Volume 2, Issue 8, August 2012 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

A Compositional Approach of Reliable and Efficient Cloud Service Selection

Parveen Dhillon*
Student of M.Tech,CSE Department
SBSCET Ferozepur
India

Vishal Arora
Assistant Professor of CSE Department
SBSCET Ferozepur
India

Abstract— The use of Cloud services became a common practice these days. There are increasing amount of Cloud services being available on internet. Cloud service is a software component invoked over the Cloud by XML and SOAP protocol used for exchange information between ends. When we work with Cloud for a single problem, we can get the number of solutions in the form of Cloud service. These Cloud services are hosted by different Cloud servers. To get an efficient and secure Cloud service user has to track all Cloud servers, but it is not feasible. In this paper we are providing an algorithm for the efficient and reliable Cloud service selection. We are using the repository model for the implementation. We will verify the deadlock free with modelling the Cloud service.

Keywords— Efficient Cloud service selection;QoS Parameters; Deadlocks; Repository Model.

I. INTRODUCTION

This Cloud Service is a software component invoked over the Cloud via an XML message that follows the SOAP. Cloud Services are based on distributed technology and provide standard means of interoperating between different software applications across and within organizational boundaries with the use of XML. Cloud Services technology is emerging as a powerful means for organizations that need to integrate their applications within and across organizational boundaries.[1] Services are described by Cloud Service Definition Language in XML format.It is a major language which is widely used and it provides a model and an XML format to describe the syntax about Cloud services.[2]

A. Cloud Service Structure

Cloud service structure has mainly five levels which includes: HTTP at lowest level then XML, SOAP, WSDL and UDDI. Among these three are major components. We are describes those below:

1) SOAP

SOAP is a XML based protocol. The way a Cloud service works is like the application used by client send the request to service provider using SOAP. The service provider process the business logic and send it back to SOAP.[3]

2) *WSDL*

The WSDL file is an XML document that describes a set of SOAP messages and provides the information necessary for a client to interact with the WS. It specifies the location of the service, the publicly available operations, functions the service exposes, data type information for all XML messages, and the communication protocol used to talk to the service.[3]

3) UDDI

The Cloud service interfaces described by WSDL may be put into a directory called UDDI. The UDDI is a central directory service where businesses can publish, register, and search for Cloud services.[3]

II. RELATED WORK

A. Cloud Service Selection Models

Maximilien and Singh proposed a multi-agent based architecture to select the best service according to the consumers' preferences. Maximilien and Singh describe a system in which proxy agents gather information on services, and also interact with other proxy agents to maximize their information.[4] The model is described in the figure 2.1 given below. In the figure we can see that client interact with Cloud services with a proxy agent in between which collect and update its information about the Cloud services.

Fig 2.1. Model Proposed by Maximilien and Singh

Liu, Ngu, and Zeng proposed a model in which selection is done at the client side. Their major selection criteria is based on the QoS based service selection. They have considered three quality criteria namely execution time, execution duration and reputation for the selection.[5]. In addition, execution price, duration, transactions support, compensation and penalty rate are the other criteria. The authors of suggest an open, fair, and dynamic framework that evaluates the QoS of the available Cloud Services by using clients' feedback and monitoring. In figure 1.2 model proposed by Liu, Ngu and Zeng is shown. Here we can see that reasoning mechanism is attached at the client side means QoS parameters are applied at client side.

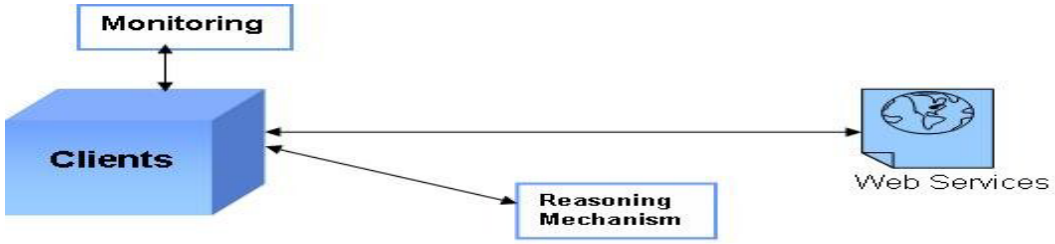

Fig 2.2 Model Proposed by Liu, Ngu and Zeng

Repository Model proposed by Abhishek Pandey and S.K. Jena uses a technique for dynamic selection of Cloud Services which will also handle the problem of redundant Cloud Services. This repository will be used to redirect the client's request. This will also provide a level of security since it will not be allowed to invoke directly by the clients.[6] This technique will prevent unauthorized access to the real services. This provision will also help to hide the systems complexity from the clients. The model is shown in the figure 2.3. In the figure we can see the repository between client and service and the QoS parameters are applied at the repository. We are going to use this model for the implementation of our algorithm.

