International Journal of Electrical and Computer Engineering (IJECE)
Vol. 10, No. 1, February 2020, pp. 396~406
ISSN: 2088-8708, DOI: 10.11591/ijece.v10i1.pp396-406

Service selection in service oriented architecture using probabilistic approach and asynchronous queues with interceptor validation

Balika J. Chelliah, K. Sathish, S. Arun Kumar
Department of Computer Science and Engineering, SRM Institute of Science and Technology, India

Article Info

Article history:

Received Feb17, 2019
Revised Aug16, 2019
Accepted Aug 30, 2019

Keywords:

Asynchronous queue
Interceptor
QoS
Sanitization
Service orientedarchitecture

ABSTRACT

In service Oriented Architecture, many services are offered with similar functionality but with different service quality parameters. Thus the service selection using a deterministic approach causes conflicts and inefficient results. We use asynchronous queue to model the service inventory architecture avoiding unnecessary locking of resources and thus allowing a provision to consumers to get their required services without intervening and with temporally decoupled fashion. Actually this kind of service selection strategy is considered in regards with game theory to eliminate fluctuations of queue length. It offers a discrete random service which is equal to some request requested by consumers, it means service can be provided based on probability mass function as a substitute of deterministic decisions for selecting a proper service provider as of the consumers. Once the request is taken out from the queue, it is delivered to the interceptor that has validation and sanitization module. It thus reduces the peak queue length and reduces periodic fluctuations in the queue length.

Corresponding Author:

Balika J. Chelliah,
Department of Computer Science and Engineering,
SRM Institute of Science and Technology,
Chennai, India.
Email: jbalika@outlook.com

1. INTRODUCTION

A Service Oriented Approach provides interactions with loosely coupled services that operate independently. SOA provides service reuse, thus avoiding starting from the scratch when upgrades and modification is needed. This saves time and money. The general principles that we follow involve loose coupling; maintaining minimal dependencies among services and only requiring them to maintain awareness of each other, discoverability; services are supplemented with communicative meta data by which they can be remotely scanned and discovered, service contract; services and consumers adhere to a defined agreement regarding service parameters and QOS. Components of SOA include services, providers, requesters, and locators.

Need of providing large services to number of consumers leads to require the web services asynchronous in nature. The way of providing asynchronous services reduce the communication overhead and services can be effectively offered even in remote area which can avoid the awaiting time for responding to a service request. By making a service call asynchronous, with a parallel distinct return message, we allow the requestor to continue execution while the provider is processing for the respond. This does not imply that synchronous service behaviour is ineffectual, just that experience shows that asynchronous service is preferred, especially when the communication costs is high or network latency is fluctuating and unpredictable.

By making service calls asynchronous, it permits providers to use number of process to switch various consumer requirements instantaneously. Asynchronous modes of operating require much more support functions for maintenance and flexibility, other than just responding quickly to the client. Provision selection in on demand environment is the practice of methods in choosing and rendering quality of services (QoS) to consumers. First, dual interfaces are specified; the requestor will request the on demand service by the way of passing request message. Interface between client and service provider enabled to accept the request and same thing is to be monitored effectively by that interface.

This paper is organized in such a way that Section 2 contains related work, Section 3 states the proposed system and its architecture, techniques and algorithm Section 4 demonstrates services selection strategy, Section 5 shows experimental analysis and in Section 6 conclusion and future work has been stated.

2. RELATED WORK

In service oriented architecture (SOA), service agents could discover numerous service providers which gives similar function with dissimilar quality of service (QoS). Cooperative mixed strategy (CMS) [1] is used to tackle the service selection problem (SSP) of time-sensitive services exhausting the theory of games that solves inconstant-sum non-cooperative n-person dynamic game. MS deals users an optimized probability mass function in its place of a deterministic choice to choose an appropriate provider from the candidates and removes the variation of queue length, and increase the whole performance of SOA significantly. A SOA modeling and design grounded on the notion of service component and standard UML modeling constructs [2] that provide the interface of a service component works better than the list of operation signatures to stipulate the complete contract between the service provider and consumer. The Web services outsourcing manager framework for dynamic business processes uses an XML-based annotation document is suggested to seize the business requirements and used to dynamically produce quest scripts for a progressive Web services discovery engine to catch Web services from UDDI registries and Web service inspection language documents [3, 4]. Service selection in dynamic environment is the treatment of techniques in choosing and offering Quality of Services (QoS) to consumers. The dynamic nature of web services is captured by an environment collecting runtime information of web services. Utilizing the statuses, availability estimation on dynamic scale [5] is proposed along with multiple QoS criteria's such as cost and performance implies Status Identification based service selection. The key requirements for commercial service oriented architectures [6] with strict latency requirements are detailed. Based on this, recognizing the capabilities of the hosting environment and effectively matching those for the wanted service configuration are required to enable hosting services with strict QoS requirements. A method [7, 8] to gather context based data of service providers is given as input to the rule based system and appropriateness of every SP is obtained as output including recommended stipulations regarding their usage. Service Oriented Architecture (SOA) which comprises of many consumers meets with reliability and availability problem. A service selection and replication based on Quality of Service (QoS) mechanism [9] is able to control service replica properly in order to, increase service instance, service replication is adopted. To control the selection and replication of service, this paper proposes. Thus the system is able to control service replica properly and force service distribution for effecting regular consumer as few as possible.

