This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE ICC 2011 proceedings

FELL: A Flexible Virtual Network Embedding Algorithm with Guaranteed Load Balancing

Sheng Zhang†, Zhuzhong Qian†, Song Guo‡, Sanglu Lu†
† State Key Lab. for Novel Software Technology, Nanjing University, China
‡ School of Computer Science and Engineering, The University of Aizu, Japan
† zhangsheng@dislab.nju.edu.cn, {qzz,sanglu}@nju.edu.cn, ‡ sguo@u-aizu.ac.jp

***Abstract*—Network virtualization has emerged as the most promising approach to overcome the current ossification of the Internet. A key problem in it is how to efficiently and effectively make use of substrate network resources by embedding multiple virtual networks with various constraints. Due to its NP-hardness, many heuristic approaches have been proposed. However, most of them restricted the solution space at the expense of limiting practical applicability and did not consider response time requirements or load balancing. In this paper, we propose *FELL*, a Flexible virtual network Embedding algorithm with guaranteed Load baLancing for the general problem. Based on simulated annealing, *FELL* can flexibly control the tradeoff between results accuracy and running time to meet various requirements of different applications by changing parameters. A novel cost criterion that reflects the impact of distribution of allocated resources for an embedding is designed to guarantee load balancing, which enables substrate network to avoid resource fragmentation. Path splitting is supported to achieve better resource utilization by making use of small pieces of available bandwidth. We also design some key functions including generating initial and neighbor solutions. The effectiveness of our algorithm is finally validated by our simulations.**

I. Introduction

The Internet has been extremely successful in supporting lots of distributed applications and serving global commerce, communication and defense. However, its rapid growth and development is now creating hurdles for its further evolution. Its multi-provider nature makes the introduction of new technologies need consensus between competing providers [1][2]. Furthermore, deploying changes to Internet Protocol requires global agreement because of its end-to-end design [1].

Recently, *network virtualization* is widely viewed as the most promising approach to overcome the current ossification of the Internet [1][2][3][4][5], and it has been investigated in a large number of projects, including PlanetLab[6], VINI[7] and CABO[3]. Network virtualization allows service providers to deploy multiple heterogeneous *virtual networks* (VNs) on shared *substrate networks* (SNs) to offer customized value-added services to end users. Traditional *Internet Service Providers* are divided into : *infrastructure providers* (InPs) who manage the substrate network, and *service providers* (SPs) who manage virtual networks by leasing resources from InPs.

One of the fundamental problems in network virtualization, is how to embed multiple VNs with resource constraints such that SN resources are utilized in a most efficient and effective manner. Known as the *virtual network embedding/mapping* (VNE) problem, it is NP-hard [8]. To cope with its NP-hardness, researchers resorted to heuristics to reduce computational time. Ricci et al. [9] assumed exclusive use of substrate nodes, proposed the *Assign* algorithm based on simulated annealing and only bandwidth constraint was considered. Lu et al. [4] considered an offline version problem based on special network topologies. Fan et al. [10] investigated the dynamic topology configuration problem in overlay networks. Zhu et al. [11] assumed substrate nodes and links have unlimited CPU and bandwidth resources, focused on load balancing and on-demand assignments. Yu et al. [12] considered path splitting and path migration. Lischka et al. [13] proposed a backtracking algorithm based on subgraph isomorphism detection, but restricted length of substrate paths . Chowdhury et al. [14] proposed a linear programming, deterministic/randomized rounding based algorithm, but added locations constraints to simplify it. Other research works, such as [15][16], focused on distributed embedding protocols/architectures.

However, most of previous works fell into two broad types: (i) greedily mapping virtual nodes or mapping virtual links by adopting shortest path algorithms [11][12], (ii) restricting the locations of virtual nodes [14] or the length of substrate paths [13]. Both of them restricted the solution space at the expense of limiting practical applicability and did not consider response time requirements or load balancing.

Also, none of existing solutions can flexibly support real applications. We observed different requirements on response time from various applications. For real-time requests, system should respond in a short time and disregard whether mapping is an optimized one, which does not matter because lifetime of such applications is often relatively short. On the other hand, for applications without requirements on response time, results accuracy is more important because they normally last a long time. Consequently, we need balance the tradeoff between running time and results accuracy.

