Research on Virtual Network Mapping Algorithm with Path Splitting Based on Sort Preprocessing

Yong Huang [1,2,3]
[1]School of Information Science and Engineering, Guangxi University for Nationalities, Nanning, China
[2] Chengdu Institute of Computer Application, Chinese Academy Of Sciences, Chengdu, China
[3]Guangxi Key Laboratory of Hybrid Computation and IC Design Analysis, Nanning, China
Email: gxunhy@163.com

Jinzhao Wu[1,2,3]
[1]School of Information Science and Engineering, Guangxi University for Nationalities, Nanning, China
[2] Chengdu Institute of Computer Application, Chinese Academy Of Sciences, Chengdu, China
[3]Guangxi Key Laboratory of Hybrid Computation and IC Design Analysis, Nanning, China
Email: jinzwu@126.com

***Abstract*—Based on the previous research, a virtual network mapping algorithm with repeatable embedding over substrate nodes is summarized, in which, the virtual nodes in the same virtual network may be assigned to the same substrate node so that some virtual links don't need to be mapped to reduce the substrate link costs and improve the mapping effectively. Additionally, in the link mapping process, path splitting is introduced to make best use of some low bandwidth to make more virtual networks mapped, which is similar to the multi-commodity flow problem. Meanwhile, we classify the virtual network requests before mapping, map the virtual networks without link splitting request firstly and assign those with it secondly. The experimental results show that the proposed algorithm and the improved scheme perform better in mapping percentage, acceptance percentage and revenue.**

***Index Terms*—Virtual network, Mapping algorithm, Path splitting, Sort preprocessing**

I. INTRODUCTION

The Internet has been a great success in the past few decades and has provided a whole new way to access and exchange information. Indeed, as the main server of communication and information in modern time, it plays a critical role in work, business, education, entertainment, and other aspects of society. However, the internet is also a victim of its own success because its size and scope make the introduction and deployments of new network technologies and services very difficult [1]. Specifically, due to the multi-provider nature of the Internet, adopting a new network architecture or modification of an existing one requires not only changes in individual routers and hosts, but also joint agreements among IPSs. In fact, the Internet can be considered to be suffering from "ossification" [1, 2], a condition where technical and technological innovation meets natural resistance.

Network virtualization provides a promising way to address the ossification of the Internet and it has recently received a considerable amount of attention in both academia and various fields of industry as an important method for designing the Internet architecture of the future [1-8]. The concept of network virtualization has been proposed as a potential solution for diversifying the future Internet architecture into separate Virtual Networks (VN), which can simultaneously support network experiments, services and architectures over a shared substrate network [2, 7, 8]. A VN is a group of virtual nodes connected by virtual links.

One of the fundamental functions of network virtualization is the mapping or assigning of substrate network resources to individual virtual nodes and links. The set of virtual nodes and virtual links of VNs should be mapped to a specific set of substrate nodes and substrate paths, respectively. A substrate path is a logical path between two substrate nodes which may be a single substrate link or a sequence of substrate links. As shown in figure 1, multiple VNs may share the same underlying physical network.

The main objective in solving the virtual network mapping problem is to make efficient use of the underlying resource, while still satisfying the set of previously defined mapping constraints (e.g. topology constraints, bandwidth, CPU capacity). A virtual network mapping algorithm should be able to handle online requests which arrive dynamically and also offer admission control, since some VN requests must be rejected or postponed to avoid violating the resource guarantees of the existing virtual network [11]. Additionally, it is very important as quickly as possible to map the VN because VN mapping is only a sub function of network virtualization, which may contain many other functional processes building on the VN mapping. A faster mapping algorithm guarantees the success of other processes in network virtualization.

doi:10.4304/jcp.8.9.2413-2420

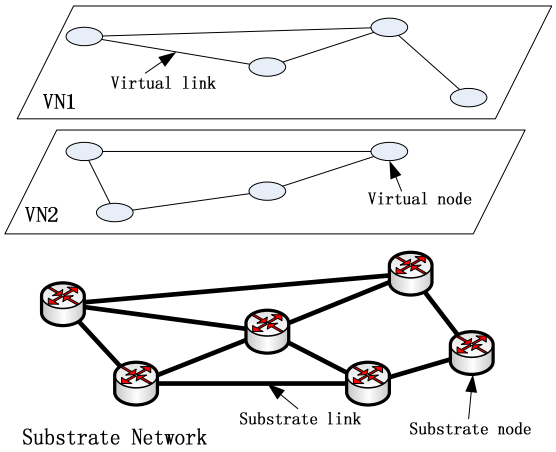

Figure 1. Mapping of VNs to a shared substrate network

In this work, based on previous research in the field of VN mapping, we summarize a general mapping algorithm as the basis of all other advanced algorithms. Due to the weaknesses in link mapping, we advocate a developed algorithm based on the K-shortest path, which searches the K-shortest path for increasing *K*, and stops the search if one with enough bandwidth capacity is found. The search process enhances the percentage of link mapping effectively. Up to now all mapping algorithms support that virtual nodes share substrate nodes, and that a substrate node may receive multiple virtual node requests, but virtual nodes in the same VN must be mapped to different substrate nodes. In fact, it is unnecessary unless there is geographical constraint. In this paper, we propose a new mapping algorithm supporting repeatable mapping over substrate nodes, in which multiple virtual nodes in the same virtual network can be mapped to the same node in the substrate network if the resource capacity is sufficient. Meanwhile, we classify the virtual network requests before mapping, map the virtual networks without link splitting request firstly and assign those with it secondly. That not only cuts down some virtual link mapping processes and reduces the mapping time, but also saves the substrate link cost and allows more virtual networks to be mapped. Our simulation experiments show that the new algorithm is much better in mapping percentage and revenue.

