Towards Optimal Resource Management for Cloud Data Centers

Shahnaza Tursunova, Tae-Ho Lee, Nodir Kodirov, and Tae-Sang Choi
Electronics and Telecommunications Research Institute (ETRI),
Yuseong-gu, Daejeon, 305-700, South Korea
{shahnaza, kthlee, nodir, choits}@etri.re.kr

Abstract—**As virtualization technology matured, the concept of virtual network environment has emerged. Because of the flexibility of virtualization, the cloud computing services can serve various needs of its customers more efficiently. However, in most current virtual network provisioning algorithms, decisions are made without any consideration of underlying network resource states. In this paper, we are focusing on combined optimization of resources in the application and network stratums, and proposing an efficient and optimized resource provisioning algorithm and show its benefits over traditional resource allocation.**

Keywords- cloud computing, graph clustering, optimal resource provisioning

I. INTRODUCTION

Cloud computing promises to reshape the way IT service is produced and consumed by virtualizing computing resources (CPU, storage, and network). Virtualization enables flexible management of the computing resources, which allows dynamic service offerings based on various needs of the cloud customers. Among many different aspects of cloud computing management issues, such as monitoring and on-line fault management, we focus on the provisioning of computing resources since we believe that the optimal resource provisioning algorithm is the first step to fully utilize the cloud computing infrastructure.

Resource provisioning in the cloud data centers falls into the broad category of the Virtual Network Embedding (VNE) problem which tries to optimally map a user's request, known as the virtual network (VN), to a substrate network (SN). This problem has been widely studied [1-7]. Moreover, QoS related parameters, such as latency, jitter, and packet loss, become more important in the cloud service provisioning. However, many decisions are made in the application space without considering underlying network and cross stratum application/network optimization (CSO) efforts were proposed to focus on this challenges [16] .

In this paper, we introduce a CSO-aware graph clustering based resource provisioning algorithm for the cloud data centers. The graph-clustering algorithm is based on the algorithm proposed in [15]. In addition, we incorporate a hierarchical clustering algorithm to extend our provisioning algorithm.

The paper is organized as follows. Related work is provided in Section II. In Section III, we introduce our provisioning management architecture and use cases of CSO-aware VN provisioning. The problem definition and the proposed algorithm are given in Section IV and V, respectively. A simulation result and analysis is provided in Section VI. Then final concluding remark with description of our future work is given in Section VII.

II. RELATED WORK

A. Cross-Stratum Optimization

Cloud applications are used to provide a wide variety of services such as video gaming, social networking, grid applications and others. Such applications make substantial bandwidth demands on the network. In addition these applications (e.g., VoIP or HDTV) may require specific bounds on QoS related parameters, such as delay and jitter. However, many end user applications and services cannot efficiently utilize the network, nor can achieve the desired QoS due to lack of cross-interaction between cloud providers and substrate network. Cross-stratum optimization (CSO) was presented to focus these challenges [16]. The main objectives of CSO are i) resource optimization, ii) quick response to changing demands, and iii) QoS provisioning through better usage of application and network information. By taking these opportunities of CSO, in this paper, we enhance resource provisioning algorithm via CSO for cloud data centers.

B. Virtual Network Embedding

Virtual Network Embedding (VNE) to a substrate network is known to be a NP-Hard problem [1]. One of the most common Virtual Network Embedding (VNE) to a substrate network (SN) is known to be a NP-Hard problem [1]. Generally, VNE problem is divided into two phases: node mapping phase and the link mapping phase. However, even the two phase approach is still computationally intractable. The node mapping onto the SN that honors the bandwidth constraints is a multi-way separator problem, which is NP-Hard [1]. In addition, the link mapping problem onto the SN is an unsplittable multi-flow problem, which again is NP-Hard [2]. In [3], the authors consider splitting of the links of the VN among multiple SN paths (a set of links). With this assumption, the link mapping problem becomes a multi-commodity flow problem. As for the node mapping, the authors propose a greedy algorithm that is optimized by taking account of

common topological structures, such as hub-and-spoke topology, within SN and VN.

Other heuristic methods are provided in [4, 5]. In [4], authors make an extension with a concept of hidden hops. The hidden hops are the intermediate substrate nodes that are on the path for a VN link. The author points out that such intermediate nodes consume resources for forwarding the packets traversing along the VN link, these consumed resources are considered using hidden hops. In [5], authors incorporate the node mapping problem into the link mapping problem, and provide two heuristic methods to solve the problem.

