A QoS Based Web Service Selection Through Delegation

G. Vadivelou, E. Ilavarasan, R. Manoharan, P. Praveen

Abstract- Service selection is essential for fulfilling the requirements of service requestors. In the service oriented environment, Quality of Services (QoS) is one of the utmost concerns for consumers during service selection. Existing web service standards do not undertake the QoS issue efficiently and the load balancing is not performed to the maximum degree. In this paper we propose a new architecture called the Delegation Web Service (DWS) for selecting the web service more efficiently and with maximum load balancing. The load balancing is achieved by grouping the web services of similar type from the registry by the DWS for each consumer's request and it is predestinated to each monitored web service. The monitoring of QoS parameters such as response time, efficiency, round trip time are done using the Web Service Distributed Management (WSDM) standard, since it has the better method and specifications.

Index Terms: Delegation Web Service, load balancing, Quality of Services, response time, Service selection, Service oriented architecture, WSDM

1. INTRODUCTION

A Web service is a programmable Web application that is universally accessible through standard Internet protocols [1], such as Simple Object Access Protocol (SOAP). Web service technology is becoming more and more popular in many practical application domains, such as electronic commerce, flow management, application integration, etc. It presents a promising solution for solving platform interoperability problems encountered by the application system integrators.

With the ever increasing number of functional similar web services being made available on the Internet, how to distinguish the best Web service from others becomes an urgent problem to be solved. Web Service Selection is a key component in service-oriented architecture [2].

- First Author- Research Scholar in Computer Science and Engineering, Bharathiyar University Coimbatore, India.
- Co-Authors- Department of Computer Science & Engineering, Pondicherry Engineering College, Pondicherry, India.

The selection of web service is usually based on the functional requirements of the consumer but those web services may not able to provide the quality the consumer expects.

Consumer requirements may include not only functional aspects of very depends on the ability to describe and to match QoS offers and demands, in addition to functional capabilities [3], [4], [5], [6].The web services has to provide a good quality of service to the consumer. The Web Service Selection based on QoS parameters become the challenging task in current trend. Quality of Service is an aggregated metric for describing characteristics of systems in areas, such as networks and distributed systems. According to Liu Sha [7], QoS based web service selection mechanisms plays an essential role in service-oriented architectures, because most of the applications want to use services that accurately meet their requirements.

To overcome the above mentioned drawbacks of previous works, a new approach has been proposed which offers a better solution for implementing web service load balancing. It also reduces the complexity of the work in selecting a particular web service and provide less components compared to the previous selector approaches. In this proposed work, each similar type of web service has one Delegation Web Service and it is predestinated thereby we can solve the problem of load balancing. It also simplifies the architecture as well as the Delegation Web Service's interface. Moreover in this paper the manageability of resources can be performed efficiently. The standard called WSDM has been provided for the resource monitoring. WSDM includes Management using Web Services (MUWS) and Management of Web Services (MOWS). The MUWS and MOWS of the WSDM standards are used to monitor the QoS of each of those similar web services consumed by the Delegation Web Service. The availability of the web service has been checked by using Web Service Ping operation which is called as a diagnostic tool. With help of these standards the accuracy of the web service can be maintained greatly.

The remainder of this paper is outlined as follows: In section 2 we briefly present the framework to monitor the QoS parameters and also the processing states of the request , in Section 3 we present the related researches, in Section 4 we describe the Delegation Web Service (DWS) architecture's concepts and the selection process steps, in Section 5 the selection algorithm based on strategy pattern is discussed, in section 6 the process to balance the load is discussed, in section 7 the implementation and its results is discussed and Section 8 concludes the paper and presents the future work.

2. BACKGROUND

2.1 WSDM Framework

2.1.1 Framework Description

The WSDM standard specifies how the manageability of a resource is made available to manageability consumers via web services. Endpoints that support access to manageable resources are called manageability endpoints. The implementation behind manageability endpoints must be capable of retrieving and manipulating the information related to a manageable resource.

