ORIENTAL JOURNAL OF
COMPUTER SCIENCE & TECHNOLOGY
An International Open Free Access, Peer Reviewed Research Journal
Published By: **Oriental Scientific Publishing Co., India.**
www.computerscijournal.org

ISSN: 0974-6471
December 2014,
Vol. 7, No. (3):
Pgs. 416-424

Enhance the Interaction Between Mobile Users and Web Services using Cloud Computing

ATUL M. GONSAI[1] and RUSHI R. RAVAL[2]

[1]Department of Computer Science, Saurashtra University, Rajkot, Gujrat, India.
[2] Department of M. Sc. (IT), GK & CK Bosamia College, Jetpur,
(Affiliated with Saurashtra University), India.

(Received: November 20, 2014; Accepted: December 16, 2014)

ABSTRACT

Day-by-day smartphone network's structures are improving in an efficient manner; they are becoming ideal users to accessing the any web resources or a service, specifically, Services which are access by Internet. Web services that are used to provide changed kind of services for an app running on smart mobile users suitable and widespread used; still there are some limitations of the current smart phone clients in common manner, like as low processing speed, limited storage capacity, less band-width, latency, and in-adequate memory. This paper gears a platform free architecture for connecting mobile users to the existing Internet based Services. In this architecture includes a cross-platform design of smart mobile users based on client services and a middleware for acquisitive the communication between mobile users and Internet based Web Services. We have used the architecture for deployed services on cloud platforms, such as "Google App Engine" (GAE) and "CloudSim" to enhance the consistency and scalability and reached up to the end-users.

Key words: Web-Services; Mobile Clients; JSON; GAE; CSV; PHP; REST.

INTRODUCTION

Mobile clients are likely to upsurge progressively from current users. As cellular network infrastructures incessantly improve, their data broadcast develops gradually affordable and available, and thus they are becoming prevalent traditions to access the Internet Based Services, mainly the Web Services that are also available in the cloud computing. Today, mobile clients' uses devices like Android, iPhone, and Black-berry, have encompassed applications that consume the web services from popular websites, such as YouTube, eBay and Msn. The share of android in smart phone market is 46%, iPhone is 35% and blackberry is 10%[1].

Though, there are difficulties in connecting smart phones to present Internet based Services. Initially, the Internet Based Services need to deliver optimization for mobile clients. For e.g. the size of the Web Services messages desires to be reduced to fit the bandwidth of mobile clients. Moreover, smart phones have to acclimate to different kinds of Web Services, for e.g. SOAP and RESTful Services. Figure1 shows a smart phone retrieving

Web Services. This paper considers how Cloud Computing[2] can help mobile clients connect to existing Web Services.

Problem definition

Accessing Web Services from a smart phone based mobile client is different compared to the standard web services developments, because of these aspects environment. A review done by Earl *et al*[3] estimated how well the current mobile clients including Android, Symbian, iPhone, and Windows Mobile, support the perception of cellular network based research. According to the review, all of these mobile platforms have certain limitations.

- The communication between user and service is established through wireless network.
- Mobile clients have limited resources like Low processing power, small screen size etc.
- Present Web Services in the Cloud doesn't support mobile clients.

There are several challenges in accessing the Internet based Services from the existing smartphone clients. The following two are the focus of this paper.

Connection loss Problem: Meanwhile the smart phone based devices are not stable and due to the mobility of the smart phones and the wire-less network setup, smart phones can be momentarily removed from the previous connected network and later may join available network.

Latency/ Bandwidth Problem: Cellular networks have a very limited bandwidth and are often billed based on the amount of data transferred. Though, even a simple SOAP message often contains a large chunk of X M L data, which consumes a lots of bandwidth and the broadcast can cause major network dormancy. In addition, the SOAP message contains mostly X M L tags that are not all necessary for mobile clients.

Table. 1: Http Methods and their Action

HTTP Methods	CRUD Operation
GET	Retrieve a Resource
POST	Create a Resource
PUT	Update a resource
DELETE	Delete a Resource

Limited resources problem: mobile clients are normally "thin clients"[4] with a less processing power. They also have limited computational power and screen size/resolution. These deficiencies are only due to mobility[5].

