Enterprise Application Integration using Publish/Subscribe REST with IP Multicast

Fadel Cheteng and Rattakorn Poonsuph
Graduated School of Applied Statistics, NIDA, Thailand
Email: fadelc99@hotmail.com, rattakorn@as.nida.ac.th

***Abstract*—Web services is a technology standardized by World Wide Web Consortium or W3C with the purpose of enabling software the ability to coordinate with one another regardless of the programming languages, technologies and platforms. Web Services is widely used in the integration of various applications and platforms in each organization so that the common data can be communicated and transferred to the silo applications on any platform. However, Web Services has significant limitations; which is its inability to indicate the latest state of the common data across applications. In order to do so, the connected application programs have to constantly retrieve the common data. This procedure not only overuses the network and causes a bottle neck issue, but it can also generally affect the response time, further causing the delay of the entire network. There were many other researches that try to resolve these problems, but were unable to resolve almost all of the aforementioned problems. Therefore, this research paper will be focusing on the network usage's most efficient method, which is to use public/subscribe Web Services on REST lightweight protocol along with the multicast data transmission that transfers a single packet of data only once to the networks so that all of the connected silo's applications will receive it simultaneously. The solution can be applied with mobile devices that maintain common data across the mobile devices.**

***Index Terms*—EAI, web services, IP multicast, REST**

I. Introduction

Information nowadays is essential day-to-day operations for any organizations. Application program is designed to process data and provides intelligence information to manage the business operations. However, the application programs have been grown organically within the organization based on their business and user's demands. Usually, there was no strategic planning or architectural blueprint to structure the application programs that are to be integrated in a unified pattern. Most of the application programs were acquired or developed independently. In other words, it was developed by the variety of programming languages, technologies and platforms without them being linked or exchanged of the common data or information. Consequently, the data and information of each application programs contains inside their applications.

Nevertheless, today's ever doubling demand for information leads to the emerging of a new digital businesses such as business to business trade (B2B) [1] with mobility platforms. Moreover, there are rapid changes in the business operations as well. An enterprise application integration or EAI [2] is, as a result, necessary and crucial for organizations these days.

Web Services [3] is a technology standardized by World Wide Web Consortium or W3C with the purpose of enabling software the ability to coordinate with one another regardless of the languages and platforms. Thus, it is widely used in the integration of various application programs and platform in each organization so that common data can be communicated and transferred to solo applications on any platform. Web Services is consisted of three main components: SOAP (Sample Object Access Protocol) [4], WSDL (Web Service Description Language) [5] and UDDI (Universal Description Discovery and Integration) [6]. All of them are in XML (Extensible Markup Language) format that are simple but self-describing messages. Their self-description means that they do not depend on any platform or programming languages. As for the application of Web Services, it is divided into two sections: Service Provider and Service Client. Firstly, the Service Client will search for the self-registered service from UDDI in order to obtain the WSDL document (or the client can get it directly from the Service Provider). WSDL is a document that informs the Service Provider; the methods that are available for the system call, including the formats and types of the parameters in which each method requires such as Endpoint. When the Service Client receives the WSDL, it will be used to make a Stub or Proxy. Then, the Web Services can be used through this newly-made stub. In 2000, Roy Fielding put forward the new type of Web Services, which is the REST or Representational State Transfer [7]. It is an application of the HTTP protocol method with the URI or Universal Resource Identifiers when operated. For instance, the GET method is used in order to extract data resource from the Web Services as indicated in the URI.

Using Web Service in the EAI context is a challenged implementation. Ranges of silo application programs in organization maintain common data such as customer

Manuscript received May 20, 2015, revised August 29, 2015.

information, product catalog, or account receivable profile. These common data could be updated in a silo application program, while others were not realized of those changed. Web Service could be used to exchange the common data among silo application programs. The point-to-point Web Service integration is to sync the common data as a direct link from one application to another. At the end, the direct links of point-to-point Web Service are all across the silo application programs, it creates burden to developer and manager in order to maintain these messed up architecture for enterprise application system. On the other hand, the public/subscribe Web Services with a coordinated broker is the most suitable integration architecture of enterprise application integrations. A publisher will send only an updated common data package to the coordination broker on a certain topic. A subscriber may then choose listen to the topics of their own interests and received updated common data packages in order to update the copy of its common data inside their internal database. The most benefits of this architecture is that the publisher and subscriber are only connected to the coordinated broker, therefore the publisher has less burden in order to send the common data directly over to each subscriber regardless number of subscribers.

