Journal of Physics: Conference Series

PAPER • OPEN ACCESS

Mobile Cloud Computing with SOAP and REST Web Services

To cite this article: Mushtaq Ali *et al* 2018 *J. Phys.: Conf. Ser.* **1018** 012005

IOP Conf. Series: Journal of Physics: Conf. Series **1018** (2018) 012005 doi:10.1088/1742-6596/1018/1/012005

Mobile Cloud Computing with SOAP and REST Web Services

Mushtaq Ali [*,1], Mohamad Fadli Zolkipli [2], Jasni Mohamad Zain [3], Shahid Anwar [4]

[1,2,4] Faculty of Computer Systems & Software Engineering. Universiti Malaysia Pahang Gambang, 26300, Malaysia
[3] Faculty of Computer & Mathematical Sciences, Universiti Teknology MARA (UiTM), 40450 Shah Alam, Selangor, Malaysia sunnyswa@hotmail.com

***Abstract*—**Mobile computing in conjunction with Mobile web services drives a strong approach where the limitations of mobile devices may possibly be tackled. Mobile Web Services are based on two types of technologies; SOAP and REST, which works with the existing protocols to develop Web services. Both the approaches carry their own distinct features, yet to keep the constraint features of mobile devices in mind, the better in two is considered to be the one which minimize the computation and transmission overhead while offloading. The load transferring of mobile device to remote servers for execution called computational offloading. There are numerous approaches to implement computational offloading a viable solution for eradicating the resources constraints of mobile device, yet a dynamic method of computational offloading is always required for a smooth and simple migration of complex tasks. The intention of this work is to present a distinctive approach which may not engage the mobile resources for longer time. The concept of web services utilized in our work to delegate the computational intensive tasks for remote execution. We tested both SOAP Web services approach and REST Web Services for mobile computing. Two parameters considered in our lab experiments to test; Execution Time and Energy Consumption. The results show that RESTful Web services execution is far better than executing the same application by SOAP Web services approach, in terms of execution time and energy consumption. Conducting experiments with the developed prototype matrix multiplication app, REST execution time is about 200% better than SOAP execution approach. In case of energy consumption REST execution is about 250% better than SOAP execution approach.

Key Words: Execution Time, Battery Consumption, Soap Webservices, Rest Webservices

1. Introduction

Over the years, businesses interacted using ad hoc methods to get advantages of the basic network infrastructure [1]. however, Web services emerging application to application systematic approach for interaction which basis on existing protocols and XML standards. Web services can be defined as to expose the information system functionalities and make them available to public through Web technologies [2]. The term Web services in typical meaning is to serve something in Web and it defines the way of interacting the Web applications to each other using the WSDL (Web Services Description Language), UDDI (Universal Description Discovery and Integration), XML (Extensible Markup Language) and SOAP (Simple Object Access Protocol) which are open standards in a network protocol backbone. XML is used to distribute the computation over network by tagging the data, SOAP is a carrying protocol used to carry the data. WSDL describes the services available such as location of the

services and methods of the services. UDDI is used for the discovery of suitable service of a particular task.

Web Services can also be defined as, an application accessible to other application over the Web. It is initially designed for application to application interaction regardless of the operating environment and developed where everyone agrees on some set of rules by which the interaction of Web Services takes place. As mobile devices by design carries limitations such as limited battery timing, slow processing, limited bandwidth [3, 4]. The similar concept can be implemented in mobile devices for computational offloading in order to address these limitations.

Amongst many other approaches of computational offloading one such approach is to utilize the idea of Web Services where the computational load is to delegate to remote servers for execution. Instead to have the clone of a mobile device or to delegate the whole application based on Virtual Machine Migrations, the mobile applications should be served as Web Services and the devices can directly approach to access those Web services without any management and overhead hurdles [5].