A state of art interceptor module and highly probabilistic queue management technique are introduced in this chapter. The lack of existing validation system that isolates the request validation from service rendering forces the queue to be overcrowded with invalidated requests and insecure hits [10-15]. The least effort to the validation of the existing system leaves the service provider insecure and busyrowding a highly rated service providers' queue will deprecate the quality of service provider which is meant to provide and vice versa. The survey shows that there is a need for such a model that deals the queue length as a random variable that most likely chooses a queue with lower length and with best effort service.

The interceptor is introduced to have an efficient flow of service without any web security threat such as passing requests from clients without proper validation. This approach further integrates a sanitizer module which discards the invalid requests sent by any client and promotes efficiency [16-20]. Further it establishes a time-sensitive service with minimum waiting time and consistent performance in quality. The interceptor allows a valid request to hit the service and the response will be given accordingly. In case of any delay the interceptor duly intimates the client and the client will be kept posted in time.

3. ARCHITECTURE OF QoS IMPROVISATION AT QUEUE LEVEL

Interceptor and probabilistic approach attempts to design an efficient flow of service without any web security threats. As it decouples the sanitizer module from the service provider, the overall burden in web service provider is reduced considerably [21-25]. This approach establishes a time sensitive service with minimum waiting time and consistent performance. In the architecture depicted in Figure 1, there is the probabilistic model which distributes the request randomly and asynchronously but always tries to insert the request into the queue which has minimal waiting time. Thus it produces normalized probabilistic model. Once the request is taken out from the queue, it is delivered to the interceptor that has validation and sanitization module.

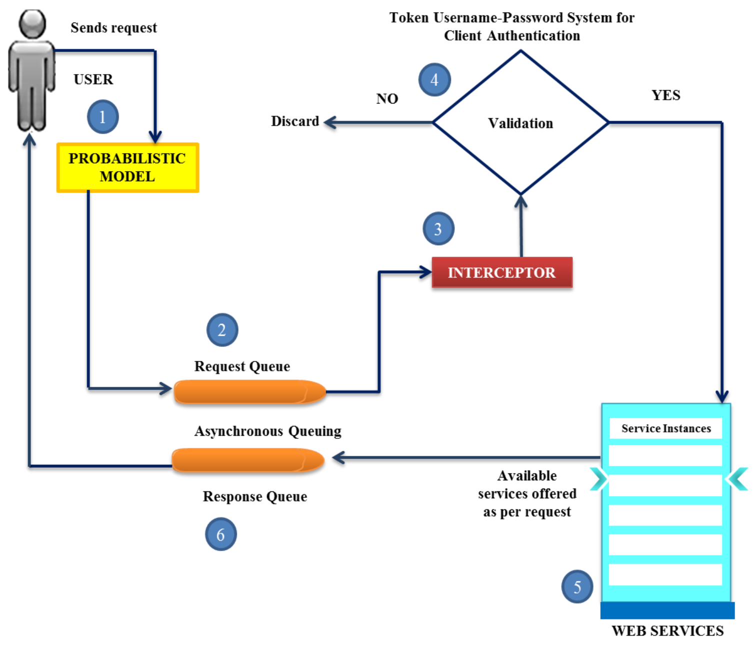


Figure 1. Architecture of service delivery with interceptor and probabilistic approach

Registration of service requestor, Signature &IP Verification, Attempts Check and Sanitizer are the processes in interceptor module. The motivation behind the interceptor module is to isolate the request validation from the service provider. This architecture shown in Figure 2 will completely allow the interceptor to be functioning independently and coherently with the service provider. Thus it reduces the burden of the service provider to a greater extent. The service provider never keeps any chores except the core business in hand. The round trip time will be stamped along with the service reply.

