Volume 3, Issue 6, June 2013 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

Methods to Ensure Quality of Service in Cloud Computing Environment

Sonal Dubey*, Sanjay Agrawal
Deptt. of Computer Engg. & Application
NITTTR, Bhopal, India

Abstract—Cloud computing systems promise to offer subscription-based, enterprise-quality computing services to users worldwide. As the demand for delivering services to a large number of users has increased, they need to offer differentiated services to users and meet their expected quality requirements. Service-Oriented Computing (SOC) enables the alignment of loosely coupled services with different levels of Quality of Service (QoS). Choosing an optimal set of services for a composition in terms of QoS is critical when many functionally alike services are available. With the arrival of Cloud Computing, the number of such services and their distribution across the network are rising rapidly. So, this paper describes EXACT and Fully Polynomial Time Approximation Scheme (FAPTS) algorithms for QoS aware service composition to optimize user's experiences for Cloud service access, introduces the concept of cloud computing, explains the QoS Aware Services Mashup (QASM) Model. One of the challenging issues of cloud services mashup is how to find the optimal paths for services, to route the service instances through the provider while meeting the applications' resource requirements so that the QoS compulsions are satisfied. However, the service routing problem aware of QoS is typically NP-hard. The purpose of this paper is to explain a QASM model to solve this problem more effectively.

Keywords—Cloud Computing, QoS, QASM, EXACT, FAPTS.

I. INTRODUCTION

With the promotion of the world's top companies, cloud computing is drawing more and more attention for providing a flexible, on demand computing infrastructure for a number of applications. As the concept of cloud computing has been accepted recently, it continues to spread widely. Cloud computing technologies have changed the way applications are developed and accessed. They are designed to run the applications as services over the Internet on a scalable infrastructure. Cloud computing is an emerging computing paradigm that may change the way how information services are provisioned. Clouds represent a new step in evolutional computing and communication technologies development chain by introducing a new type of services and a new abstraction layer for the general services virtualisation (similar to utilities) and mobility. Cloud computing has been defined by NIST as a model for enabling convenient, on-demand network access to a shared pool of configurable computing resources that can be rapidly provisioned and released with minimal management effort or service provider interaction. The main characteristics of Cloud computing are on-demand self-service, broad network access, resource pooling, rapid elasticity and measured service [1]. The cloud model is composed of five essential characteristics, three service models, and four deployment models.

A. Essential Characteristics:

- On-demand self-service: A consumer can unilaterally provision computing capabilities, such as server time and network storage, as needed automatically without requiring human interaction with each service provider.
- Broad network access: Capabilities are available over the network and accessed through standard mechanisms that promote use by heterogeneous thin or thick client platforms (e.g., mobile phones, tablets, laptops, and workstations).
- Resource pooling: The provider's computing resources are pooled to serve multiple consumers using a multi-tenant model, with different physical and virtual resources dynamically assigned and reassigned according to consumer demand.
- Rapid elasticity: Capabilities can be elastically provisioned and released, in some cases automatically, to scale rapidly outward and inward commensurate with demand.
- Measured service: Cloud systems automatically control and optimize resource use by leveraging a metering capability at some level of abstraction appropriate to the type of service (e.g., storage, processing, bandwidth, and active user accounts).

The service of the cloud computing is divided into three main categories: Infrastructure-as-a-Service (IaaS), Platform-as-a-Service (PaaS), and Software-as-a-Service (SaaS). laaS delivers basic storage and computes capabilities as standardized services over the network. The network devices and other systems are cooperated and made available to handle workloads. Data and storage on the clouds deal with reliable access to data with the dynamic size, weighing resource usage according to the requirements. laaS provides managed and scalable resources to the user, and also provides enhanced virtualization capabilities.