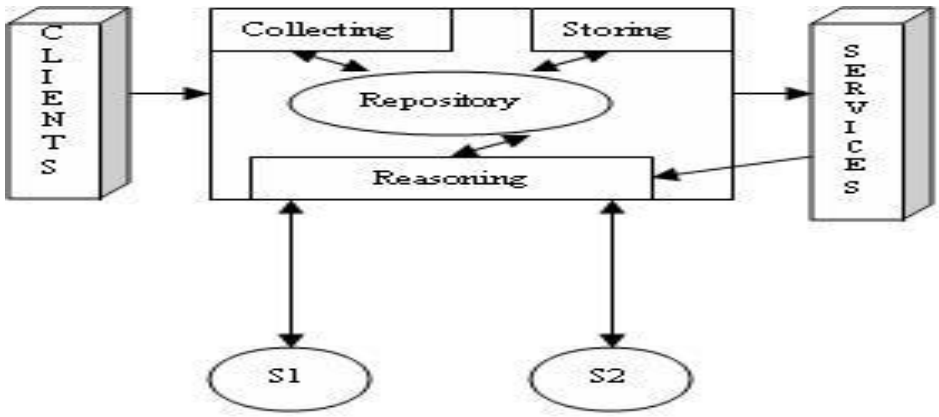

Fig 2.3 Repository Model

A. QoS Parameters

QoS parameters or the quality of service parameters are the parameters which control the quality of the process we are performing. In this paper we are concerned with the Cloud services. So quality of Cloud service selection can be measured by the QoS parameters related to it. Normally quality of service parameters vary from very simple parameters like availability to very sophisticated like atomic transaction. We have discussed in A part that measuring of QoS parameters can be done at different points according to the model we are using. The different pints where QoS parameters can be calculated are:[7] Cloud service, Cloud service framework, server, application server, server, client

and between client and server. In our implementation as we are using repository model QoS parameters are applied between client and server i.e. network.

Here we are talking about the efficiency and reliability so these are the major QoS parameters we are concerned for. Lets discuss one by one:

Efficiency: Efficiency is described differently in different situations e.g. speed, output, time. As we are concerned with the selection process so in our case we are measuring the efficiency in form of response time.

Reliability: In general reliability is defined as the ability of a system or component to perform its required functions under stated conditions for a specified period of time. Again it can have many parameters like message ordering in network, duplication elimination etc. The parameter which we are using for reliability is deadlock.

Other parameters which we are using is the very basic parameter Availability which is must to apply in case of Cloud service selection. P.W. Chan and Michael R Lyu gave the algorithm for reliable Cloud service composition and verified it using petri nets[8]. This algorithm composes the Cloud service by using WSDL file and then verify that it is deadlock free using petri nets.

III. PROPOSED WORK

In this paper we are proposing an algorithm for Cloud service selection. The Cloud services on which parameters are to be applied will be taken together and all the three parameters are applied one by one. On the basis of results we can select the best Cloud service. Results are calculated dynamically on line directly without composing the Cloud services but for the verification of deadlock we compose our own Cloud service so that we can show that algorithm is working.

A. Algorithm

In the algorithm given below Cloud services are taken together in form of array and three parameters are applied. We will see how we apply the parameters on by one.

1) *Availability:* We will read the WSDL file of the Cloud services one by one. Then a method is invoked dynamically. If we get the response it shows that Cloud service is available.
2) *Deadlock:* To check whether the Cloud service is deadlock free or not, we first check is there any shared resource among Cloud service methods. Then we check the resources needed by the method we need to invoke are locked by some another method or not. If the resources are locked then this method is put in wait state. A timer is attached which calculate the wait time. If the wait time exceeds a threshold value, it means that deadlock has occurred. If the resources are not locked then our method lock the resources and enter the critical section, which means there is no deadlock.
3) *Response Time:* We send the request dynamically to invoke the Cloud service and calculate the time it takes to respond. This gives our response time.