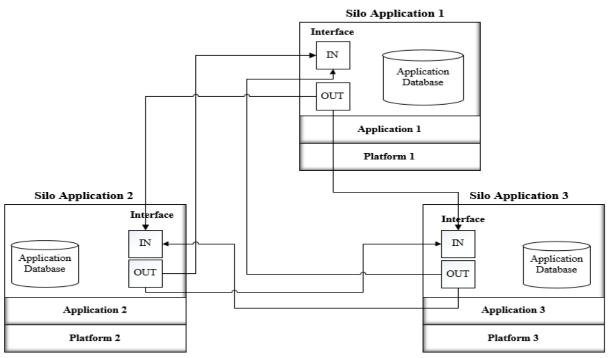

Figure 1. Point-to-Point web services for enterprise application integration

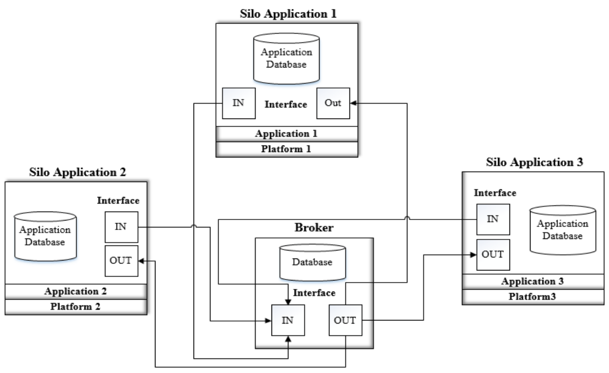

Figure 2. Publish/Subscribe web services for enterprise application integration

One significant limitation of enterprise application integration using Web Services is its inability to indicate the latest state of the common data amongst silo application program. In order to do so, the connected application programs have to constantly retrieve the common data from the source application programs. This procedure not only overuses the network and may cause a Bottle neck issue when the size of the data is too large or when there are too many clients, but it can also generally affect the response time causing the delay of the entire network. Hence, there were researches done in order to get rid of the issues that will be covered in more details in the paper.

II. Related Works

The following research aims to resolve the aforementioned problems and have adapted the Publish/Subscribe architecture to their resolutions. In 2003, Brenna and Johansen [8] used this procedure with WAIF Proxy. Firstly, the Service Client registers the WAIF Proxy device and retrieves the certain data to his or her own device. Then, if there were data alterations on the Service provider, the data would be transferred to the Service Client's registered device as seen in Fig. 3 (a.). However, the disadvantage of this method is that it requires the user to have the WAIF Proxy to be constantly retrieving the updated data. If the data size is too large and being retrieved too many times, the bottle neck problem would occur to the Service Provider's device. In 2009, Feng, Xue and Zhang [9] adapted the UDDI system to their work. They started from having the Service Providers register the UDDI and the Service Clients would come register for the topics of their own interests as seen in Fig. 3 (b.). When there is any updating in the data, the Service Providers would send signals to the UDDI which would then be transferred and indicated to the clients so that they could retrieve the data from the Service Providers. However, this procedure causes complication as each application program interface requires its own queuing system. In 2010, Skjervoid, Hafsøe, Johnsen, and Lund [10] utilized a central broker called Delay and Disruption Tolerant SOAP Proxy (DSProxy). DSProxy is in both sides of the Service Providers and the Service Clients. The DSProxy in the Service Providers' side would retrieve data from the Service Providers to constantly examine them for any specific changing information. If any updated data was found, a signal would be sent to the DSProxy in the Service Clients' side in order to inform them of data alteration and the information retrieval from the Service Providers' side. This method helps lessen the unnecessary use of the network. However, its practicality faces problems as the protocol is not of standard protocol as SOAP/REST protocol and therefore has limitation in the application program implementation in the real world.