PaaS provides computational resources via a platform that applications and services can be developed and hosted. It facilitates development and deployment of applications the underlying infrastructure. Additionally, PaaS makes use of dedicated APls to control the behavior of a server hosting engine, which executes and replicates the execution according to user requests. SaaS provides services using a cloud infrastructure, and it offers implementations of specific functions and processes capabilities. SaaS delivers applications to the user's browser. Furthermore, it helps the organizations with limited resources able to deploy and maintain the software meanwhile they reduce the energy consumption and expenses. In Cloud computing, resources can be either externally owned (public Cloud – as provided by Google and Amazon) or internally owned (private Cloud). Public Clouds offer access to external users who are typically billed on a pay-as you-use basis. The private Cloud is built for the access within the enterprise where the users can utilize the facility without any charge [1]. In the virtualization-based Cloud computing environment, Cloud service provisioning consists of two key steps. First, the Cloud and network resources are abstracted as Cloud and network services that form a virtual service network. In this step, network virtualization provides an effective schema to enable resource abstraction. Second, the service provider composes service components from these abstracted services as Cloud services and provides such composed Cloud services to end users. The network performance has a significant impact on Cloud service provisioning, and in many cases networks become a bottleneck that limits Clouds from supporting high-performance applications. Therefore, the optimization for service provisioning is a significant problem to be resolved in Cloud computing [2], for which we have explained the EXACT and FPTAS algorithms.

In this paper, we investigate an optimization issue for Cloud service provisioning, that is, QoS-aware service composition upon network virtualization in Cloud computing. In order to optimize the Cloud service provisioning, in this paper we describe an exact and approximation algorithms to resolve QoS-aware service composition (QSC). The remaining paper is ordered as follows. Section II presents the related work in this area, Section III describes EXACT and FAPTS algorithms. Section IV gives the detail of the QASM model in cloud computing environment and section V concludes the paper.

II. Related Work

In order to provide Cloud service to end users, the Cloud computing infrastructure must be first abstracted through virtualization as a set of virtualized services, then the service components from these virtualized services need to be further composed as a Cloud service. However, existing researches do not really take the network performance in offering Cloud service into account, and few of them employed network virtualization [3]. On the other hand, composing appropriate service components into a Cloud service that meets multiple QoS requirements is a challenging optimization issue since such a problem can be basically addressed as Multi-Constraints Path problem (MCP), which is known to be NP-hard [4]. To handle this problem, much research progress has been made toward designing efficient algorithms recently. Xue *et al.* [5], [6] defined a different version of MCP and presented a set of approximation algorithms for each problem, which are known as the best among the aforementioned results. Huang et al. [7] gave a heuristic for Multi-Constrained Optimal Path (MCOP) selection, in which a nonlinear combination technique and a geographical version of a well-known process in solving Delay Constrained Least Cost (DCLC) were introduced. The results obtained, show that the proposed heuristic can achieve a good trade-off between execution time and quality of the path, and it needs less time to compute a path compared with Xue's algorithm in [5]. Other progress toward efficient QoS routing algorithm can be found from the surveys given in [8], [9], and [10]. In [11], Yuan studied the multi constrained path problem and proposed a limited granularity heuristic and a limited path heuristic on the basis of extended Bellman-Ford algorithm. Mieghem and Kuiper [12] provided four concepts of exact QoS routing and developed an exact algorithm relies on their observation that NP-hard behavior of QoS routing would not happen in practical networks.

The above mentioned works mainly address the problem from QoS routing perspective, which cannot be applied directly in virtualization-based Cloud computing for QoS guaranteed service composition. This is because network virtualization brings new challenges that a virtualized service network may actually span a great scale of heterogeneous networks with diverse network resources in different domains. In this paper, we investigate an optimization issue for Cloud service provisioning, that is, QoS-aware service provisioning upon network virtualization in Cloud computing.

III. Algorithms For Qos Aware Service Provisioning In Cloud Computing Environment

The main optimization focus for Cloud service provisioning is how to compose a sequence of service components from virtualized services into the Cloud service and provide it to end users. We address the problem of QoS-aware service composition as follows.

