Available Online at www.ijcsmc.com

International Journal of Computer Science and Mobile Computing

A Monthly Journal of Computer Science and Information Technology

ISSN 2320–088X

IJCSMC, Vol. 2, Issue. 4, April 2013, pg.177 – 185

RESEARCH ARTICLE

Enhancing Interaction between Smartphones and Web Services on Cloud for Improved Bandwidth and Latency

Mandeep Singh[1], Kanwalvir Singh Dhindsa[2]
[1]Assistant Professor, Chandigarh Engineering College, Landran, India
[2]Associate Professor, Baba Banda Singh Bahadur Engineering College, India

[1] mandeepsingh22@yahoo.com; [2] kdhindsa@gmail.com

Abstract— As cellular network infrastructures are improving day by day; they are becoming the ideal clients to access the any Web resources, especially Internet Based Services. However, Smartphones have certain limitations in connecting smartphone based devices to existing Internet based Services. This paper mainly focuses on focuses on the following limitations: connection loss, bandwidth, latency, and limited resources. This paper implements a platform independent architecture for connecting smartphones to the existing Internet based Services. The architecture includes a cross platform design of smartphone based service client and a middleware for increasing the interaction between mobile clients and Internet based Web Services. The architecture can be deployed on Cloud Platforms, like CloudSim and Google App Engine to enhance the scalability and reliability.

Key Terms: - Smartphones; CloudSim; XML; PHP; Apache; Application Server; Recess PHP Platform; Representational State Transfer; RRJSON

I. INTRODUCTION

Smartphones are expected to increase gradually from current users. As cellular network infrastructures continuously improve, their data transmission becomes increasingly available and affordable, and thus they are becoming popular ways to access the Internet Based Services, especially the Web Services that are also available in the cloud. Today, smartphone devices like iPhone, and Android, have included applications that consume the web services from popular websites, such as YouTube, Msn and eBay. The share of android in smartphone market is 46% and iPhone is 35%.

However, there are problems in connecting smartphones to existing Internet based Services. Firstly, the Internet Based Services need to provide optimization for mobile clients. For example, the size of the Web Services messages needs to be reduced to fit the bandwidth of mobile clients. Secondly, smartphones have to adapt to different kinds of Web Services, for example, SOAP and RESTful Services. Figure 1 shows a smartphone accessing Web Services. This paper investigates how Cloud Computing [1] can help smartphones connect to existing Internet based Services.

II. PROBLEM DEFINITION

Accessing Internet Based Services from a smartphone based client is different compared to the standard web services scenarios, due to these factors environment. A survey done by Earl et al. [2] evaluated how well the current smartphones including Android, iPhone, Symbian (S60), and Windows Mobile, support the concept of mobile network based research. According to the survey, all of these mobile platforms have certain limitations.

- Smartphones have limited resources like Processing power, screen size etc.
- The communication between client and service is established through wireless network.
- Existing Web Services in the Cloud do not support smartphones.

Fig 1: Smartphone Accessing the Internet Web Services

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

a) Loss of connection Problem: Since the smartphone based devices are not stable and due to the mobility of the smartphones and the wireless network setup, smartphones can be temporarily removed from the previous connected network and later may join network.

b) Bandwidth/Latency Problem: Cell networks have a very limited bandwidth and are often billed based on the amount of data transferred. However, even a simple SOAP message often contains a large chunk of XML data, which consumes a lot of bandwidth and the transmission can cause major network latency. In addition, the SOAP message contains mostly XML tags that are not all necessary for mobile clients.

c) Limited resources problem: Smartphone clients are normally "thin clients" [3] with a less processing power.

They also have limited screen size and computational power. These shortcomings are only due to mobility [4].

III. ARCHITECTURE FOR SMARTPHONE DEVICES

Middleware Architecture [5] is mainly used in Distributed Computing system. Distributing Computing Systems [6] "consist of multiple processors that do not share primary memory, but sending messages over network". Mobile clients are distributed computers that connect to the middleware. In Emmerich's paper [7], he defined four requirements for general middleware.

Network communication: Hosts who need to communicate with each other involves some transport layer (TCP and UDP) and marshaling, a process of converting data structure to transferable format.

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

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.

Heterogeneity: Components in a distributed system can be implemented with different languages and deployed on different platforms. Thus, the design needs to consider a heterogeneous environment.

Middleware Architecture is often used to extend functions for thin clients, like mobile devices. Uribarren et al. [8] proposed a middleware for adaptation in mobile environments. The proposed middleware hides the complexity of deploying ubiquitous applications. Applications are automatically moved between different platforms.