Formal algorithm is given below:

```
Algorithm(I,O)
/* I represent the Input Arrary with N services and O represents the Output list*/
{
1.  Collect all the Webservices I(1)…I(n) and perform the Composition on it as the Single Unit.
2.  Select the Common Method Call for the Webservice call Method Fun
3.  Derive the process for  all webservices in output List O(i)
4.  for all (Out in O(I))
5.  {
6.  if (Available(O(i))/* Check the Webservice is available*/
7.  {
8.  if (SharedResource(O(i))=true)
9.  {
10. if(Critcal_Section(SharedResouce(O(i))=true)
11. /* the resource is already occuipied by some other process and it cannot continue with it*/
```



```
12.  {
13.  while(wait)
14.  {
15.  if(Wait(ResourceRelease())>Thresholdvalue)
16.  {
17.       return “Deadlock Occur”;
18.  }
19.  }
20.  Enter_CriticalSEction(O(i));
21.  Lock(O(i));
22.  }
23.  Store Response time in list R
24.  R(i)=Responsetime(O(i));
25.  }
26.  else
27.  {
28.  print(“Service is Not Available”);
29.  }
30.  }
31.  max=infinity;
32.  j=0;
33.  for(Res in R)
34.  {
35.  if(!Deadlock(R(i) )
36.  {
37.  if(ResponseTime(Res)<max)
38.  {
39.  max=ResponseTime(Res);
40.  j=I;
41.  }
42.  }
43.  return J;
44.  }
45.  Exit.
```

IV. ALGORITHM IMPLEMENTATION RESULTS

In this section we show the results calculated on basis of algorithm given above. We have four Cloud services related to the weather forecasting. Firstly we apply first parameter that is we check the service is available and check its response time and whether it is deadlock free or not. We plot the response time in form of a graph response time in milliseconds vs Cloud services. Also all the parameter results are shown in form of a table.
On basis of these QoS parameters we can rank the Cloud services and find the most efficient and reliable among them.

For implementing the algorithm we are taking the example of weather forecasting Cloud services. Here we have four weather forecasting Cloud services with different Cloud methods. Firstly we check the availability of all the four Cloud services and after that we calculate the response time. Response time of these services is ploted in graph below.

