Graph Clustering based Provisioning Algorithm for Optimal Inter-Cloud Service Brokering

TaeSang Choi, Younghwa Kim, Sunhee Yang

Electronics and Telecommunications Research Institute
138 Gajeongno, Yuseong-gu
Daejeon, 305-700, S.Korea
{choits, yhwkim, shyang}@etri.re.kr

Abstract— **As virtualization technology matured, the concept of virtual network environment has emerged. One of the major beneficiary of the virtualization technology is the cloud computing technology. Cloud computing technology is intended to service various needs of its customers, and the flexibility of virtualization technology is its main enabler. This paper is related to efficient management of cloud computing technology, in particular optimal planning of cloud resources distributed across multiple cloud service providers. We proposed a graph clustering based provisioning algorithm and state its benefits over traditional provisioning algorithms based on the minimum cut from graph theory.**

Key words: cloud computing, graph clustering, minimum cut, 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; thus, making dynamic service offerings to the cloud customers.

Among many different aspects of management issues of cloud computing, such as monitoring, on-line fault management, we focus on the optimal provisioning of computing and network resources since we believe that the efficient provisioning algorithm is the first step to fully utilize the inter-cloud computing infrastructures.

Resource provisioning in inter-cloud environment falls in the broad category of the virtual network embedding problem (VNE). It tries to optimally map a user's virtual network (VN) to an underlying physical substrate network (SN). This problem has been widely studied [1]-[6].

In this paper, we introduce a graph clustering based resource provisioning algorithm. Instead of using the minimum cut that only considers sum of the edge weights of the cutset, the graph clustering considers the orders of the sets that are being cut apart, yielding often in more significant separations. In addition, we consider a hierarchical clustering algorithm to further enhance the provisioning algorithm.

The paper is organized as follows. Related work is provided in Section II. In section III, we introduce our provisioning management system architecture. Section IV provides the problem formulation. Our proposed algorithm is given in Section V. Then final concluding remark with description of our future work is given in Section VI.

II. RELATED WORK

Virtual Network Embedding (VNE) to a substrate network (SN) is known to be a NP-Hard problem. One of the most common approaches is dividing the VNE problem 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 substrate network that honors the bandwidth constraints is a multi-way separator problem, which is NP-Hard. In addition, the link mapping problem onto the substrate network is an Unsplittable Multi-Flow Problem, which again is NP-Hard [1].

In [2], the authors create a more virtual network (VN) friendly network, which allows a split of a logical link into multiple substrate links. With this relaxation, the link mapping problem becomes a multi-commodity flow problem. As for the node mapping, the authors proposes a greedy algorithm but taking account of the structure of both networks (virtual network and substrate network). Contrast to [2], [3, 4] tries to heuristically solve the Unsplittable Multi-Flow Problem, which is a MIP problem. In [3], authors make an extension with a concept of hidden hops. The hidden hops are the intermediate links that are on the path of a VN link. The author considers resource required to forward packets in such nodes using the concept of hidden hops. In [4], authors incorporate the node mapping problem into the link mapping problem, and provides 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, federation of multiple cloud computing data centers is frequent; thus, a VNE problem among multiple SN needs to be investigated. Perhaps the most apparent problem when having multiple SN is scalability. In [5], 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. By introducing intelligence to every management systems, the management systems under the administrator management system operate autonomously.

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 among SN providers. In [6], 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.

Another approach is using graph analysis techniques, mainly the minimum k cuts. In [9-10], the authors use minimum k cut algorithm to distribute the VN across multiple SNs. Since inter-cloud bandwidth is known to be more expensive than the intra-cloud bandwidth, the minimum cut algorithm, which minimizes the edge weights of the cutset, is a logical choice. However, since, minimum k cuts often is result in a highly unbalance cuts, the load distribution among multiple SNs may not be even [11].