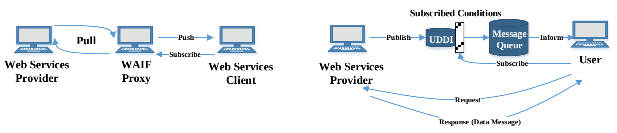

Figure 3. Publish/Subscribe architecture used in troubleshooting. a) WAIF Proxy and b) UDDI registration system.

Later in 2012, there was a research done by Thanisa Noomnon [11] that can eliminate the bottle neck problem and can also offer a better network usage by following a Push-Based architecture as seen in Fig. 4. The Push-Based architecture is consisted of Broker, Service Provider and Service Client. The Broker is used to set up a Topic and create a WSDL document which is considered as a canonical message for the chosen topic so that the Service Provider and Service Client can use them together.

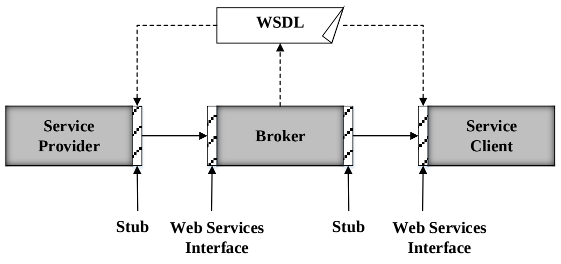

Figure 4. Overview of calling web service using Push-Based method

Meanwhile, the Service Provider would bring in the WSDL document as a client and transform it into a stub. The Service Provider swaps the role by transferring the data over to the Broker's interface. Simultaneously, the Service Client would also bring in the same WSDL document as the Service Provider but it would set up its own Web Service interface endpoint as a data receiver in order to update their internal data. After that, the Service Client would register by notifying the Endpoint of his Web Service to the Broker. But consequently, whenever the Service Provider makes any alteration to the data, the updated data from Stub would transfer the data onto the Broker's interface. Then, the Broker would send the data to each registered Service Client directly. With this procedure, the bottle neck issue would be eliminated of as the application programs on the Service Provider constantly transfer the data to the Service Clients so that they receive the latest edited data. So the constant pooling of the updated data on the Provider Client is no longer needed.

Upon consideration, it is noticeable that the Broker has to keep sending out the same data packet to a registered Service Clients. However, if the number of the clients was too high or the sizes of the data were too large, the network overusing issues would follow.

III. Research Conceptual Model

This research comes up with the practical solution of using the network in the best efficient way possible by having Broker send one set of data one time to the IP Multicast [12] in which will be received by the Agent in the recipient's side before transferring onto the Service Client as seen in Fig. 5.

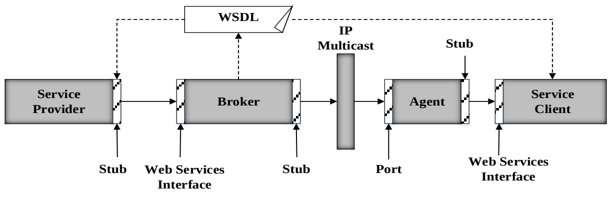

Figure 5. Components of web service using IP multicast.

Additionally, most of the previous researches typically use SOAP protocol as the data transmission between Service Provider and Service Client. REST protocol, however, has a better performance because it bears minimal overhead on top of HTTP. This means that the performance of the network (i.e. bandwidth). Since REST uses standard HTTP it is much simpler in creating clients, developing APIs. REST also permits many different data formats that have been beneficial. JSON, for example, usually is a better fit for exchanging data and parses much faster than XML in SOAP.

The research objective is to quest for the best solution using Web Service as the integration platform. So that we applied the push-based method [11] of inversion of Web services and improving them with REST protocol along with IP Multicast.