The focus of the WSDM architecture is the manageable resource. The manageable resource must be represented as a Web service. In other words, management information regarding the resource must be accessible through a web service endpoint. To provide access to a resource, this endpoint must be able to be referenced by an endpoint-reference (EPR). The EPR provides the target location to which a manageability consumer directs messages. The manageable resource may also direct notifications of significant events to a manageability consumer, provided the consumer has subscribed to receive notifications. Thus, WSDM covers three modes of interaction between a manageable resource and a manageability consumer. These modes of interaction are as follows:

- A manageability consumer can retrieve management information about the manageable resource. For example, the consumer can retrieve the current operating status of the manageable resource or the current state of the process running on the manageable resource.
- A manageability consumer may affect the state of some manageable resource by changing its management information.
- A manageable resource may inform, or notify, a manageability consumer of a significant event. This mode of interaction requires the manageability consumer to subscribe to receive events on a desired topic.

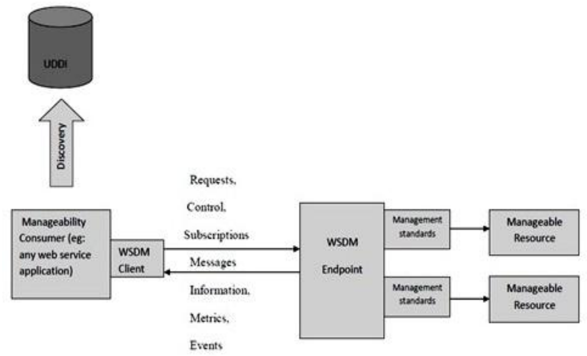


Fig. 1: WSDM Framework

2.2 WSDM SPECIFICATIONS

WSDM consists of two specifications: Management of Web Service (MOWS) and Management using Web Service (MUWS).

WSDM-MUWS

MUWS enables management of distributed information technology (IT) resources using Web services. Many distributed IT resources use different management interfaces. By leveraging Web service technology, MUWS enables easier and more efficient management of IT resources. This is accomplished by providing a flexible, common framework for manageability interfaces that leverage key features of web services protocols. Universal management and interoperability across the many and various types of distributed IT resources can be achieved using MUWS.

The types of management capabilities exposed by MUWS are the management capabilities generally expected in systems that manage distributed IT resources. Examples of manageability functions that can be performed via MUWS include:

- monitoring the quality of a service
- enforcing a service level agreement
- controlling a task
- managing a resource lifecycle

WSDM-MOWS

MOWS provide mechanisms and methodologies that enable manageable web services applications to interoperate across enterprise and organizational boundaries. It is used to publish Web Service's QoS parameters. WSDM's Management of Web Services specification extends MUWS to define how to specifically manage a resource that is a web service. Web services, like other resources, have identity, metrics, configuration, and other capabilities to enable management. For web services the compose-ability characteristic is especially interesting because it allows the business function of a service and the management function for a service to be composed together into a single service.

2.3 REQUEST PROCESSING STATES

A web service endpoint accepts and processes messages targeted at it requests. Every request goes through a number of states (e.g. received, processing, completed or failed) as defined by the [WSLC] and extended here.

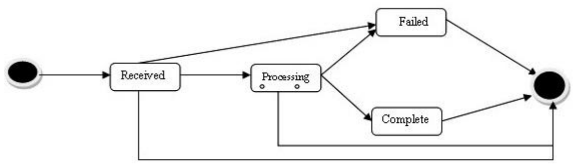


Fig. 2: Request processing states

Following is a list of elements corresponding to the top-level states of the request processing state model (Fig 2).

Request Received State

This element corresponds to the Received top-level state which means that the web service endpoint has accepted a request to perform one of the service's functional responsibilities. This state represents the earliest point at which the manageability provider knows that the request was dispatched to the web service endpoint being managed.