III. PROVISIONING MANAGEMENT ARCHITECTURE

This section presents an overall management architecture followed by the description of provisioning management and their relationship. Fig. 1 depicts a scenario for virtual cloud network provisioning multiple substrate cloud.

In this scenario a Virtual Network (VN) Creator is a user, who requests provisioning for VN G^V with N^V nodes and E^V edges among the nodes. A VN Provider is an analog of service providers, and is responsible for the virtual network topology analysis; it splits the VN into number of clusters, which is described in the next section. Moreover, the VN Provider makes a decision about which cluster should be mapped to which substrate cloud. VN is provisioned across multiple infrastructure providers, which are managed by substrate cloud managers. Every cloud manager should keep up-to-date information about cloud network resources in the Resource Data Base (DB). The resources are number of nodes and links, node CPU, and link bandwidth. When a VN $G^V(N^V, E^V)$ request is received from the VN Creator (step 1 in Fig. 1), the Topology Analyzer analyzes the VN topology, and *i*-level hierarchy of clusters is generated for the requested G^V network (step 2). This hierarchy consists of *i* number of virtual clusters or sub-VNs. Each sub-VN information is passed to each cloud for the isomorphism detection (step 3).

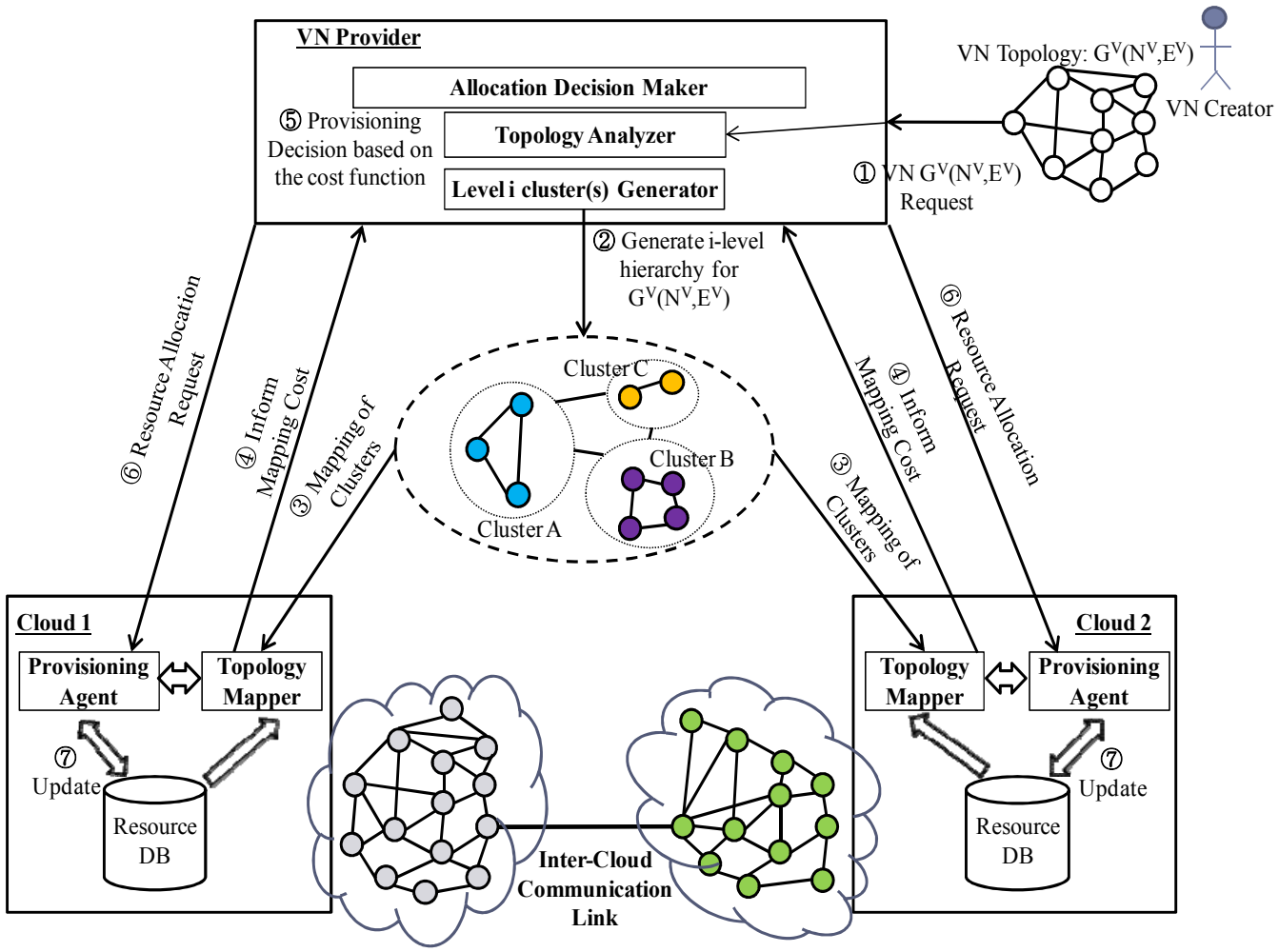

Fig. 1 Virtual Cloud Network provisioning scenario across multiple substrate clouds

The Topology Mapper of the substrate cloud manager maps G_i^V into cloud topology considering resource constraints of nodes (eg., CPU, memory) and links (eg., link bandwidth), and calculates the cost of node and edge mapping. A number of isomorphic cloud subgraphs - sub-VNs can be found; therefore, a set of mapping information will be generated, and including mapping cost be forwarded to the Allocation Decision Maker (step 4). Consecutively, Decision Maker calculates mapping cost function based on the possible minimum allocation cost of nodes, edges and inter-cloud communication link cost (step 5). Among