Each mobile application considered to be logically divided into two parts, one local part to reside in mobile device such as UI and the other should reside at remote server as a Web Service such as decoding or complex computational part. The existing mobile Web services are designed simply to serve the consumer on their mobile phones [6]. The intension of this work is to provide a self-configuring and self-provisioning service to address the complexity of traditional computational offloading techniques which are resources intensive and energy draining [7]. For the first time, we going to simulate and test the concept of Web services as a computational offloading solution for mobile computing in our lab environment. One closely related work done by [8], where they focused on to choose a best implantation structure for Web services amongst SOAP and REST in terms of services utilization. However, we differ here as our focus is to minimize the communication overhead by testing the developed prototype in our lab where we installed a middle layer next to remote server. It is to reduce the transmission distance to a single hop in order to address the limitations of mobile devices such as execution time delay and energy consumption through computational offloading.

To implement the concept of offloading through Web services, two types of load carrier protocols existed [9]. One SOAP (Simple Object Access Protocol), which is a standard object-oriented technology relying on the existing protocols such as SMTP and HTTP for messages negotiations and delegations. It uses XML format for the transmission of data through Web [10]. On the other hand, REST (Representational State Transfer Protocol) used for building up RESTful Web services. REST is categorized a resources oriented technology and developed by Roy Fielding [11] and it is an architectural style of World Wide Web which communicates over HTTP protocols. Uniform Interfaces which operates on URL resources, provides basis to RESTful Web. Both REST and SOAP are used to implement Web Services, however, each one has its own discrete features. We claim, that RESTful Web services carries a lightweight nature which are best fit for mobile computing and to address the limitations better than SOAP Web services.

Rest of the paper structured as, Section II provides the architectural comparison of SOAP and REST. Section III presenting the implementation methods of both the Services. Section IV discusses the results. Section V gives the future work and concludes the paper.

2. Architecture of Soap and Rest

A general overview of a SOA (Service Oriented Architecture) common both for SOAP and REST given in Fig. 1. WSDL a language used to create services description for a Service provider in the Service Directory. Services Directory which is used by the services providers to publish the services they offer. SOAP / REST is used as a protocol / architectural style by both the Services Providers and Consumers then to talk to the Services Directory. Next, based on the description given in Service Directory an XML message (XML a standard language used where both software can communicate in the same language to each other) is formed and then the Services Consumer requests to Service provider for permitting those particular Services. Both, REST and SOAP carrying their explicit pros and cons. For secure services utilization SOAP is considered the perfect solution, while for scalable and lightweight nature REST is the right choice [12].

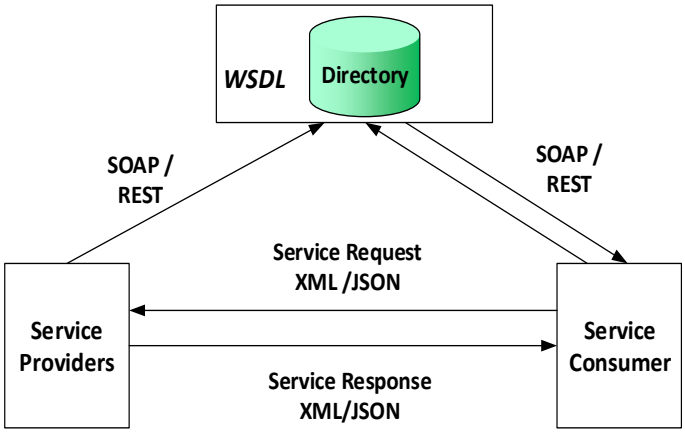

Figure 1: Service Oriented Architecture

Table 1. Presents a glance of comparison of both the REST and SOAP. Based on SOAP and REST, the web services having their own distinct features that turn the web service to a more or less suitable for a certain type of application and environment.

RESTful Web Services using a Unique URI (Uniform Resource Identifier) to describe the Resource. Using HTTP four basic operations such as GET, PUT, POST and DELETE are performed, while SOAP using a SOAP envelop which carriers the details of operations such as, starting of message, end of message, and the actual message in the form of XML.