When designing distributed systems, scalability should be the primary concern. Rajive et al. [9] did research on investigating scalable middleware to support mobile Internet applications

The proposed middleware solutions for Smartphone based devices mostly focus on application and content adaptation. Coordination, scalability, reliability, and heterogeneity are four fundamental requirements for general middleware as well as middleware for mobile device [10]. Scalability can be achieved with distributed middleware. 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 of the middleware.

IV. PROPOSED ARCHITECTURE OF MIDDLEWARE

The middleware that is proposed will act as proxy that is hosted on the Cloud platforms which provide mobile clients access to Cloud services. The middleware architecture will improves interaction between mobile clients and Cloud Services, for example, adaptation, optimization and caching. The middleware also provides extended functions to mobile clients. In general, the architecture enhances the functionality, reliability and compatibility of the interaction between smartphones and Cloud Services.

In order to overcome the problems listed 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 middleware. When the mobile clients are not able to connect to the middleware, 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 middleware transforms Simple Object Access Protocol to a much light- weight format RJSON through RESTful Internet Web Services. Transferring SOAP to light-weight protocols, like RESTful, reduces processing time as well as the size of the messages [11].

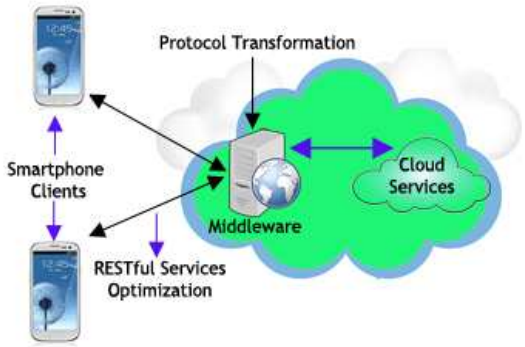

Fig 2: Architecture for Smartphones Middleware

Result optimization: Result optimization reduces the size of the service results, thus reduces the bandwidth used to interact with internet Services. The middleware converts the format of service results from XML to RJSON and removes unnecessary data from the original service result. Less data transferring also reduces network latency.

V. EXPERIMENTAL SETUP

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 main features. The architecture consists of three parts, the mobile clients, the middleware and the Cloud services. Since Cloud services are usually controlled by service providers, the middleware performs all the necessary adaptation to the mobile clients.

A. Middleware Architecture

The middleware is responsible for consuming the Cloud Services whether they are SOAP or RESTful services like RJSON and delivers the service result to the smartphone. On the smartphone, users can define Web Services and later execute the pre-defined Web Services.. The middleware provide RESTful interface for the mobile clients. Figure 4.2 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 middleware, the follow steps are involved in the middleware.

- The Smartphone sends a HTTP GET request with an identifier of a Web Services to the middleware.
- The middleware deals with interactions to the Web Services (and generates SOAP client if necessary).
- The middleware extracts (RJSON or XML parsing) the required service results from the original service result and form a new service results in RJSON format.
- The middleware stores a copy of result with the service ID in the database and returns the optimized result to the mobile client

.

B. Mobile Client implementation on middleware in Smartphones

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, location service and file system access.

To verify the smartphone based client design, I have integrated the design with an existing online application in php application for Punjabi newspaper portal, called Sadelok Newspaper as shown in Figure 3. The application is re-implemented with the mobile client design on smartphone. Using the application, the viewers can access the Web Services of the portal which is hosted on cloud through the smartphones.

The client application can be divided into three layers, User Interface (UI), controller and cache manager. The UI layer has two implementations, native UI and embedded browser UI. Figure show how they look like on the device. Figure show the architecture of both implementations. The controller is the key coordinator among the UI, middleware, and cache manager. The controller creates the UI and gets data from the RESTful client or cache manager. If network connections are not available, the controller passes cached data to the UI components.

Fig 3: Layout on Android Browser of Sadelok Application

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 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 middleware. When the data arrives, the UI gets updated through the callback. With this model, the native UI can be updated as soon as the data changes. The embedded browser needs to wait for the data to arrive, because the native library cannot receive a JavaScript callback. The embedded browser also cannot be updated automatically when the data changes.

The entire middleware application is hosted on a Google Application Engine GAE uses the Services oriented architecture. The middleware architecture is implemented as a Core PHP Web application. The application uses the RESTful Internet Service interfaces to mobile clients, since RESTful web services are more suitable for mobile devices [12]. Because the middleware uses RESTful and RJSON API of core PHP libraries. It has to be deployed on Apache Web server container.