of calculated cost functions for the different clouds, the cloud with the minimum provisioning cost function will be chosen for the resource allocation, and Decision Maker sends a resource allocation request to that cloud. The cloud's Provisioning Agent allocates the resources to the cloud network, and updates the resource DB with the allocation parameters and the available resource information.

IV. Problem Formulation

In this section, we formally describe the resource provisioning problem. Symbols used throughout the paper are described in Table 1.

The Substrate Network (SN) is made up of N domains that are linked by inter-domain links. On top of this topology, virtual networks (VNs) are mapped. The topology of a VN is denoted as G^V, and its partitioned sub-VNs are denoted as G_i^V. Note that by definition of partitioning, the relation between G^V and G_i^V is shown in (1).

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

TABLE 1. SYMBOLS

Symbol	Description
$G^{SN}(V,E)$	Topology of the Substrate Network (SN)
$G_i^{SN}(V,E)$	Topology of the clouds or Domains(*i=1…N)* of SN
$G^V(V^V,E^V)$	Topology of the Virtual Network (VN)
$G_i^V(V_i^V,E_i^V)$	Topology of sub-VN_i, *i=1…K*
w	Edge capacities w:E->R^+
u	Node capacities u:V->R^+
X_i^j	Used Edge capacity of cloud i by sub-VN j
Y_i^j	Used Node capacity of cloud i by sub-VN j
$Z_{ij}^{k,l}$	Used inter-cloud bandwidth between cloud i and j by the inter-cluster link between sub-VN k,l
$f_i(.)$	Cost function for Node Utilization of SN_i $f_i(): V \rightarrow R_0^+$
$g_i(.)$	Cost function for Link Utilization of SN_i $g_i(): E \rightarrow R_0^+$
h_{Li}	Cost function for Inter-Domain Utilization of between SNi and SNi+1 $h_{Li}(): R_0^+ \rightarrow R_0^+$

The cost function for using the nodes, links are defined as f_i:*V*->R_0^+ and g_i:*E*->R_0^+, respectively. We assume that these functions are convex and monotonically increasing. The reason for such properties is to penalize any VN that is trying to use larger amount of resource of SN. Such kind of penalty is common in everyday life to discourage overuse, such as the electricity bill.

