Modern Applied Science; Vol. 12, No. 3; 2018
ISSN 1913-1844 E-ISSN 1913-1852
Published by Canadian Center of Science and Education

Web Services: A Comparison of Soap and Rest Services

Festim Halili[1] & Erenis Ramadani[1]

[1] Faculty of Natural Sciences and Mathematics, Department of Informatics, University of Tetova, Macedonia

Correspondence: Erenis Ramadani, Faculty of Natural Sciences and Mathematics, Department of Informatics, University of Tetova, Macedonia. E-mail: festim.halili@unite.edu.mk, erenisramadani@gmail.com

Received: December 12, 2017 Accepted: December 23, 2017 Online Published: February 28, 2018

doi:10.5539/mas.v12n3p175 URL: https://doi.org/10.5539/mas.v12n3p175

Abstract

The interest on Web services has been growing rapidly in these couple of years since their start of use. A web service would be described as a method for exchanging/communicating information between devices over a network. Often, when deciding which service would fit on the architecture design to develop a product, then the question rises which service to use and when?

SOAP (Simple Object Access Protocol) and REST (Representational State Transfer) are the two most used protocols to exchange messages, so choosing one over the other has its own advantages and disadvantages. In this paper we have addressed the differences and best practices when to use one over the other.

Keywords: SOAP, REST, XML, HTML, Client/Server

1. Introduction

In these days of modern technology development, there are a lot of ways that can be used to create enterprise applications. The choice of selecting one over the other should be solely based on technical arguments and their capabilities delivered by each alternative. Thus, web services have gained an enormous popularity in how devices communicate between each other. Briefly, web services (Mironela, 2009) are self-describing and modular business applications exposing business logic as a service over the Internet. They are delivered through programmable interfaces, while their functionality can be consumed and invoked through their IP address. (Wagh, 2012; Halili & Dika, 2012)

There are a lot of technologies that can make this communication possible, such as RMI (Remote Method Invocation), CORBA or DCOM. But, when it comes to security or compatibility, these technologies seems to cause a lot of troubles. Instead, the modern technology is generally based on two new models: SOAP and REST. (Tihomirovs & Grabis, 2016)

Both, SOAP and REST are based on service-oriented architecture. Their development process includes something called Web API, which represents the interface for consuming their service.

Various applications such as conferencing, web or social applications can be developed using these web services, because there is not required any prior knowledge before usage, what makes them platform independent and loosely coupled. (Adamopoulos, 2014)

SOAP is designed to be a lightweight, platform independent protocol for decentralised, distributed environment which uses Internet and XML to exchange information between nodes. It represents a messaging protocol, which uses XML to define the communication and HTTP to transmit these messages. It is a stateless, one-way message communication between nodes or devices, from sender to receiver. (Halili et al., 2012)

In the other hand, REST represents a client-server architecture where the client sends the requests, while the server processes them and returns the responses. It was introduced in 2000, by Roy Fielding. Unlike SOAP, REST services does not limit itself to XML, instead, it also supports JSON (JavaScript Object Notation), plain text, etc. (Halili & Kasa, 2011)

In this paper we will compare the two services and explore the differences that they have in the underlying technology, implementation, strength and weaknesses.

The paper is organized as follows. In the sections 2 and 3 I will provide information about SOAP and REST web services, their advantages and disadvantages. In section 4 will be shown a comparison table, and the paper is closed by providing a conclusion about which one is more preferable for usage.

2. SOAP

Simple Object Access Protocol (SOAP) is a messaging protocol that allow applications to communicate using HTTP and XML. It represents a fundamentally stateless, one-way message exchange paradigm between nodes. By combining one-way exchanges with features provided by the underlying transport protocol and/or application specific information, SOAP can be used to create more complex interactions such as request/response, request/multiple response, etc. (Mumbaikar & Padiya, 2013; Halili, Rufati & Ninka, 2013)

The process of invoking web services is very important; therefore the SOAP protocol is established to exchange messages between service providers and consumers. It is a structured XML message format for exchanging data in a distributed environment. It uses an underlying transport protocol (HTTP, SMTP etc) through binding. By the time of writing this paper there are two version of SOAP: SOAP version 1.1 and SOAP version 1.2 which has brought some new benefits: It is cleaner, faster, it has better web integration and more it is versatile.