Request Processing State

This element corresponds to the Processing top-level state which means that the web service endpoint is doing some internal processing/execution to fulfill the requested function. This state represents the earliest point at which the application module or business logic begins processing the request. For example, if the application server queues the request before dispatching it to the business logic, the time difference between "request received" and "processing" will include the duration the request was queued.

Request Completed State

This element corresponds to the Completed top-level state which means that the web service endpoint successfully completed requested function returning results to the requester.

Request Failed State

This element corresponds to the failed top-level state which means that the web service endpoint encountered an error and didn't complete the requested function, returning error/fault to the requester.

The RequestProcessingStateType XML Schema type is declared as follows.

```
<xs:complexType
name="RequestProcessingStateType">
<xs:complexContent>
<xs:extension base="muws:StateType"/>
</xs:complexContent>
</xs:complexType>
```

An instance of the request processing state information represented in XML may appear as shown in the following example

```
<my:RequestProcessingStateInformationElement
xsi:type="mows:RequestProcessingStateType">
<my-soap:SerializationState>
<mows:RequestProcessingState/>
</my-soap:SerializationState>
</my:RequestProcessingStateInformationElement
```

3. RELATED WORKS

According to Liu Sha [7], the Web service selection is usually driven only by functional requirements, which can’t guarantee the real-time validity of the web services selected. So he proposed a QoS based Web Services Selection Model (WSSM-Q) to provide QoS support for service publishing and selection. In the model, the QoS of web services is managed, including defining the QoS model, collecting the QoS information, computing and maintaining the QoS data. Upon the QoS management, the web services that match the requirements of consumers are ranked for selection according to the overall QoS utility. In [8], Web service architecture employs an extended UDDI registry to support service selection based on QoS, but only the certification approach is used to verify QoS and no information is provided about the QoS specification.

According to Diego Zuquim Guimarães Garcia and Maria Beatriz Felgar de Toledo [9], an extended web service architecture can be used to support QoS management. In this approach, QoS information derived from policies based on WS-Policy is encapsulated inside QoS Policy structures stored in UDDI registries. Each element of a QoS Policy structure can be associated with a Technical Model (tModel) structure, which allows specification, standardization and reuse of QoS-related concepts. Furthermore, the extension allows the use of brokers to facilitate service selection according to functional and non-functional requirements, and monitors to verify QoS attributes. According to S.Ran [10], a component called certifier and a new extension of UDDI is modeled. The certifier’s role is to verify service provider’s QoS claims and the new UDDI registry is a repository of registered web services with lookup facilities. The new registry differs from the current UDDI model by having information about the functional description of the web service as well as its associated quality of service registered in the repository. Lookup could be made by functional description of the desired web service, with the required quality of service attributes as lookup constraints. The new role in this model is the Web service QoS certifier that does not exist in the original UDDI model. The certifier verifies the claims of quality of service for a web service before its registration.

Y.Lee [11] introduces WSQMS (Web Service quality Management System) to measure QoS of web services. It further introduces WSQDL (Web Service Quality Description Language). It advocates the use of the QoS enhanced UDDI for web service selection. Tao Yu et al. [12] design the service selection algorithms to meet the end-to-end QoS constraints. Their works are not focused on the trustworthiness of QoS criteria of a service. Although the global quality constraints can be satisfied, service selection may not be locally optimized. Therefore, good component service often fails to exert its potential and embody its personality. The work was proposed in the reference [13], which presented a model of reputation-enhanced QoS-based web service discovery that combines an augmented UDDI registry to publish the QoS information and a reputation manager to assign reputation scores to services. However, it only described abstract service matchmaking, ranking and selection algorithm. The references [14], [15] introduces WS QoS (Web Service QoS), a framework for QoS based selection and monitoring of web services. It advocates the use of a Web Service Broker for QoS based web service selection. Frameworks to support QoS verification with the goal of guarantying quality levels are described in [16], [17], [18].