As VNE is often treated as an optimization problem, the embedding *cost* of a VN request is a deterministic factor to the final embedding. Most of existing algorithms apply a very simple embedding cost proportional to the allocated resources for a VN. Even worse, load balancing is always neglected in these studies, resulting in fragmentation of residual SN resources [12] after embedding some VN requests. In such circumstances, some parts of SN become excessively loaded while others are idle. This unbalanced resource allocation

978-1-61284-233-2/11/$26.00 ©2011 IEEE

TABLE I
NOTATIONS IN THIS PAPER

Substrate Network		Virtual Network	
G^s	Substrate network	G^v	Virtual network
N^s	Nodes set	N^v	Nodes set
E^s	Links set	E^v	Links set
A_c	CPU capacity	C_c	CPU constraint
A_b	Bandwidth capacity	C_b	Bandwidth constraint
P^s	Paths set		

causes inefficient resource utilization and new VN requests to be rejected even though residual resources are sufficient.

This motivates us to propose *FELL*, a Flexible virtual network Embedding algorithm with guaranteed Load baLancing. Simulated annealing [17], a type of metaheuristics, was originally developed for *Very Large Scale Integration* design and now widely used in optimization problems because of its simplicity and controllability, is adopted as the basic framework of *FELL*. We use its parameters to control the tradeoff between results accuracy and running time. To guarantee load balancing, a novel cost criterion that reflects the impact of resources allocation of an embedding is designed to conduct embedding process. Path splitting is supported to make SN accept more VN requests. We also design some key functions including generating initial and neighbor solutions.

II. Network Model And Problem Formulation

A. Elementary Notations

Commonly, we use a weighted undirected graph to model a network, vertices represent nodes while edges represent links. Each vertex is associated with a CPU capacity/constraint while each edge is associated with a bandwidth capacity/constraint. The notations are summarized in Table I for reference. The notation system in this paper is similar to that in [12][14]. The principle behind these notations is that superscript indicates SN/VN and subscript indicates CPU/bandwidth. Fig. 1 is an example of these notations. The corresponding number near each vertex/edge is the CPU/bandwidth capacity/constraint.

B. Virtual Network Embedding

The residual CPU resource $R_c(n^s)$ of a substrate node n^s and residual bandwidth resource $R_b(e^s)$ of a substrate link e^s are defined as: $R_c(n^s) = A_c(n^s) - \sum_{\forall n^v} f_c(n^v, n^s)$ and $R_b(e^s) = A_b(e^s) - \sum_{\forall e^v} f_b(e^v, e^s)$, where $f_c(n^v, n^s)$ denotes the CPU resources of n^s allocated to n^v and $f_b(e^v, e^s)$ denotes the bandwidth resources of e^s allocated to e^v. Note that $f_c(n^v, n^s)$ is always equal to $C_c(n^v)$, but $f_b(e^v, e^s)$ is not so, because that a virtual link may be mapped into several substrate paths. Since bandwidth is a concave type metric, the residual bandwidth $R_b(p^s)$ of a substrate path p^s is defined as: $R_b(p^s) = min_{e^s \in p} R_b(e^s)$.

The embedding of a single VN request have two main parts: *node mapping* and *link mapping*. The node mapping $\mathcal{M}_N : N^v \rightarrow N^s$ maps a virtual node to a substrate node, subjecting to: $\forall n^v, m^v \in N^v$, (i) $\mathcal{M}_N(n^v) \in N^s$, (ii) $\mathcal{M}_N(n^v) = \mathcal{M}_N(m^v)$ iff $m^v = n^v$, (iii) $R_c(\mathcal{M}_N(n^v)) \geq C_c(n^v)$. $\mathcal{M}_N(n^v)$ is seen as the substrate host of n^v.