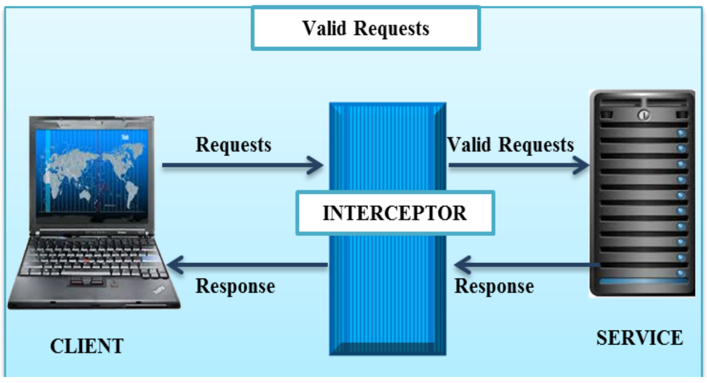


Figure 2. Interceptor module process valid requests in web service

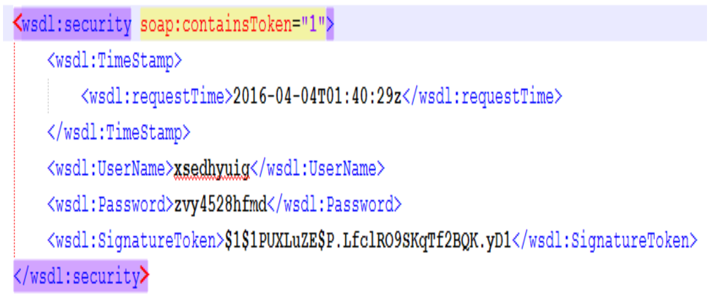
3.1. IP registration

The request Time (login time of the req) is a new tag which is enclosed for this and includes Token that takes values from 0 to 1, SignatureTokenSTclient. It performs the registration of the user with credentials and consequent signature validation.

```
If contains_Token == 0 Then
        DoRegistration(user_name,pwd);
Else if contains_Token == 1
        DoSignature_Validation(user_name, pwd,STclient)
        Do Registration(user_name,pwd)
While (Login_IP:LIP,Login_Time:LT,Token/seed,)
        {
        Generate Token
        Appends LT, LIP and Token / seed in
whitelist
        }
```

The cryptographically protected pseudo random number producer generates the token and the whitelist is populated with either the token or seed. The token is obtained by the client which calculates the Signature_Token as follows:

Signature_Token is computed by the MD5 algorithm with Token and pwd as parameters.



```
<wsdl:security soap:containsToken="1">
    <wsdl:TimeStamp>
        <wsdl:requestTime>2016-04-04T01:40:29z</wsdl:requestTime>
    </wsdl:TimeStamp>
    <wsdl:UserName>xsedhyuig</wsdl:UserName>
    <wsdl:Password>zvy4528hfmd</wsdl:Password>
    <wsdl:SignatureToken>$1$1PUXLuZE$P.LfclRO9SKqTf2BQK.yD1</wsdl:SignatureToken>
</wsdl:security>
```

Whitelist may further comprises additional attributes like username, password, Timestamp skew – TSS, No of Attempts – NoA, Threshold limit –TL, Interval – IT, Last Request Time – LRT. The default token validity is 12 hours and it will expire in case of NoA exceeding the threshold. Using a signature Token is better than the conventional authentication schemes, which aids in dealing of DOS. Although, occurrences of the demands are augmented by the invader, it is constrained that the mark/kernel would stand usable simply for the specified amount of period or until it's alive.

3.2. Interceptor module algorithm

The invalid request and corresponding error message are sent back to the client. Here for the signature verification module MD5 algorithm is used that takes token and password as parameters to generate the signature token as signature validation, which takes user name and password and gives yes to client considered as a token from the white list. White list gives the permitted tokens and black list lists non permitted tokens. If the token is not in the white list the request will be denied with the error message otherwise sent to IP verification module.