The works discussed above may have certain limitations and involves complexity in monitoring the QoS parameters efficiently and also the load balancing may not be provided to maximum degree. So a new approach has been proposed to overcome the limitations of the existing works.

4. DELEGATION WEB SERVICE ACHITECTURE

4.1 Concept

The new approach that is the Delegation Web Service as selector architecture is predestinated to implement web service load balancing. In this approach a Delegation Web Service for each web service type is made, as this simplifies the architecture as well as the Delegation Web Service's interface. However, one Delegation Web Service can also be used for multiple web service types. The Delegation Web Service does not implement any functional parts. It delegates the functional request to corresponding web services. This DWS is used to perform the load balancing factor in an efficient way by grouping all the similar web services requested by the consumer into its register module and there by assigning the priority value to all the grouped web services and it will use the particular web service with the highest priority value. In case if that particular service gets overloaded then immediately it will use the next service in the prioritized order from the register module of DWS. Moreover it will consume functional as well as non-functional parts from the used web services. Non-functional parts will include MOWS and web service Ping, as well as MUWS from multiple MUWS web services. The Delegation Web Service will then decide which web service it delegates to. The decision is based on the consumer's functional requirements and the non-functional requirement that is the QoS preferences.

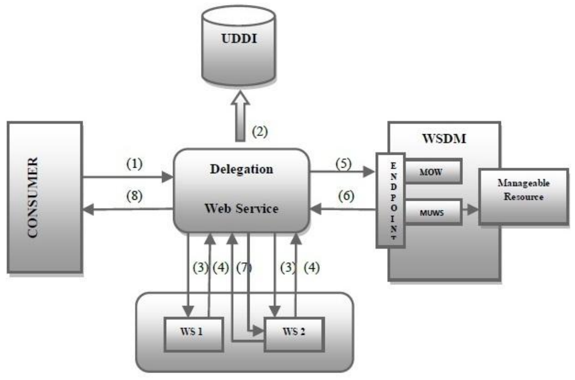


Fig. 3: Proposed Architecture

4.2 Selection Process Steps

Following are the steps to select the best Web Service using the Delegation Web Service:

(1) In step 1 the consumer requests the Delegation Web Service (DWS) for the best web service by giving the functional requirements along with the QoS preference.
(2) In step 2 the DWS look for similar web services according to the functional requirements of the consumer from the UDDI registry.
(3) In step 3 the DWS send the request to publish the QoS parameters of each web services.
(4) In step 4 the web services publish their QoS parameters to DWS.
(5) In step 5 DWS request the WSDM to monitor the QoS parameters of the consumer's preference of each web services by using the selection algorithm which is described below.

(6) In step 6 the WSDM response with the QoS metrics value and report the best Web Service.
(7) In step 7 the request will be sent to the best web service that has been reported by the WSDM and that particular Web Service will then respond to the DWS.
(8) In step 8 the best web service meeting the requirement and the QoS preference will be sent to the consumer.

5. SELECTION ALGORITHM

```
Input: Web Services ($W_{I}, W_{I+1}$............$W_n$) with QoS constraints

Output: Best performing Web Service $W_k$ satisfying QoS constraints

Begin

DecisionContext → Best Web Service

//check for service identification

If $W_i$ !Ɛ $S_L$<Service List>

AddService($W_i$) to $S_L$ // adding service to the service list $S_L$

Assign $W_i$ = {EPR, IP, EPRMUWS} // assigning endpoint reference, IP
and the MUWS EPR to the added Web Service

While ( ServiceList != empty) do

ServcieList→null throw

List is Empty Else

getstrategy(QoS)  //Getting the QoS parameter to be monitored with the
constraint

DecisionStrategy(QoS for all $W_k$ ($S_L$))  // QoS monitoring of the Web
Services in the list

Compare (Result (QoS ($W_k$($S_L$)) ) )  // To compare all the web services
result of the QoS from the list

Select ($W_k$($S_L$)) // Selecting the best Web Service

Return Best WebService($W_k$)  // returning the best web services
satisfying the QoS constraints

End
```

