bridge server transforms the SOAP services into the RESTful services; the server handler executes the SOAP services using normal HTTP web request to consume the SOAP service based on its provided WSDL then send the result back to the mobile clients through RESTful service.
2. Result Optimization: the raw service is formatted in a structure that may be not suitable for the mobile clients; the bridge server Optimizer receives the raw results and extracts the result core converting it into JSON format. For example, the user may only need 5 instead of 10 news stories. Second, the original data format may also not be efficient for mobile clients. The result optimizer first extracts the required part of data from the raw response, and then makes a copy of the extracted result in various formats, for example, mobile HTML for mobile browsers and JSON for native mobile applications. The bridge server also caches these copies of result in the service repository.
3. Client Subscription: the Bridge Server also allows results caching; after optimizing the results the Bridge Server saves a local copy from it to be pushed back to the client in case of connection error or client's starting up.

After the result processing at the Optimizer, it is returned to the Service Repository to be cashed at the system storage to be used when needed and the service calling is recorded in the database then the result is sent back to the mobile client for user usage.

Like most bridge server, scalability is always a major concern; the approach is to take advantage of the cloud platforms to host the bridge server. Microsoft Azure is the cloud platform examined in this research. It has a very complex service model and performance characteristics.

3.3 Mobile Client Design

The proposed mobile client architecture is a hybrid solution which according to the comparative study provide in section 2.4 combines both native and embedded browser application. Figure 4 is the overview of the client architecture. It follows a basic Model View Controller pattern. The User Interface is designed within the embedded browser using HTML, CSS and JavaScript. When the UI components need service data, they invoke the custom JavaScript libraries to pull the data from local cache. If the local cache does not contain a recent copy of inquired data, the RESTful client interacts with the middleware to get the data. The data are then passed to the data module and stored in the local file system. Note that the data passed to the embedded browser is in JSON format which can be easily parsed by JavaScript.

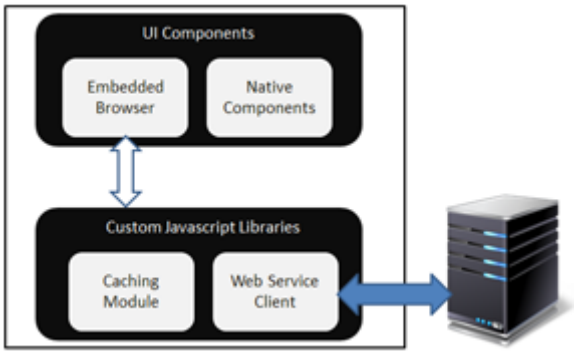

Figure 4: Mobile client architecture

The separation of UI components and the client makes the architecture platform independent. To change the application to a pure native application, the embedded browser UI can be replaced by native UI and the client can be reused. The

RESTful client can also implement push technology. Push technology enables a server to push content to the clients, in order to optimize the data traffic, energy and bandwidth used.

4. Proposed System Implementation

This section covers the implementation of the system architecture for mobile cloud computing platform described in the system architecture section; In order to gain better understanding to the architecture a prototype for both bridge server and mobile client was implemented.

4.1 Bridge Server Implementation

The Bridge Server is implemented based on Microsoft Azure Cloud (see Figure 5) to be able to use the features of scalability and interoperability which are required by the platform architecture to support various mobile platforms. The Mobile Client is developed via Phone Gap; it is an open source framework for building cross platform mobile applications using HTML5, Javascript and CSS, the main purpose of using this framework is to solve the problem of building a separate application for each device such as iPhone, Android, Windows Mobiles and more.

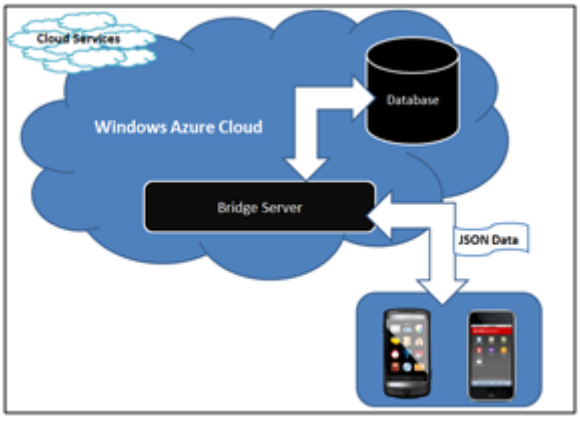

Figure 5: Bridge server implementation

The Bridge Server architecture is implemented as Microsoft .Net web application hosted on Microsoft Azure cloud. The application mainly exposes Repository Windows Communication Foundation (WCF) RESTful service layer to be consumed via mobile clients as RESTful service is more suitable to the mobile usage; the Repository Service layer mainly depend on exchanging JSON data through RESTful services with the mobile clients in order to maintain processing efficiency with minimum amount of exchanged data.