All of the works described above addresses the VNE problem for a single SN case. In cloud computing environment, for example, a federation of multiple cloud computing data centers is frequent; thus, a VNE problem among multiple SN needs to be investigated. Scalability problem must be solved in such problems. In [6], the authors structure a provisioning management system in a hierarchical structure, i.e. a management system in the higher level of hierarchy manages all management system under its administrative domain. In this structure, by introducing autonomy to each management system, the upper-level management system can delegate some of the decisions to the management systems under its governance.

Another approach for multi SN VNE problem is using theories from the field of economics. The basic idea is to model the interactions between the customer and the SN providers and interactions among SN providers. In [7], two-step auction model is used to describe the interactions. In the first step, a Vickrey auction is used so that the customer can learn about the price for embedding his/her virtual network. Then a second one-time sealed auction is performed to actually determine which SN provider wins the bid.

III. PROVISIONING MANAGEMENT ARCHITECTURE

A. Proposed architecture

Before presenting proposed CSO based provisioning management architecture, we clarify our terminology. In the previous section, we used terminologies - *SN*, *inter-SN* - on purpose since the VNE problem has often been described with such terminologies. From now on, we use the term *domain* to represent each SN. In addition, the SN will have a new definition as a *CSO agent*, who is responsible for mapping of AS request to residual resources of network stratum (NS); intra-SN and inter-SN bandwidth will be denoted as *intra-domain* and *inter-domain* bandwidth. In cloud specific terms, a domain is similar to a cloud data center (CDC) and inter-domain to the inter-CDC. All these elements abelong to NS. In our architecture, there are three players - *VN creator*, *CSO Manager*, and *CSO Agent*. A VN creator is a customer who wants to create a VN. CSO Manager is the manager who generates the policy about how to provision incoming VN requests based on received AS constraints from VN creator and informs the CSO agents about the policy that it generates. The main responsibility of the CSO agents is to provision a VN or some part of the VN request according to the policy that VN provider sends.

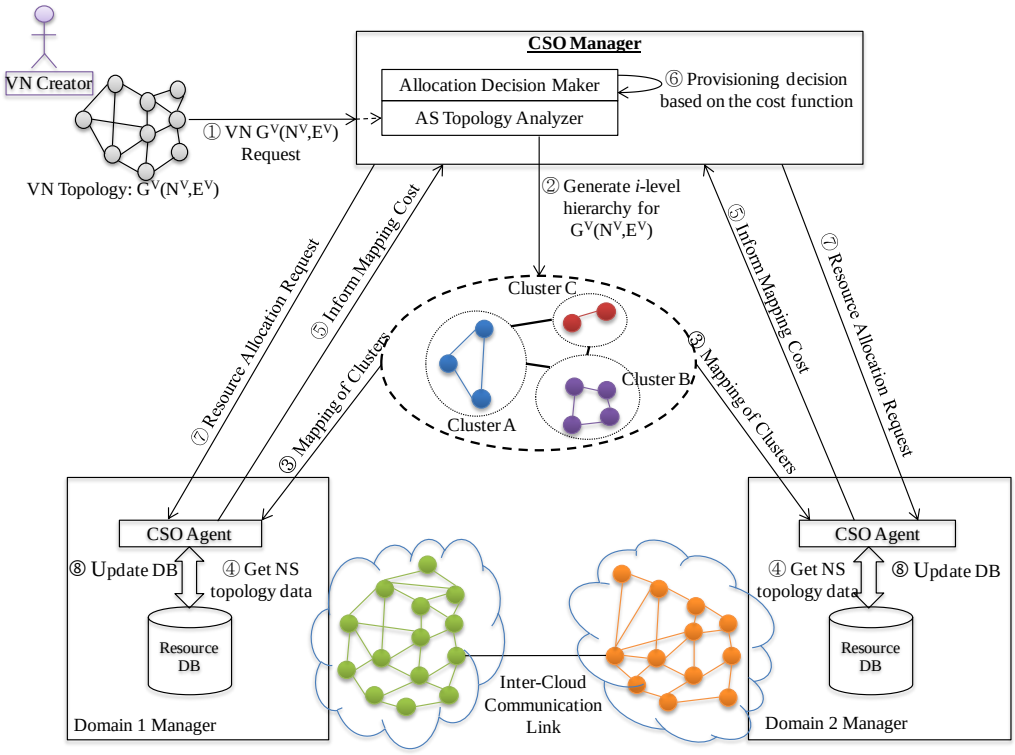

Figure 1. CSO-aware Virtual Cloud Network provisioning scenario across multiple domains