3.3. Signature verification

It validates the authenticity of the client by verifying the value in the Signature_Token tag obtained through the client request.

```
        Signature_Validation(user_name,password, STclient)
{
1. Consumea kernel from the Whitelist
2. If  kernel
spawn the tokens
```

```
    Else if Token
Do stagenumber 3
3. STINT = MD5 (pwd+Token )
4. If STclient = ST_INT {
Print success to client
Do IP_verfication(IPclient)
}
Else {
Print the Unacceptable Desires of fault dispatch
}
}
```

The choice of choosing a hash algorithm like MD5 is transforming the calculation simpler than retaining a pair of private and public keys that are encrypted by a further cryptographic algorithm.

3.4. IP verification

It checks whether the IP address of client is valid by using either whitelist or blacklist. The blacklist holds a set of client's IP address that made threat to the service provider. In IP verification, first step is signature validation which is followed by IP verification module. Considering IP white list and black List, any IP which is in the white list will be allowed otherwise it will be declined. If IP address is in IP white list and not in the IP black list then it is a success, it is an otherwise invalid request. Attempt check module finds the running time of any attempt and if the running time is greater than the constant time (allowed time) then the request will be invalid.

```
IP_Verfication(IPclient){
1. AssumeIpwhitelist and IPblacklist.
2. IfIPclient = =IpwhitelistorIPclient != IPblacklist {
Print Success
Do Attempts_Check()
}
Else {
Print Invalid Requests of Error message
}
}
```

3.5. Efforts plaid

Accessing demands on the specific period of value associated with the information depends proceeding with following limitations.

a. References of times must be represented in standard time format (UTC).
b. The recommended time refrences should be in Xml Schema Defnition:date Time format. ValueType must be used for other attribute.
c. The limit of threshold TL is defined as

$$T_L < \sum_{i=0}^{t1} (Requests)\, t_1, IP \quad (for\ heavy\ traffic)$$

$$T_L < \sum_{i=0}^{t1} (Requests)\, t_1, IP/\, n \quad (Average\ traffic)$$

$$T_L < \sum_{i=0}^{t1} (Requests)\, t_1, IP, k \quad (Requests\ from\ same\ IP)$$

n	*-number of requested URI in t1 time period*
k	*- Specific k URI from similar IP.*
TSS	*- maximum tolerance limit for the clock skewed between the sender and the recipient.*
IT	*- permitted time gap from* LT to *LRT.*

```
AttemptsCheck(){
    INT - CT
    If [CT < (LT – TSS * constant value)] {
    Print Error message to client as invalid request
    }
    Else{
    If (LT - LRT) > IT {
    If(NoA< TL)
     Call Sanitizer()
    Else{
    Print Error message to client as invalid request
     } } } }
```

3.6. Sanitizer

The schema caching process is maintained by the sanitizer which is the process of maintaining indigenous replicas of the representation. System can be supposed to be updated regularly when there is any change. This sanitizer can be easily manipulated with the help of name space particularly naming the folders. On considering the schema, the unrestrained incidence must be limited till further end, since that is the main reason for coercive parsing attack. Thus, these incidents would likely be detained with respect to the kind of application in use. As namespace injection attacks are likely to occur an each namespaces and related tags are monitored and these details are enlisted inside the whitelist in order to provide authentication. The predefined input policies of the ports and services should be enforced so that improved security will be enforced. The XSD Path is extracted from XML form of service request and checked up if empty, in case the value is null, the request will be forbidden and the error message will be conveyed to the client to avoid the parameter tampering attack. Further the threshold of the magnitude of the service request message is enforced and any message that exceeds the stipulated size would be rejected. Thus the coercive parsing attack is handled. Therefore the sanitizer module authenticates the data in contradiction of the XML stipulations and upholds the data correctness, completeness and structure.

3.7. Algorithm for probabilistic queue selection

The probabilistic approach tends to provide the probability distribution for service selection. It is useful in time-sensitive services. Probabilistic approach ensures best management of queues. Providing balance among providers helps retain quality parameter for further segments. Users send the requests which will be distributed through probabilistic model with optimal queue length and it helps you to choose the less crowded queue and that is given to the interceptors which checks the registration and verifies IP for sanitization. Crowding a highly rated service providers' queue will deprecate the quality that the service provider promises to provide and vice versa.

A synchronous queuing provides a synchronous service illusion that hides the validation logic for the service call into a shared call and then phases the inbound request on a service message queue for processing at a later period. If the service request validates then it can be placed on the queue for processing and a 'success' response issued to the requester immediately. A failure will cause an error response to the service consumer. The message is then abandoned and not placed in the queue. Both the request and response operations occur independently and thus provide both the server to return to their core processing after sending a request/response message. They can interact using messages in the form of non-blocking interrupts, thus providing much user experience.