The remainder of this paper is organized as follows. Section 2 discussed the related work and section 3 presents the network model and formalizes the VN assignment problem. VN mapping algorithms are given in Section 4. In section 5, we evaluate the performance of the proposed mapping approach in different network settings and conclude the paper in section 6.

II. Related Work

VN mapping or embedding is a key area of research in the network virtualization [2, 20]. In order to maximum the number of co-existing VNs it is very important to determine how to map a VN request onto the physical network. But the mapping problem, with constraints on nodes and links, can be reduced to the NP-hard problem even when all requests are known in advance [2, 20].

The VN mapping problem shares similarities with the virtual circuit routing and virtual private network (VPN) design problems. All of them involve finding paths for source/destination pairs. In both the routing and VPN design problem, locations of source/destination pairs that need to be connected are given as input parameters. In network virtualization, however, the mapping between the VN and the substrate network could be arbitrary, and this extra degree of freedom increases the complexity of the problem. Furthermore, in VPN design as well as the static routing problem, only link utilizations are considered, while in the problem of VN assignment we could consider the mapping of both nodes and links.

Previous research has proposed many approaches solving the VN mapping problem from different perspectives [8-15, 18, 19]. Reference [8] studied the problem of determining dynamic topology reconfiguration for service overlay networks with dynamic communication requirement, and the ideal goal was to find the optimal reconfiguration policies that could minimize the potential overall cost of using an overlay. A solver for the network testbed mapping problem, building on simulated annealing, was explored in [9], which was in production use on the Netbed shared network testbed. In [11] and [15], the authors proposed heuristic greedy algorithms to maintain low and balanced stress among all substrate nodes and links during the VN assignment process. The overall objective was to achieve a near optimal VN mapping solution. In [11], virtual link embedding was simplified by allowing the substrate network to split a virtual link over multiple substrate paths. Besides, the multi-agent based approach was introduced to ensure distributed negotiation and synchronization between the substrate nodes [12]. Authors in [10] solved the VN mapping problem by introducing the traffic model, which involved selecting a network topology comprising virtual links and virtual routers, where the links were provisioned with sufficient capacity to accommodate any traffic pattern permitted by the given constraints. In addition, the integer and mixed integer programming approaches were applied to the VN assignment problem [14]. In [13], authors proposed a backtracking algorithm based on a subgraph isomorphism search method that mapped nodes and links during the same stage. In fact, the algorithm in [13] only found a feasible solution, which might not be the optimal solution because the search process didn't build on the heuristic idea.

All the previous mapping algorithms assigned the virtual nodes in the same VN onto different substrate nodes while the virtual nodes from different VNs were allowed to be assigned to the same substrate node if the constraint was satisfied. In fact, it is unnecessary to limit the nodes in the same VN to be mapped into different nodes unless there is the location constraint. In our work, we present a new mapping algorithm allowing the repeatable assignment over the substrate nodes even in the same VN mapping process.

III. Network Model And Mapping Problem Formulation

Substrate Network Model

The substrate network can be represented by a weighted undirected graph $G_s = (V_s, E_s)$, where V_s denotes the set of substrate nodes and E_s is the set of substrate links between nodes of the set V_s. Each substrate node $v_s \in V_s$ is associated with a capacity weight value $C(v_s)$ which refers to the available CPU capacity of node v_s. Similarly, each substrate link $e_s \in E_s$ is associated with a capacity weight value $W(e_s)$ which denotes the available bandwidth capacity of e_s. The right side of Fig.2 shows a substrate network, where the numbers over the links represent the available bandwidths and the rectangles are the CPU resources available at the nodes.

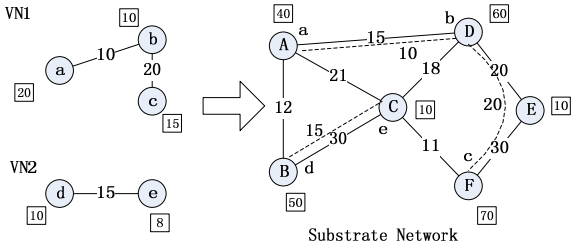

Figure 2. An example of VN mapping

Virtual Network Model

Customers should send requests to the substrate provider to set up on-demand VN topologies with different capacity parameters over the shared substrate. We denote a virtual network request by a weighted undirected graph $G_v = (V_v, E_v)$, where V_v and E_v have the similar definition in the substrate network model. In the same way, each virtual node $v_v \in V_v$ is associated with a required CPU capacity denoted by $C(v_v)$ and each virtual link $e_v \in E_v$ is associated with a capacity weight value $W(e_v)$, which refers to the required bandwidth capacity of e_v. Fig.2 depicts two VN requests, where the numbers over the links represent the required bandwidths and the rectangles are the required CPU resources at the nodes.