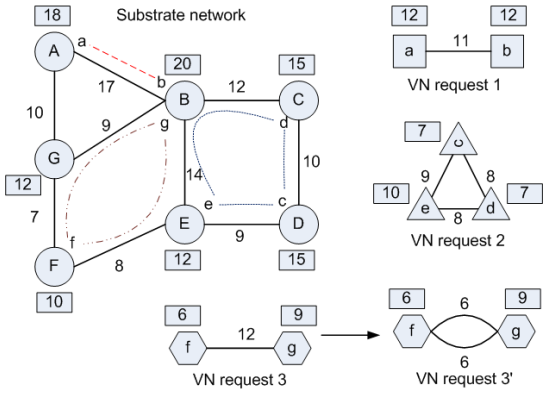

Fig. 1. An example of virtual network embedding

For link mapping, we take path splitting into account. Path splitting (or multi-path [18]) has been propounded as an effective way to aggregate high bandwidth [19][20], increase resiliency against failures to offer reliable adaptive routing [21], it was also investigated in network virtualization relevant literature recently [12][14]. Path splitting based link mapping $\mathcal{M}_E : E^v \rightarrow (P^s)^{maxSplit}$ maps a virtual link to several substrate paths, subjecting to : $\forall e^v = (m^v n^v) \in E^v$, (i) $\mathcal{M}_E(m^v n^v) \in (P^S(\mathcal{M}_N(m^v), \mathcal{M}_N(n^v)))^{maxSplit}$, (ii) $\sum_{p \in \mathcal{M}_E(e^v)} R_b(p) \geq C_b(e^v)$, where $maxSplit$ is the number of substrate paths that a virtual link is mapped into. Fig. 1 is an example for node and link mapping. For VN request 3, the virtual link between f and g is mapped into two substrate paths $F-G-B$ and $F-E-B$, i.e., $maxSplit = 2$.

C. Problem Formulation

Virtual network embedding is done by InPs. From their standpoints, a natural objective is to increase *revenue* and to decrease *cost*. However, a SP is only willing to pay for fees proportional to their required resources, so the revenue of embedding a VN request can be defined as: $\mathcal{R}(G^v) = \omega_c \sum_{n^v \in N^v} C_c(n^v) + \omega_b \sum_{e^v \in E^v} C_b(e^v)$ (following the definition in [14]), where ω_c, ω_b are the corresponding weights for CPU and bandwidth respectively.

We can see from above definition that given a VN request, revenue is fixed. In order to get more profit, an InP should try to make SN accept more VN requests. To this end, the *cost* of embedding single VN request should be minimized. Previous definitions of the embedding cost mainly focused on minimizing the amount of resources allocated to virtual links (Note that resources allocated to virtual nodes is constant), they seldom considered impact of the distribution of allocated resources. We will propose a novel cost criterion for embedding cost, which reflects both the amount of allocated resources to a VN and impact of the allocated resources distributions in next section.

The VNE problem can be formulated as: given a SN G^s and a set of VN requests which arrives independently, find an embedding for the latest arrived VN request G^v to minimize its embedding cost.

III. A Flexible Virtual Network Embedding Algorithm with Guaranteed Load Balancing

Simulated annealing (Details of simulated annealing can be found in [17][22][23]) is a metaheuristics method which optimizes a problem by iteratively improving a candidate solution with regard to a given measure of *expense*, which determines how much expense we pay for a solution. (Note that the embedding *cost* is different from this *expense*, the former is to quantify the economic cost of an embedding from infrastructure providers' standpoint while the latter is to evaluate how "good" a solution is from the standpoint of our algorithm.) With the help of reheating [23], the basic framework of *FELL* is as Algorithm 1 shows.

The tradeoff between running time and results accuracy can be achieved by decreasing $maxAnneal$, $maxReheat$ if application is ephemeral and requires a short response time and increasing them if application will exist long enough. ρ is typically a value between 0.95 and 1 and the probability p is defined as $p(\mathcal{M}, \mathcal{M}', T) = e^{-\frac{cost(\mathcal{M}')-cost(\mathcal{M})}{T}}$ according to [17][23][22].

A. Supporting Path Splitting

Path splitting enables better substrate resource utilization and makes SN accept more VN requests. It also provides fault-tolerance and quick reactive recovery by reassigning resources between the set of substrate paths that a virtual link is mapped into. Fig. 1 shows an example. After embedding VN request 1 and 2, SN can not accept VN request 3 beacause there is not any substrate link that has a more than 12 units bandwidth. But we can get 6 units bandwidth from $F-G-B$ and another 6 units from $F-E-B$ to form 12 units bandwidth (VN request 3') if we supports path splitting. Path splitting is easy to implement in *FELL*. When a VN request arrives, we change it into a multi-graph as input of *FELL* by adding additional edges between virtual nodes according to $maxSplit$.

B. Generating Initial Temperature and Solution

Kirkpatrick [17] suggested that the initial temperature is set to be a value that results in average acceptance probability equals to 0.8 for bad mutatings from initial solution. Hence, we randomly generate lots of cost-increasing mutatings and compute the average increase in cost $\Delta cost = cost(\mathcal{M}') - cost(\mathcal{M})$, then the initial temperature $T = -\frac{\overline{\Delta cost}}{ln(0.8)}$.