A. Problem Formulation

A service network formed by interconnecting service components can be modeled as a directed graph G (V,E) with n vertices and m edges. Each vertex (for example v) is associated with a capacity weight c (cv for vertex v) denoting its serving capability in service Sh, $1 \leq h \leq H$, and each edge $e \in E$ is associated with K weights w = (w1, w2, · · · , wK) representing QoS parameters. Let Req = {C, W} be a user request, where C = (C1, C2, · · · , CH) denotes the requirements on computing capacities of H virtualized services, i.e., the constraints on the vertices, W = (W1,W2, · ··,WK) denotes the K QoS end-to-end constraints on paths. Let p in the service network is a series of tandem services [2].

B. Algorithm1. EXACT

```
Input:
  Graph: $G$ ($V, E, w, W, c, C$);
Output:
  Path set: Pareto minimum path set $MP$;
```

```
1: To each vertex $v \in Sh$, prune $v$ and its connected edges if $c_v < C_h$ ,$1 \le h \le H$;
2: for $k = 1$ to $K$ do
3:  Apply Dijkstra to calculate the shortest path $P_k^L$ according to weight $wk(e)$ on each edge in $G(V, E)$;
4:    if $Wk < wk(P_k^L)$ or $\max_{1\le i\le K, i\ne k} wi\{(P_k^L)/Wi\} > 1$
          then
5:          return Invalid request, Exit;
6:      end if
7: end for
8: Compute a new weight $w_M(e) = \sum_{k=1}^{K} wk(e) / Wk$ for each edge $e \in E$;
9: Apply $\kappa$-shortest paths algorithm in terms of $wM(e)$ on each edge to find the first $\kappa$ paths $P^M_j$, $1 \le j \le \kappa$ from source to destination, $MP \leftarrow \{P^M_j \mid 1 \le j \le \kappa\}$;
10: To all paths in $MP$, remove the path which is dominated by any other;
11: return $MP$;
```

The algorithm named EXACT is shown in Algorithm 1, which involves five steps.

The first step (line 1 in Algorithm 1) prunes the topology according to the constraints on the vertex. This is a required step because c_v denotes the serving capability of a vertex $c_v \in S_h$ in the service network and $c_v < C_h$ means such a node cannot meet the service requirement; therefore should be eliminated in the initial stage. In addition to this, the notation G (V, E) is still used here to denote the pruned topology.

In the second step (from line 2 to line 7), the invalid request is blocked according to the end-to-end constraints on the path. If the constraints imposed by a user are so rigid that the network resources cannot be satisfied, this set of constraints would be treated as an invalid request, and it should be blocked in advance. Though blocking the invalid requests in the first step, the weights on each edge can be aggregated by a linear combination manner, i.e. $w_M(e) = \sum_{k=1}^{K} wk(e) / Wk$ for each edge $e \in E$, in the third step (line 8) of Algorithm 1.

In the fourth step (line 9), an existing κ-shortest paths algorithm is applied in terms of new weight $w_M(e)$ on each edge to find first κ shortest paths from source (service entrance portal) to destination (service exit portal). The obtained κ shortest paths form the MP initially.

The fifth step examines MP generated from the fourth step, and eliminates the path which is dominated by any other. That is, if there is a path P^M_j in MP such that each weight of it (for example delay), is inferior (equal or greater) than that of any other path in MP, then P^M_j should be removed from MP. On the basis of above five steps, the Pareto minimum paths set can be eventually obtained [2].