Additionally, each VN request has an arrival time and a life time. The arrival time is the time a VN request should be mapped onto the substrate network and the life time denotes the time period a VN could last on the substrate network. When the life time of VN is over, the substrate resource occupied by the corresponding VN would be released so that other VN requests can be assigned. Therefore, VN mapping is a dynamic process.

Mapping Problem Formulation

Based on the substrate and VN model, the challenge is to find the best mapping between the virtual graph G_v and substrate graph G_s so that the required resources in G_v can be satisfied. The mapping is defined as follows:

$M: G_v \mapsto (V_s', P')$, where $V_s' \subseteq V_s$, denotes the set of substrate nodes that virtual nodes are mapped to $P' \subseteq P$, here, P denotes the set of loop-free substrate path which may contain multiple links and P' denotes the objective set virtual links are mapped to. Generally, a mapping process contains two phases:

1) *Node Mapping*: All virtual nodes should be mapped onto the substrate nodes and the formalization is denoted as $M^V: V_v \mapsto V_s$, subject to $M^V(v_v) \in V_s, \forall v_v \in V_v$, $RC(M^V(v_v)) \geq C(v_v)$ in which, $RC(v_s)$ refers to the residual resources of substrate node v_s and can be defined as:

$$RC(v_s) = C(v_s) - \sum_{M^V(v_v)=v_s} C(v_v) \quad (1)$$

2) *Link Mapping*: Each virtual link must be mapped onto a substrate path which connects two substrate nodes corresponding to two virtual nodes connected by the virtual link. Link mapping can be denoted as $M^E: E_v \mapsto P$, subject to

$M^E(uv) \in P(M^V(u), M^V(v)), \forall uv \in E_v$, where both u and v denote the virtual nodes and uv denotes a virtual link connecting the virtual node u and v. $P(M^V(u), M^V(v))$ refers to the substrate path that virtual link uv is mapped onto. Note that the bandwidth capacity of every link in the substrate path $P(M^V(u), M^V(v))$ must be more than or equal to $W(uv)$.

In Fig.2, after successful mapping of VN1 request, virtual nodes *a*, *b*, *c* are mapped at the substrate nodes *A*, *D*, *F*, respectively and corresponding virtual links (*a*, *b*), (*b*, *c*) are assigned onto the substrate paths (*A*, *D*) and (*D*, *E*, *F*). All required capacities of VN1 request are satisfied. Particularly, both links in the path (*D*, *E*, *F*) must satisfy the required bandwidth of the virtual link (*b*, *c*). The request of VN2 is mapped in the same way.

Objectives

From the substrate network provider's point of view, the main interest of VN mapping is to assign VN requests as much as possible at the minimum cost. Since VN mapping is the foundation of network virtualization, the time the VN mapping process takes is a key factor impacting virtualization, and the faster mapping algorithm is preferred.

To measure the VN mapping revenue, as in previous research, we denote the revenue of serving the VN request $G_v(t)$ at time *t* by $R(G_v(t))$, which is defined as:

$$R(G_v(t)) = \alpha_1 \cdot \sum_{e_v \in E_v} W(e_v) + \alpha_2 \cdot \sum_{v_v \in V_v} C(v_v) \quad (2)$$

Here, $W(e_v)$ and $C(v_v)$ are the bandwidth and CPU requirements for the virtual link e_v and the virtual node v_v respectively. α_1 and α_2 are the weight of virtual links and virtual nodes respectively, and the values are determined by the real applications. At time *t*, the

larger $R(G_v(t))$ is, the more virtual resource is mapped, and the larger revenue the substrate providers achieve. However, during a long period, VN request assignment is a dynamic process. We introduce the long-term average revenue R/T to measure the mean revenue in the long period, which is defined as:

$$R/T = \lim_{T\to\infty} \frac{\sum_{t=0}^{T} R(G_v(t))}{T}. \quad (3)$$

In addition, with the same revenue, the mapping at the lower cost is more qualitative. To express the quality of VN mapping, we introduce the revenue-to-cost ratio R/C defined as:

$$R/C = \frac{\sum R(G_v)}{\sum Cost(G_v)}. \quad (4)$$

Here, $Cost(G_v)$ denotes the cost mapping VN request G_v, and is given as following:

$$Cost(G_v) = \alpha_3 \cdot \sum_{e_v \in E_v} (W(e_v) \cdot length(M^E(e_v))) + \alpha_4 \cdot \sum_{v_v \in V_v} C(v_v), \quad (5)$$

where, $length(M^E(e_v))$ refers to the length of substrate path that virtual link e_v is mapped to. α_3 and α_4 have the similar definition as α_1 and α_2 in equation (1).