Architecture for mobile client devices

Middle ware Architecture[6] is primarily used in Distributed Computing system (DCS). DCS[7] "consist of multiple processors that don't share primary memory, however messages sending over the network"; Mobile users are distributed computers that connect to the middle-ware. In Emmerich paper[8], he stated four requirements scenario for general middle ware.

Heterogeneity: Components in a distributed system can be implemented with different languages and deployed on different-different platforms. Therefore, the design needs to study a heterogeneous location.

Fig. 1: Mobile clients accessing the Internet web services[19]

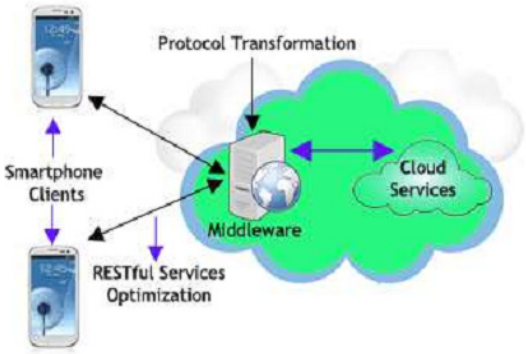


Fig. 2: Architecture for Mobile Clients Middle Ware

Network communication: Masses who need to communicate with each-other involves some transport layers (T C P and U D P) and arranging, a method of transforming data structure to transferable format.

Reliability: Requests maybe lost during the network transmission. The middleware needs to deploy error detection and correction mechanisms to enhance reliability.

Scalability: Distributed systems deal with client interactions and also interact between distributed components. Changes in the allocation of components could affect the system architecture, which refers as transparency in the reference model of open distributed processing

Co-ordination: Since distributed systems have multiple points of control, different components need to coordinate and collaborate through synchronization.

Middleware Architecture is often used to extend functions for thin clients, like mobile devices. Uribarren *et al*[9] proposed a middleware for adaptation in mobile environments. The proposed middle-ware hides the complexity of deploying ubiquitous Apps. Applications are automatically moved between different platforms.

When designing distributed systems, scalability, should be the major concern issue. Rajive *et al*[10] did research on investigating scalable middle ware to support mobile Internet applications.

The future middle ware solution for mobile clients mostly focuses on application and content adaptation. Heterogeneity, Network communication, reliability, and co-ordination are four fundamental