The above algorithm is based on the strategy pattern to select the best Web Service by monitoring the QoS parameters specified by the user. The strategy pattern based selection algorithm yields easy to implement algorithms and to use multiple diverse selection algorithms. The decision context of the algorithm is to return the best web service to the Delegation Web Service. At first it checks for the service identification to see that particular service belongs to the service list if not it will added to the service list. Then the end point reference will be assigned to it and also the corresponding MUWS address to which it should get monitored.

After that get the QoS parameters to be monitored for all the similar web services which have been sent by the Delegation Web Service and those corresponding parameters will be then monitored. Then the result of the monitored parameters of all the web services in the list will be compared and the decision will be made to select the best web service. After the decision the result will be sent to the decision context and then it will be responded to the Delegation Web Service. Multiple algorithms can be defined for the same QoS parameter depending on the constraints.

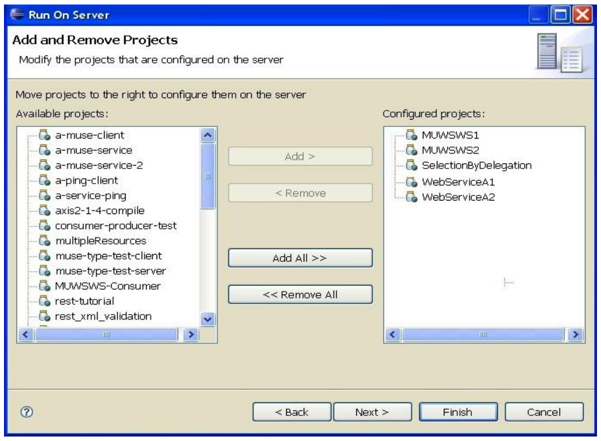
6. MANAGEMENT OF BALANCING THE LOAD

In the proposed work the load balancing factor is concerned greatly. For every consumer request the DWS look into the registry and group all the similar web services meeting the functional requirements. In this approach for each type of web services a Delegation Web Service is provided and also it is predestinated in sending the request to the web services to implement web service load balancing to maximum. Each web service will be provided with some threshold value and by reaching above the value will make the service to get overloaded. Since for each type a DWS is provided it is easy to balance the load when it gets overloaded by making the next subsequent monitored web service to be available.

From the consumer the functional requirement needed will be requested to DWS along with the QoS preferences. Then the DWS identifies the type of service based on the functional requirement requested by the consumer and will look into the registry by specifying the name of the service. The Find module of the UDDI registry will display all the services of the similar type. The DWS will then group all the similar web services into its registry module. Then each of these similar types of web services will be monitored and assigned the priority value so that the DWS will be able to predestinate the request of the consumer's to the web service. This mechanism provides the maximum degree of load balancing compared to the other approaches.

7. IMPLEMENTATION AND EXPERIEMENTS

The proposed system is implemented using the Eclipse IDE and Apache Tomcat server as the service deploying platform. Apache Muse a java based implementation is used to implement the WSDM standards. In the Apache Muse framework first create a WSDL specifying the Delegation Web Service interface and then create DelegationAsSelectorResource.rmd. Secondly add the Operation getBestWS, for returning the Address of the web service with the best QoS. The DelegationAsSelector.wsdl needs to contain all functional operations, of the web service it delegates to. The code is then imported to the Eclipse and then being tested with the TestServlet to be run on the server as shown in the Fig 4.