There are three main types of SOAP Nodes:

- SOAP Sender – Generates and transmits a SOAP message,
- SOAP Receiver – Receives and processes the SOAP message and it also may generate SOAP response, message or fault as a result, and
- SOAP Intermediary (Forwarding or active) – It is both, a SOAP receiver and a SOAP sender. It receives and processes the SOAP header blocks targeted at it and resends the SOAP message towards an SOAP receiver. This process is illustrated in the figure below:

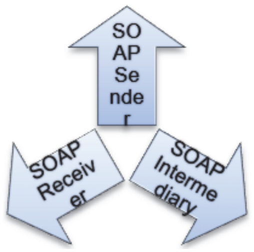


Figure 1. SOAP Nodes

The SOAP message has a structure, which is characterized with two SOAP-specific sub-elements within the overall SOAP Envelope (env:Envelope), namely a SOAP Header (env:Header) and a SOAP Body (env:Body).

As mentioned before in this paper, SOAP is a lightweight independent protocol. It is independent and lightweight because it does not matter what OS or what platform is the service used from: it responds in the same way in any platform or OS. All this is possible because of XML and HTTP protocols.

There are two types of SOAP messaging requests: Remote Procedure Call (RPC) and Document request. Each of them are treated in the following subsections.

2.1 Remote Procedure Call

A Remote Procedure Call represents execution of a procedure in another remote address, usually on another computer in the same network, which is previously coded and it is called as a normal procedure local call. Thus, the programmer will only have to develop the code once, and it does not matter if the call is performed in local or remote circumstances.

This procedure represents a client-server model interaction, which is implemented through a request/response methodology. These requests and responses are formatted in XML Usually, this communication is synchronous, which means that when a request is sent, the app is blocked until the response is processed and returned.

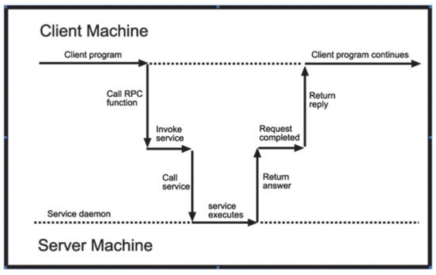
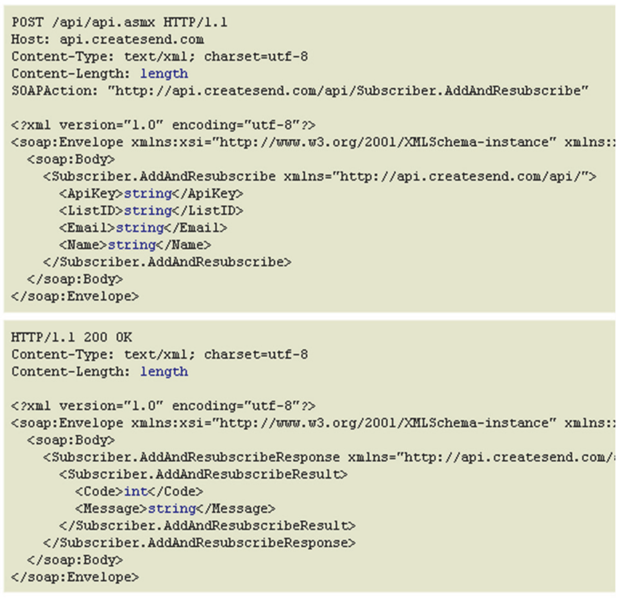


Figure 2. RPC Lifecycle

2.2 Document Requests

While transmitting information from the client to server or vice versa through document requests, the XML document is passed in the body of the SOAP message instead of as parameter.