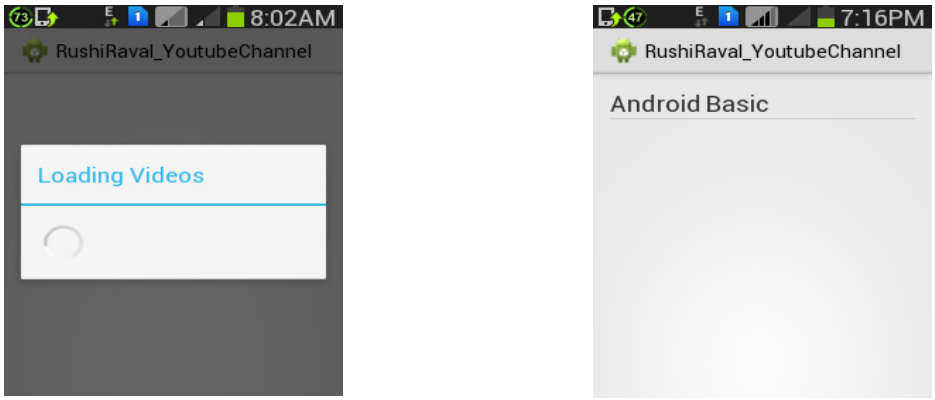


Fig. 3(1): Layout on Android Mobile Clients

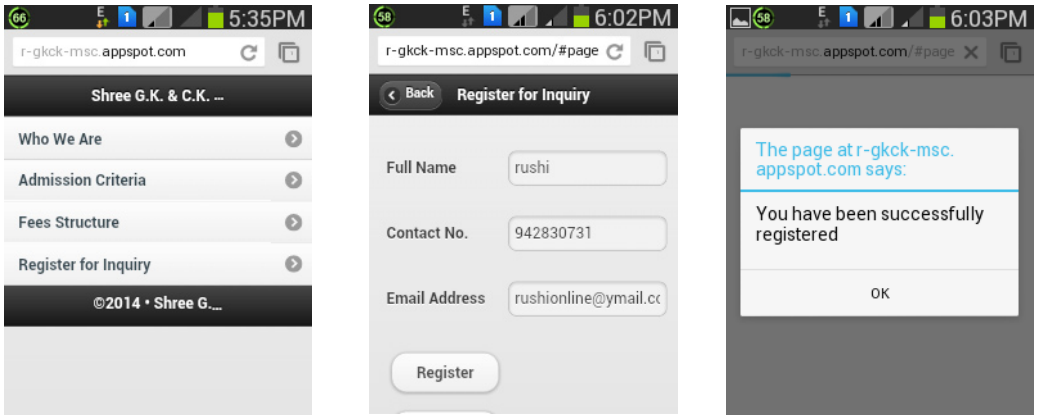


Fig. 3(2): Student admission procedure at college using Mobile App

requirements for general middleware as well as middle ware for mobile device[11]. Scalability can be achieved with distributed middle ware. Context can help middleware to adapt to the heterogeneous environment. However, the goal of the paper is to use middleware to improve the interaction between mobile clients and internet services as well as use Cloud platforms to improve the scalability and reliability of the middleware.

RESTful web service

RESTful[12] is a software application architecture modeled after the way data is accessed, modified and represented on the web. In the REST architecture, data and functionality are considered as resources, and these resources are retrieved with use of Uniform Resource Identifiers (URIs), typically links on the web. The resources are acted upon by using a set of modest, well defined processes. The REST architecture is fundamentally client-server based architecture, and is designed to using as a state less communication protocol, usually HTTP. In the R E S T architecture, users and servers interchange representations of resources with use of an identical protocol and interface. These principles encourage REST applications to be simple, lightweight, and have high performance.

RESTful web services[12] are web applications built upon the REST architecture. They expose resources (data and functionality) through web URIs, and use the four main HTTP methods to C.R.U.D. which are known as create, retrieve, update, and delete resources. RESTful WS typically map the four main HTTP methods to the so called C.R.U.D. operations: create, retrieve, update, and Del, below table shows a representing of HTTP methods to these C.R.U.D. operation.

Proposed architecture for middle ware

The middle ware that is proposed will act like as proxy that is hosted on the Cloud platforms which provide mobile clients access to Cloud services. The middle ware architecture will improves interaction between mobile clients and Cloud Services, for e.g., adaptation, optimization and caching. The middle ware also provides extended functions to mobile clients. In general,

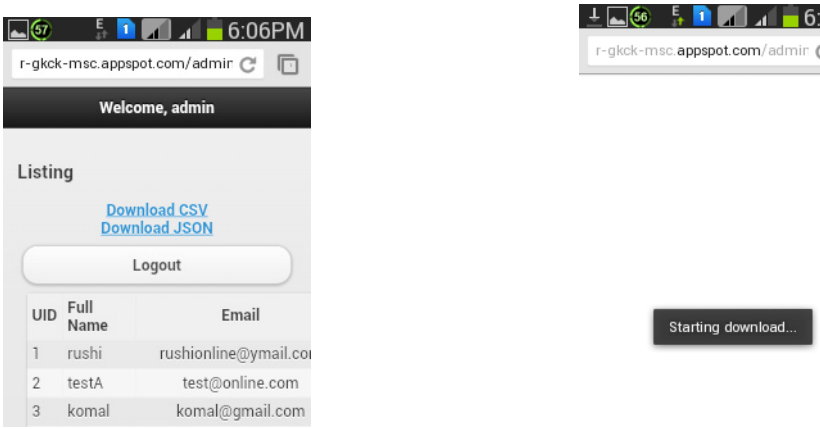


Fig. 3(3): Admin can download in JSON and/or CSV format

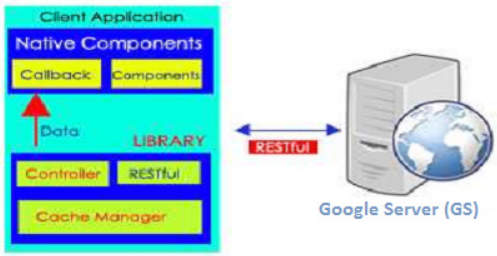


Fig. 4: Native Smart phone implementation

the architecture enhances the dependability, functionality, and compatibility of the interaction between mobile clients and Cloud Web Services.