Since the candidate solution mutates for a sufficient long time and arrives at a feasible solution at last, we can randomly generate an initial solution even though it is unfeasible.

C. Generating Neighbors

Function *neighbor* is required to alter current candidate solution in some particular ways and generates a slightly different new one. It should be kept simple, otherwise simulated annealing will consume additional time. We also parameterize it for adjusting performance and supporting uniformly random moves within the whole solution space. We accomplish it by:

Node remapping: (i) randomly choose a virtual node n_c^v; (ii) randomly choose integer h such that $1 \leq h \leq maxHop$; (iii) let n_c^v randomly walk h hops away from its current corresponding substrate host, arrive at a new one, and use it as its new substrate host.

Link remapping: Suppose the chosen node in node remapping is n_c^v, its original host is n_{old}^s and its new host is n_{new}^s. For each virtual link e^v with an end point n_c^v, if n_{new}^s is not a node on the original path $\mathcal{M}_E(e^v)$, we find a new path (by random or BFS [24]) from n_{old}^s to n_{new}^s with bandwidth requirement $C_b(e^v)$ and extend the original path by adding the new generated path. If not, we use a part of the original path as the new path.

Algorithm 1 Basic Framework of *FELL*

```
initial temperature T, initial random solution M
for reheat= 1 to maxReheat do
    for anneal= 1 to maxAnneal do
        M' ← neighbor(M)
        if expense(M')<expense(M) then
            M ← M'
        else
            do M ← M' with a probability p
        end if
        T ← ρT
    end for
    T ← 2T
end for
return M
```

D. Designing Embedding Cost and Solution Expense

For the embedding cost, previous works mainly thought that it is proportional to the amount of allocated resources, then the common definition of embedding cost $\mathcal{C}(G^v)_{com}$ for G^v is:

$$\mathcal{C}(G^v)_{com} = \alpha_c \sum_{n^v \in N^v} C_c(n^v) + \alpha_b \sum_{e^v \in E^v} \sum_{e^s \in E^s} fb(e^v, e^s)$$

where $fb(e^v, e^s)$ denotes the bandwidth resource allocated on e^s for e^v. α_b, α_b are corresponding weights.

But they did not consider fragmentation of SN resources. After embedding some VN requests, residual SN resources become fragmented, some parts of SN become excessively loaded while some others are idle. This unbalanced resources distribution leads to inefficient resource utilization and further makes some VN requests can not be accepted although there are enough substrate resources. To avoid this, we add another item[1] to the above definition.

$$\mathcal{C}(G^v)_{new} = \alpha_c \sum_{n^v \in N^v} C_c(n^v) + \alpha_b \sum_{e^v \in E^v} \sum_{e^s \in E^s} fb(e^v, e^s) + \alpha_b^d \sum_{e^v \in E^v} \sum_{p \in \mathcal{M}_E(e^v)} D_b(p)$$

where $e^v = (m^v n^v)$ and p is one of the substrate paths that e^v is mapped into. Let $p = (n_1^s, n_2^s, ..., n_k^s)$ (here $n_1^s = \mathcal{M}_N(m^v)$,

[1]This new item is inspired by the *WCETT* path metric proposed in [25] in wireless mesh network.

$n_k^s = \mathcal{M}_N(n^v))$ and $Var[x_1, x_2, .., x_m]$ be the mathematical variance of m variables $(x_1, x_2, ..., x_m)$, then

$$D_b(p) = Var[\{R_b(n_1^s n_2^s), R_b(n_2^s n_3^s), ..., R_b(n_{k-1}^s n_k^s)\}]$$

is the mathematical variance of the residual bandwidth of links on p. A small variance indicates that the residual resources are evenly distributed, i.e., load balancing is achieved. Hence, the smaller $\mathcal{C}(G^v)_{new}$ the better.

The solution expense function, used to evaluate the candidate solution, should reflect the following principles: (i) A solution which uses less substrate resources (i.e., the length of substrate path is averagely shorter) incurs less expense than that uses more resources; (ii) A solution which evenly allocates the required resources (i.e., guarantees load balancing) incurs less expense than that causes hot spots or links; (iii) A solution which does not meet the constraints of a VN request incurs much more expense than that satisfies.