A sample of SOAP message's envelope is:

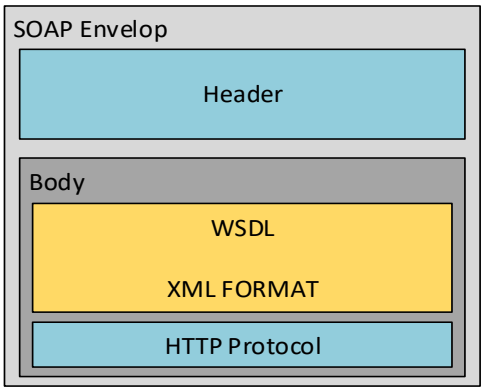

Figure 2: SOAP Envelope

Sample SOAP Message

```
<?xml version="1.0?>
<S:Envelope xmlns:S="http://
schemas.xmlsoap.org/soap/envelope/"
xmlns:SOAP-ENV="http://
schemas.xmlsoap.org/soap/envelope/">
…….
Messge information goes here
…….
</S:Envelope>
```

Figure 3: Sample SOAP Message

Every SOAP Envelope is the mandatory part of a SOAP message. It is the first root element which decides what should be sent to whom and what should be the rout of transmission. Each SOAP envelope having one head and one body element as shown in Fig. 2. The body of each SOAP envelope extricates between a SOAP request and SOAP Response. A typical SOAP request and SOAP response of a SOAP message shown in Fig 3. The message body contains the Name-Space of the Service and then two arguments arg0 and arg1 to the Server in a SOAP Request. Both the arguments contain 2, 2 which sent to the Service where both the arguments are taking as an input for a matrix multiplication service which generates two random matrices of size 2x2.

SOAP Request

```
<?xml version="1.0" encoding="UTF-
8"?><S:Envelope xmlns:S="http://
schemas.xmlsoap.org/soap/envelope/"
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/
soap/envelope/">
    <SOAP-ENV:Header/>
    <S:Body>
        <ns2:MatrixMulti xmlns:ns2="http://
javabrains.ali.org/">
            <arg0>2</arg0>
            <arg1>2</arg1>

        </ns2:MatrixMulti>
    </S:Body>
</S:Envelope>
```

Figure 4: SOAP Request

SOAP Response

```
<?xml version="1.0" encoding="UTF-
8"?><S:Envelope xmlns:S="http://
schemas.xmlsoap.org/soap/envelope/"
xmlns:SOAP-ENV="http://
schemas.xmlsoap.org/soap/envelope/">
<SOAP-ENV:Header/>
<S:Body>
<ns2:MatrixMultiResponse
xmlns:ns2="http://javabrains.ali.org/">
<return>
<result>
<item>1625.0</item>
<item>2200.0</item>
</result>
<result>
<item>1232.0</item>
<item>2241.0</item>
</result>
</return>
</ns2:MatrixMultiResponse>
</S:Body>
</S:Envelope>
```

Figure 5: SOAP Response

REST	**SOAP**
Representational State Transfer	Simple Object Access Protocol
Suitable for Point to Point Communications	Designed for Distributed Computing
Minimize middle-wares, only HTTP needed	Needed middleware /tooling support
URL typical carrying the operation to be performed	Content of messages decides the operations
Security Issues	Secure
Not details description of WSDL services	Well-defined services structure
No Payload constraints	Payload need to comply with the SOAP policy
Built-in Error Handling	No Error Handling
Relying on HTTP transport model	SMTP and HTTP both are reliable layers for transmitting SOAP
Simple and Light	Complex and Heavy
An Architectural Style	A Protocol
SOAP can't use REST	REST can use SOAP as it is an architectural style and can use any protocol like SOAP, HTTP
Services Interface needed for exposing the business logic	URI needed to expose business logic
JAX-WS is the API in JAVA used	JAX-RS is Java API
Needed high bandwidth and resources	Needed less bandwidth and resources
Own Security defines	Inherit from underlying layer
Permit XML format only	Support different formats such as HTML, JSON, XML and Plain-text