Ⅳ. Virtual Network Mapping Algorithms

A. Basic Mapping Algorithm

Most mapping algorithms use greedy approaches in two stages, solving the node mapping in the first stage and doing the link mapping in the second one. The motive of the greedy idea is to map the VN to the substrate nodes and links with the maximum substrate resources so as to minimize the use of the resources at the bottleneck nodes/links. In this paper, we use the basic VN assignment scheme in [15] as a building block for all other advanced algorithms. In the node mapping stage, we select the maximum available resource node in substrate network first. Note that for a substrate node v_s, we do not use CPU capacity alone as the metric, because we not only want to make sure that there is enough CPU capacity available, but also take bandwidth capacity into consideration to prepare for the subsequence link mapping stage. Therefore, we define the amount of available resources for a substrate node v_s by:

$$AR(v_s) = RC(v_s) \cdot \sum_{e_s \in L(v_s)} RW(e_s), \quad (6)$$

where $RC(v_s)$,defined in equation (1), denotes the residual resource of substrate node v_s , and $RW(e_s)$ denotes the residual bandwidth of substrate link e_s . $L(v_s)$ is the set of all adjacent substrate links of v_s .

The intuition behind this is that virtual nodes with higher degree will setup more virtual links. All of these virtual links will go through some of the substrate link in the neighborhood of the corresponding substrate node. Therefore, the above matching tends to reduce the congestion and enhance the success of mapping.

After all virtual nodes are mapped, we need to map the virtual links using the shortest-path algorithm [16] to select the path between corresponding substrate nodes. The basic algorithm is described in Algorithm 1.

Algorithm 1 Basic VN mapping algorithm

1. Initial $V_A = \phi$ // V_A is the set of selected substrate nodes
2. $V_A = V_A \cup \{\max_{v \in V_s} AR(v)\}$
3. For i = 2,…, $|V_v|$
   a. $V_A = V_A \cup \{\min_{v \in V_s - V_A} \pi(v)\}$.
4. Assign nodes in V_A to virtual nodes so that virtual nodes with higher degree are mapped to substrate nodes with higher *AR*.
5. Find the shortest-path for all virtual links.

B. Link Mapping Based on K-shortest Path

In basic algorithm, to reduce the substrate link cost, link mapping is built on the shortest path algorithm. However, the shortest path selected in step 5 may not meet the required bandwidth capacity of the virtual link, which may lead to VN mapping failure. To improve the percentage of link mapping, the K-shortest path algorithm [16] is introduced to map the virtual links. For each virtual link of VN request, we search the K-shortest paths for increasing *K*, and stop the search when we find one with enough bandwidth capacity to map the corresponding virtual link. Thus, link mapping is efficiently improved so that the percentage of VN successful assignment can be enhanced. The developed algorithm [21] based on K-shortest path is described in Algorithm 2.

Algorithm 2 VN mapping algorithm based on K-shortest path

1. Initial $V_A = \phi$, K=1 // V_A is the set of selected substrate nodes
2. $V_A = V_A \cup \{\max_{v \in V_s} AR(v)\}$
3. For i = 2,…, $|V_v|$
   a. $V_A = V_A \cup \{\min_{v \in V_s - V_A} \pi(v)\}$.
4. Assign nodes in V_A to virtual nodes so that virtual nodes with higher degree are mapped to substrate nodes with higher *AR*.
5. For each virtual link of the request, search the K-shortest path for increasing K until a path with enough bandwidth capacity is found and K reset to 1.

C. Repeatable Mapping over Substrate Nodes

Up to now, all VN mapping algorithms restrict each virtual node in the same VN to a different substrate node, but allow the virtual nodes in different VNs to be mapped to the same substrate node. In fact, it is unnecessary to restrict the nodes in the same VN to be mapped to the different substrate nodes unless there is a requirement of geographical location for the corresponding virtual nodes. Furthermore, the algorithm with the repeatable mapping over substrate nodes need not map the virtual links between the corresponding virtual nodes because the communication bandwidth in the substrate node can be considered as infinity.

In this subsection, a new mapping algorithm[21] is proposed to support repeatable mapping on the substrate nodes no matter whether the virtual nodes are in same VN or not. To reduce the substrate links cost, we map the virtual nodes in the same VN to the same substrate nodes as far as possible and make the virtual nodes with higher required resource be mapped to the substrate nodes with larger *AR* (defined in equation (6)). We introduce the aggregating resource of virtual nodes to denote the required resource combining the CPU capacity and bandwidth of virtual links going through the corresponding virtual node. The aggregating resource of virtual node v_v is defined as follows:

$$AR(v_v) = C(v_v) \cdot \sum_{e_v \in L(v_v)} W(e_v) . \quad (7)$$

During the link mapping stage, based on the K-shortest path algorithm, we only map the virtual links between the virtual nodes mapped to the different substrate nodes. The detail description of the algorithm is given in Algorithm 3.