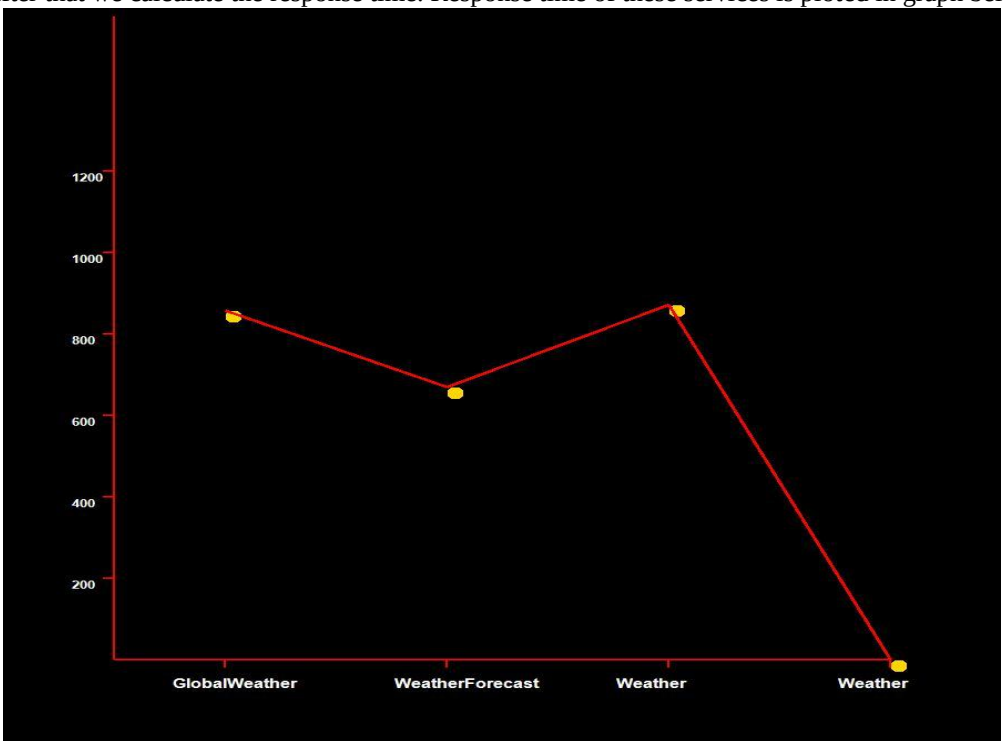

Fig 4.1 Graph Response Time vs Cloud Services

Fig 4.1 is a graph showing response time in milliseconds on y axis and weather Cloud service on x axis.
Here we can see that the first Cloud service GlobalWeather has the highest response time then WeatherForecast and so on. Similarly response time of all four Cloud services can be compared.

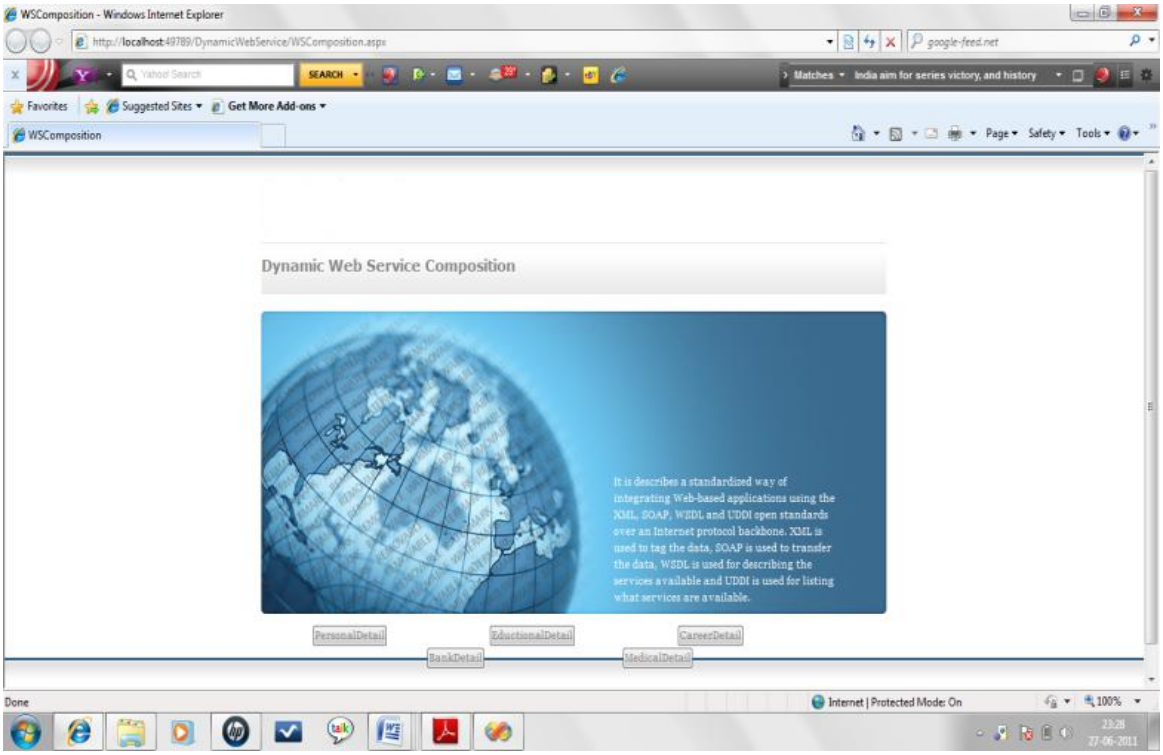

Fig 4.2 Home Page Dummy Cloud Service

Now we apply the remaining parameters which verify the Cloud service as reliable service i.e. deadlock free. For this firstly we check whether there is some shared resource in all four Cloud services. We don't find any shared resource

in any of the Cloud service. If there is no shared resource then there is no deadlock in any of the Cloud service. The algorithm terminates here and the results are shown in table 1.

To show the unreliable Cloud service we created our own Cloud service with shared resources. This Cloud service has a common database as shared resource. It is about retrieving the data against some member or say student. Data stored is like personal details, medical detail, educational detail etc. It is a dummy Cloud service designed to show the shared resource and to perform a check for deadlock. We are applying only that part of algorithm on dummy Cloud service which is related to the deadlock i.e. from line 1 to line 22. Our Cloud service is shown in fig 4.2.