4. SERVICES SELECTION STRATEGY

In service oriented architecture, service providers offer different quality of service (QoS), thus they encounter decision conflicts in selection to obtain an optimal service quality that result in poor performance.

4.1. Problem with pure strategy

The arrival of the series of service requests that come to web service environment follows the distribution of Poisson with as mean arrival rate. Every request of user is meant to select one service provider among the m parallel items $[M_i]_{i=1}^{m}$for finishing the demands. The user spares T_p time to finish planning routines such as binding provider and transmitting data. The parameters T_p, l, and μ which are presumed to be not a variable during the queuing duration of the service requests differ based on these requests and responses. Any request j needs the necessary details at time t of service level parameter and queue length from the broker before the selection of service provider.

The non-probabilistic strategy J is to select the service provider by which the request might anticipate to be completed in the minimum duration:In order to anticipate the request to be completed within the minimum duration, any pure strategy is selected as following:

$$\min T_i = (L_i + 1) / \mu_i$$

By assuming that the J chooses the service provider M1 and the ratio L_i/μ_i where, L_1/μ_1, i=2,3,...,m at t.

During evaluation of expected time, every request will choose the one with the lowest expected time. Considering the time period when M_1 acquires the position of the best provider, say D_t just before the interval t, the requestors of the service are incoming while this recess must have completed the similar opportunity to tie M1. Similar quality of service is not delivered when the queue length is lengthier than the queue at the time J arrives to queue on time $t+T_p$, It discovers the finding of queue is lengthier than its anticipation and it may not deliver the similar quality of service. Thus it causes periodic fluctuation in the queue length and causes a drastic increase in the peak queue length.

4.2. Optimizing quality parameter

Our top priority is to optimize the QOS parameter of service queues so as to provide a retain of quality of the service providers. A more balanced distribution using probabilistic approach can drastically utilize the remaining service providers of mediocre quality parameter as well as improve the efficiency of best by reducing its queue peak length.

5. EXPERIMENTAL RESULTS

We are considering three service providers which provide different quality parameter with the top one providing the best quality parameter and the bottom one providing least. We provide a distribution of service requests using probabilistic approach to each of the service provider. It is well noted that any number of request may arrive at a particular interval of time. The utility of asynchronous queues can further provide improve the user response operation.

5.1. Simulation setup

For our ease of evaluation we have chosen 3 service providers each with their dedicated asynchronous processing with the first one providing the highest service quality and the last providing with least service quality. Thus to asses them we generate a rational number of requests at every interval and record the requests distribution as well as the queue length of the providers. Figure 3 show therequests distribution based on QOS parameter.

To further illustrate we provide a queue length distribution line of service providers. The simulation is extended with the further setup that the 1000 queues are set up with its initial random queue length of mean 10 and SD3 is chosen from normal distribution. The web service request arrival follows the exponential distribution.

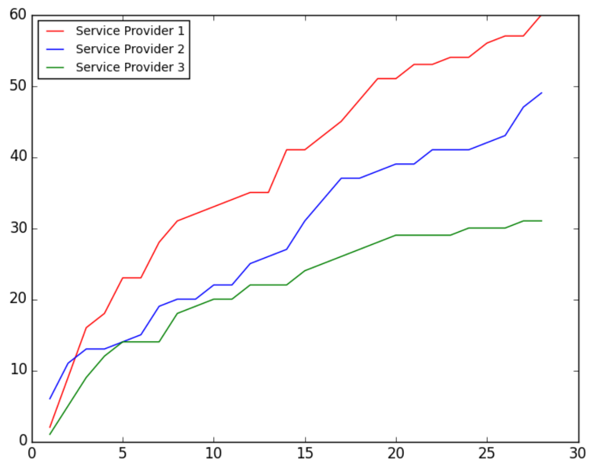


Figure 3. Requests distribution based on QOS parameter

At any moment to enque a service request, a random queue length is chosen from exponential distribution in which lambda value is the inverse of mean queue lengths. The simulation scenario is set to be Monte- Carlo simulation where the random sampling from probabilistic distribution is presumed to have a replica of real world. A million service requests were generated and the growths of the queues are recorded after every 1000 requests.