Algorithm 3 VN mapping algorithm with repeatable mapping over substrate node

1. Sort(V_s) according to *AR* defined in equation (6) from high to low;
2. Sort(V_v) according to *AR* defined in equation (7) from high to low;
3. For each virtual node $v_v \in V_v$:
   a. i =0; v_s = *get* (V_s, i);
   b. if: $RC(v_s) \geq C(v_v)$, do mapping $M: v_v \mapsto v_s$, update available CPU capacity;
   c. else: check if v_s has assigned the virtual nodes in the same VN. Yes, resort(V_s), go to step a;
   d. No, i ++, v_s =get(V_s, i), go to step b;
4. For each virtual link between the virtual nodes mapped to the different substrate nodes, search the K-shortest path for increasing *K* until a path with enough bandwidth capacity is found.

D. *Path* Splitting Based on Sort Preprocessing

In practice, some virtual network link to allow diversion request is marked as SVN (Splitting VN); while some restrictions due to the application request to the virtual network is not allowed to link diversion, marked UVN (Un-splitting VN). In general, the link resources are relatively scarce or sporadic cases, UVN mapping is more difficult compared to SVN. So, when a large number of simultaneous requests VN mapping, the first classification of the VN request, the first mapping UVN request, the request after mapping SVN, you can ensure that physical resources are relatively abundant UVN in the case of the map. SVN request because the link can take advantage of scattered resources, so after receiving a request UVN, the physical network can still receive requests for SVN. Thus, by request of the classification VN, VN can effectively improve the rate of acceptance. The algorithm is described as algorithm 4.

Algorithm 4 VN mapping algorithm with sort preprocessing

1. For time t, the requests of VN are classified into UVN collection and SVN collection;
2. For each request of UVN, algorithm 2 is executed;
3. For each request of SVN, algorithm 3 is executed.

V. PERFORMANCE EVALUATION

We have implemented a discrete event simulator in JAVA platform to evaluate the performance of our algorithms. The actual characteristics of substrate and virtual network are not well understood since network virtualization is still an open field. Therefore, similar to [11], we utilize a synthetic network to study the trends and quantify the benefits of repeatable mapping over substrate nodes. In this section, we evaluate the performance of the above mapping algorithms in different simulation settings. We first observe the performance affected by VN settings, and then present the evaluation results in the condition that a large number of VNs arrive and leave. Finally, we present the advantages of the strategy which utilizes node repeatable mapping in the case of path-splitting.

A. *Effects of Virtual Network Settings*

1) *Simulation Settings*

This experiment focuses on the performance of algorithms affected by the single VN settings, such as VN size, required resource and connectivity. Due to the low percentage of link mapping in the basic algorithm, we cannot clearly get the mapping performance in the single VN mapping process. In this experiment, we only compare the performance of the algorithm based on K-shortest path and algorithm with repeatable mapping over substrate nodes.

Substrate network topology is generated by the GT-ITM tool [17] and consists of 100 nodes and 495 links, which corresponds to the scale of a medium-sized ISP. The CPU capacity of nodes and bandwidth of the links follow a uniform distribution from 0 to 100 units. The VN resources including both CPU and bandwidth follow a uniform distribution from 0 to β units, and β is determined by the concrete settings. α_1, α_2, α_3, α_4 and δ are set to 1. To reduce the effects of random factor, 100 VNs are generated in each setting, and we evaluate the average performance measured by revenue-cost ratio (*R/C*-ratio, defined in equation (4)) and runtime. The larger the *R/C*-ratio is, the better the algorithm is. In the algorithm based on K-shortest path, the *R/C*-ratio is less than or equals to 1, but may be more than 1 for the algorithm with repeatable mapping over substrate nodes.

2) *Effects of VN Split Radio*

In this sub experiment, the bandwidth capacity of virtual links follows a uniform from 0 to 10, i.e. β =10. The connectivity of VN (i.e. probability that there is a virtual link between a pair of virtual nodes) is fixed at 0.5. The size of VN is set to 10, and each size includes 100 VNs. The relation between VN request acceptance Radio and VN split radio is shown in Fig.3, where split denotes the algorithm based on K-shortest path, and NodeRep

denotes algorithm with repeatable mapping over the substrate nodes.

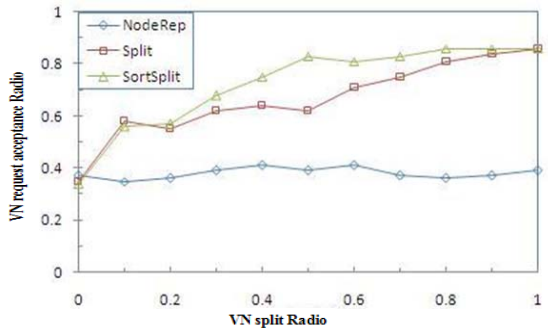

Figure 3. VN split radio vs. *acceptance radio*

3) *Effects of VN Required Resources*

We are interested in the algorithm performance influenced by the VN required resources including CPU capacity and bandwidth. In this sub experiment, VN size is fixed to 10, and the connectivity is set to 0.5 again. We increase β_1 (upper limit of VN resources) from 10 to 100 and generate 100 VNs for each β_1 value and evaluate the average performance.