In order to overwhelm the difficulties stated in the previous sections, the Cloud Computing architecture provides the following features to enhance the interaction between mobile clients and Web Services.

No Loss of connection: Client and middleware caching -Copies of service results are stored on both mobile clients and the middle ware. While the mobile clients are not able to connect to the middle ware, the client-side cache is used. When the middleware to server connection is not available, the middleware returns its cached data to the mobile clients.

Bandwidth/Latency: Protocol transformation – Protocol transformation reduces the latency as well as bandwidth of the client to service interaction. The middle ware transforms Simple Object Access Protocol (SOAP) to a much light- weight format JSON through Restful Internet Web Services. Transferring SOAP to light-weight protocols, like Restful, reduces processing time as well as the size of the messages[13].

Optimization of Result: Result optimization reduces the size of the service results, thus reduces the bandwidth used to interact with internet based services. The middle ware converts the format of service results from X M L to JSON and removes unnecessary data from the original service result. Less data transmitting also reduces network latency.

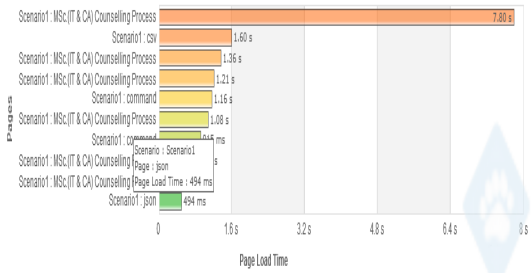
implementation of restful web service

The goal of the middleware cloud architecture is to provide a proxy for mobile clients connecting to Cloud services. Figure 2 shows an overview of the middleware cloud and its key features. The architecture consists of three parts, the mobile-clients, the cloud services and the middle ware. Since Cloud services are usually controlled by service providers, the middle ware performs all the necessary adaptation to the mobile clients.

Middle ware Architecture

The middle ware is responsible for consuming the Cloud Services whether they are SOAP or REST services like JSON and delivers the service result to the mobile clients. On the smart phone, mobile clients can define Web Services and later execute the pre-defined Web Services. The middle ware provide REST interface for the mobile clients. Fig. 4 indicates how to consume/execute a pre-defined Web Services. Note that the execution starts with a HTTP GET request whose URL path contains the resource identifier to the web. When Web Services are executed through the middle ware, the following steps are involved in the middle ware.

- The mobile clients send a HTTP GET request with an identifier of a Web Services to the middle ware.
- The middle ware deals with interactions to the Web Services (and generates SOAP client if necessary).
- The middleware extracts (JSON or X M



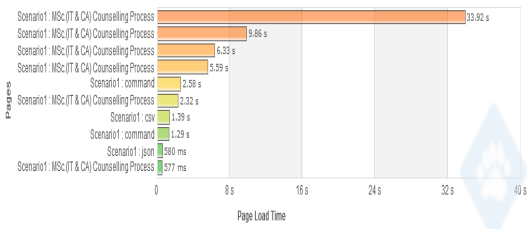
Scenario Completion Time

Average	2 m 11.68 s
Maximum	2 m 11.68 s
Minimum	2 m 11.68 s

Page Load Time

Average	844 ms
Maximum	7.80 s
Minimum	12 ms

Fig. 5(1): Performance testing for JSON, CSV Loading Time and Scenario Completion Time Comparison for 1 User[16]



Scenario Completion Time

Average	2 m 59.44 s
Maximum	3 m 18.96 s
Minimum	2 m 22.56 s

Page Load Time

Average	2.75 s
Maximum	46.92 s
Minimum	5 ms

Fig. 5(2): Performance testing for JSON, CSV Loading Time and Scenario Completion Time Comparison for 20 User[16]

L parsing) the required service results from the original service result and form a new service results in JSON format.

