Comparison of Soap and Rest Services

Thanvendra Eswar Amara

Abstract

As the digital world grows picking the best way for devices to communicate online is essential for creating effective products. In this study we dive into two key methods: SOAP (Simple Object Access Protocol) and REST (Representational State Transfer). These methods act like different languages devices use when talking over the internet. We explore what makes each of them great and where they might have limitations, helping to figure out which one fits best for different situations.

We take a close look at important elements like XML which structures data and HTML the language of web pages. Understanding how devices like your computer or phone interact as clients and servers is also crucial. This study provides clear guidance on when it's smarter to use SOAP or REST in various setups giving practical tips for making the right choice in designing web-based systems.

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

1. Introduction

In today's tech world when we build business apps we have lots of ways to do it. The key is to pick the right one based on tech reasons and what each option can do. That's where web services come in – they help devices talk to each other. Basically web services are flexible business apps that share their functions online. You can use them through special interfaces and access them using their internet addresses.

In the past, we had other ways for devices to talk like RMI or DCOM but they often caused problems with security and compatibility. Nowadays we mostly use two methods: SOAP and REST.

Both SOAP and REST are built on a way of designing services using something called Web APIs to use their services. They can be used for all kinds of apps like meetings, websites, and social apps. What's great is you don't need to know a lot beforehand, making them work on different systems easily.

SOAP is like a lightweight, all-purpose language for devices to talk in a decentralized way. It uses the internet and XML to share information between devices. It's like sending messages from one device to another without keeping track of past conversations.

On the other hand REST is like a conversation between a customer and a waiter. The customer asks for something, and the waiter brings it. This was introduced by Roy Fielding in 2000. Unlike SOAP, REST can use different formats like JSON or plain text.

This paper aims to compare these two services, looking at the tech they use, how they work, and what they're good and not so good at. We'll talk about SOAP and REST web services in mentioning their pros and cons and will show a table comparing them. We'll finish by deciding which one is better to use.

2. PROBLEM DEFINITION

Designing an API technically might seem simple, but creating one that stays easy to manage works well and can grow as needed is super important. Sometimes, picking an API style without thinking about its good and bad points leads to problems like needing to redo things, slow performance, or not being able to handle a lot of users. In this Term paper, a comparison will be made on different API Architectures that are widely used. This comparative analysis can aid and assist in choosing the right API architecture when designing solutions on the web.

3. Soap

Simple Object Access Protocol (SOAP) helps apps talk to each other using a language called XML over the internet. It's like sending messages from one place to another without expecting an immediate reply. However, with some tricks it can handle more complicated conversations like asking for something and getting a response back.

When apps need to communicate SOAP makes sure they can share information properly. It organizes this information in a specific way using a structured format called XML and works with different internet methods like HTTP or SMTP.

At the moment, there are two versions of SOAP: SOAP 1.1 and SOAP 1.2. The newer version SOAP 1.2 is better in many ways. It's cleaner faster works more smoothly on the internet and can handle more types of tasks.

SOAP Sender: Creates and sends a SOAP message.

SOAP Receiver: Gets and deals with the SOAP message. It might also create a reply or error message.

SOAP Intermediary: Acts as both a SOAP receiver and sender. It gets and handles specific parts of the SOAP message then passes the message along to another SOAP receiver. This process is shown in the figure below:-

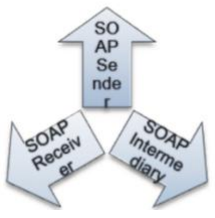


Figure 1:SOAP Nodes

The SOAP message is made up of two parts: The SOAP Header and the SOAP Body both found in the SOAP Envelope.

As we mentioned earlier SOAP is a simple and flexible way for different systems to communicate. It doesn't care which operating system or platform you are using it works the same way for all. This reliability happens because SOAP uses XML and HTTP protocols.

There are two kinds of SOAP requests :

- Remote Procedure Call
- Document request

3.1 Remote Procedure Call

When you do a Remote Procedure Call it's like asking another computer to do a specific job for you just as if you were doing it on your own computer. So a programmer writes code for a task and it can be used whether it's done on the same computer or a different one.

This works a bit like asking for something from a server computer. They talk by sending messages back and forth using a special format called XML. Usually this talking happens one step at a time when one message is sent the program waits until it gets a response before it does anything else.

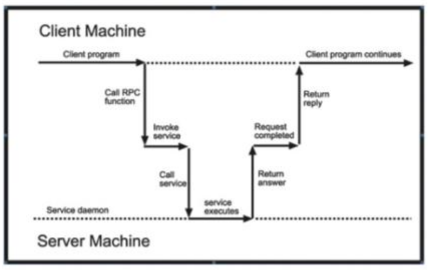


Figure 2 : RPC Lifecycle

3.2 Document Requests

When data moves between the client and server using document requests the XML document is carried within the body of the SOAP message rather than being passed as a parameter.