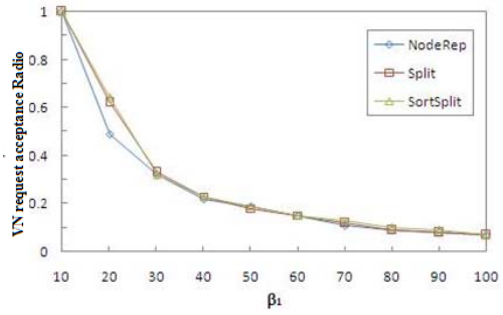

Figure 5. β_1 vs. *acceptance* ratio

In Fig.5, the benefit of NodeRep is weakened with the increasing β_1. The ratio of NodeRep is the same times as that of Kshort with β_1 =10. As β_1 increases, CPU capacities are increased so that the probability of repeatable mapping over the substrate nodes declines. So the trend of decrease is the same as the three algorithms.

We increase β_2 (upper limit of link resources) from 10 to 100 and generate 100 VNs for each β_2 value and evaluate the average performance.

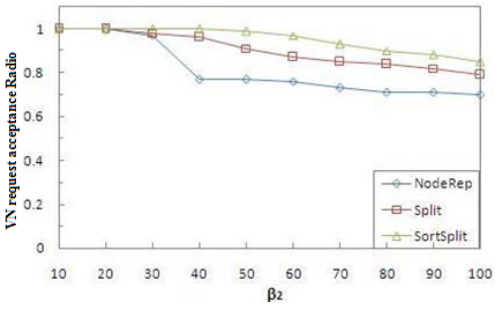

Figure 6. β_2 vs. *acceptance* ratio

Fig.6 shows that acceptance radio declines as β_2 increasing for all algorithms. However, no matter what values β_2 is set to 10 or 20 , NodeRep takes the less time than Kshort.

4) *Effects of VN Connectivity*

This sub experiment focuses on the performances influenced by VN connectivity. The VN size is fixed to 10, and we evaluate the performances with VN connectivity increasing from 0.1 to 1 in two cases that β_2 =15 and β_2 =30, which β_1 is fix to 10.

Fig.7 depict that Kshort algorithm has the similar simulation results with β_2 =15 or 30. For NodeRep, *the* ratio decreases slightly if β_2 =15 and strongly if β_2 =30 However, the acceptance radio of SortSplit keeps stable with β_2 =10 and 30. Therefore, NodeRep algorithm runs more effectively for the VNs with lower required resources, which is consistent with the results in Fig.5. From Fig.7, we can see that NodeRep performs better for VNs with dense connectivity when the required resources of VNs are low and has effective performance for sparse VNs with high required resources. No matter what settings of VNs, NodeRep algorithm performs better than Kshort.

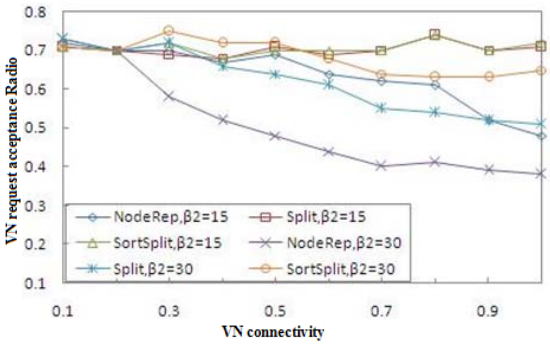

Figure 7. VN connectivity vs. acceptance radio

Dynamic Evaluation

1) *Simulation Settings*

In this experiment, we study the performances of 3 algorithms in the case that a large number of VN requests arrive and leave as the time windows are going. The substrate network is the same as that in section 5.1, consisting of 100 nodes and 495 links. CPU capacities and bandwidth follow a uniform distribution from 0 to 100. VN size is uniformly distributed from 2 to 10. The average VN connectivity is fixed to 0.5. VN requests arrive in a Poisson process with an average rate of 5 VNs per 10 time windows, and each one has an exponentially distributed lifetime with an average of μ =100 time windows. We run all of our simulations for 5000 time windows, which correspond to about 2500 requests on average in one instance of simulation.

2) *Evaluation Results*

We use several performance metrics for evaluation purposes, including the VN request acceptance ratio, *R/C*-ratio and average revenue over time (defined in equation

(3)). We denote the basic algorithm in subsection 4.1 as Base here.