Therefore, the solution expense function includes: (i) the embedding cost defined above, i.e., $\mathcal{C}(G^v)_{new}$; (ii) the penalty of violating QoS constraints. Then it can be defined as:

$$expense(\mathcal{M}) = \beta(\alpha_c \sum_{n^v \in N^v} (C_c(n^v) - Allocated(n^v)) + \alpha_b \sum_{e^v \in E^v} (C_b(e^v) - Allocated(e^v))) + \mathcal{C}(G^v)_{new}$$

where β denotes penalty coefficient.

IV. Performance Evaluation

In this section, we first describe simulation settings, then present comparison method and some numerical results.

A. Simulation Settings

As network virtualizaiton is still an open field, our simulations follow the similar settings to [12][14]. We use the GT-ITM tool [26] to generate the SN topologies, which are configured to have 50 nodes. The values of CPU and bandwidth capacity of SN are generated uniformly from [50,100]. The arrivals of VN requests are modeled as a Poisson process with an average rate of 2 VN requests per minute. For each VN request, the number of nodes is uniformly generated from [2,10] and each pair of them are connected with probability 0.5. The CPU and bandwidth constraints of VN are uniformly generated from [0,20] and [0,50] respectively. Its lifetime is assumed to be exponentially distributed with an average of 180 seconds.

B. Simulation Results

For previous works [10][9][4][11][13], it is difficult to compare our algorithm with them because problem formulations varies a lot. Algorithm in [12] has a similar performance to *R-ViNE* [19], so we only compare *FELL* with the latest *R-ViNE* [14]. Recall that an InP's revenue is proportional to the allocated resources for all VN requests, so we use node/link utilization as the comparison metrics. Figures about link utilization are omitted because of space limitation and its similarity to node utilization. Node utilization is measured using stress defined in [11][14]. Every point is the average of 100 times of comparisons. Some key observations are summarized in the following:

(1) FELL could have better results if we sacrifice time efficiency. Fig. 2 shows the comparison between *FELL* and *R-ViNE*. We can see that the increase of running time controlled by parameters R (R for $maxReheat$) and A(A for $maxAnneal$) makes the results of our algorithm change from worse (the green line) than *R-ViNE* to better (the blue line) than *R-ViNE*. Fig. 3 reports that increasing $maxReheat$ and $maxAnneal$ improves embedding results but brings more running time. Therefore, *FELL* can flexibly meet various requirements of different applications without major changes.

(2) It does not mean that the larger $maxHop$ the better. Fig. 4 shows the node utilization for varying $maxHop$. The cases of $maxHop = 3, 4, 5$ were also conducted in simulations, we omit them due to the same reasons mentioned above. Fig. 4 tells that SN accepts more VN requests when $maxHop = 2$ than $maxHop = 10$, which conflicts with intuitions. The main reason is that the simplest neighbor function brings no restrictions and makes the candidate solution walk randomly within the whole solution space.

(3) A larger $maxSplit$ makes no difference when the small piece of bandwidth is smaller than the minimal residual bandwidth of all substrate links. It is intuitive to see that, when $maxSplit$ is getting larger, pieces that the bandwidth constraints are divided into are getting smaller, consequently, the VN requests are getting much more easily accepted. However, when the small piece of bandwidth is smaller than the minimal residual bandwidth of all substrate links, a larger $maxSplit$ makes no differences. Fig. 5 illustrates the effect of $maxSplit$. The results are close, because the range of bandwidth constraints in simulations is small to some extent that $maxSplit = 2$ is enough.

(4) Random is better than BFS to neighbor function. It is also because that Random gives much higher probability that the candidate solution walks uniformly and freely than BFS, as Fig. 6 shows.

(5) A proper ratio of α_b to α_b^d is better. Fig. 7 shows that $\alpha_b = 2, \alpha_b^d = 1$ is better than other two settings. The main reason is that a large α_b makes the embedding cost primarily depend on allocated bandwidth resources while a large α_b^d make the embedding cost be greatly determined by distribution of allocated resources, i.e., load balancing or not. Hence, a proper ratio can achieve better results.

V. Conclusions

In this paper, we propose *FELL* for virtual network embedding problem. Based on simulated annealing, parameters are used to achieve flexibility. A novel embedding cost criterion is designed to guarantee load balancing for the purpose of avoiding resource fragmentation. Some key functions including generating neighbors are designed. Path splitting is also supported to enable better resource utilization. The effectiveness of our algorithm is confirmed by simulations.