For instance let's say there's a service called Purchase Order that needs an XML document as its input. When sending a request through a SOAP message asking for the Purchase Order action the SOAP message needs to contain the purchase order document as its input. Once this request reaches the server it's immediately processed. After processing, the server sends back another XML document as a response which might contain various details connected to that purchase.

4. REST

Representational State Transfer(REST) focuses on how information moves between clients and servers while managing their states. It relies on a client/server style and centres around transferring resources. Each resource is identified by a unique Uniform Resource Identifier usually representing a document capturing that resources state.

Compared to SOAP, REST is lighter and doesn't need extra things like headers in messages. Instead it handles data in a simpler way, often using JSON a language easy for both humans and computers. JSON is estimated to be around a hundred times faster than XML.

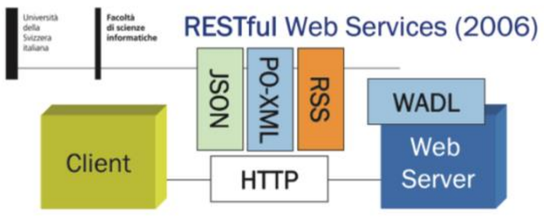

Università della Svizzera italiana
Facoltà di scienze informatiche
RESTful Web Services (2006)
JSON
PO-XML
RSS
WADL
Client
HTTP
Web Server

Figure 3: Architecture of RESTful web services, and the communication between Client and Server

Designing RESTful Web Services involves key principles. Addressability means representing datasets as resources identified by URIs. Statelessness requires each transaction to stand alone not dependent on past ones as all needed data is in the request. Uniform interface means using a fixed set of HTTP methods for access. Adhering to these principles ensures simplicity and lightness in REST applications.

In simpler terms in REST architecture when a computer in one place say Tetovo, talks to another like computer B in Berlin about a resource on Computer C in New York they use 'nouns' (URIs) and 'verbs' (HTTP methods). URIs act like names for trillions of concepts worldwide stored in everyone's heads and files.

Back in beginner school we learned about nouns and verbs in grammar exercises. Verbs describe actions related to nouns. In REST architecture verbs (loosely) represent actions that apply to URIs or in simpler terms to the things identified by those URIs. In REST there are four main universal verbs as shown in the figure below.

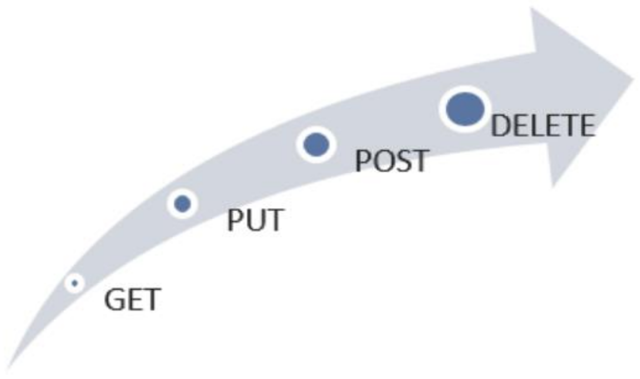


Figure 4: Basic methods of RESTful architecture

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.