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

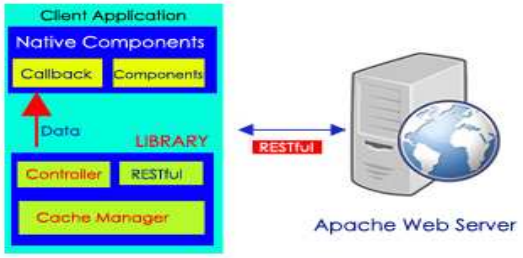

Fig 4: Native Smartphone Implementation

The middleware still has a RESTful interface to mobile clients, but the Google Application Engine platform itself is a Web application server which can only handle server requests. The Apache HTTP client library is not supported on the Google Application Engine, due to the restrictions from the provider. Instead the middleware constructs and sends HTTP requests through the URL fetch service which implements the PHP RESTful Framework interface.

To enhance the interaction between mobile clients 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.

To use the Cloud platform as a way to improve scalability and reliability of the middleware

- The middleware can be implemented on CloudSim and Google Application Engine.
- Cloud platform improves the scalability and reliability of the middleware.

VI. CASE SCENARIOS

The middleware is implemented as a standard PHP Web Application. The middleware uses the *PHP* 5.4.13 standard, so it can be deployed in most Apache server containers. The PHP RESTful Framework of Recess [14] interface implements the pseudo-Restful based web service on a RESTful PHP Framework / MySQL stack.). The middleware also uses the MySQL database to interact with the MySQL Community Server 5. In the following experiments, the middleware is deployed in three platforms, Webserver with Apache Platform, virtual machine and Google Application Engine. Because Application Engine uses Google's internal structure, its hardware specification is not known.

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 Tsung is used as

a load generator. The simulator for the cloud is CloudSim [13]. Tsung is responsible for generating and sending HTTP requests to the middleware in a specified rate. Tsung calculates the mean of response times every 10 seconds based on its log file. The load generator runs on the standard server for the Sadelok News Portal.

The mobile client is implemented on Android.. The Android device used is using Android Version 4.1. The build-in Apache HTTP client is used to send HTTP request. Both of them are connected to the Internet through wireless 802.11g. The client uses the IO libraries from PHP and in build Browser Support.

A. Consuming Sadelok Web Services through the Middleware

This experiment compares the overhead associated with different WS interactions. Sadelok News Portal provides both SOAP and RESTful WS interfaces for their News service. Their RESTful WS return result in either XML or RJSON format. The tested WS is "id", which returns an article under that id. The maximum list size is 100 and the keyword used is "Android". The middleware is run on the standard server.

The Sadelok_id":"51 segment of J SON and XML result were taken for the tests for the Web Services.

The size of the J S O N result is about 121 KB and the size of the X M L result is about 170 KB. The load generator sends HTTP request at the rate of 1 request per 10 second (exponential distribution, mean 0.1request/s), so the middleware does not overload.

B. Enhancing Interaction between the Client and Middleware over Cloud

- *Consume Sadelok RESTful Web Services directly with RJSON result.*
- *Consume Sadelok RESTful Web Services directly with XML result.*
- *Consume Sadelok RESTful Web Services through the middleware with RJSON result. The middleware forwards the complete result. (no parsing involved)*
- *Consume Sadelok RESTful Web Services through the middleware with XML result.*
- *Consume Sadelok RESTful Web Services through the middleware with RJSON result. The middleware returns the optimized result in RJSON format.*

.

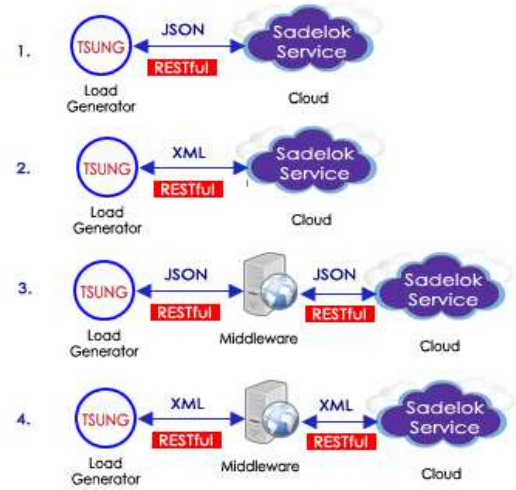

Fig 5: Assessing Sadelok Application through smartphone

- *Consume Sadelok RESTful Web Services through the middleware with XML result. The middleware returns the optimized result in RJSON format.*
- *Consume the sadelok portal through the SOAP WS through the defined could middleware.*

Direct accessing versus Accessing through middleware: Comparing the experiments in Figure 5 and Figure 6 in experiments 2,1, 4 and 5, whether the Sadelok services id return RJSON or XML, the middleware the overhead is mainly caused by network latency between the client and middleware.