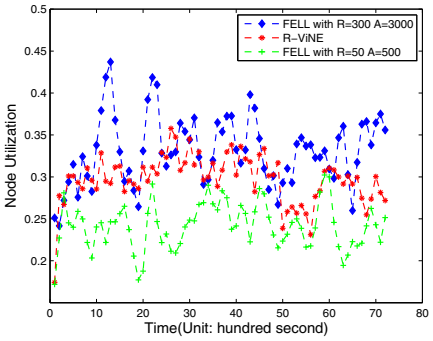

Fig. 2. *FELL* vs. *R-ViNE*

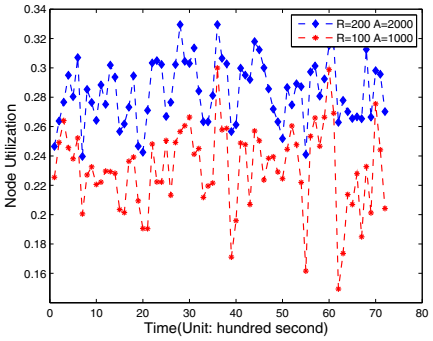

Fig. 3. Effect of $maxReheat$ & $maxAnneal$

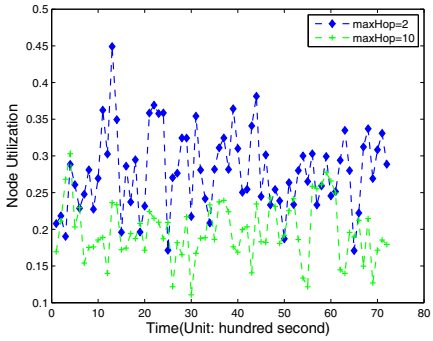

Fig. 4. Node utilization for varying $maxHop$

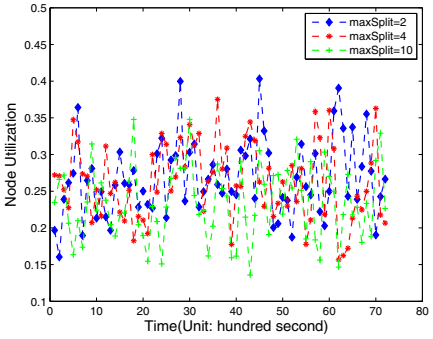

Fig. 5. Node utilization for varying $maxSplit$

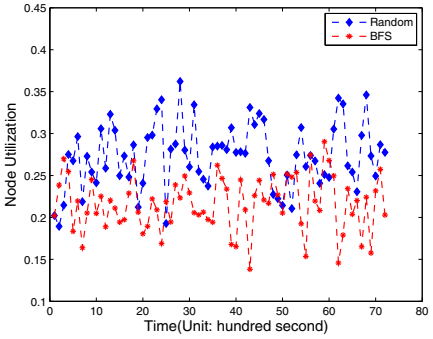

Fig. 6. Comparison of different ways to generate solution neighbors

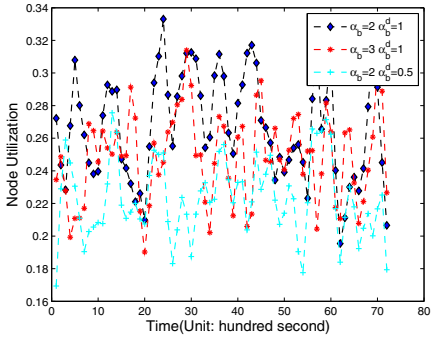

Fig. 7. Effect of ratio of α_b to α_b^d

ACKNOWLEDGMENT

This work is partially supported by the National Natural Science Foundation of China (No. 61073028, 60721002); the National Basic Research Program of China (973 Program, No. 2009CB320705) and Jiangsu Science and Technology Support Program (No. BE2010179).

REFERENCES

[1] J. Turner and D. Taylor, "Diversifying the internet," in *IEEE GLOBECOM 2005*, vol. 2, 2-2 2005, pp. 6 pp. –760.

[2] N. M. M. K. Chowdhury and R. Boutaba, "A survey of network virtualization," *Computer Networks*, vol. 54, no. 5, pp. 862–876, 2010.

[3] N. Feamster, L.-X. Gao, and J. Rexford, "How to lease the internet in your spare time," *SIGCOMM Comput. Commun. Rev.*, vol. 37, no. 1, pp. 61–64, 2007.