In this figure we can see that we have five methods in one Cloud service which are personal details, educational detail, medical detail, bank detail and career detail. Here the database is shared by all the five Cloud service, when one method lock the database another cannot use that database neither for writing nor to read the database. We set a threshold value, if the wait state exceed that threshold time value we consider there is a deadlock. Two instances of the Cloud service are taken to make a check for the deadlock. Both instances are same as shown in fig 4.2.

TABLE 1

RESULT AFTER APPLING ALL THREE PARAMETERS

	Global Weather	**Weather Forecast**	**Weather by Zip**	**Weather**
Cloud Service Name	http://www.webservicex.net/WS/WSDetails.aspx	http://www.webservicex.net/WeatherForecast.asmx	http://ws.cdyne.com/WeatherWS/Weather.asmx	http://www.deeptraining.com/webservices/weather.asmx
Cloud Method	GetcitiesByCountry	GetWeatherByPlaceName GetWeatherByZipCode	GetCityForecastByZip GetCityWeatherByZip	GetWeather
Availability	Available	Available Available	Available Available	Available
Response Time in milliseconds	858	670 419	873 0	0
Shares Resources	No	No	No	No
Deadlock	No	No	No	No

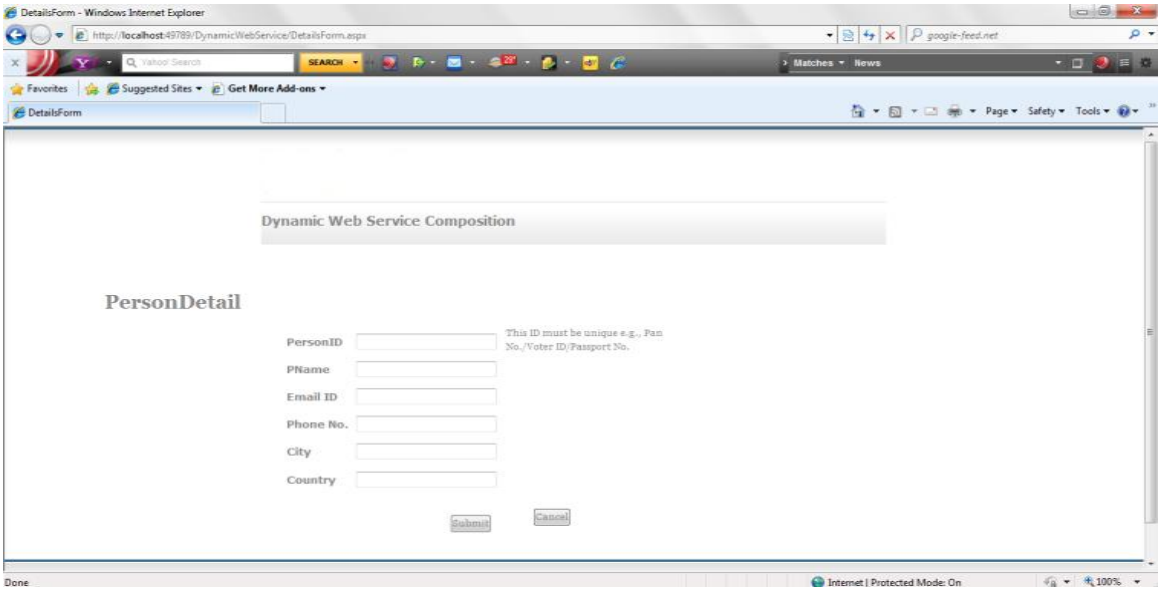

Fig 4.3 Form displayed after personal detail Cloud method in first instance of Cloud service

When we click the personal detail in one instance we get the form shown in fig 4.3. To check for deadlock we use any other Cloud method in second instance say we click on educational detail then the algorithm check for the shared resources i.e. database but the resources are locked by personal detail method so a deadlock occurred there as shown in fig 4.4.

In fig 4.4 deadlock message is shown in the highlighted region. But if this method is used after the threshold time value there is no deadlock. In this way deadlock function is applied on the Cloud service and the algorithm is verified.