- The middleware stores a replica of result with the service_ID in the database and returns the enhanced result to the mobile client.

Mobile client implementation on middle ware in mobile clients

In order to implemented the proposed smartphone based architecture with client side native libraries. Smartphones has an embedded browser which includes JavaScript libraries that implement several common functionalities of the client side browser, for example, access a file system and location service.

To verify the smart phone based client design, we have develop the design with an idea of generate a list of you tube channel as shown in Figure 3.1. The application is getting response with use of JSON parsing and a restful WS with the mobile client design on smart phone. Using the application, the mobile clients can access the Web Services of the portal which is hosted on Google cloud through the smart phones.

The mobile client application can be divided into three layers; Controller, User interface (UI), and Cache Manager. The UI layer has two enactments, embedded browser UI and native UI. Figure show how they look like on the device. Figure also shows the architecture of both applications. The controller is the key manager among the UI, middleware, and cache manager. The controller creates the U I and gets data from the RESTful client or cache-manager. If that time network connections are not accessible, the controller passes cached data to the UI components.

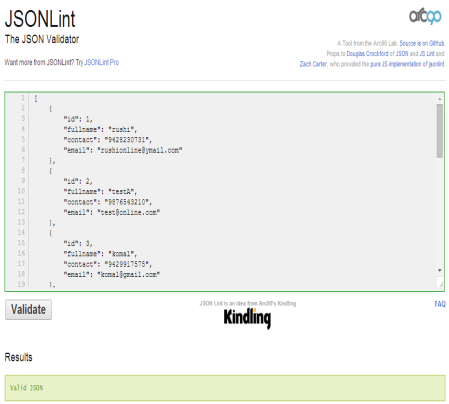


Fig. 5 (3): Validate JSON data using "JSONLint" Tool[17]

Furthermore, we have developed one more mobile/website app design with an idea of admission procedure of college and day-to-day generate a list of students with the full information which was given by the students/parents, this information was accessed by an admin, and admin can also download that data in JSON format as well as in CSV format from their mobile or computer system as shown in Figure 3.2, 3.3. The application is getting response with use of PHP, JSON parsing with the mobile client design on smart phone. By use of this app, the students/parents can access the Web Services of the portal which is hosted on Google cloud server through the use of smart phones.

This mobile client app can be divided into three layers; Google App Engine[20], User interface (UI), and Google Cloud Server. The UI layer has embedded browser UI. Figure show how they look like on the device. The GAE and GCS are the key coordinators among the UI, middle-ware. The GAE manage the user requests and gets data from the mobile client and storage is on Google Cloud Server (GCS). If network connections are not obtainable, the control passes cached data to the U.I. components.

Otherwise, it invokes the RESTful client to get data from the middleware. The cache manager then saves recent received data on a local file system. With the native UI, the client interacts

Fig. 5(4): JSON data parser testing valid data[18]

with the middleware asynchronously. When the native UI requires data, it passes a callback to the controller and continues to receive UI events as shown in Figure 4.

The controller starts a new thread to interact with the middle ware. When the data arrives, the U.I. gets updated through the notification. With this model, the native UI can be updated as soon as the data changes. The android app needs to wait for the facts and figures to reach, because the native library can't receive a Java Script notification. The android app also cannot be updated automatically when the data changes.

The entire middle ware application is hosted on a Google Application Engine GAE uses the Services oriented architecture. The middle ware architecture is implemented as an android application. The application uses the RESTful Internet Service interfaces to mobile clients, since RESTful web services are more suitable for mobile devices[14], Because the middle ware uses RESTful and JSON API of core Android libraries.

The middle ware also uses a popular JSON client library which provides functions of composing custom HTTP requests, sending and receiving HTTP requests and responses. The middle ware architecture expects the Web Services to return JSON responses, so that results can be extracted using the android build-in library. User defined tasks, service actions, parameters and results are Java objects which map to database entities using the android API.