5.2. Results and discussion

Providing a balance of distribution of requests among the providers helps them to retain their quality parameter for further segments. Moreover it also makes use of quality of other queues than simply targeting the best provider. A probabilistic approach can ensure the best management of queues and eventually reduces peak queue length and unnecessary congestion.

5.2.1. Service system rendering performance

As in Figure 4 the performance sustains throughout the number of requests to the service provider increases. With the interceptor module, as the overcrowding requests will be validated before the service provider, the performance endures.

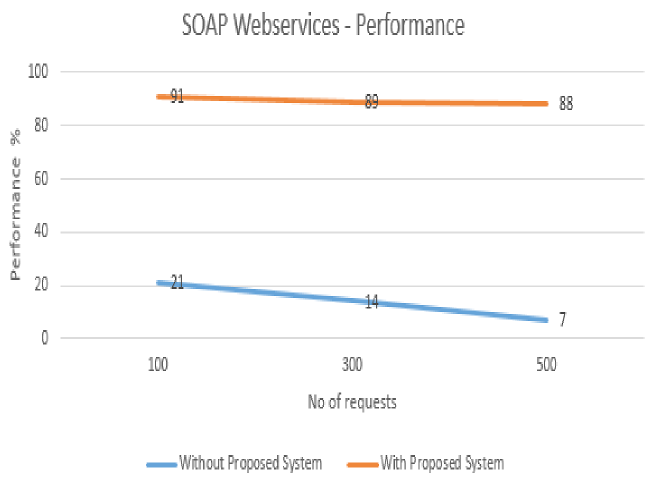


Figure 4. Performance comparison of interceptor system

As depicted in Figure 5 CPU usage is very low compared to the existing system and maintained below 25%. It is inevitable for the time sensitive service. It helps to manage the best possibility and provide the balance and helps to retain the quality parameters for the segments.

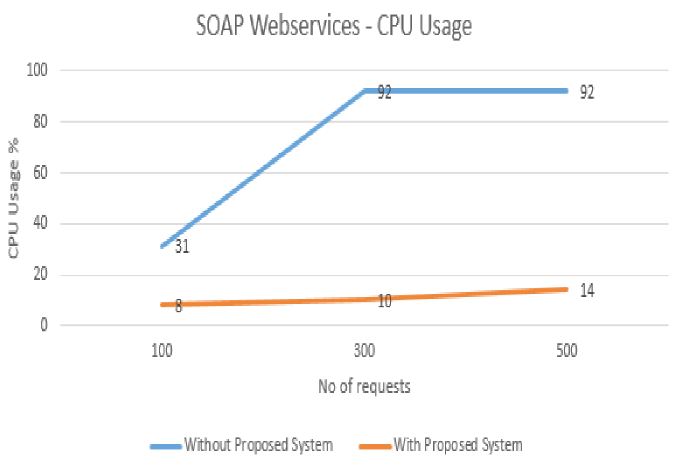
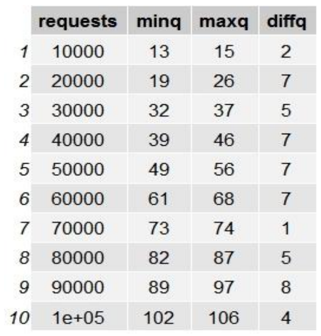


Figure 5. CPU usage comparison of interceptor system

Depending on the quality of parameter we can calculate our sample P_i. For our experiment we chose to provide the top provider with 0.58, second with 0.3, and the last one with 0.16 so as to prevent excessive fluctuation in the queue length. The difference between the maximum queue length and minimum queue length never had shown considerable deviation from the normal course. It means the requests are evenly queued as shown in the following figure and the sample of recorded requests. The length difference has the distribution from 2 to 10 that is really 0.025% of error in queuing. Thus the plot Figure6 shows our state of art probabilistic approach outperforms the pure strategy and consequently improves the quality of service.



	requests	minq	maxq	diffq
1	10000	13	15	2
2	20000	19	26	7
3	30000	32	37	5
4	40000	39	46	7
5	50000	49	56	7
6	60000	61	68	7
7	70000	73	74	1
8	80000	82	87	5
9	90000	89	97	8
10	1e+05	102	106	4

Figure 6. Queue difference distribution for 1 million request in probabilistic approach

5.3. QoS analysis-availability comparison

As depicted in Figure 7 using the Queue Optimization (QO) approaches improves the availability ratio in the range of 67% to 91%. The reliability is improved to the extent by the introduction of sanitizer and probabilistic approach. Since the interceptor keeps most of the invalid service requests away, the availability is drastically increased by 24% to 91%. The access count matches to the new high because of the probabilistic approach. The introduction of interceptor module and a stochastic model promise improvement in QoS parameters. The consistency in rendering SOAP services is maintained during scale up. The stochastic model assures better queue with minimal queue length. The Measured QoS Availability is increased up to 90% as in Figure 8.