Run On Server

Add and Remove Projects

Modify the projects that are configured on the server

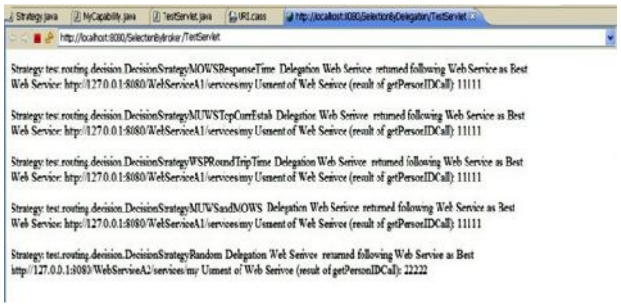
Move projects to the right to configure them on the server

Available projects:
- a-muse-client
- a-muse-service
- a-muse-service-2
- a-ping-client
- a-service-ping
- axis2-1-4-compile
- consumer-producer-test
- multipleResources
- muse-type-test-client
- muse-type-test-server
- MUWSWS-Consumer
- rest-tutorial
- rest_xml_validation

Configured projects:
- MUWSWS1
- MUWSWS2
- SelectionByDelegation
- WebServiceA1
- WebServiceA2

Fig. 4: Testing of the services using the Test Servlet

Then the result of the best web service for different QoS parameters in the service list based on the proposed selection algorithm will be displayed as shown in the Fig. 5

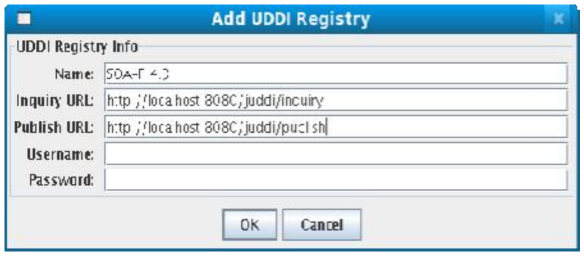


http://localhost:8080/SelectorByBroker/TestServlet

```
Strategy: test.routing.decision.DecisionStrategyMOWSResponseTime Delegation Web Service returned following Web Service as Best
Web Service: http://127.0.0.1:8080/WebServiceA1/services/my Usment of Web Seivce (result of getPersonIDCall): 11111

Strategy: test.routing.decision.DecisionStrategyMUWSTcpCurrEstab Delegation Web Serivce returned following Web Service as Best
Web Service: http://127.0.0.1:8080/WebServiceA1/services/my Usment of Web Seivce (result of getPersonIDCall): 11111

Strategy: test.routing.decision.DecisionStrategyWSPRoundTripTime Delegation Web Serivce returned following Web Service as Best
Web Service: http://127.0.0.1:8080/WebServiceA1/services/my Usment of Web Seivce (result of getPersonIDCall): 11111

Strategy: test.routing.decision.DecisionStrategyMUWSandMOWS Delegation Web Serivce returned following Web Service as Best
Web Service: http://127.0.0.1:8080/WebServiceA1/services/my Usment of Web Seivce (result of getPersonIDCall): 11111

Strategy: test.routing.decision.DecisionStrategyRandom Delegation Web Serivce returned following Web Service as Best
http://127.0.0.1:8080/WebServiceA2/services/my Usment of Web Serivce (result of getPersonIDCall): 22222
```

Fig. 5: Result of the Best Web Service in the Service List

For inquiry and publishing of the service UDDI browser and jUDDI registry is used. The jUDDI is the java implementation of the UDDI registry. The module publishes the service in the UDDI registry with the WSDL file and extracts the QoS information of the service. The service can be published and inquired by specifying the URL into the registry information module as shown in the Fig. 6



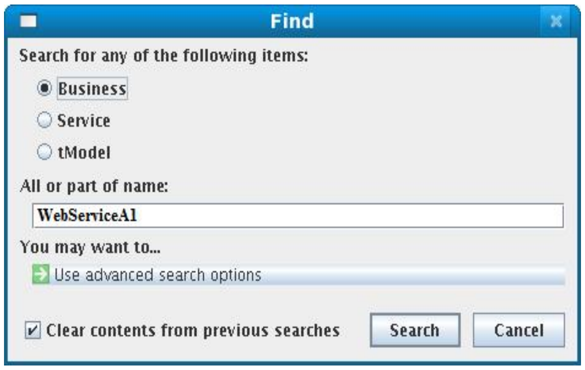
Add UDDI Registry

UDDI Registry Info

Name:	SOA-F 4.3
Inquiry URL:	http://localhost:8080/juddi/inquiry
Publish URL:	http://localhost:8080/juddi/publish
Username:	
Password:	