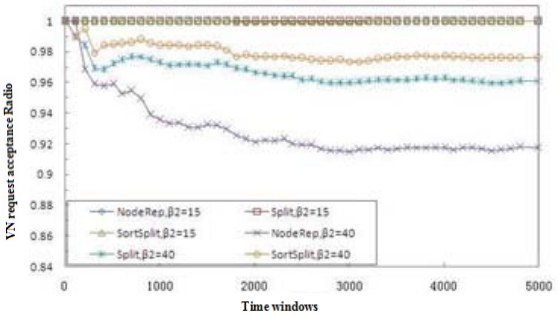

Figure 9. VN mapping percentage of algorithms

Fig.9 depicts the VN request acceptance ratio of three algorithms with β_2 ={15, 30}. With β_2 =15, both Kshort and NodeRep algorithms keep 100% VN acceptance, while Base algorithm only has acceptance percentage between 50% and 60%. The reason is that Base algorithm assigns the virtual link to the shortest substrate path, which may not satisfy the bandwidth capacity of the corresponding virtual links so the percentage of VN mapping is low. With β_2 =30, the mapping percentage of all three algorithms is less than that with β_2 =15, among which NodeRep keeps about 90%, Kshort reduces to about 80% and Base is only less than 40%. From Fig. 9 we can see that the scheme of repeatable mapping over substrate nodes and link mapping based on K-shortest path improves the percentage of VN mapping effectively.

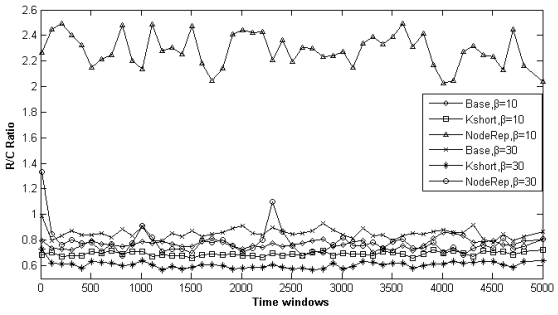

Fig.10 *R/C*-ratio of algorithms

In Fig.10, with β =10, NodeRep algorithm performs much better, and Kshort has slightly larger R/C-ratio than Base. However, with β =30, Base algorithm has the best performance in *R/C*-ratio because larger β leads to the smaller probability of repeatable mapping over nodes in NodeRep algorithm, so the link cost would increase. However, Base algorithm selects the shortest path to map the virtual link so that the link cost is the lowest, which leads to a high *R/C*-ratio. In fact, *R/C*-ratio may hide the performance of mapping percentage. Base algorithm has a low percentage of VN mapping, but performs best in *R/C*-ratio with β =30.

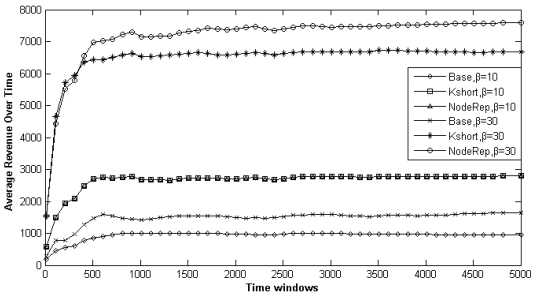

Fig.11 Average revenue over time

Fig.11 illustrates the average revenue over time. Bigger revenue shows that more virtual resources are mapped and that the corresponding mapping algorithm performs better. With β =10, Kshort and NodeRep algorithm achieve the same revenue because they assign the 100% VN, which is shown in Fig.9, while Base algorithm gain the lowest revenue due to the lowest percentage of VN mapping. With β =30, NodeRep achieves the highest revenue, and Base gains the lowest revenue, and the revenue of Kshort is intermediate. For these three algorithms, although the percentage of VN mapping is lower with β =30 than with that of β =10, the revenue becomes higher with β =30. In other words, higher revenue does not mean higher percentage of VN mapping. From Fig.11, we can see that NodeRep performs best in the metric of average revenue with β =10 or 30.

VI. CONCLUSIONS

Network virtualization has been proposed as a promising way to overcome the current ossification of the Internet by allowing heterogeneous VNs to coexist on a shared substrate network. A major challenge in network virtualization is the VN mapping problem that deals with efficient assignment of virtual nodes and links onto the substrate network resources.

This paper introduces the several mapping algorithms firstly. Then, by relaxing the restraint on the node mapping, we propose a new mapping algorithm allowing sort preprocessing before VN to be mapped to a substrate node, if CPU capacity is sufficient. This scheme not only saves much time in selecting nodes, but also effectively reduces the substrate link cost to allow more VNs to be assigned. Our simulations show that new algorithm performs much better in the metrics of mapping percentage, runtime and VN revenue.

ACKNOWLEDGMENT

We would like to thank Dr. Li Wen as well as the reviewers for useful discussions and comments. This work was supported in part by the National Science Foundation of China (Grant No. 61063039, 11061004, 60873118 &60973147) and Natural Science Foundation of Guangxi Province in China (Grant No. 2010GXNSFB013052 & 2011GXNSFA018154) and Scientific Research Fund of Guangxi University for nationalities; the Program of Science and Technique

Foundation of Guangxi Province (GXST11107006-30), the Doctoral Fund of Ministry of Education of China under Grant No. 20090009110006, the Science and Technology Foundation of Guangxi under Grant No. 10169-1, and Guangxi Scientific Research Project No.201012MS274.

REFERENCES

[1] T. Anderson, L. Peterson, S. Shenker, J. Turner. "Overcoming the Internet impasse through virtualization," *In Proceedings of ACM Workshop on Hot Topics in Networks*, 2005, pp.34-41.