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. 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 $x>=0, i, j$ [6].

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, 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 virtual network embedding algorithm is a NP-Hard problem; thus, only heuristic solutions exist.

Among many heuristic approaches, our provisioning algorithm is based on a graph isomorphism detection algorithm and VN topology partitioning. Before describing the details of the provisioning algorithm, the motivation behind the VN partitioning is provided.

A. Motivation Behind VN Partitioning

Readers may get confused, at first, since if VN is split into multiple sub-VNs, this will incur inter-domain link costs, which is about a magnitude higher than the intra-domain link. However, one very obvious case where VN partitioning is necessary is when the VN requires more resource (Node computing power, intra-domain BW) than what a single SN domain can provide.

Another case is when the user that requests for the VN asks for some of the VMs to be placed in certain clouds. In this case, VN partitioning is necessary to accommodate the request.

Last case is more subtle—the cost of partitioning and then provisioning is lower than provisioning the entire VN on a single SN domain. As an example, consider a SN with two clouds: cloud A that is very cheap but has limited amount of available resource and cloud B that is expensive but has abundant amount of resource. Suppose that a VN to be mapped onto this SN is too big for cloud A to handle but is small enough for cloud B. In this case, despite the expensive inter-cloud link cost between the two clouds, the provisioning the VN over two clouds may be optimal.

The next natural question is how to split the virtual networks. The most obvious way is to use minimum cut of graph. The minimum cut of a graph is a cut that minimizes the sum of weights that are in the cutset. Minimum cut has been studied extensively in the graph theory community, and there are efficient algorithms [7], [8].

In the resource provisioning setting, the topology can be viewed as an undirected graph with the link bandwidth between the nodes as the weights of the graph. The minimum cut algorithm is provides a mean to partition the VN into two disjoint sub-VNs with the minimum inter-domain bandwidth.

Even after partitioning, the partitioned VNs may still be too resource hungry. In such case, via the minimum *k* cut, the VN can be partitioned into k partitions [9]. The minimum *k*-cut algorithm is known to be NP-complete if *k* is part of the input [1]. However, there are several approximation methods with an approximation ratio *2-2/k*. One popular approximation algorithm is using Gomory-Hu Tree [12].

Although minimum cut and minimum *k* cut minimizes the inter-domain bandwidth among partitioned sub-VNs, 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 placed in one side of the partition [11]. In such case, the sub-VNs with more nodes may be too resource hungry to be provisioned in any single cloud, thus, requiring additional partitioning and higher inter-cloud link cost.

B. Graph Clustering

Graph clustering provides a mean to analyze the graph based on other attributes, such as size of partitions, in addition to the cutset 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 other words, quality of intra-clusters, are expansion and conductance as shown in (3) and (4) respectively for the graph *G(V,E)*. In (3) and (4) *w(u,v)* is the edge weight between node *u* and *v*. In (3), *(S,$\overline{S}$)* is a cut of the graph. In (4), *c(S) = c(S,V)* = $\sum_{u\in S}\sum_{v\in V} w(u,v)$

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

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

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. [13]

Two particular type of clustering that we are interested in are the hierarchical clustering and parametric clustering. There are two general approaches to hierarchical clustering depending on whether the hierarchy is created bottom-up or top-down: they are referred as agglomerative clustering or divisive clustering respectively. The divisive clustering is of our interest, since it can naturally be mapped to our provisioning algorithm. If a VN topology or a sub-VN topology requires more resource than a single SN can provide, the divisive clustering provides a mean to split the VN or the sub-VN into smaller sub-VNs.

Parametric clustering allows 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, it allows evolution of the graph. Since the seminal work by Gallo et all. in [GGT], the parametric clustering with weights of the links to the source and sink (with some restrictions described in (5)) of the parameter has been extensively studied. In (5), source node is denoted as *s* and sink node as *t*.