The workflow between the actors is described in Fig. 1. Once the CSO Manager receives a VN request, it initially asks every CSO Agent to check whether it can accommodate the VN request. Each CSO Agent runs a feasibility check and informs the expected cost of the provisioning to the CSO Manager. The details of the feasibility check and the cost computation is described in the later sections. Based on the cost information, the CSO Manager decides whether to assign the provisioning request to one of its CSO Agents, or to partition the VN request into multiple partitions and re-request every CSO Agents to perform a feasibility check for each of the partitioned VN requests.

The decision to further partition the graph is determined based on a greedy algorithm, i.e., the CSO Manager keeps partitioning the VN request until the cost of allocation partitioned VN requests becomes more expensive. This process is bounded by the number of nodes in the VN request; however, the partitioning process may be stopped after a predetermined number of partitioning.

B. Use case of CSO-aware VN provisioning

A CSO aware VN provisioning enables optimal placement of VN based on precise AS topology information provided by VN creator and NS topology information obtained by CSO Agent. It is possible to further improve QoS for the applications which are run on particular VN, if more detailed description of AS information is given.

In the first scenario it is assumed that VN is used to provide voice over IP (VoIP) services to the end user. Thus, with pre-knowledge about VN usage, VN creator will include application type information A^V to the VN topology request $G^V(N^V,E^V,A^V)$. It is generally known that voice application is very sensitive to the network latency compared to bandwidth and packet loss [21]. When a CSO Manager requests a CSO agent of each domain manager about the cost of VN allocation, the CSO agent will include the network latency which domain is capable to provide (together with VN allocation information). If VN is going to be allocated by partitioning, the highest value of the network latency (between all available partitions) is going to be chosen for entire VN. Based on the responses provided by domain CSO agents, Allocation Decision Maker of the CSO manager will decide optimal allocation with smallest value of the network latency. This allocation can guarantee provisioning of QoS by the VN to be the best possible allocation for providing VoIP service.

Similar to above scenario, there can be other ways of improving QoS of application via CSO-aware VN allocation at cloud. Above scenario is used to explicitly illustrate advantage of CSO-aware cloud application. Situation gets even interesting when VN is *heterogeneous*, i.e., VN is used for several application at the same time (such as for online gaming or video conferencing). In those cases, the CSO manager should make multi-criteria decision or put preference to one type of the service over the others. It could be more desirable to split VN to small pieces based on group of VMs to be running a particular type of application only. But, more information should be provided by AS (i.e., by VN creator). As requirements are not clearly stated (so far) for those cases, we limit the scope of this paper for the simple scenario, which is described in following sections.

IV. PROBLEM DEFINITION

The substrate network (SN) is made up of N domains that are linked by inter-domain links, and each domain consists of computing servers and links that inter-connect the servers. The resources of the SN is described primarily by the bandwidth of each inter-domain link and the resources of each domain refers to CPU power, and memory and storage capacity of each compute server and bandwidths of the links that interconnect the computing servers.

The virtual networks (VNs) that are mapped on top of the SN consists of nodes described in terms of required CPU power, memory, and storage capacity, and links that interconnects the nodes, which is described by the required bandwidths. The topology of the VN is denoted as G^V, and its partitioned VNs are denoted as G_i^V. Note that by definition of partitioning, the relationship between G^V, G_i^V, G_j^V is shown in (1).

$$\begin{cases} G_i^V \cap G_j^V = \theta \ \ \forall i, j, i \neq j \\ \quad \cup G_i^V = G^V \end{cases} \tag{1}$$

The cost for mapping a node of a VN to a substrate node and for mapping a link of a VN to substrate links are defined as $f_i : V \to R^+$ and $g_i : V \to R^+$, respectively. We assume that these functions are convex and monotonically increasing. The reason for such properties is to penalize any mapping that uses a large amount of a single SN resource. In another words, such cost functions promote load-balancing. In addition, such kind of penalty is common in everyday life to discourage overuse, such as the electricity.