[4] J. Lu and J. Turner, "Efficient mapping of virtual networks onto a shared substrate," *Washington University, Tech. Rep. WUCSE-2006-35*, 2006.

[5] T. Anderson, L. Peterson, S. Shenker, and J. Turner, "Overcoming the internet impasse through virtualization," *Computer*, vol. 38, no. 4, pp. 34 – 41, april 2005.

[6] PlanetLab, "http://www.planet-lab.org/."

[7] VINI, "http://www.vini-veritas.net/."

[8] D. G. Andersen, "Theoretical approaches to node assignment," Dec. 2002, unpublished Manuscript.

[9] R. Ricci, C. Alfeld, and J. Lepreau, "A solver for the network testbed mapping problem," *SIGCOMM Comput. Commun. Rev.*, vol. 33, no. 2, pp. 65–81, 2003.

[10] J. Fan and M. H. Ammar, "Dynamic topology configuration in service overlay networks: A study of reconfiguration policies," in *IEEE INFOCOM 2006*, april 2006, pp. 1 –12.

[11] Y. Zhu and M. Ammar, "Algorithms for assigning substrate network resources to virtual network components," in *IEEE INFOCOM 2006*, april 2006, pp. 1 –12.

[12] M. Yu, Y. Yi, J. Rexford, and M. Chiang, "Rethinking virtual network embedding: substrate support for path splitting and migration," *SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 2, pp. 17–29, 2008.

[13] J. Lischka and H. Karl, "A virtual network mapping algorithm based on subgraph isomorphism detection," in *ACM VISA 2009*. New York, NY, USA: ACM, 2009, pp. 81–88.

[14] N. Chowdhury, M. Rahman, and R. Boutaba, "Virtual network embedding with coordinated node and link mapping," in *IEEE INFOCOM 2009*, 19-25 2009, pp. 783 –791.

[15] C. Marquezan, J. Nobre, L. Granville, G. Nunzi, D. Dudkowski, and M. Brunner, "Distributed reallocation scheme for virtual network resources," in *IEEE ICC 2009*, 14-18 2009, pp. 1 –5.

[16] I. Houidi, W. Louati, and D. Zeghlache, "A distributed virtual network mapping algorithm," in *IEEE ICC 2008*, 19-23 2008, pp. 5634 –5640.

[17] S. Kirkpatrick, "Optimization by simmulated annealing: quantitative studies," *Journal of Statistical Physics*, vol. 34(5/6), pp. 975–986, 1984.

[18] J.-C. Chen and S.-H. Chan, "Multipath routing for video unicast over bandwidth-limited networks," in *IEEE GLOBECOM 2001*, vol. 3, 2001, pp. 1963 –1967 vol.3.

[19] X. Chen, M. Chamania, A. Jukan, A. Drummond, and N. da Fonseca, "Qos-constrained multi-path routing for high-end network applications," in *IEEE INFOCOM Workshops 2009*, 19-25 2009, pp. 1 –6.

[20] T. Anjali, A. Fortin, G. Calinescu, S. Kapoor, N. Kirubanandan, and S. Tongngam, "Multipath network flows: Bounded buffers and jitter," in *IEEE INFOCOM 2010*, 14-19 2010, pp. 1 –7.

[21] W. Zhang, J. Tang, C. Wang, and S. de Soysa, "Reliable adaptive multipath provisioning with bandwidth and differential delay constraints," in *IEEE INFOCOM 2010*, 14-19 2010, pp. 1 –9.

[22] I. H. Osman, "Metastrategy simulated annealing and tabu search algorithms for the vehicle routing problem," *Ann. Oper. Res.*, vol. 41, no. 1-4, pp. 421–451, 1993.

[23] A. Anagnostopoulos, L. Michel, P. V. Hentenryck, and Y. Vergados, "A simulated annealing approach to the travelling tournament problem," in *IJCAI*, 2003, pp. 1357–1358.

[24] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, 2nd ed. The MIT Press, September 2001.

[25] R. Draves, J. Padhye, and B. Zill, "Routing in multi-radio, multi-hop wireless mesh networks," in *ACM MOBICOM 2004*. New York, NY, USA: ACM, 2004, pp. 114–128.

[26] E. Zegura, K. Calvert, and S. Bhattacharjee, "How to model an internetwork," in *IEEE INFOCOM 1996*, vol. 2, 24-28 1996, pp. 594 –602 vol.2.