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

In [14], the author proposes a parametric clustering algorithm based on Gomory-Hu tree. In addition, the authors extend their algorithm to hierarchical clustering. By adding an artificial sink and connecting it with every node of graph with the parameter, λ, as the link weights, they derive some nice clustering properties based on λ:

- 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, ∞), an hierarchical clustering can be formed as shown in Fig. 1.

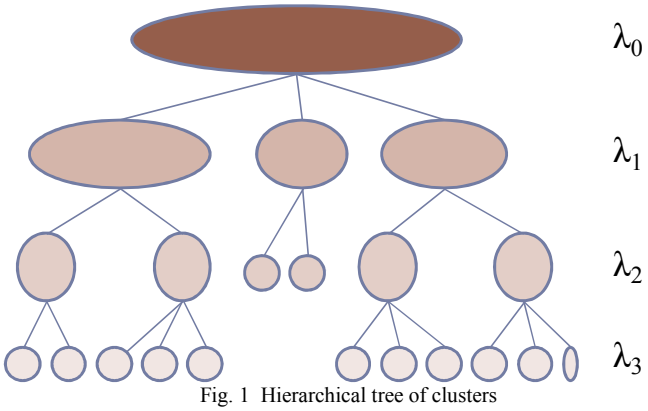

Fig. 1 Hierarchical tree of clusters

C. Proposed Algorithm

In this section proposed algorithm to deal with the problems mentioned in the Section IV-A is described. In the proposed algorithm, hierarchical graph clustering technique is integrated to generate potential virtual clusters. Algorithm 1

and Algorithm 2 depict the pseudocodes for the proposed algorithm.

The proposed algorithm consists of following major steps:

Step 1. Finding all breakpoints $\lambda \in (0, \text{maxCut}]$. In order to find all breakpoints of given G^V graph, we find *T*, a Gomory-Hu tree of $G'^V = G^V + t$ graph, and get the number of sub-VNs. If, number of sub-VNs is changed comparing to previous number, current lambda is added to the breakpoint set.

Algorithm 1.
GraphClustering((GV(NV, EV), lambda)

```
G'^V = G^V + t;
foreach node in N^V do
    Connect t with node;
    e(t, node) = lambda;
end
GTree = MinimumCutTree((G'^V(N'^V, E'^V));
foreach connection in t do
    cluster_set[connection] = connection.components;
end
remove t from GTree;
return cluster_set;
```

Algorithm 2.
HierarchicalGraphClusteringAlgorithm($G^V(N^V, E^V)$, $G^S(N^S, E^S)$)

```
// finds all available breakpoints based on binary search
breakpoint_set = GetAllBreakpoints (G^V(N^V, E^V));
foreach lambda in breakpoint_set do
    cluster_set = GraphClustering((G^V(N^V, E^V), lambda);
    foreach set in cluster_set do
        map_set = Subgraph Isomorphism((G^V(N^V, E^V), (G^S(N^S, E^S));
        cost[lambda] = minimum_cost(map_set);
        if (cost[lambda] < cost[lambda-1])
            lambda_optimum = lambda;
        end
    end
end
total_cost =cost[lambda_optimum]+interdomain_communication;
return total_cost, M(G^V, G^S)
```

Step 2. For each $\lambda \in (0, \text{maxCut}]$

1. Virtual cloud clusters: G_i^V clusters will be constructed with respect to current λ, where *i* is a number of virtual cloud clusters on λ breakpoint.
2. Isomorphism detection: every cluster in G_i^V cluster set goes through isomorphism detection, to get the possible provisioning cloud domain, if cloud domain can provide required resources for G_i^V.
3. Minimum cost of mapping: in real scenarios, usually G^S is much more larger than requested G_i^V, thus a number of isomorphic cloud subgraph can be result of isomorphism. The cloud subgraph with the minimum provisioning cost will be chosen, and the cost of mapping will be calculated for the virtual nodes and links.
4. Optimum breakpoint: if provisioning cost of current breakpoint is less than provisioning cost of previous breakpoint, current breakpoint becomes an *optimum breakpoint*.