Fig. 6: The info window for publish/inquiry of service

After the service has been published it can be inquired at any time for the functional implementation of the service. The customer queries the Find module for services with functional and QoS requirements. The Delegation Web Service gets functional matched services from the registry and then requests those web services for the QoS attributes of each Web services and then it will be monitored by the WSDM framework. The Find module of the registry provides the search with three options as shown in the Fig. 7



Search for any of the following items:
[✓ Business] Service tModel
All or part of name: WebServiceA1
You may want to...
Use advanced search options
[✓] Clear contents from previous searches

Fig. 7: The Find window to search any published services

The previous work [7] is taken for experiment and compared with the proposed work. Both the works have been experimented in their own form of selection process. Then the experimental result of both the works is calculated by evaluating the parameters such as responsiveness of the web service selection per request and efficiency. These parameters are defined as follows:

Efficiency: Refers to the overall quality of the service selection and the load balancing factor.

Responsiveness: Defines the time taken to response for each consumer's request/ second

The comparison graph of efficiency for the proposed architecture and the previous work [7] is shown below (Fig. 8) in which for example the consumer 1 gets the efficiency of 94.5 percent with the proposed work whereas the same consumer gets the efficiency with 92.54 percent. The Fig. 9 shows the responsiveness of the web service selection by the selector to the consumer's request.

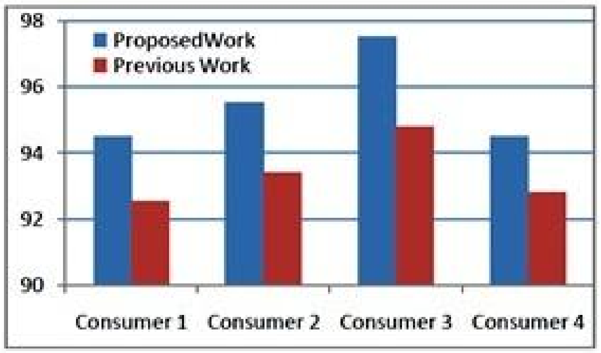


Fig. 8: Efficiency of the web service using proposed and previous techniques

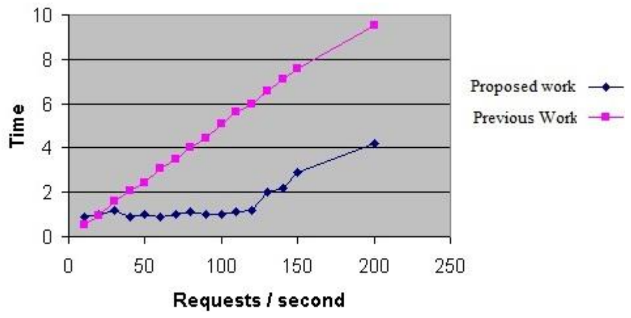


Fig. 9: Responsiveness of web service selection to customer's request

8. CONCLUSION

Web Service Selection based on QoS value is an important research area. The selection of web service are usually based on the functional requirements of the consumer but those web services are not able to provide the accuracy in their services The decision should always be based on QoS parameters important to the specific consumer. The proposed Delegation Web Service as selector architecture hides the web services and also it is predestinated to implement web service load balancing. For each type of Web Services a Delegation Web Service is provided for better load balancing. The QoS monitoring is done efficiently by using the WSDM and the best web service is selected based on the selection algorithm being proposed. Future and upcoming work is to include more QoS parameters to get monitored by the WSDM and to have one Delegation Web Services for multiple web services types.