For example, a service named PurchaseOrder expects a document (XML document) as the input message. When the request is sent through SOAP message, requesting the PurchaseOrder operation, it must contain a purchase order document as input in the SOAP message. The requests is processed as soon as it reaches the server, and when processing is done, another XML document is returned as response, which might contain any kind of information related to that purchase.



```
POST /api/api.asmx HTTP/1.1
Host: api.createsend.com
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://api.createsend.com/api/Subscriber.AddAndResubscribe"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:
  <soap:Body>
    <Subscriber.AddAndResubscribe xmlns="http://api.createsend.com/api/">
      <ApiKey>string</ApiKey>
      <ListID>string</ListID>
      <Email>string</Email>
      <Name>string</Name>
    </Subscriber.AddAndResubscribe>
  </soap:Body>
</soap:Envelope>
```

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:
  <soap:Body>
    <Subscriber.AddAndResubscribeResponse xmlns="http://api.createsend.com/
      <Subscriber.AddAndResubscribeResult>
        <Code>int</Code>
        <Message>string</Message>
      </Subscriber.AddAndResubscribeResult>
    </Subscriber.AddAndResubscribeResponse>
  </soap:Body>
</soap:Envelope>
```

Figure 3. Example of Document request and response

2.3 RMI vs Document Request

RPC interface is meant to be relatively static, which means that any changes on its interface requires additional changes in the implementation of that service. Changing its description would also crash the platforms relying on the service. While, using document messaging requires less such limitations, and any change that might occur in XML schema would not break the structure of the service implementing it. This, because the response is an XML document instead of a structured return value.

Other advantages of Document requests over RPC are reliability, scalability and performance. When the call is performed from the client, it does not have to block the whole process waiting for a response, and can easily be processed asynchronously or placed in the queue. It is more reliable, because the response is sent to the client even in the case that, for some reason, the client might have gone offline after performing the request. It improves scalability because, depending on the architecture the application is build, processing the response can very easily be spread over more instances in the infrastructure.

3. Restful Web Services

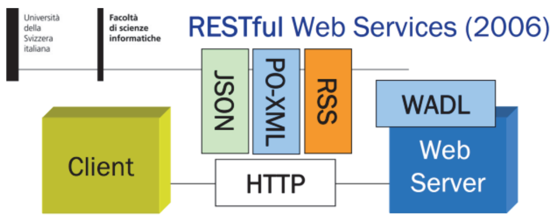

Università della Svizzera italiana
Facoltà di scienze informatiche
RESTful Web Services (2006)
JSON
PO-XML
RSS
WADL
Client
HTTP
Web Server

Figure 4. Architecture of RESTFul web services, and the commuunication between Client and Server

REST - (Representational state transfer) as the name implies, it has to do with client and server relationship and how state is stored. REST architecture is based on the client/server architecture style. Thus, the requests and responses are built based on the transfering process of the resources. All resources are identified by unique Uniform Resource Identifier (URI), which typically represents a document that captures the state of the resource. Generally, the REST style architecture is much lighter compared to SOAP. It does not require formats like headers to be included in the message, like it is required in SOAP architecture. In the other hand it parses JSON – a human readable language designed to allow data exchange and making it easier to parse and use by the computer. It is estimated to be at around one hundred times faster than XML.