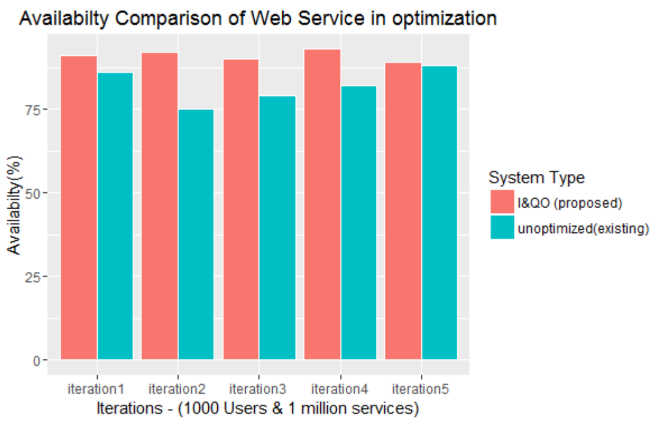


Figure 7. Availability comparison of web service in optimization

5.4. QoS analysis-reliability comparison

The reliability is Figure 8 increased from the average of 55% to the average 86% by the introduction of interceptor module and probabilistic queue mechanism. This integrated approach ensures the consistent reliability and provides the assurance in the quality standpoint. Avoiding unusual requests and placing them in the queue is likely to have minimal waiting time. This improves the stewardess into the greater level so that the bench marks of service level agreement would be maintained at par. The probabilistic approach does its part well and altogether it produces 31% increase in the reliability.

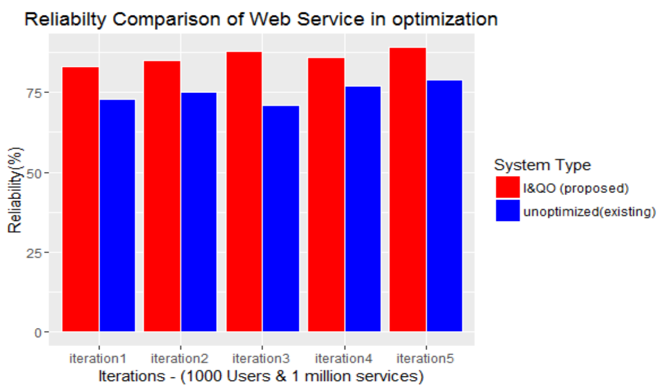


Figure 8. Reliability comparison of web service in optimization

6. CONCLUSION AND FUTURE WORK

Every user wishes to have their request processed faster and have the best response. Pure strategy using deterministic approach may cause poor performance. The interceptor module approach and probabilistic queue approach altogether improve the web service system performance at greater level. The separation of service request validation from the business entity ameliorates the performance of the system directly. The likelihood approach places the request into better queues that helps the request likely to not wait in a longer queue. Overall, the reliability and availability benchmarks are improved by the probabilistic approach and interceptor module.

Deterministic approach in service selection causes to choose the best service provider and may not result in its specified quality when there occurs numerous simultaneous requests because of the queue fluctuation. Probabilistic approach provides a probability for service selection and thus reduces queue fluctuation to prevent and loss of quality. This is very useful in time-sensitive services which require minimum waiting time and consistent performance in quality.

REFERENCES

[1] Temglit, N., Chibani, A., Djouani, K., & Nacer, M. A. “A Distributed Agent-Based Approach for Optimal QoS Selection in Web of Object Choreography,” *IEEE Systems Journal*, 12(2), 1655–1666. doi:10.1109/jsyst.2016.2647281,2018

[2] Zhu, W., Yin, B., Gong, S., & Cai, K.-Y., “An Approach to Web Services Selection for Multiple Users,” *IEEE Access*, 5, 15093–15104. doi:10.1109/access.2017.2722228 ,2017

[3] Tang, M., Zheng, Z., Kang, G., Liu, J., Yang, Y., & Zhang, T. “Collaborative Web Service Quality Prediction via Exploiting Matrix Factorization and Network Map,” *IEEE Transactions on Network and Service Management*, 13(1), 126–137. doi:10.1109/tnsm.2016.2517097, 2016.