When the Service Repository receives a mobile client request, it passes the request to Executor module which sends HTTP Web Requests for external cloud services using .Net WebClient library for initiating the requests and receiving results; then the results is delivered to the Service Optimizer module which uses JSON.Net library for converting SOAP XML results to small sized JSON objects to be used at the mobile client after cashing the results at the database storage via the Service Repository.

4.2 Mobile Client Implementation

The mobile client is developed via PhoneGap framework which is open source mobile development framework which builds applications for mobile devices using JavaScript, HTML5 and CSS3, instead of lower-level languages such as Objective-C, C/C++ or C#. The resulting applications are hybrid based, meaning that produced applications are neither fully native applications nor purely web based applications; the output application is developed once and through a quick conversion process, it could be deployed at multiple mobile platforms such as Android, BlackBerry OS and Windows Phone OS.

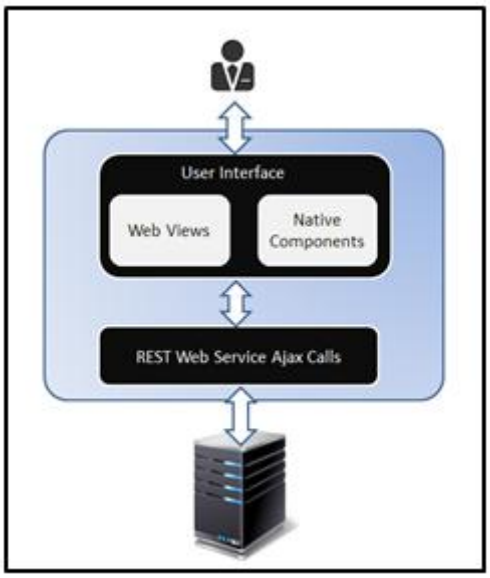

Figure 6: Mobile client implementation

The mobile client as shown in figure 6 sends web requests which access the Bridge Server exposed WCF REST services through GET / POST / PUT / DELETE verbs. Those web requests are sent using JQuery library by calling Ajax method which performs an asynchronous HTTP Ajax request. The optimized JSON formatted results will be displayed at the web view of the mobile client.

5. System Evaluation

This section summarizes the performance benchmark results of the prototype implementation of Mobile Cloud Computing Platform architecture. In order to obtain the benchmark two experiments have been developed.

5.1 Experiment 1: Bridge Server Evaluation

The main goal of this experiment is to evaluate the Bridge Server and its ability for interaction with both SOAP and REST services. eBay API provides both types of the web service in both XML and JSON formats; “Find Products” is used for the test; it searches for stock product information, such as information about a particular kind of mobile or camera. Also, retrieves up to 200 eBay listings associated with a product. The size of JSON result is 11.3 KB, XML SOAP size is 14 KB and the XML REST is 13.3 KB. To better evaluate the Bridge Server the following sub-experiments are done:

1. Consume RESTful service directly without the Bridge Server obtaining JSON results.
2. Consume RESTful service directly without the Bridge Server obtaining XML results.
3. Consume RESTful service through the Bridge Server obtaining JSON results.

4. Consume RESTful service through the Bridge Server obtaining XML results.
5. Consume RESTful service through the Bridge Server with processing obtaining JSON results.
6. Consume RESTful service through the Bridge Server with processing obtaining XML results.
7. Consume SOAP service through the Bridge Server.

The results obtained from the previously mentioned experiments are shown in table 1. Each experiment is tested ten times calculating highest, lowest and mean values for response time.