```
{
        "firstname" : "John",
        "lastname" : "Smith"
}
```

Figure 5. A simple JSON document

There are several principles that designing RESTFul Web Service requires. Addressability is a REST principle where the datasets are modeled to operate as URI marked resources. Statelessness is another principle that the designer of a REST service will have to follow. This means that every transaction must be independent and must not be related to any previous transaction, as all the data required to perform and process the request are contained on that request, thus, the server will not have to maintain client session data. Uniform interface requires that an interface is uniform and standard used to access the resources, i.e. using fixed set of HTTP methods. If the service designer holds to these principles, than it is almost guaranteed that the REST application will be simple and lightweight.

In other words, how can we define the basics of the problem in REST architecture? Imagine there is a computer A in Tetova, it interacts with computer B in Berlin related to a resource available on Computer C in New York. None of them belongs to the same domain. In the REST style there are defined nouns and verbs. URI's are the equivalent of nouns and there are trillions of nouns for the entire concept in all the heads and files of all the users in the world.

Back when we studied in the beginner school, we in grammatical exercises we have studied about Nouns and verbs, the verbs describe an action related to nouns, whilst in REST architecture style verbs (loosely) describe actions that are applicable to nouns or with other words URI's. There are four universal verbs introduced in REST like described in Figure 6.

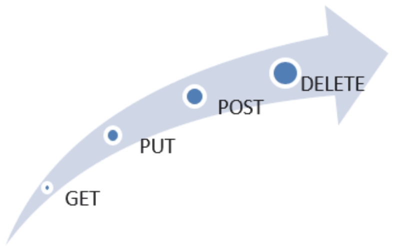


Figure 6. Basic methods of RESTFul architecture

The web application which follows the REST architecture we call it as RESTful web service. Restful web services uses GET, PUT, POST and DELETE http methods to retrieve, create, update and delete the resources. (Sinha et al., 2014)

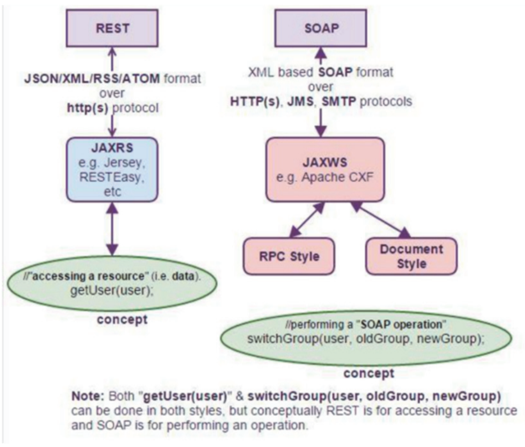

REST
JSON/XML/RSS/ATOM format over http(s) protocol
JAXRS e.g. Jersey, RESTEasy, etc
//"accessing a resource" (i.e. data). getUser(user);
concept
SOAP
XML based SOAP format over HTTP(s), JMS, SMTP protocols
JAXWS e.g. Apache CXF
RPC Style
Document Style
//performing a "SOAP operation" switchGroup(user, oldGroup, newGroup);
concept
Note: Both "getUser(user)" & switchGroup(user, oldGroup, newGroup) can be done in both styles, but conceptually REST is for accessing a resource and SOAP is for performing an operation.

Figure 7. Flowchart showing hor REST and SOAP access methods

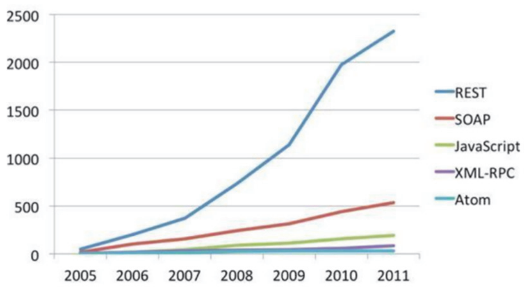


Figure 8. Several platform usage over years

REST is becoming the go to for system interaction which includes the usage of RESTFul web services mostly the way cloud providers expose their services. In the present days, we can easily conclude that most of the new projects are based on RESTFul architecture, in order to create and provide professional services. Not only the tech giants like Facebook, Google or Twitter use REST these days. This, because thanks to the REST architecture, every application is able to scale horizontally in the easiest possible way.

3.1 Advantages of Rest

- REST uses smaller message format and provides cost efficiency over time and better performance because of the JSON messages with which makes the communication. and there is no intensive processing required
- Learning curve is reduced
- It supports stateless communication
- It’s simple to learn and implement
- Efficiently uses HTTP verbs
- Light bandwidth since its passes message is JSON
- (JavaScript Object Notation) format also it can use
- multiple other formats
- For security it uses HTTP standards
- REST can be consumed by any client
- It makes data available as resource. (Kumari, 2015)

3.2 Disadvantages of REST

- It's not suitable for large amount of data
- Comparative SOAP it does not cover all varieties of web service standards like Security, Transactions etc.
- REST is not reliable
- REST requests (especially GET) are not suitable for large amount of data
- Latency in request processing times and bandwidth usage
- REST APIs end up depending on headers for state (such as to route subsequent requests to the same back-end server that handled the previous update, or for authentication.) Use of headers is clumsy and ties the API to http as a transport.

4. Comparison between the Two Services

Following is shown a table containing information about both, SOAP and REST web services, where can easily be seen their comparison.

Table 1. Comparison between SOAP and REST (Wagh & Thool, 2015)

SOAP	REST
Changing services in SOAP web provisioning often means a complicated code change on the client side.	Changing services in REST web provisioning not requires any change in client side code.
SOAP has heavy payload as compared to REST.	REST is definitely lightweight as it is meant for lightweight data transfer over a most commonly known interface, - the URI
It requires binary attachment parsing.	It supports all data types directly.
SOAP is not a wireless infrastructure friendly.	REST is a wireless infrastructure friendly.
SOAP web services always return XML data.	While REST web services provide flexibility in regards to the type of data returned.
It consumes more bandwidth because a SOAP response could require more than 10 times as many bytes as compared to REST.	It consumes less bandwidth because it's response is lightweight.
SOAP request uses POST and require a complex XML request to be created which makes response-caching difficult.	Restful APIs can be consumed using simple GET requests, intermediate proxy servers / reverse-proxies can cache their response very easily.
SOAP uses HTTP based APIs refer to APIs that are exposed as one or more HTTP URIs and typical responses are in XML / JSON. Response schemas are custom per object	REST on the other hand adds an element of using standardized URIs, and also giving importance to the HTTP verb used (i.e. GET / POST / PUT etc
Language, platform, and transport agnostic.	Language and platform agnostic
Designed to handle distributed computing environments.	Assumes a point-to-point communication model - not for distributed computing environment where message may go through or more intermediaries.
Harder to develop, requires tools.	Much simpler to develop web services than SOAP
Is the prevailing standard for web services, and hence has better support from other standards (WSDL, WS) and tooling from vendors.	Lack of standards support for security, policy, reliable messaging, etc., so services that have more sophisticated requirements are harder to develop.

5. Implementation of a Demonstration REST Service

As support for all the job done about this research paper, there has been created a sample web service aiming to illustrate the whole concept.

Because of the space consumed by demostrating it, the SOAP approach of the service is not included on this paper.

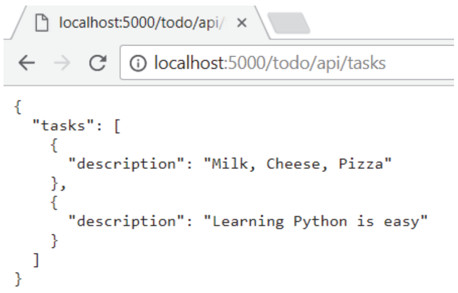
The service is done by following the REST principles. The service creates a ToDo list, which can provide all of the "ToDos" or give you details about a particular item in the list. The service is done by using Flask, a lightweight Python framework, but it can be done in any other language that supports web. (Grinberg, 2013)