Finding the optimum breakpoint at the worst case may take $|N^V|$ iterations, where each virtual node is considered as a single virtual cluster.

Step 3. Calculate the cost function of $M(G^V, G^S)$. Obtained information (node mapping cost, edge mapping cost and virtual cloud topology) along with the inter-cloud communication cost will be used to calculate the cost function of mapping using Equation (2).

Step 4. Update the substrate network information for the future requests.

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 and 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 [15]. The *Barabasi-Albert* model is one of the methods for generating random graphs that follow a power law [16]. 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.

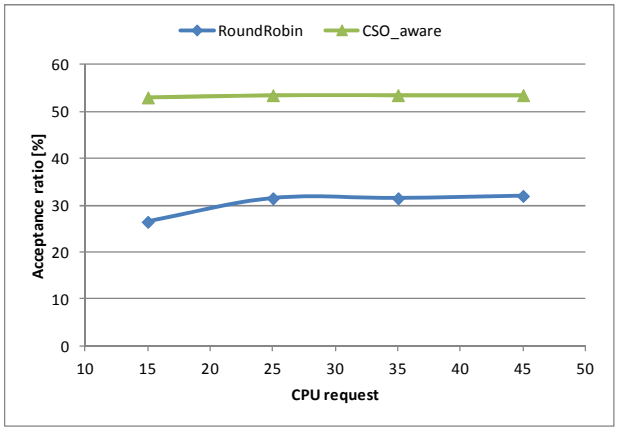

Figure 1. Simulation result

From the results, we can see that the acceptance ratio of the round-robin algorithm which is the one used by the referenced related work in the section II 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 graph clustering accepts more than 50% of VN requests. The reason for this is that our proposed algorithm, based on 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 AND FUTURE WORKS

In this paper, we proposed a provisioning algorithms for cloud data center based on graph clustering algorithm. Our algorithm performed significantly better than the traditional round robin algorithm that does not have any notion of partitioning the VN request. 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.

ACKNOWLEDGMENTS

This research was funded by the MSIP (Ministry of Science, ICT & Future Planning), Korea in the ICT R&D Program 2013

REFERENCES

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

[2] 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.

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

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

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

[6] F. Zaheer, J. Xiao, R. Boutaba, “Multi-provider Service Negotiation and Contracting in Network Virtualization”.

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

[8] 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.

[9] Y. Xin, I. Baldine, A. Mandal, C. Heermann, J. Chase, A. Yumerefendi, “Embedding Virtual Topologies in Networked Clouds,” in Conference in Future Internet 2011, June 13-15, 2011, Seoul, Korea.

[10] I. Houidi, W. Louati, W. Ameur, D. Zeghlache, “Virtual network provisioning across multiple substrate networks,” in Computer Networks, Vol. 55, Issue 4, pp. 1011-1023, Mar. 2011.

[11] J. Wang, H. Peng, J. Hu, C. Yang, “A Graph Clustering Algorithm Based on Minimum and Normalized Cut,” in ICCS 2007, Part I, LNCS 4487, pp.497-504.

[12] H. Saran, V. Vazirani, “Finding k-cuts within twice the optimal,” in Foundations of Computer Science, 1991. Proceedings., 32nd Annual Symposium on, pp.743-751, 1-4 Oct. 1991.

[13] S. Schaeffer, “Graph Clustering,” in Computer Science Review, Vol. 1, No. 1, Aug. 2007, pp.27-64.

[14] G. Flake, R. Tarjan, K. Tsioutsiouliklis, “Graph Clustering and Minimum Cut Trees,” in Internet Mathematics Vol. 1, No. 4, 2004, pp.385-408.

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

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