TABLE 1. A COMPARISON OF SOAP AND REST

Both the randomly generated matrices are multiplied in the service and the resulted matrix is returned in the response as shown in Fig. 4. The XML tags <ns2:MatrixMulti…. </ns2:MatrixMulti> shows the SOAP request in Fig. 3 while <ns2:MatrixMultiResponse… </ns2:MatrixMultiResponse> shows the SOAP response in Fig 4. The resulted matrix should be of the same dimension 2x2. The returned Items are 1625, 2200 in the first row and 1232, 2241 are in the second row of the resulted matrix.

3. Implementation

The argument we test here that, offloading in mobile cloud using SOAP web services is heavyweight and resources intensive. SOAP needed more Turnaround Time (TT) and mobile battery consumption than REST. For that reason, to empirically test the intensiveness of SOAP against REST we planned to conduct the experiments in real lab environment.

In addition, a Server-Socket connection used to implement mobile Web serveries. There are five building blocks in Mobile Web Services: *Web Service-Servlet:* is used to deploy new services to mobile device and also invokes the demanded service. *HTTP Listener:* It has to listen to incoming requests coming through Server-Socket and accept incoming requests from client. *Request Handler:* It is responsible for processing the requests. The way this block work is different in both kind of frameworks. For instance, In SOAP-base framework the request handler unpacks the HTTP POST requests and extract the inside SOAP envelope then it dispatches the same envelope to message parser.

Conversely, request handler for RESTful Web Services extract directly the HTTP request and send it to message parser. Parser Module: it gets the required information of a Web Service such as name, URL and parameters of a Service. The extracted information is then send to Service-Servlet. This is performed different in two frameworks. In SOAP-based Web services, the parser de- serializes the object and by using KSOAP2 and kXML2 (Open source APIs for SOAP parsing) maps the data types to Java Objects. In RESTful Web Services, we used to create our own manual string manipulator which do Parsing. Response Composer: It interpret the results and then send it back to client.

Experiment Setup:

The experimental emulation environment planned. The setup composed of remote surrogate machine running Microsoft Windows 10 Professional 32-bit operating system with Intel® core(TM) i7-3770 @ 3.40GHz speed and 4.0 GB RAM capacity and a Mobile client device Samsung Galaxy A5 runs Android 5.0.2(Lollipop) OS with Quad Core 1.2 GHz Cotex-A53 Processor, 16GB memory 2GB RAM and 2300mAh battery. A D-Link wireless Wi-Fi modem / Access Point with a physical layer data rates 54 Mbps used to connect the remote server machine and mobile devices. Mobile device accesses the wireless network via Wi-Fi wireless network connection of radio type 802.11g, with the available physical layer data rates of 54Mbp.

Two components of the proposed framework developed. One for Client side running at mobile device while the other runs at surrogate servers. The Server-side application (Web service) deployed by configuring a Glassfish server at surrogate using Web Services Application tools in Eclipse. Also, Android Developers (Java based Android Software development tool kit) deployed for the development of client-side application in Intellij IDE 15 Community environment. (Android Debug Bridge) ADB plugin used to develop the prototype android application.

Two porotype applications named SOAP-app and REST-app developed which required to get two numeric integer values as arguments, pass the arguments to Web Server. Web Server randomly generates two matrices of the same input size and multiply both. The resulted matrix gets back to client device. The execution time and battery consumption tested for both. A stop watches Time Left used to read TT while Power Tutor to read the energy consumption.

4. Result and Discussion

As the matrices multiplication is a computational intensive task, a porotype developed to multiply two randomly generated matrices ranged between 300*300-400*400. Each computational intensity varies from the previous one with an addition of 10. The total 10 different computational intensities ran. Though, different parameters such as, fluctuating bandwidth and loss of network signals, remote server

load and so on, may affect the result of executing a task at remote servers, therefore each intensity iterated for 5 times in order to eliminate ambiguity in the observations. For further validation mean value of the collected 5 sample values calculated. The execution time while executed the prototype through SOAP web service is shown in Figure 6.