C. Algorithm2. FPTAS

```
1: Eliminate the edges whose bandwidth is smaller than $r0$ or delay is greater than $\theta_\Sigma$ ;

2: if $W1 \ge cmax$ && $W2 \ge dmin$ && $W3 \le bmax$ then   goto Step 3;
   else
       Output Invalid requests, exit;
   end if

3: For each $e \in E$ in $G(V,E)$, compute the new weights $w_k^N(e) = \lceil w_k(e)/ W_k . (n-1)/ e \rceil$ , $k = 1, 2$, and set $W^N_1 = W^N_2 = \lceil (n-1)/ e \rceil$;

4: Extend the graph $G(V,E)$ to a directed graph $G^N(V^N,E^N)$ with vertex set $V^N = V \times \{0, 1, \ldots\ldots, \lceil (n-1)/ e \rceil\}$ and edge set $E^N$. To a given edge $(u, v) \in E$, $E^N$ contains direct edges from vertex $(u, \delta)$ to vertex $(v, \lambda)$ such that $\lambda = \delta + w^N_2(u, v)$ as well as directed edges from $(v, \delta)$ to vertex $(u, \lambda)$ such that $\lambda = \delta + w^N_2(u, v)$. All such edges between $(u, \cdot)$ and $(v, \cdot)$ have the same length $w^N_1(u, v)$.
Moreover, $E^N$ contains zero-length edges from $(u, \delta)$ to $(u, \delta+1)$, where $0 \le \delta \le \lceil (n-1)/ e \rceil - 1$;

5: Calculate the shortest path $p^N$ in $G^N$ from $(s, 0)$ to $(t, \theta^*)$ where $\theta^*$ is the smallest integer such that $\theta^* \le \lceil (n-1)/ e \rceil$;

6: Output $p^N$ by omitting the latter components of each extended node on the finding path;
```

The approximation algorithm named Fully Polynomial Time Approximation Scheme (FPTAS) is presented in Algorithm 2, which has three key stages. The first stage is to eliminate the topology and filter the invalid service requests in Step 1 and Step2 as the same functionality with the initial phase of Algorithm 1. While in the second stage, each weight on the link in $G(V,E)$ is reformulated to be a new weight $w_k^N(e)$ in Step 3 such that $w_k^N(e) = \lceil w_k(e)/ W_k . (n-1)/ e \rceil$, $1 \le k \le 2$; thus the MCOP problem is transformed to be an easy one. The last stage of FPTAS, it applies a graph-extending based dynamic programming process [18] to solve such simplified problem in polynomial time.

In the last stage of FPTAS, Step 4 extends the pruned topology to a directed graph $G^N(V, E)$. Each vertex of G has been extended to a vertex group in which the adjacent vertices are connected by a directed edge with length zero. For example, to a given node u in G, there exists a directed zero-length edge from node (u, δ) to node $(u, \delta + 1)$ in G^N.

Among the vertex groups, they are also connected by directed edges. For example, (*u*, ・) and (*v*, ・) are connected by many edges, moreover, such edges have the same length $w^{N}{}_{1}(u, v)$ [13].

IV. QOS AWARE SERVICES MASHUP (QASM) MODEL

Mashup is a Web-based network resource that composes existing services resources, be it content, data or application functionality, from more than one resource in enterprise environments by empowering the actual end-users to create and adapt individual information centric and situational applications. In internet there are maybe many available web services with various QoS (Quality of Service) providing the same functionality specific to a specific task. So a selection needs to be made. Therefore services mashup have to search for an optimal set of services to construct a composite service and result in a best QoS, under user's QoS constraints and resource requirements [14].

A. Related terms:-

- Abstract Service. Abstract service has function descriptions without implementation and standard service interfaces across different service providers. The user can directly name the requested distributed application. An abstract service is corresponding to a directed acyclic graph (DAG) of workflow tasks.
- Service Instance. Service instances are concrete services published by service providers. They could give the function implementation specified by abstract services. And some service instances may have the same function, but different QoS of all candidate service instances, according to the abstract service path.
- Service Function Graph. Constructing abstract services as a workflow to fulfil user's requirement in functionality will obtain a service function graph. In the service function graph, there are service portals that serve as entrance/exit points.