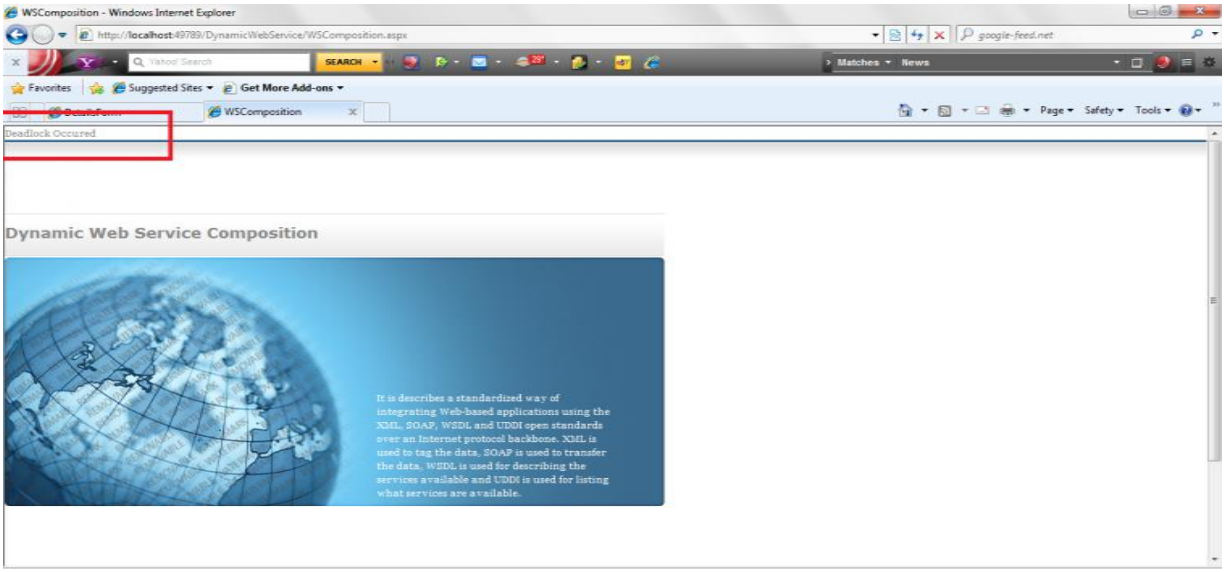

Fig 4.4 Deadlock message when educational detail method is invoked in second instance

V. CONCLUSION

All In this paper we surveyed Cloud service selection models and give an algorithm for dynamic Cloud service selection. It facilitates in selection efficient and reliable Cloud service. Selection is done on basis of QoS parameters where efficiency is ensured by Availability and Response Time and reliability is verified by proving the service deadlock free. We studied various QoS parameters to choose for reliability. Repository model is used to make the selection dynamic. Further efficiency can be increased by making slight change in repository model by introducing cache concept.

REFERENCES

[1] Xin Dong Alon Halevy Jayant Madhavan Ema Nemes Jun Zhang "*Similarity Search for Cloud Services*." Proceedings of the 30th VLDB Conference, Toronto, Canada, 2004.

[2] Erik Christensen, Francisco Curbera, Greg Meredith, and Sanjiva Weerawarana. *Cloud services description language (wsdl)* 1.1. http://www.w3.org/TR/wsdl.

[3] Vassiliki Diamadopoulou, Christos Makris, Yannis Panagis, Evangelos Sakkopoulos"*Techniques to support Cloud Service selection and consumption with QoS characteristics.*" Journal of Network and Computer Applications 31 (2008) 108–130

[4] E. Maximilien and M. Singh. " *Towards Autonomic Cloud Services, Trust and Selection."* ICSOC'04 pages 212–221, November 2004.

[5] Y. Liu, S. Ngu, and L. Zeng."*QoS Computation and Policing in DynamicWeb Service Selection"* WWW2004, (8 pages), May 2004.

[6] Abhishek Pandey, S.K.Jena "*Dynamic Approach for Cloud Services Selection*" Proceedings of the International MultiConference of Engineers and Computer Scientists 2009 Vol I IMECS 2009, March 18 - 20, 2009, Hong Kong.

[7] Dr. IIavarasan Egambaram , G. Vadivelou, S. Prasath Sivasubramanian "*QOS Based Cloud Service Selection*". *www.ubicc.org.*

[8] Pat. P. W. Chan, Michael R. Lyu "*Dynamic Cloud Service Composition: A New Approach in Building Reliable Cloud Service*" 22nd International Conference on Advanced Information Networking and Applications IEEE 2OO8.