[4] Hashmi, K., Malik, Z., Erradi, A., & Rezgui, A. “QoS Dependency Modeling for Composite Systems,” *IEEE Transactions on Services Computing*, 1–1. doi:10.1109/tsc.2016.2589244, 2016.

[5] Saleem, M. S., Ding, C., Liu, X., & Chi, C.-H. “Personalized Decision-Strategy based Web Service Selection using a Learning-to-Rank Algorithm,” *IEEE Transactions on Services Computing*, 8(5), 727–739. doi:10.1109/tsc.2014.2377724,2015.

[6] Hu, Y., Peng, Q., Hu, X., & Yang, R. “Time Aware and Data Sparsity Tolerant Web Service Recommendation Based on Improved Collaborative Filtering,” IEEE Transactions on Services Computing, 8(5), 782–794. doi:10.1109/tsc.2014.2381611, 2015.

[7] Ricardo J. Rabelo, OvidiuNoran and Peter Bernus, “Towards the Next Generation Service Oriented Enterprise Architecture,” IEEE 19th International Enterprise Distributed Object Computing Workshop, 2015

[8] SutheeraPuntheeranurak,"SOA selection and replication based on QoS," TENCON 2014 - 2014 IEEE Region 10 Conference, 2014.

[9] SaurabhShrivastava and Ashish Sharma, “An approach for QoS Based Fault Reconfiguration in Service Oriented Architecture”, *IEEE International Conference on Information Systems and Computer Networks*, 2013

[10] MadihaWaris , Dr. Shoab Ahmad Khan and Muhammad ZamanFakhar, “Factors Effecting Service Oriented Architecture Implementation”, *IEEE Science and Information Conference* ,2013

[11] Haiqing Bu, “Metrics for Service Granularity in Service Oriented Architecture”, *IEEE International Conference on Computer Science and Network Technology*, 2011

[12] Neelavathi, Dadapeer and K. Vivekanandan, “Service selection based on status identification in SOA”, IEEE Electronic Computer Technology (ICECT), *3rd International Conference* on (Volume: 6), 2011.

[13] AryaAmini, Soheila Sadjedy. “SOMGIS- A Service Oriented Model for GIS Software Architecture,” *International Conference on Computer Engineering and Technology*, 2010.

[14] Mohammad Kazem HAKI, Maia Wentland Forte, “Service Oriented Enterprise Architecture Framework,” IEEE 6th World Congress on Services, 2010.

[15] Siani Pearson and Tomas sander, “A mechanism for policy-driven selection of service providers in SOA and cloud environments,” *IEEE 10th Annual International Conference on New Technologies of Information Systems (NOTERE)*, 2010

[16] Jan Christian Lang, Thomas Widjaja, Peter Buxmann, Wolfgang Domsche and Thomas Hess, “Optimizing the Supplier Selection and Service Portfolio of a SOA Service Integrator,” *IEEE Hawaii International Conference on System Sciences*, 2008

[17] JuhaSavolainen and Anssi Karhinen, “Matching Service Requirements to Empirical Capability Models in Service-Oriented Architectures,” *Annual IEEE International Computer Software and Applications Conference*, 2008.

[18] M. Rosen, Service-Oriented Architecture and Design Strategies, Jun Wiley 2008.

[19] YiminShen, Yushun Fan. "Cooperative Mixed Strategy for Service Selection in Service Oriented Architecture" *IEEE International Conference on Systems, Man and Cybernetics*, 2007.

[20] Z. Stojanovic, A. Dahanayake, H. Sol. “Modeling and design of service-oriented architecture,” *IEEE International Conference on Systems, Man and Cybernetics*, Oct. 2004.

[21] L.-J. Zhang, B. Li, C. Tian, H. Chang, “On demand web services-based business process,” *IEEE International Conference on Systems, Man and Cybernetics*, Oct. 2003.

[22] P. Bernus, L. Nemes, and G. Schmidt, “Handbook on Enterprise Architecture,” Berlin: Springer, 2003

[23] D. Gross, C. M. Harris, *Fundamentals of Queuing Theory*. New York: John Wiley & Sons, 1985, pp. 79.

[24] Web Service Level Agreement language, (WSLA language), http://www.research.ibm.com/wsla

[25] Service-Oriented Architecture, http://en.wikipedia.o- rg/wiki/Service-oriented_architecture