```
tasks = [
 {
"id": 1,
        "title": "Buy groceries",
        "description": "Milk, Cheese, Bread",
        "done": False
 },
 {
"id": 2,
        "title": "Learn Python",
        "description": "Learning Python is interesting",
        "done": False
 }
]
```

The snippet represents the sample data that will be used to demonstrate the service.

The very first method to be treated is *get_tasks* which as a result, returns the list of all tasks. In REST terms, it is a *GET* method. Following are represented the code and the execution of the same.

```
function get_tasks:
  if(loggedIn)
    return tasks
  else
    return error:authentication failed
```



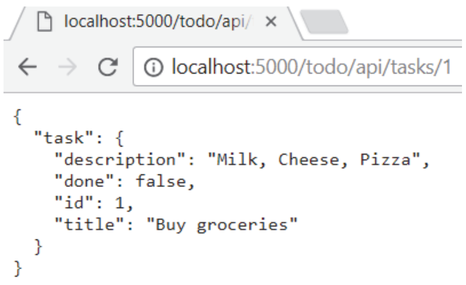
localhost:5000/todo/api/tasks

```
{
  "tasks": [
    {
      "description": "Milk, Cheese, Pizza"
    },
    {
      "description": "Learning Python is easy"
    }
  ]
}
```

Figure 9. The response for get_tasks method

If the user needs only a particular item of the list, REST principles provide a simple solution for that. Instead of getting all items, and then cycling the whole array to find the particular item, REST services offer the item to be accessed directly. get_task(:id) is such a method, where id represents the id of the item the user needs.