The middle ware still has a RESTful interface to mobile clients, but the Google Application Engine (GAE) platform itself is a Web application server which can only handle server requests. The middle ware constructs and sends HTTP requests through the URL fetch service which implements the android RESTful framework interface.

To enhance the communication between mobile users and Web Services

- Evaluate the cross-platform capability of the mobile clients design.
- Implement the mobile client in different models.
- Consume RESTful WS through the middleware.
- Transfers SOAP WS to RESTful Services to be consumed by mobile clients.
- Reduce bandwidth consumption of mobile clients.
- Push updates to mobile clients in real-time. Using the Cloud platform as a way to improve consistency and scalability of the middle ware
- The middleware can be implemented on "CloudSim" and Google Application Engine "GAE".
- Cloud platform improves the scalability and consistency of the middleware.

Performance evaluation of web services

The middleware is implemented one as a standard android application and other one as a standard PHP with Python application. For first application the middleware uses the Android 2.2 or above version, so it can be run on most of the android devices.

And for another application PHP5.4.13 standard and Python 27, API version 1, so it can be deployed in Google server containers. The middleware also uses the file handling requests to interact with the Google App Engine and also with Google server. In the following experiments, the middleware is deployed in two platforms, Webserver with Apache Platform (Local), and Google Application Engine (GAE), since Application Engine uses Google's internal structure, its h/w specification is unidentified.

Because some experiments require simulating a large number of mobile clients and calculating the response times, a real mobile device is not capable of doing such task. A performance testing tool called "LoadUIWeb" [16] is used as a load generator. The emulator for the cloud is CloudSim[13]. LoadUIWeb is responsible for generating and sending HTTP requests to the middleware in a specified rate. LoadUIWeb calculates the mean of response times every 5 seconds based on its log file scenario. The load generator runs on the standard server for the College Portal. The mobile client is applied on Android. The Android device's using

Android Version 4.2. The build-in Apache HTTP client is used to send HTTP request. Together are connected to the Internet through wireless 802.11g standard. The client uses the I/O libraries from PHP and in build Browser Support.

Consuming College admission Web Services through the Middleware

This experiment compares the overhead associated with different WS interactions. College Admission Portal provides RESTful WS interfaces for their student admission service. Their RESTful WS return result in either JSON or CSV format. The tested WS is returned the result in following manner.

In above scenario, we have set some parameters like as User count: 1, Workstation: Virtual, Browser: Google Chrome, Connection speed: Maximum, Load: Steady Load, Max page load time: 5 second.

For next scenario, we have set some parameters like as User count: 20, Workstation: Master, Browser: Google Chrome, Connection speed: Maximum, Load: Steady Load, Continuous Load: Enable and set to 10 seconds.

The middle ware is run on the standard server. The "UID", "Full Name", and "E-mail" segments of JSON and CSV result were taken for the tests for the WS.

The size of the JSON result is about 129 KB and the size of the CSV result is about 190 KB. The load generator sends HTTP request at the rate of 1 request per 5 second so the middleware does not overload.

Validate JSON and CSV Data

We have also tested generated data which is in JSON and CSV format whether they generate valid data or not, for that we are using online tool "JSONLint" provided by arc90 Lab[17].

Enhancing Interaction between the Client and Middleware over Cloud

Consume "r-gkck-msc" RESTful Web Services directly with JSON result and CSV result.

Consume "r-gkck-msc" RESTful Web Services through the middleware with JSON result. The middleware forwards the whole result. (No parsing involved in this scenario)

Consume "r-gkck-msc" RESTful Web Services through the middleware with CSV result.

Consume "r-gkck-msc" RESTful Web Services through the middleware with JSON result. The middleware returns the optimized result in JSON format.

Consume "r-gkck-msc" RESTful Web Services through the middleware with CSV result. The middleware returns the optimized result in CSV format.

CONCLUSIONS

As service clients, smart phones basically have unique properties like they are portable and small. They are personal devices with numerous sensors. However, these smart-phones have limitations, for e.g., minor bandwidth, loss connectivity and less processing power. On the other hand, the existing services are normally designed for immobile clients. For e.g. SOAP is a protocol which involves a lot of X M L parsing. To overcome the limitations, this paper presents the Middleware for the mobile based architecture for connecting mobile device to the existing Cloud Services.