[2] N. Chowdhury, R. Boutaba. "A Survey of Network Virtualization," *Computer Networks*, vol.54(5), pp.862-876, 2010.

[3] J.Carapinpa, J. Jimenez. Network Virtualization – A View from the Bottom. *In Proceedings of 1st ACM workshop on Virtualized infrastructure systems and architectures*, pp.73-80, 2009. [doi: 10.1016/j.comnet. 2009.10.017]

[4] R.Bless, C. Werle. "Network Virtualization from a Signaling Perspective," *In Proceedings of the International Workshop on the Network of the Future*, pp.1-6, 2009.

[5] E. Keller, R. Lee, J. Rexford. "Accountability in Hosted Virtual Networks," *In Proceedings of 1st ACM workshop on Virtualized infrastructure systems and architecture*, pp.29-36, 2009.

[6] G. Schaffrath, C. Werle, P. Papadimitriou, et al. "Network Virtualization Architecture: Proposal and Initial Prototype," Proc. 1st ACM workshop on Virtualized infrastructure systems and architectures, pp. 63-71, 2009.

[7] N. Feamster, L. Gao, J. Rexford. "How to Lease the Internet in Your Spare Time," *ACM SIGCOMM Computer Communication Review*, vol. 37(1), pp.61-64, 2007. [doi: 10.1145/1198255.1198265]

[8] J. Fan, M. Ammar. "Dynamic Topology Configuration in Service Overlay Network: A study of Reconfiguration Policies," *In Proceedings of IEEE INFOCOM*, pp.1-12, 2006.

[9] R. Ricci, C. Alfeld, J. Lepreau. "A Solver for the Network Testbed Mapping Problem," *ACM SIGCOMM Computer Communication Review*, vol 33(2), pp.65-81, 2003. [doi: 10.1145/956981.956988]

[10] L. Jing, T. Jonathan. "Efficient Mapping of Virtual Networks onto a Shared Substrate," Technical Report, WUCSE-2006-35, 2006.

[11] M. Yu, Y. Yi, J. Rexford, M. Chiang. "Rethinking Virtual Network Embedding: Substrate Support for Path Splitting and Migration," *ACM SIGCOMM Computer Communication Review*, vol.38 (2), pp.17-29, 2008. [doi: 10.1145/1355734.1355737]

[12] I. Houidi, W. Louati, D. Zeghlache. "A Distributed Virtual Network Mapping Algorithm," *In Proceedings of ICC*, pp.5634-5640, 2008.

[13] J. Lischka, H. Karl. "A Virtual Network Mapping Algorithm based on Subgraph Isomorphism Detection," *In Proceedings of 1st ACM workshop on Virtualized infrastructure systems and architectures*, pp. 81-88, 2009.

[14] N. Chowdhury, M. Rahman, R. Boutaba. "Virtual Network Embedding with Coordinated Node and Link Mapping," *In Proceedings of IEEE INFOCOM*, pp.783-791, 2009.

[15] Y. Zhu, M. Ammar. "Algorithms for Assigning Substrate Network Resources to Virtual Network Components," I*n Proceedings of IEEE INFOCOM*, pp.1-12, 2006.

[16] T.H. Cormen, C.E. Leiseron, R.L. Rivest, C.Stein. *Introduction to Algorithms. Second Edition*. Cambridge MIT press, 2001.

[17] E. Zegura, K. Calvert, S. Bhattacharjee. "How to Model an Internetwork," *In Proceedings of IEEE INFOCOM* , pp.594-602, 1996.

[18] A. Haider, R. Potter, A. Nakao. "Challenges in Resource Allocation in Network Virtualization," *In Proceedings of the 20th ITC Specialist Seminar on Network Virtualization*, 2009.

[19] C. C. Marquezan, J. C. Nobre, L. Z. Granville, et al.. "Distributed Reallocation Scheme for Virtual Network Resources," *In Proceedings of ICC*, pp1-5, 2009.

[20] I. Houidi, W. Louati, D. Zeghlache, S. Baucke. "Virtual Resource Description and Clustering for Virtual Network Discovery," *In Proceedings of ICC*, 2009.

[21] Li Wen, Wu Chunming, et al. "Virtual Network Mapping Algorithm With Node Repeatable Embedding and Link Splitting," *Telecommunications Science*, vol. 10, pp.114-120, 2010.

Yong Huang received the B.S. and M.S. degrees from Nanjing University of Science and Technology, Nanjing, China, in 2001 and 2004, respectively. Then, he received the ph.D degrees from Zhejiang University, Hangzhou, China, in 2009.

He is currently an Associate Professor of computer science, Guangxi University for nationalities, Nanning, China. His research activity mainly focuses on information security, formal methods, and network system.

Jinzhao Wu received the B.S. and M.S. degrees from Lanzhou University, Lanzhou, China, in 1988 and 1991, respectively. Then, he received the ph.D degrees from Chinese academy of sciences, Beijing, China, in 1994.

He is currently a Professor of computer science, Guangxi University for nationalities, Nanning, China. His research activity mainly focuses on information security, formal methods, and hybrid system.