Lastly, the cost function for inter-domain link utilization is denoted by h_{Li}. We also assume that h_{Li} is convex and monotonically increasing for the same reason as the other two cost functions. In addition, since the inter-domain links are more expensive than the intra-domain links (generally a magnitude higher), we let $h_{Li}(x) > g_j(x)$ for all $x \geq 0, i, j$ [7].

Finally, the objective of the provisioning algorithm is to minimize (2). Essentially it is a scaled sum of all the cost functions where α_i, β_i, and γ_i are the weight factors that can be adjusted for each domain.

$$C = \sum\sum \alpha_i f_i\left(X_i^j\right) + \beta_i g_i\left(Y_i^j\right) + \sum\sum h_{L_{ij}}\left(Z_{ij}^{kl}\right) \tag{2}$$

V. PROPOSED ALGORITHM

As mentioned earlier, the provisioning algorithm or the VNE problem is a NP-Hard problem; thus, only heuristic solutions exist. Among many heuristic approaches, our provisioning algorithm is based on graph isomorphism detection algorithm and graph clustering algorithm.

A. Motivation for using graph clustering

Readers may get confused, at first, since splitting VN into multiple sub-VNs will incur inter-domain link costs, which is about a magnitude higher than the intra-domain links. However, there are three cases that may require VN splits. One very obvious case is due to resource constraints of the SN. i.e. VN requires more resource than what a single SN offers. Another case is when a user asks some parts of his/her VN to be provisioned in different SNs. The last case is more subtle — the cost of provisioning partitioned VNs is cheaper than provisioning the entire VN as a whole. As an example, consider a SN that consists of two domains where domain A is very cheap but has limited amount of resources and domain B is very expensive but has abundant amount of resources. Suppose that a VN to be provisioned in such environment is too big for domain A. In such case, partitioning may offer a cheaper provisioning cost despite the expensive inter-cloud link cost.

The next natural question is how to split the virtual networks. The most obvious ways is to use the minimum cut from graph theory. The minimum cut of a graph is a cut that minimizes the sum of weights that are in the cutset. Thus, if the bandwidths between the nodes in a VN were used as the edge weights, the minimum cut would minimize the inter-partition bandwidth. The minimum cut algorithm has been studied extensively in the graph theory community, and there are efficient algorithms [8-9]. There may be cases where a VN is split not once but multiple times; and the minimum-k cut algorithm could be used to partition a VN into k partitions. The minimum k-cut algorithm is known to be NP-complete if *k* is part of the input [2]. However, there are several approximation methods with an approximation ratio of 2-2/k; and one popular algorithm is using a Gomory-Hu Tree [13].

Although minimum cut and minimum k cut minimizes the inter-domain bandwidth among partitioned sub-VNs, the partitioning based on a minimum cut has some weaknesses. One of such weakness is that it often causes an unbalanced partition. In other words, most of the nodes are grouped into one side of the partition [12]. In such case, the sub-VNs with more nodes may require too much resource, which requires additional partitioning, which in turn translates to higher inter-cloud link cost.

B. Graph Clustering

Graph clustering provides a mean to analyze a graph based on other attributes, such as size of partitions, in addition to the edge weights. In a general setting, the main idea behind the clustering algorithm is to find groups with similar elements and separate non-similar elements. Some of the criteria that can be used to measure similarity or in another words, quality of intra-clusters, are expansion and conductance as shown in (3) and (4), respectively. In (3) and (4) *w(u,v)* is the edge weight between node u and v; in (3), $\left(S, \bar{S}\right)$ is a cut of the graph and in (4), $c(S) = c(S,V) = \sum_{u \in S} \sum_{v \in V} w(u,v)$.