The projected mobile-client design is mobile platform independent. The mobile client provides an interface for users to define services and consume them through the middle ware. It interrelates with the middle ware through RESTful WS interface. The mobile client has been applied as on Android platform. The smart phone based design involves native as well as browser based applications. For better compatibility aspects, the interface can be applied on implanted browser with use of HTML, CSS and JavaScript, while the actual client component is implemented in platform reliant on specific language, the server side scripts can run on the application server.

The middleware provides a medium for the smart phones to access the Cloud Services. The middleware also provides result optimizations which extract the required data from the original service results.

REFERENCES

1. Charles Arthur "Nokia revenues slide 24% but Lumia sales rise offers hope". *The Guardian*. (Retrieved 19 July 2013).
2. M.A. Vouk, "Cloud computing: - Issues, research and implementations," Information Technology Interfaces, 2008. I T I 2008. 30th International-Conference on, 2008, pp. 31–40.
3. E. Oliver, "A survey of platforms for mobile-networks research," *SIGMOBILE Mobile Compute Comm. Revised,* **12**; 56–63 (2008)
4. M. Al-kistany, Sumi, "Adaptive wireless thin client model for mobile -computing," *Wireless Comm. Mobile-Computing.,* **9**; 47–59 (2010).
5. M. Satyanarnynnan, "*Mobile-computing," Computer,* **26**; 81-82 (1993).
6. D.E. Bakken and M. API, Middle ware, 2001.
7. H.E. Bal, J.G. Steiner, and A.S. Tanenbaum, "Programming languages for distributed computing systems (DCS)," *A C M Compute Surveys.,* **21**; 261–322 (1989).
8. W. Emmerich, "Software engineering and middle-ware: a roadmap," ICSE '00: Proc. of the Conference on The Future of Software Engineering, New York, NY, USA: A C M, 2000, pp. 117–129.
9. A. Uribarren, J. Parra, J.P. Uribe, M. Zamalloa, and K. Makibar, "Middle ware for Distributed Services and Mobile Applications," Inter Sense '06: Proceedings of the 1st International conference on Integrated internet ad-hoc and sensor-networks, New York, NY, USA: A C M, 2006.
10. T. Phan, R. Guy, and R. Bagrodia, "A Scalable, Distributed Middleware Service Architecture to Support Mobile Internet Applications," WMI '01: Proceedings of the 1st workshop on Wireless mobile internet, New York, NY, USA: A C M, 2001, pp. 27–33.
11. P. Bellavista, A. Corradi, R. Montanari, and C. Stefanelli, "A mobile computing middle ware for location and context-aware internet data services," *A C M Trans. Internet Technology,* **6**; 356–380 (2006).
12. Fielding R., "Architectural Styles and the Design of Network-based Software Architectures," Ph. D. Dissertation, University of California, Irvine, California, USA, 2000.
13. Feda AlShahwan, Evaluation of Distributed SOAP and RESTful Mobile Web-Services, *International Journal on Advances in Networks and Services,* **3** (3-4) (2010).
14. R. Deters, "SOA's Last Mile-Connecting Smartphones to the Service Cloud," Cloud Computing, IEEE International Conference on, 80-87 (2011).
15. CloudSim: A Framework For Simulation And Modeling Of Cloud Computing Infrastructures And Services : http://www.cloudbus.org/cloudsim/
16. Performance testing tool Web UI, JSON, CSV data, LoadUIWeb Tool downloaded from http://loaduiweb.org/
17. Valid JSON data using online tool access from http://jsonlint.com/
18. JSON data parser testing valid data using online tool access from http://jsonviewer.stack.hu/
19. M. Singh, K. S. Dhindsa, "Enhancing Interaction between Smartphones and Web Services on Cloud for Improved Bandwidth and Latency", *IJCSMC*, **2**; 177-185 (2013).
20. Google App Engine accessed via https://developers.google.com/appengine (2014)