Java Script Object Notation vs. Extensible Markup Language

Compare the RJSON experiment in Fig 5 and XML experiments in Figure 6 interactions utilized by RJSON have less response time than XML. It is because the XML messages are very large which causes transmission delay of packets therefore slowing down the system.

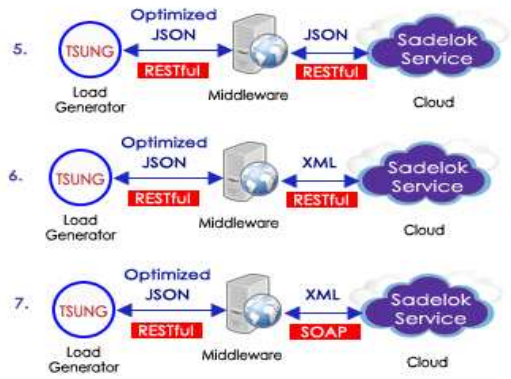

Fig 6: Assessing Sadelok Application through smartphone

Optimized versus Non-optimized Protocols:

Compare the results of experiments in Figure 5 i.e. experiments no 2 & 3, result optimization with RJSON reduces the response time a little. It certainly adds a little overhead on the response time. Because the middleware does not do any processing of the service results,

Comparing the experimental results of 4 and 5 result optimization with XML adds a little overhead .The middleware adds overhead with parsing and extracting data from the original result.

Table 1. Comparison of SOAP, XML and Recursive Java Script Object Notation Response Time

S.No	Protocol Used	Lowest (ms)	Avg (ms)	Highest (ms)
1.	RJSON direct	0.051	0.093	0.177
2.	RJSON **M**iddle**w**are with RJSON.	0.048	0.109	.238
3.	RJSON middleware optimized.	0.056	0.11	0.277
4.	XML direct	0.055	0.111	0.317
5.	XML middleware	0.058	0.128	0.388
6.	XML middleware optimized	0.06	0.187	0.469
7.	SOAP middleware	0.117	0.299	0.534

RESTful versus SOAP Protocols:

As the experiment 7 indicated, Simple Object Access Protocol has higher response times than the rest of the experiment with RJSON. SOAP is verbose protocol which means more data needs to be transferred.

Fig 7: Concurrent requests for RJSON and SOAP

Multiple concurrent Requests comparison for RJSON and SOAP

As shown in Figure:7 multiple request from an smartphone were thrown at the middleware and major parameters that were taken in consideration were memory consumption ratio, Response time ratio, Message Length ratio. Processing SOAP requires comparatively heavy-weight.

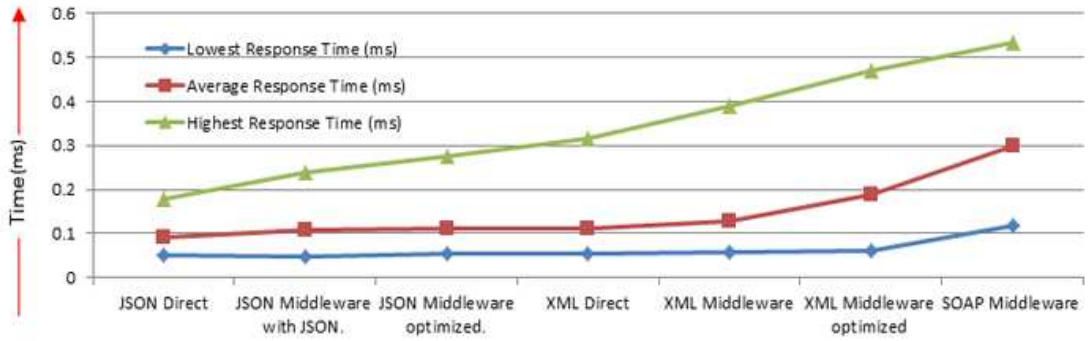

Fig 8: If necessary, the images can be extended both columns

Messages and parsers and they have more response time. We can see that the communication delay is directly proportional to the size of transferred message, which is certainly larger for SOAP than RJSON.