Graph clustering with low conductance is believed to be superior to minimum cuts because it takes into account the orders of the sets that are being cut apart, yielding often in more significant separations. Unfortunately, the clustering algorithm based on low conductance is a NP-Hard problem; and, there are many heuristic methods [14].

$$\min_{S} \Psi(S) = \frac{\sum_{u \in S, v \in \bar{S}} w(u,v)}{\min\left\{|S|, |\bar{S}|\right\}} \tag{3}$$

$$\min_{S} \Phi(S) = \frac{\sum_{u \in S, v \in C-S} w(u,v)}{\min\left\{c(S), c(C-S)\right\}} \tag{4}$$

Two particular types of clustering that we are interested in are the hierarchical clustering and parametric clustering. Hierarchical clustering provides a convenient method to split the graph into multiple pieces at a cheap computational cost because of the recursive nature of the hierarchy [15]. For our purpose, if a VN topology or a sub-VN topology requires more resource than what a single SN can provide, the VN can be divided into sub-VNs as many time as necessary based on the hierarchical clustering.

Parametric clustering offers a convenient method to generate a hierarchical tree of clusters [18]. The idea behind parametric clustering is to allow some or all edges to be a function of some variable. Although this does not consider the entire evolving nature of the structure of the graph (since new nodes and edges are not dynamically added or deleted), it allows to describe evolution of the graph. The simplest and most studied case of parametric clustering is making only the links to the source and sink node parametric as shown in (5).

$$\begin{cases} w(s,v) \text{ is a nondecreasing function of } \lambda \text{ for all } v \neq t \\ w(v,t) \text{ is a nonincreasing function of } \lambda \text{ for all } v \neq s \\ w(u,v) \text{ is constant for all } u \neq s, v \neq t \end{cases} \tag{5}$$

The graph clustering algorithm that we use is based on [15], where the author proposes a parametric clustering algorithm based on Gomory-Hu tree. The author constructs a parametric graph by adding an artificial sink to a graph and connecting it with every node of the graph with the edge value of λ, as the parameter. The following is some of the interesting properties of the graph clustering that the authors prove [15], which is useful for our proposed algorithm.

- λ serves as an upper-bound for inter-cluster capacity and a lower-bound for intra-cluster capacity which is described in equation (6), where $s \in S, t \notin S, P \cup Q = S, P \cap Q = \varphi$

$$\frac{c(S,V-S)}{|V-S|} \leq \lambda \leq \frac{c(P,Q)}{\min\left(|P|,|Q|\right)} \tag{6}$$

- Extending the first bullet, with monotonically increasing value of λ between (0, ∞). The λ values at which the number of clusters changes are referred as breakpoints.

C. Proposed Algorithm

A more detailed description of our algorithm, which was initially explained in Section 3, is provided in this section. The proposed algorithm consists of the following major steps:

Step 1. CSO manager (AS Topology Analyzer) analyzes the VN provided by the VN customer — computes all breakpoints of the VN graph using the parametric graph analysis and constructs the hierarchical tree of clusters.

Step 2. CSO manager identifies the partitioned sub-VNs for a given break point λ.

Step 3. Every CSO Agent:

1. Runs an isomorphism detection algorithm to find a suitable mapping for each sub-VN;
2. Computes a cost vector for embedding each sub-VN.

Step 4. CSO Manager computes the allocation vector i.e. which CSO Agent embeds which sub-VN.

1. Sort the cost matrix, where each column represents sub-VNs in decreasing order of resource demand, and each row represents SNs in increasing order of available resource;
2. Starting from the sub-VN that requires the most amount of resource, allocate it to the SN with the least amount of available resource that can accommodate the request.
3. Compute the cost for embedding the sub-VNs to domains and the inter-domain bandwidth cost, if the VN splitting were used, based on (2).

Step 5. Iterate Step 2 to Step 4 until the total cost is no longer decreasing.

The inter-domain bandwidth cost computation, step 4-3, requires a link-mapping algorithm. As described in the related work, such problem is NP-Hard, thus, in our algorithm, we use the *k*-shortest path algorithm to find the link mappings, and compute the associated cost. The maximum number of iterations for Step 2 to 4 is $|N^V|$, where each node in the VN is considered as a single sub-VN.

VI. Simulation and Analysis

In this section, we present the preliminary simulation result of our work. The full simulation with larger number of nodes in both VN an SN is in progress at the time of writing this paper.

Substrate Network. For our simulation, we chose to use an extended star topology, since a star topology is the most popular one in the current cloud data centers. Our substrate network has four domains, each of which is connected to the aggregation switch/router, hence completing the first layer of the star. Each domain is again a star topology with 50 nodes connected to the Top of Rack switch.