The execution of lowest intensity 300*300 executed in approx. 30 seconds while the highest intensity 400*400 executed in 140 seconds. Similarly, the execution of all the intensities through REST web services is as shown in Figure 7, the lowest intensity 300*300 executed in approx. 20 seconds while the highest intensity in 38 seconds.

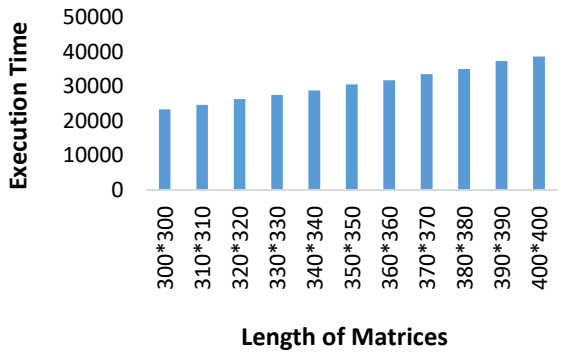

Figure 6: Execution Time of SOAP Web services

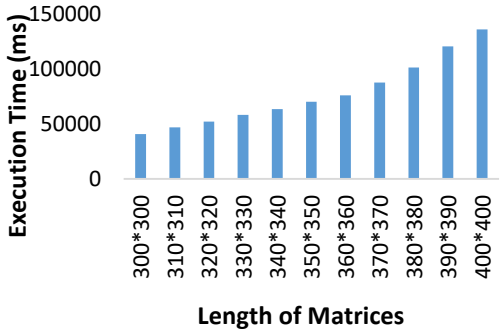

Figure 7: Execution Time of REST Web services

The comparison of both the services as shown in Figure 8 illustrates that for the same data to transmit and execute with the same lab test using same computing devices needed more than double execution time to execute by SOAP web services as compare to REST web services.

The execution time of SOAP based Web services is highest than the RESTful Web services and it is due the heavyweight process of un-wrapping the SOAP envelope for each message request and then the SOAP object de-serializes and mapping the data types of actual messages (XML based) into the Java objects. Conversely, the execution time of RESTful Web services is almost half of the execution time of SOAP web services and the reason is, the parsing of heavyweight process eliminated by a lightweight parsing procedure and to extract the hidden messages. As, REST no need to use additional libraries for parsing SOAP response and utilizes less resources compare to SOAP. Also, SOAP needed to have a contract between remote system and client device in order to make the communication secure and effective it need to define schemes and implement policies to establish a secure and smooth flow of data.

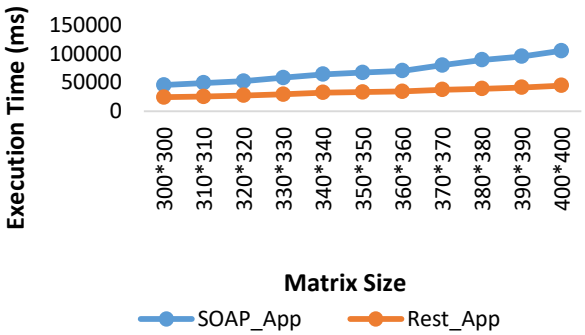

Figure 8: Comparison of ET between SOAP and REST Web services

Similarly, the energy consumption cost of the same input arguments of executing through SOAP web services is shown in Figure 9. The energy consumption cost in Joule (J) of the lowest computational intensity 300*300 is approx. 30J while the same for the highest intensity 400*400 is 60J.