9. REFERENCES

[1] Ryman, "Simple object access protocol (SOAP) and Web services", *Proceedings of the 23rd International Conference on Software Engineering (ICSE 2001)*, Toronto, Ontario, Canada, pp. 689, 2001.

[2] Berbner, T. Grollius and N. Rep, "An approach for the Management of Service oriented Architecture (SoA) based Application Systems", *in Proceedings of the Enterprise Modelling and Information Systems Architectures, Oct.2005*.

[3] G. Alonso, F. Casati, H. Kuno, and V. Machiraju. "WebServices: Concepts, Architectures and Applications", Springer, 2004.

[4] M. P. Papazoglou and D. Georgakopoulos. "Service oriented computing" - guest editorial. *Communications of theACM*, 46(10):24–28, 2003.

[5] H. Ludwig, "Web services QoS: external SLAs and internal policies or: how do we deliver what we promise?", In *Proc. of the Int'l Conf. on Web Information Systems Engineering Workshops*, pages 115–120. Springer, 2004.

[6] E. Lee, W. Jung, W. Lee, Y. Park, B. Lee, H. Kim, and C. Wu, "A framework to support QoS-aware usage of Web services", In *Proc. of the Int'l Conf. on Web Eng.*, pages 318–327.Springer, 2005..

[7] Liu Sha, Guo Shaozhong, Chen Xin and Lan Mingjing, "A QoS based Web Service Selection Model", *in Proceedings of the IEEE International Forum on Information Technology and Applications, pp.353-356, 2009.*

[8] M. A. Serhani, R. Dssouli, A. Hafid and H. Sahraoui, "A QoS broker based architecture for Efficient web services selection", *in Proceedings of the IEEE International Conference on Web Services ICWS, Proceedings, pp. 113-120,2005.*

[9] Diego Zuquim Guimarães Garcia and Maria Beatriz Felgar de Toledo, "A Web Service Architecture Providing QoS Management", *in Proceedings of the Fourth Latin American Web Congress, 2006*.

[10] S. Ran, "A Model for Web Services Discovery with QoS", *in Proceedings of the ACM SIGecom Exchanges, pp.1–10, 2003*

[11] Y. Lee, "Quality Context Composition for Management of SOA Quality", 2008 *IEEE International Workshop on Semantic Computing and Applications, pp. 117–122, Sept. 2008.*

[12] Tao Yu, Kwei-Jy Lin, "Service Selection Algorithms for Web Services with End-to-end QoS Constraints", *In Proc. Of the IEEE International Conference on E-Commerce Technology, 2004.*

[13] Z. Xu, P. Martin, W. Powley and F. Zulkernine,"Reputation-enhanced QoS-based web services discovery,"*In Proc. of the IEEE Intl. Conf. on Web services, pp.249-256, 2007.*

[14] M. Tian, A. Gramm, H. Ritter, J. Schiller, "Efficient Selection and Monitoring of QoS-aware Web services with the WS-QoS framework", *In proceedings of the 2004 IEEE/WIC/ACM International Conference on Web Intelligence, pp.152-158, Sept. 2004.*

[15] D. A. DMello, V. S. Ananthanarayana and Santhi T, "A QoS Broker Based Architecture for Dynamic Web Service Selection", *in Proceedings of the IEEE Second Asia International Conference on Modelling & Simulation, pp.101-106, 2008.*

[16] H. Ludwig, A. Keller, A. Dan, R. P. King, and R. Franck, "Web Service Level Agreement Version 1.0Specification*", IBM, 28th Jan, 2003.*

[17] S. Ran., "A framework for discovering Web services with desired quality of services attributes", *In Proc. of the Int'l Conf.on Web Services, pages 208–213. CSREA Press, 2003.*

[18] Z. U. Singhera "Extended Web services framework to meet non-functional requirements", In *Proc. of the Symposium on* Applications *and Internet, pp-334-340, 2007.*