The end user request speciation is represented by Req= $\{S_{req,} Q_{req}\}$ where S_{req} is a set of services which have to be traversed in a particular order and Q_{req} is a set of QoS constraints, the problem of QoS aware services composition is to compose a service path $p = s_1 \rightarrow s_2 \rightarrow \ldots. \rightarrow s_n$, from s_0 (entrance portal) to s_{n+1} (exit portal), such that QoS constraints ("(1)" and "(2)"), i.e. $RT^{target} \leq RT_p$, $A^{target} \leq A_p$ and also resource requirements ("(3)" and "(4)") i.e. $CR_{si} \leq 1$, $R_{Ij} \leq 1$ are satisfied. The QoS constraints and resource requirements can be defined as follows:

$$RT_p \overset{\Delta}{=} \text{response time} = \sum_{i=0}^{n+1} RT_{si} \quad (1)$$
$$A_p \overset{\Delta}{=} \text{availability} \prod_{i=0}^{n+1} A_{si} \quad (2)$$

where, $A_s =$ time that service is available / total time monitored

$$CR_{si} \overset{\Delta}{=} \text{CPU ratio} = CPU_{si\ required} / CPU_{si\ available} \quad (3)$$
$$BR_{Ij} \overset{\Delta}{=} \text{Bandwidth ratio} = BW_{Ij\ required} / BW_{Ij\ available} \quad (4)$$

where I_j is the service link on the edge $I_j = (s_{i,} s_j) \in E$, $s_{i,} s_j \in V$.

The QASM model includes three mapping steps, illustrated by Fig. 1 (a). For each end user request, the abstract services first maps(mapping-1) it to a composite service template, and then maps the template to an instantiated service path.

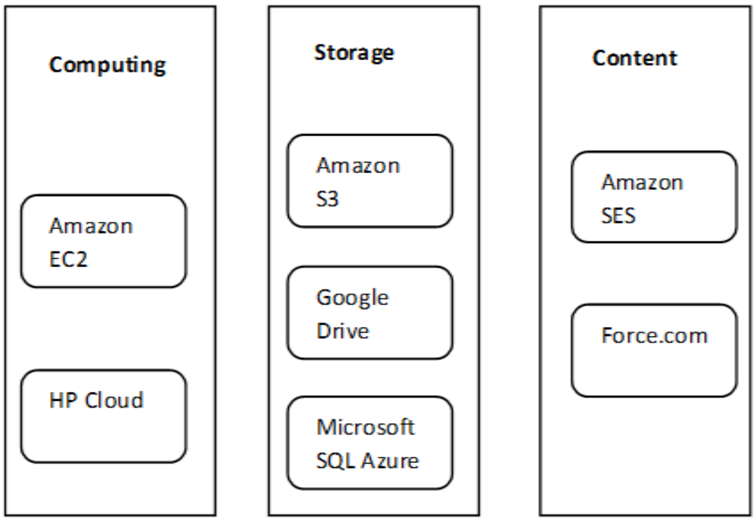

Fig. 1(a). Services instances for all required abstract services

Fig. 1(b) start from the source service; check the QoS consistency between the current examined service instance and all of its predecessors on the service path. If the QoS constraints ("(1)" and "(2)") predecessor satisfies the current examined service and then add a directed edge from the current examined service to the predecessor. We assume that the sink service is set as the end user's QoS requirements.