On the other hand, the execution of computational intensities through REST web services gives the results as shown in Figure 10. The energy consumption cost of the lowest intensity 300*300 is approx. 10J while the same of highest intensity 400*400 is approx. 20J. The comparison of both the services as given in Figure 11 shows that executing the same intensity by the both the services, the SOAP web services hit the battery more compare to the REST services.

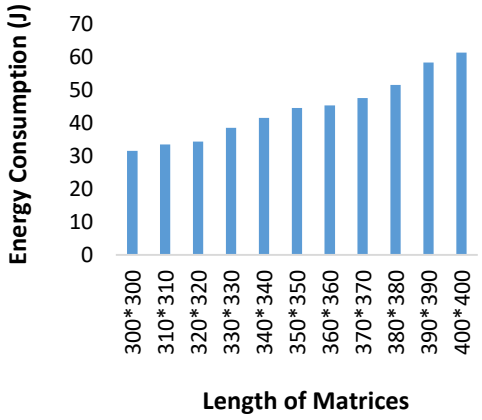

Figure 9: Energy Consumption Cost of SOAP Web Services

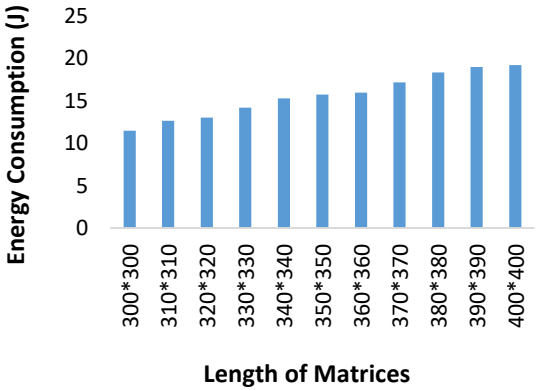

Figure 10: Energy Consumption Cost of REST Web Services

By executing the same intensity in order to identify the energy intensive service in two, it is clear from Figure 11 that SOAP Web Services drain power of mobile device more than double compare to REST services. The energy consumption of lowest intensity 300*300 of REST Web services starting from approx. 10J and reaches up to 18J, while the same of SOAP Web Services starting from 30J of the lowest computational intensity and reaches up to 60J for the highest one.

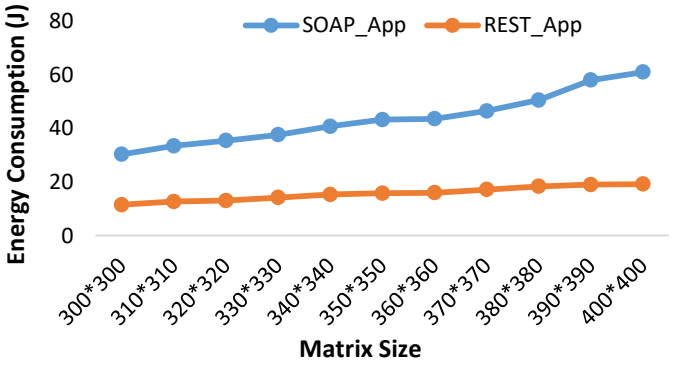

Figure 11: Comparison of EC between SOAP and REST Web Service

5. Conclusion and Future Work

Web utilities, where application to application interactions takes place are used to let electronic devices communicate with each other and initially Web Services designed to support interoperability of machine-to-machine interactions using World Wide Web. Combining Web services with mobile computing drives towards an approach where the limitations of mobile devices can be addresses. Web services basis on two types of technologies, SOAP implemented and REST implemented. Both, SOAP and REST carries their unique features which makes them applicable for different scenarios. In case of mobile computing, the intension of this work is to use Web services and delegate the computational intensive tasks to remote servers utilizing both type of Web Services, in order to identify that which

service is more lightweight and suitable for mobile computing, as a result to minimize resources consumption at mobile device.