Virtual Network. We chose to use the random graphs generated by the *Barabasi-Albert* model for the virtual network request [19]. The *Barabasi-Albert* model is one of the methods for generating random graphs that follow a power law [20]. Since such graphs may be used to describe the characteristics of the World-Wide-Web (www) links and social networks (e.g., Facebook), and such applications are most common at cloud data center applications, we chose to use the *Barabasi-Albert* model for generating random virtual network requests. For the simulation, we let each VN request to have 15 nodes with random link weights based on a uniform random distribution between [50, 100] Mbps.

Mapping cost function. The mapping cost of nodes and links are defined as the sum of the inverse of the remaining resource after provisioning the requested VN. This cost model severely penalizes a provisioning when it tries to allocate a VN to a substrate network that has a limited amount of resource. In addition, we define the inter-cluster communication link cost to be ten times higher than the intra-cluster communication link.

For the analysis, we compare our algorithms to existing Round Robin algorithm. The Round Robin algorithm does not consider any partitioning, and it will simply try to map the entire VN to one of the domains, that is, does not consider network stratum information. In our implementation, the Round Robin algorithm will try to the map the VN to the domain that has the largest amount of CPU resource.

Fig. 2 shows the ratio of the number of VNs accepted to the total number of VNs requested. For this simulation, the substrate network had 4 domains with 50 nodes each of which had 200 units and each intra-domain link had bandwidth of 1Gbps and 10Gbps of bandwidth for inter-domain links.

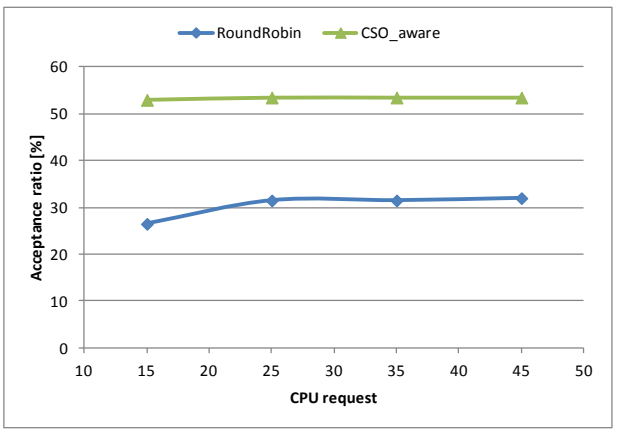

Figure 2. Simulation result

From the results, we can see that the acceptance ratio of the round-robin algorithm is not more than 35%, since it tries to map whole request into one domain, without any clustering mechanism or any consideration on NS information, and rejects the request if none of the domains could support the request. Our proposed algorithm based on CSO and graph clustering accepts more than 50% of VN requests. The reason for this is that our proposed algorithm, based on CSO and graph clustering, considers the number of nodes within each sub-VN as well as the sum of the weights of the cutset, and performed better than the Round-Robin provisioning algorithm of optimal utilization of CDC resources.

VII. CONCLUSION

In this paper, we proposed a provisioning algorithms for cloud data center based on graph clustering algorithm and cross-stratum optimization. Our algorithm performed significantly better than the traditional round robin algorithm that does not have any notion of partitioning the VN request, nor CSO-aware. In addition, proposed the graph clustering algorithm considers the number of nodes within each cluster as well as the weights of the cutset. Moreover, by considering application and network stratum information, proposed algorithm provides optimum and QoS-guaranteed resource allocation.

So far, we have only simulated our algorithm in a small scale cloud data centers. At the time of writing this paper, we are increasing the scale of our simulation to match a size of a mid-size cloud data centers to produce more realistic scenario. Moreover, provisioning of heterogeneous VN request while optimizing network resource usage and considering QoS of user application is still open issue.

ACKNOWLEDGMENT

This research was supported in part by KCC (Korea Communications Commission), Korea, under the "Novel Study on Highly Manageable Network and Service Architecture for New Generation" support program supervised by the KCA (Korea Communications Agency) (KCA-2011-10921-05003).

REFERENCES

[1] D. Andersen, Theoretical approaches to node assignment. Unpublished manuscript: <http://www.cs.cmu/edu/~dga/papers/andersen-assign.ps>, 2002.