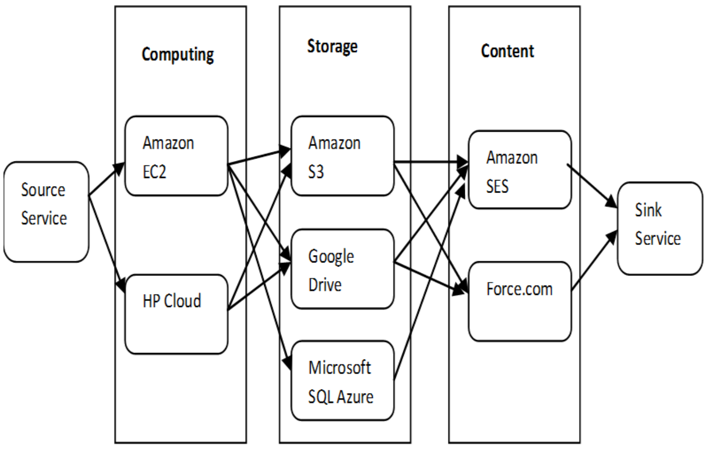

Fig. 1(b). Add edges between two QoS consistent service instances

The mapping from the user request to different composite service templates (mapping-1) is constrained by the user's application-specific quality requirements and different pervasive client devices, such as PDAs and cell-phones. If multiple QoS consistent service paths exist, the service composer should choose the one which has the minimum aggregated resource requirements so that the overall workload of a cloud computing system is minimized. In order to guarantee a successful service delivery, the resource requirements of the instantiated service path have to be satisfied. For ease, we only consider the CPU resource for end hosts and bandwidth for overlay links. In "(3)", the $CPU_{si\ required}$ represents the required CPU resource for running a new process. The $CPU_{si\ available}$ represents the available CPU resource in the physical hosting environment of s_i. The smaller the CR_{si}, the more advantageous we choose the service instance in terms of CPU load balancing because we start the new service process on a lightly loaded host. Similarly, we define the term bandwidth ratio ("(4)") for the service link I_j.

The mapping from the composite service template to an instantiated service path (mapping-2) is constrained by QoS constraints (e.g., availability, response time) and resource availability conditions (e.g., computing, network transport).

The "cost value" on the edge $I_j = (s_i, s_j) \in E$, $s_i, s_j \in V$ is defined using "(1)" ~ "(4)". Intuitively, the ratio w_p / R (w represents any of the above attributes and R is the total value) represents the normalized cost of selecting a portion of the service path in terms of one specific factor (e.g., QoS constraints or resource requirements). For each link in the cloud computing system, we also use QoS constraints or resource conditions to represent it QoS attributes. For any path *p* from the entrance point to exit point, the following cost function is used to evaluate the feasible path:

$$C(p) \overset{\Delta}{=} \lambda_1 (w_{1p} / R_1) + \lambda_2 (w_{2p} / R_2) + \ldots + \lambda_k (w_{kp} / R_k) \quad (5)$$

where w_{1p} is the summation of *i*-th dimension QoS attribute along p, R_i is the total constraint of *i*-th dimension QoS value. $\lambda_1 \in [0,1]$ denotes the weight of each QoS attribute which reflects their importance. The path with the minimum cost function value Min (C(p)) has the biggest possibility of being a feasible path. Run the Dijkstra-like algorithm to find the shortest path, which is returned as the result of the QoS aware services mashup, illustrated by Fig. 2 (thick line).

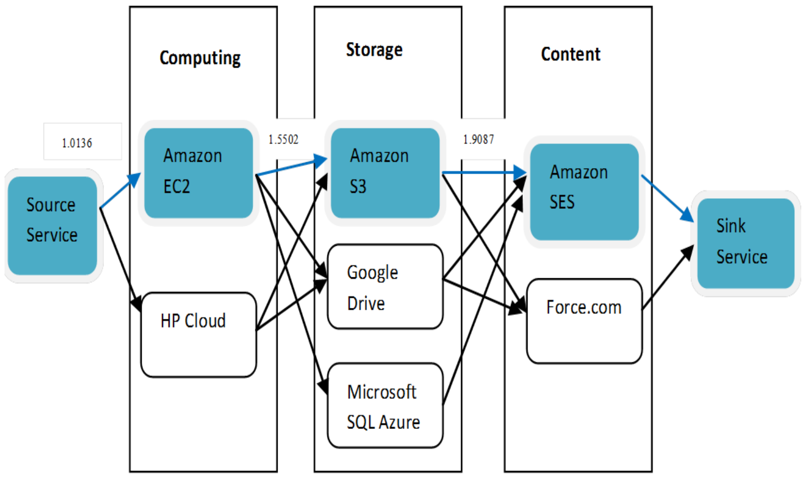

Figure 2 QASM generate the consistent resource shortest service path

Thus, the final result generated by the QASM model is a QoS consistent service path satisfying the end user's QoS constraints and also has the minimum aggregated resource requirements.