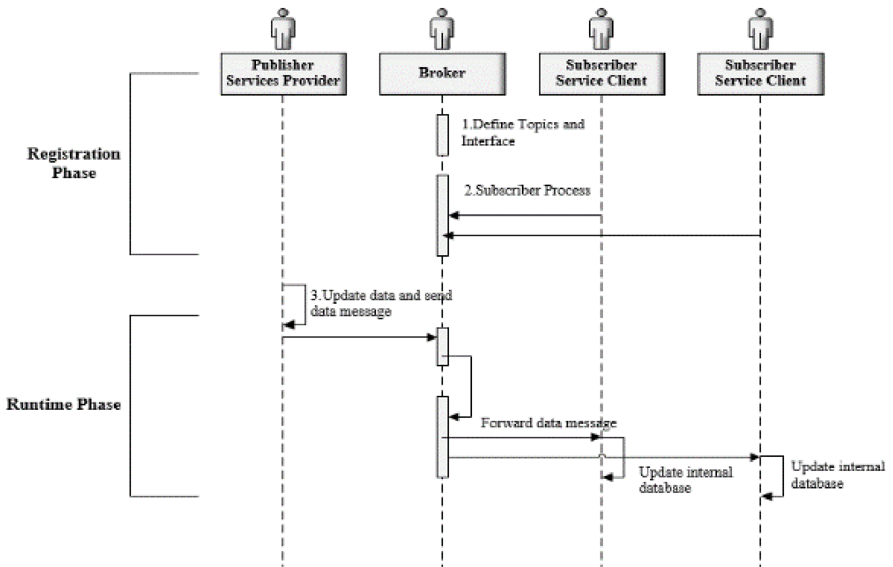
Publisher Services Provider
Broker
Subscriber Service Client
Subscriber Service Client
Registration Phase
1.Define Topics and Interface
2.Subscriber Process
3.Update data and send data message
Runtime Phase
Forward data message
Update internal database
Update internal database

Figure 6. Sequence diagram implementation Push base/Subscribe SOAP & REST IP multicast

IV. Research Methodology

The research based on real experimental of the research concepts. More than 30 servers on MS Windows platform have been used in the research experiment which was set up as Services Providers and Service Clients. The implementation of the Publish/Subscribe REST with IP Multicast in the push-based architecture is set up based on the following assumptions: First, there is only one operation in a unique topic that both Service Providers and Service Clients are used to exchange the common data. The common data is global information that is resided within all Service Providers and Service Clients. Secondly, there can be multiple Services Providers in a topic and there is no Services Provider that can also be a Service Clients of the same time. Lastly, a program of Service Providers and Service Clients has been developed by using a standard SOAP protocol and REST protocol. However, there are internally modifying these standard protocols to adapt the transmission of data package using IP Multicast.

There are two phases for the experimental implementation, which is the registration phase and the run-time phase. The registration phase is for announcing of a topic which the common data cloud be exchanged. The registration defines the WSDL topics for SOAP protocol or Web-API method for the REST protocol that used as a common interface for both Service Providers and Service Clients. The next step is the subscription process. The interested Service Clients must subscribe at least one topic to the broker for received a message from Service Providers. For the run-time phase, the Service Providers submits the updated data portion of common data to the broker, where then the broker forwards those updated data portion to all the subscribers.

V. Experimental Result

The research experiment is operated by using IP Multicast and is divided into two parts. The first part is an experiment using SOAP Web Services to compare the former data transfer with Multicast. The result has seen in Fig. 7. The larger the number of the Service Clients' devices is the better uses of multicast network. The comparison on the 30 service clients using SOAP base in Inversion Web Services is around 1,792.50 milliseconds and using IP Multicast is approximately 668.00 milliseconds by average, which is 62.73% better.

The second part of the experiment is an IP Multicast data transfer comparison between SOAP and REST Web Services. The performances of REST Web Services are found to be working better than SOAP around 73.9%.

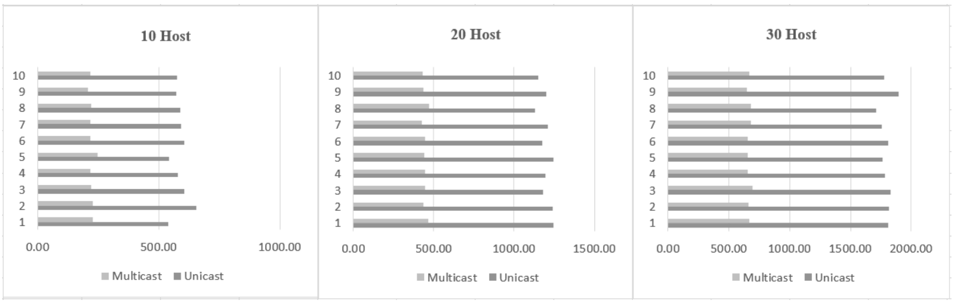

Figure 7. Comparison between existing data transmission method and multicast IP method.