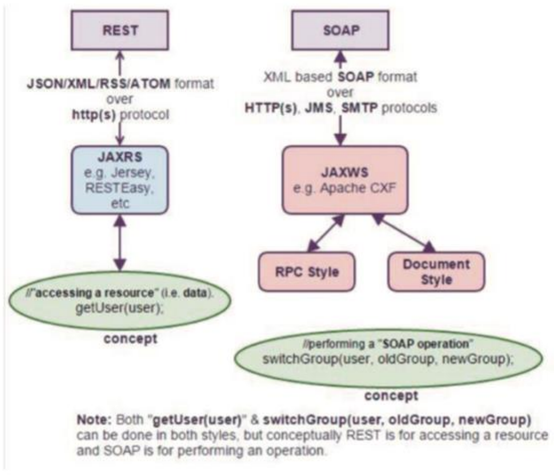
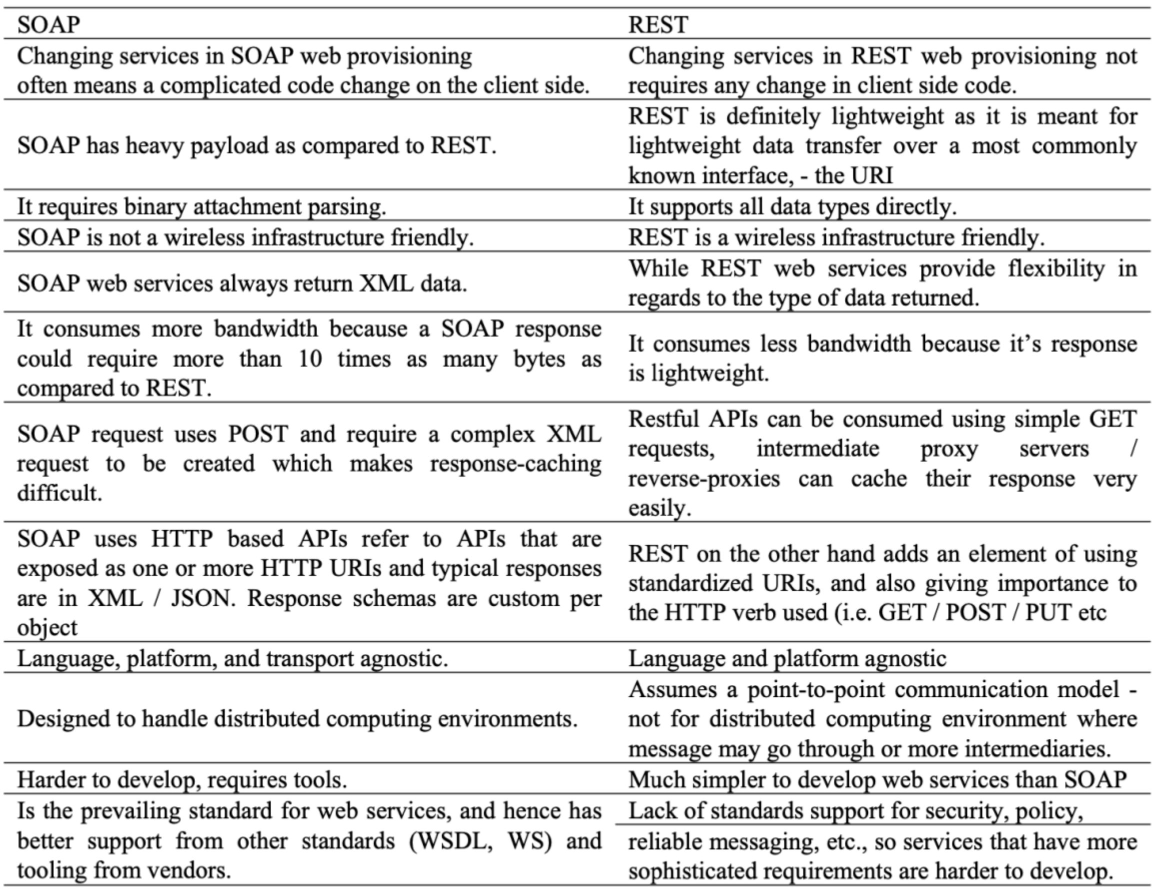

REST
JSON/XML/RSS/ATOM format over http(s) protocol
JAXRS e.g. Jersey, RESTEasy, etc
//"accessing a resource" (i.e. data). getUser(user);
concept
SOAP
XML based SOAP format over HTTP(s), JMS, SMTP protocols
JAXWS e.g. Apache CXF
RPC Style
Document Style
//performing a "SOAP operation" switchGroup(user, 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 5: Flowchart showing hor REST and SOAP access methods

REST has become the preferred method for system interaction especially through RESTful web services commonly used by cloud providers. Nowadays, it's safe to say that many new projects rely on RESTful architecture to offer top-notch services. It's not just tech giants like Facebook, Google, or Twitter; REST is widely adopted. This popularity owes to REST enabling applications to easily scale horizontally making growth smoother for any app.

5. 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.



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.

6. Conclusion

REST defines the architectural style of the World Wide Web. Despite comparisons highlighting its underlying technologies, processes, ease of use, and design, it's evident that RESTful services stand out.

However, not everyone can easily handle REST. Sometimes people label basic HTTP APIs as RESTful web services when they're not really close to being RESTful. Dealing with REST can be tough especially in the early stages of design.

Yet as businesses and technology evolve embracing REST pays off. Though it might require some adjustments along the way especially in the beginning its long-term benefits become apparent. If you're looking for something quick and easy REST might not be the best fit. But if you need a robust system that stands the test of time REST is the way to go.

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., 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.
- https://www.researchgate.net/publication/323456206_Web_Services_A_Comparison_of_Soap_and_Rest_Services