V. CONCLUSION

This paper investigated the problem of selecting an optimal sequence of infrastructure resources to form an end-to-end path for QoS provisioning in cloud computing environment The paper's main contributions include explaining QoS aware services mashup model and describing two efficient algorithms for selecting an optimal sequence of infrastructure resources for end-to-end QoS provisioning. EXACT and FPTAS algorithms are general and efficient; thus are applicable to practical Cloud computing systems.

REFERENCES

[1] Peter Mell, Timothy Grance, "The NIST Definition of Cloud Computing", NIST (National Institute of Standards and Technlogy) Special Publication 800-145.

[2] Jun Huang, Yanbing Liu, Qiang Duan, "Service Provisioning in Virtualization-based Cloud Computing: Modeling and Optimization", Globecom, 2012.

[3] Q. Duan, "Modeling and Performance Analysis on Network Virtualization for Composite Network-Cloud Service Provisioning," in Proc. of SERVICES, 2011, pp. 548–555, July 2011.

[4] Z. Wang and J. Crowcroft, "Quality-of-service routing for supporting multimedia applications," IEEE J. Sel. Areas. Commun., vol. 14, no. 7, pp. 1228–1234, Sept. 1996.

[5] G. Xue, A. Sen, W. Zhang, J. Tang and K. Thulasiraman, "Finding a path subject to many additive QoS constraints," IEEE/ACM Trans. Netw., vol. 15, no. 1, pp. 201-211, Feb. 2007.

[6] G. Xue, W. Zhang, J. Tang and K. Thulasiraman, "Polynomial time approximation algorithms for multi-constrained QoS routing," IEEE/ACM Trans. Netw., vol. 16, no. 3, pp. 656-669, Jun. 2008.

[7] J. Huang, X. Huang, and Y. Ma, "An Effective Approximation Scheme for Multi constrained Quality-of-Service Routing," in Proc. IEEE GLOBECOM 2010, Miami, Florida, pp. 1–6, Dec. 2010.

[8] F. A. Kuipers, P.V. Mieghem, T. Korkmaz and M. Krunz, "An Overview of Constraint-Based Path Selection Algorithms for QoS Routing," IEEE Com. Mag., vol. 40, no. 12, pp. 50–55, Dec. 2002.

[9] Z. Tarapata, "Selected multi criteria shortest path problems: an analysis of complexity, models and adaptation of standard algorithms," Int. J. Applied Math. and Comp. Sci., vol. 17, no. 2, pp. 269–287, July 2007.

[10] R. G. Garroppo, S. Giordano and L. Tavanti, "A survey on multi constrained optimal path computation: Exact and approximate algorithms," Compt. Netw., vol. 54, no. 17, pp. 3081–3107, Dec. 2010.

[11] X. Yuan, "Heuristic Algorithms for Multi constrained Quality-of-Service Routing," IEEE/ACM Trans. Netw., vol. 10, no. 2, pp. 244–256, Apr. 2002.

[12] P. V. Mieghem and F.A. Kuipers, "Concepts of Exact QoS Routing Algorithms," IEEE/ACM Trans. Netw., vol. 12, no. 5, pp. 851–864, Oct. 2004.

[13] Jun Huang, et al., "Novel End-to-End Quality of Service Provisioning Algorithms for Multimedia Services in Virtualization-based Future Internet", IEEE Transactions On Broadcasting.

[14] Yee Ming Chen, Yi Jen Peng, "A QoS aware services mashup model for cloud computing applications" Journal of Industrial Engineering and Management, JIEM, 2012 – 5(2): 457-47.