To test the effective approach amongst SOAP and RESTful web services, we conducted the experiments in real scenario. Prototype applications SOAP-App and REST-App developed to empirically evaluate the intensity of each Web service in terms of execution time and energy consumption with mobile computing. The prototype applications consist of two components, one component run local whereas the next component run at remote server. The local component consist of UI which get input from users and sends to remote part of the app to execute. The experimental data collected in REST and SOAP both the approaches separately with 10 different computational intensities (300*300-400*400). For validation purposes, each intensity in both the scenarios iterated 5 times and a mean value calculated of execution time and energy consumption. The results show that due to heavyweight parsing procedure of SOAP Web services REST Web services are more effective to use in mobile computing. The execution time recorded in our experiments of REST Web services is nearly 200% better than SOAP while the energy consumption cost of executing same application of both scenarios, REST Web services 250% effective than SOAP Web services.

It is intended to explore more feature of both the Web services and consider maximum parameter which can affect mobile computing with Web services will be publish in the next extended article with complete details. Further, the experiments conducted with one single model of mobile device Samsung Galaxy A5, which will be extended to any different model of mobile device for expanding the scope of the research.

ACKNOWLEDGMENT

This work is fully funded by University Malaysia Pahang under Short Term Research Grant, ref: RDU 140391. I acknowledge the support of the Faculty of Computer Systems & Software Engineering, University Malaysia Pahang by providing a suitable environment for this research.

REFERENCES

[1] F. Curbera, M. Duftler, R. Khalaf, W. Nagy, N. Mukhi, and S. Weerawarana, "Unraveling the Web services web: an introduction to SOAP, WSDL, and UDDI," *IEEE Internet computing,* vol. 6, pp. 86-93, 2002.
[2] G. Alonso, F. Casati, H. Kuno, and V. Machiraju, "Web services," in *Web Services*, ed: Springer, 2004, pp. 123-149.
[3] M. Ali, J. M. Zain, M. F. Zolkipli, and G. Badshah, "Mobile cloud computing & mobile battery augmentation techniques: A survey," in *Research and Development (SCOReD), 2014 IEEE Student Conference on*, 2014, pp. 1-6.
[4] M. Ali, J. M. Zain, M. F. Zolkipli, and G. Badshah, "Mobile cloud computing & mobile's battery efficiency approaches: A Review," *Journal of Theoretical and Applied Information Technology,* vol. 79, p. 153, 2015.
[5] S. Anwar, Z. Inayat, M. F. Zolkipli, J. M. Zain, A. Gani, N. B. Anuar, *et al.*, "Cross-VM Cache-based Side Channel Attacks and Proposed Prevention Mechanisms: A survey," *Journal of Network and Computer Applications,* vol. 93, pp. 259-279, 1/09/2017 2017.
[6] P. Farley and M. Capp, "Mobile web services," *BT Technology Journal,* vol. 23, pp. 202-213, 2005.
[7] M. Shiraz, M. Sookhak, A. Gani, and S. A. A. Shah, "A study on the critical analysis of computational offloading frameworks for mobile cloud computing," *Journal of Network and Computer Applications,* vol. 47, pp. 47-60, 2015.
[8] F. AlShahwan and K. Moessner, "Providing soap web services and restful web services from mobile hosts," in *Internet and Web Applications and Services (ICIW), 2010 Fifth International Conference on*, 2010, pp. 174-179.

[9] M. Lanthaler and C. Gütl, "Towards a RESTful service ecosystem," in *Digital Ecosystems and Technologies (DEST), 2010 4th IEEE International Conference on*, 2010, pp. 209-214.
[10] K. Devaram and D. Andresen, "SOAP optimization via parameterized client-side caching," in *Proceedings of the IASTED International Conference on Parallel and Distributed Computing and Systems (PDCS 2003)*, 2003, pp. 785-790.
[11] R. T. Fielding, "Architectural styles and the design of network-based software architectures," University of California, Irvine, 2000.
[12] C. A. Binildas, M. Barai, and V. Caselli, "Service Oriented Architecture with Java," *Using SOA and web services to build powerful Java applications. Birmingham,* 2008.