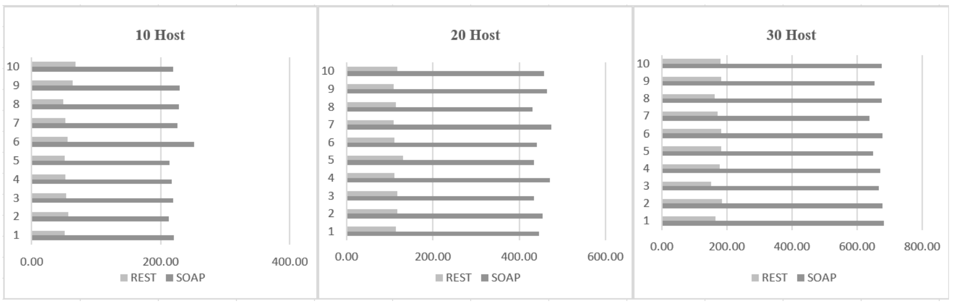

Figure 8. Comparison between SOAP and REST while applied multicast IP method to both of them.

VI. Conclusion

There is significant evidence to prove that the REST protocol and IP Multicast enhances the response time of an enterprise application integration based on Web Services. Publish/subscribe REST protocol with IP Multicast push-based model is the network usage's most efficient method. The experimental results showed that a significant number of a reduction in the network bandwidth and improvement in response time in exchange of updated information to all publisher service providers and subscribed service clients.

The experiment model is not limited to enterprise application integration area. But it can also be used in mobile devices as well. Currently, from the widespread use of mobile devices in a country. Each mobile device is considered as silo application that maintains its own data. In order to exchange the data automatically with their families, friends, colleagues, etc. such as contact information, or available inventory of products, this approach may be used to enhance the network bandwidth in distributing the common data among all mobile devices.

ACKNOWLEDGMENT

The author would like to thank the anonymous reviewers for valuable, detailed, and careful comments and feedback.

REFERENCES

[1] D. Nedbal, "Guiding B2B integration of business process and services: A process model for SMEs," in *Proc. International Conference on Emerging Data and Web Technologies, 2011.*

[2] D. S. Linthicum, *Enterprise Application Integration, Boston: Addison Wesley, 1999.*

[3] (February 10, 2013). World Wide Web Consortium (W3C). Web Services. [Online]. Available: http://www.w3schools.com/webservices/default.asp

[4] (February 10, 2013). World Wide Web Consortium (W3C). [Online]. Available: http://www.w3schools.com/webservices/ws_soap_intro.asp

[5] (February 10, 2013). World Wide Web Consortium (W3C). WSDL. [Online]. Available: http://www.w3schools.com/webservices/ws_wsdl_intro.asp

[6] (February 10, 2013). World Wide Web Consortium (W3C). [Online]. Available: http://www.w3schools.com/webservices/ws_wsdl_uddi.asp

[7] R. T. Fielding, "Architectural styles and design of network-based software architectures," Ph.D. dissertation, University of California, Irvine.

[8] L. Brenna and Johansen, "Engineering push-based web services," *International Journal of Web Services Practices*, 2005.

[9] X. Feng and T. Zhang., "Research on data exchange push technology based on message-driven," *International Join Conference on Artificial Intelligence,* 2009.

[10] E. Skjervold, F. T. Johnsen, and K. Lund, 2010, "Enabling publish/subscribe with COTS web services across heterogeneous networks," in *Proc. IEEE International Conference on Web Services.* Miami, FL, 2010, pp. 660-668.

[11] T. Noomnonda, "Inversion of web service invocation using publish/subscribe push-based architecture," Ph.D. dissertation, *National Institute of Development Administrator (NIDA),* 2012.

[12] D. Makofske and K. Almeroth, "Multicast sockets practical guide for programmers," San Francisco: Morgan Kaufmann, 2002.

Fadel Cheteng received the degree in computer sciences from Ramkamheng University. He is now working at Miracle Advance Technology Ltd., Thailand, as Team Leader.

Rattakorn Poonsuph received the Sc.D. (Computer Science) from University of Massachusetts Lowell. He is an Assistant professor in the School of Applied Statistics, National Institute of Development Administrator (NIDA), Thailand.