Parsing Time and Bandwidth Comparison of SOAP and RJSON:

JavaScript Object Notation and Extensible Markup Language are two widely used formats for transferring the Internet web services messages. Major providers of cloud give a preference of using either one of them. I have used the Java Document object model parser for the Extensible markup language and RJSON parser for RJSON and I have accessed sadelok.com which have returned 25 requests in both XML and RJSON. We can see from the Table 2 that the message which uses XML is larger and slower as compared to one which used the RJSON.

	Format	**Message size (KB)**	**Average parsing time (ms)**
Android	XML	56.2	386.4
Android	RJSON	34.1	48.08

Table 2 Parsing times for the XML and RJSON message over 25 independent requests from Android

First, comparing the size, the size of XML chunk taken is 56.2 KB and 34.1 KB for RJSON. To represent the same information, the XML format requires more bandwidth. Second, considering the parsing time, parsing XML message is more resource consuming than parsing RJSON message on both Android and Blackberry. The slowness is not only due to the size, but also the complexity of parsing. Finally, RJSON format also has very stable parsing time. However, it is very difficult to represent complex data structure in RJSON format.

VII. CONCLUSION

As service consumers, smartphones basically have unique properties like they are small and portable. They are personal devices with various sensors. However, these smart phones have limitations, for example, small bandwidth, loss connectivity and less process power. On the hand, the existing services are normally designed for stationary clients. For example, SOAP is a protocol which involves a lot of XML 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 proposed mobile client design is mobile platform independent. The mobile client provides an interface for users to define services and consume them through the middleware. It interacts with the middleware through RESTful WS interface. The mobile client has been implemented on Android platform. The smartphone based design involves native as well as browser based applications. For better compatibility, the interface can be implemented on embedded browser with HTML, CSS and JavaScript, while the actual client component is implemented in platform dependent language, the server side scripts can run on the application server..

The middleware provides a medium for the smartphones to access the Cloud Services. To support existing SOAP web services, the middleware does the protocol transformation from the SOAP to RESTful Web services and XML message to RJSON format. The middleware also provides result optimizations which extract the required data from the original service results.

REFERENCES

[1] M.A. Vouk, "Cloud computing: Issues, research and implementations," Information Technology Interfaces, 2008. ITI 2008. 30th International Conference on, 2008, pp. 31–40.
[2] E. Oliver, "A survey of platforms for mobile networks research," SIGMOBILE Mob. Comput. Commun. Rev., vol. 12, 2008, pp. 56–63.
[3] M. Al-kistany, Sumi, "Adaptive wireless thin-client model for mobile computing," Wirel. Commun. Mob. Comput., vol. 9, 2010, pp. 47–59.
[4] M. Satyanarnynnan, "Mobile computing," Computer, vol. 26, 1993, pp. 81-82.
[5] D.E. Bakken and M. Api, Middleware, 2001.
[6] H.E. Bal, J.G. Steiner, and A.S. Tanenbaum, "Programming languages for distributed computing systems," ACM Comput. Surv., vol. 21, 1989, pp. 261–322.
[7] W. Emmerich, "Software engineering and middleware: a roadmap," ICSE '00: Proceedings of the Conference on The Future of Software Engineering, New York, NY, USA: ACM, 2000, pp. 117–129.
[8] A. Uribarren, J. Parra, J.P. Uribe, M. Zamalloa, and K. Makibar, "Middleware for Distributed Services and Mobile Applications," InterSense '06: Proceedings of the first international conference on Integrated internet ad hoc and sensor networks, New York, NY, USA: ACM, 2006.
[9] T. Phan, R. Guy, and R. Bagrodia, "A Scalable, Distributed Middleware Service Architecture to Support Mobile Internet Applications," WMI '01: Proceedings of the first workshop on Wireless mobile internet, New York, NY, USA: ACM, 2001, pp. 27–33.
[10] P. Bellavista, A. Corradi, R. Montanari, and C. Stefanelli, "A mobile computing middleware for location- and context-aware internet data services," ACM Trans. Internet Technol., vol. 6, 2006, pp. 356–380.
[11] Feda AlShahwan, Evaluation of Distributed SOAP and RESTful Mobile Web Services, International Journal on Advances in Networks and Services, vol 3 no 3 & 4, year 2010.
[12] R. Deters, "SOA's Last Mile-Connecting Smartphones to the Service Cloud," Cloud Computing, IEEE International Conference on, 2011, pp. 80-87.
[13] CloudSim: A Framework For Modeling And Simulation Of Cloud Computing Infrastructures And Services : http://www.cloudbus.org/cloudsim/
[14] REST and Web Services: In Theory and In Practice Paul Adamczyk, Patrick H. Smith, Ralph E. Johnson, Munawar Hafiz