Table 2: Bridge server evaluation results

Experiment	Low(s)	Mean(s)	Highest(s)
Consume RESTful service directly without the Bridge Server obtaining JSON results	0.6	1.0	1.8
Consume RESTful service directly without the Bridge Server obtaining XML results.	0.5	1.3	3.2
Consume RESTful service directly through the Bridge Server obtaining JSON results.	0.5	0.9	1.7
Consume RESTful service directly through the Bridge Server obtaining XML results.	0.6	1.4	4.9
Consume RESTful service through the Bridge Server with processing obtaining JSON results.	0.6	1.2	3.1
Consume RESTful service through the Bridge Server with processing obtaining XML results.	0.6	1.7	5.5
Consume SOAP service through the Bridge Server obtaining XML results.	0.8	3.1	6.1

However there is a time delay in the processes related to the bridge server compared with the direct consumption but also a reduction in the response size as the optimization also increases the execution time. SOAP services have a higher response time than RESTful services. JSON has lower response time than XML as XML has more size than the JSON.

5.2 Experiment 2: Mobile Client Evaluation

In order to evaluate the mobile client the experiment will be tested at both windows phone (Windows Phone 7 version) and android (4.0 Ice Cream Sandwich version) mobile clients generated by PhoneGap. Each mobile client sends 10 HTTP/GET to the Bridge Server Restful service which returns both JSON and XML responses with fixed size 14 KB calculating the mean and standard deviation values for response time between sending the request till results UI displaying.

Table 3: Mobile client evaluation results

Experiment	Format	Mean(s)	Highest(s)
Consume Bridge Server RESTful service through Android mobile client	XML	98.7	29.6
	JSON	23.3	8.1
Consume Bridge Server RESTful service through Windows Phone mobile client	XML	178.6	38.5
	JSON	112.4	24.2

Table 3 shows that experiment results even when the message size is fixed the response time for JSON data is faster than XML data because the parsing time needed for XML is much larger from the time JSON parsing time although according to experiment 1 always the JSON message smaller than XML message. Hence the optimum choice for the mobile client to communicate with the Bridge Server is JSON based RESTful services.

6. Conclusion

In this paper an enhanced mobile cloud computing platform is introduced describing the interaction problems between cloud and mobile devices which are the motive for exploring this space. The main purpose for providing this platform is to adopt could services to be consumed via mobile devices.

The prototype implementation for the platform demonstrated the advantages and disadvantages of the platform architecture through the evaluation of all the options provided by the platform prototype. The evaluation also showed that the server ability to communicate with other cloud services besides exposing the service layer for the mobile clients. Mobile clients are able to consume both XML and JSON result formats, demonstrating that the JSON is suitable for mobile consumption more than XML.

Future work in this area has to include more wide scale development for the prototype. Also automatic user service definition option should be developed to allow the user to interact with other cloud services which are not pre-defined by the platform.

7. References

[1] M. Satyanarayanan, “Fundamental challenges in mobile computing,” in Proceedings of the 5th annual ACM symposium on Principles of distributed computing, pp. 1-7, May 1996

[2] J. Jing, A. S. Helal, and A. Elmagarmid, “Client-server Computing in Mobile Environments,” ACM Computing Surveys (CSUR), vol. 31, no. 2, pp. 117–157, 1999.

[3] ABI Research. http://www.abiresearch.com/. 2010

[4] World Wide Web Consortium, “Report from the W3C Workshop on Binary Interchange of XML Information Item Sets”, Sep. 2003, http://www.w3.org/2003/08/binary-interchangeworkshop/

[5] World Wide Web Consortium, Web Services Description Language (WSDL) 1.1, Mar. 2001, http://www.w3.org/TR/wsdl.

[6] World Wide Web Consortium, "XML-binary Optimized Packaging (XOP)", Aug. 2004, http://www.w3.org/TR/2005/REC-xop10-20050125/

[7] E. Serin and D. Brutzman, "XML Schema-Based Compression (XSBC)", http://www.movesinstitute.org/xmsf/projects/XSBC/03Mar_Serin.pdf

[8] Michael Armbrust and Armando Fox, "Above the Clouds: A Berkeley View of Cloud Computing", http://www.eecs.berkeley.edu/Pubs/TechRpts/2009/EECS-2009-28.html

[9] Web Services Activity http://www.w3.org/2002/ws/