```
function get_task(id)
  declare task from tasks where task.id is equal to id
    if task not null
      return task
    else
      return error: not found
```



```
{
  "task": {
    "description": "Milk, Cheese, Pizza",
    "done": false,
    "id": 1,
    "title": "Buy groceries"
  }
}
```

Figure 10. The response for get_task(:id)

Other features like adding items to the list or deleting items from the list, work in pretty much the same way. To put a new item, the same url as get_tasks is user, but instead, the method is set to POST, and a JSON formatted document containing task details is required as parameter. To delete an existing item from the list, requires DELETE method type and specifying the id of the task to delete.

6. Conclusion

REST defines the architectural style of the World Wide Web. After all the comparison and stating their underlying technologies, process, ease of use, and their design it's safe to say that RESTful web service will continue to dominate more and more the technology space in the coming years when it comes to building an backend RESTFul api. The RESTFul web services are lightweight, easy to consume, self descriptive, fast, have support for all data types, consumes less bandwidth and it is more simpler to develop and maintain.

It is also worth mentioning that, REST is not a service that everyone could deal with. Let us take for example some basic HTTP APIs that someone builds and just calls it RESTFul web service, but in fact, it is not even close to REST. Sometimes, REST can be really hard to deal with, especially in the very early stages of designing. But, after growing up in both, technical and economical way, it pays off with the small changes required to adapt to evolution. If you need something done quickly and easily, don't bother about getting REST right. It's probably not what you're looking for. If you need something that will have to stay online for years or even decades, then REST is for you.

References

Adamopoulos, V. T. (2014). *The Effectiveness of Promotional Events in Social Media*. Retrieved from https://www.misrc.umn.edu/wise/2014_Papers/94.pdf

Grinberg, M. (May, 2013). *Designing an RESTful API with Python and Flask.* Retrieved from https://blog.miguelgrinberg.com/post/designing-a-restful-api-with-python-and-flask

Halili, F., & Dika, A. (2012). *Choreography of Web Services and Estimation of Execution Plan*. In Proc. Book of IEEE International Conference of Information Technology and e-Services (ICITeS), pp.168-174, Sousse, Tunis, March 2012. ISBN: 978-1-4673-1166-3, IEEE CN: CFP1245S-ART.

Halili, F., & Kasa, M. (June 2011). *Analysis and Comparison of Web Services Architectural Styles, and Business Benefits of their Use*. In Proc. Book of International Conference of Information Technologies and their importance in the economic development, 701-712, Tirana. ISBN: 978-99956-59-13-4.

Halili, F., Idrizi, F., Lena, U., & Kasa, M. (February, 2012). Towards the UML Design of Web Services. In *International Journal of Science, Innovation and new Technology (IJSINT), 1*(3), 73-78, ISSN: 2223-2257.

Halili, F., Rufati, E., & Ninka, I. (2013). *Service Composition Styles – Analysis and Comparison Methods*, in Proc. Book of IEEE CICSyN2013 5th International conference on Computational Intelligence, Communication Systems and Networks, pp.278-284, 5-7 June, Madrid, Spain. ISBN: 978-0-7695-5042-8, BMS Number: CFP1381H-CDR.

Kumari, V. (May 2015). Web Services Protocol: SOAP vs REST. *International Journal of Advanced Research in Computer Engineering & Technology (IJARCET), 4*(5).

Mironela, P. (2009). *The Importance of Web Services using the RPC and REST Architecture*, IEEE, International Conference on Computer Technology and Development, 2009.

Mumbaikar, S., Padiya, P. (May 2013). Web Services Based On SOAP and REST Principles. *International Journal of Scientific and Research Publications, 3*(5), ISSN 2250-3153.

Sinha, R., Khatkar, M., Gupta, S. C. (September 2014). *Design & Development of a REST based Web service platform for mobile applications integration on Cloud.* IJISET - International Journal of Innovative Science, Engineering & Technology, (7).

Suda, B. (2003). *SOAP Web Services*. University of Edinburgh, School of Informatics, Computer Science.

Tihomirovs, J., & Grabis, J. (2016). *Comparison of SOAP and REST Based Web Services Using Software Evaluation Metrics.* Riga Technical University, 19, 92-97P.

Wagh, K. (2012). *A Comparative study of SOAP Vs REST Web Services Provisioning Techniques for Mobile Host*, Journal of Information Engineering and Applications, ISSN 2225- 0506 (Online), 2(5), 2012.

Wagh, K., & Thool, R. (2015). A Comparative Study of SOAP Vs REST Web Services Provisioning Techniques for Mobile Host. Marathwada Mitra Mandals Institute of Technology Pune, India - Shri Guru Gobind Singhji Institute of Engineering & Technology, Nanded, India.