[2] S. Kolliopoulos, C. Stein, "Improved approximation algorithms for unsplittable flow problems," in *Proc. IEEE FOCS*, 1997, pp. 426-435.

[3] M. Yu, Y. Yi, J. Rexford, M. Chiang, "Rethinking Virtual network Embedding: Substrate Support for Path Splitting and Migration," *ACM SIGCOMM Computer Communications Review*, Vol. 38, Issue 2, April 2008.

[4] J. Botero, X. Hesselbach, A. Fischer, H. Meer, "Optimal Mapping of Virtual Networks with Hidden Hops," Telecommunication Systems, 2011, pp.1-10. In Proc. *IEEE Network Operations and Management Symposium (NOMS)*, 2010.

[5] M. Chowdhury, M. Rahman, R. Boutaba, "ViNEYard: Virtual Network Embedding Algorithms with Coordinated Node and Link Mapping," *IEEE/ACM Transactions on Networking*, no.99, 2011, p. 1-14.

[6] H. Moens, J. Famaey, S. Latre, B. Dhoedt, F. Turck, "Design and Evaluation of a Hierarchical Application Placement Algorithm in Large Scale Clouds," In *Proc. of 12th IFIP/IEEE International Symposium on Integrated Network Management*, 2011, pp. 137-144.

[7] F. Zaheer, J. Xiao, R. Boutaba, "Multi-provider Service Negotiation and Contracting in Network Virtualization" In *Proc. IEEE Network Operations and Management Symposium (NOMS)*, 2010.

[8] M. Stoer, F. Wagner, "A Simple Min-Cut Algorithm," *Journal of the ACM*, Vol. 22, No. 4, July 1997, pp. 585-591.

[9] D. Karger, C. Stein, "A New Approach to the Minimum Cut Problem," *Journal of the ACM*, Vol 43, No. 4, July 1996, pp. 601-640.

[10] Y. Xin, I. Baldine, A. Mandal, C. Heermann, J. Chase, A. Yumerefendi, "Embedding Virtual Topologies in Networked Clouds," In *Proc. Conference in Future Internet*, June 13-15, 2011, Seoul, Korea.

[11] I. Houidi, W. Louati, W. Ameur, D. Zeghlache, "Virtual network provisioning across multiple substrate networks," *International Journal of Computer Networks*, Vol. 55, Issue 4, pp. 1011-1023, Mar. 2011.

[12] J. Wang, H. Peng, J. Hu, C. Yang, "A Graph Clustering Algorithm Based on Minimum and Normalized Cut," In *Proc. of ICCS*, Part I, LNCS 4487, 2007, pp. 497-504.

[13] H. Saran, V. Vazirani, "Finding k-cuts within twice the optimal," In *Proc. 32nd Annual Symposium on Foundations of Computer Science*, pp. 743-751, 1-4 Oct. 1991.

[14] S. Schaeffer, "Graph Clustering," In *Computer Science Review*, Vol. 1, No. 1, Aug. 2007, pp. 27-64.

[15] G. Flake, R. Tarjan, K. Tsioutsiouliklis, "Graph Clustering and Minimum Cut Trees," In *Internet Mathematics*, Vol. 1, No. 4, 2004, pp. 385-408.

[16] Young Lee and et. al., "Research Proposal for Cross-Stratum Optimization (CSO) between Data Centers and Networks," draft-lee-cross-stratum-optimization-datacenter-00, January 2011.

[17] D. Dhody, "Cross Stratum Optimization enabled Path Computation," draft-dhody-pce-cso-enabled-path-computation-00, February 2012.

[18] G. Gallo, M.D. Grigoriadis, and R.E. Tarjan, "A Fast Parametric Maximum-Flow Algorithm and Applications," *SIAM Journal of Computing*, 18, 30–55. 1989.

[19] R. Albert and A.L. Barabasi, "Statistical mechanics of complex networks,"*Reviews of Modern Physics*, Vol. 74, Issue 1, January 2002, pp. 47-97.

[20] A. Clauset, C. R. Shalizi, M. E. J. Newman, "Power-Law Distributions in Empirical Data," *SIAM Review*, Vol. 51, Issue 4, pp. 661-703.

[21] Neal Seitz, "ITU-T Standards for IP-based Networks," *IEEE Communication Magazine*, Vol. 41, Issue 6, June 2003.