Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2014, Article ID 271493, 10 pages
http://dx.doi.org/10.1155/2014/271493

Research Article

Topological Embedding Feature Based Resource Allocation in Network Virtualization

Hongyan Cui,[1,2] Shaohua Tang,[1,2] Fangfang Sun,[1,2] Yue Xu,[1,2] and Xiaoli Yang[1,2]

[1] *Key Laboratory of Network System Architecture and Convergence, Beijing University of Posts and Telecommunications, Beijing 100876, China*
[2] *State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing 100876, China*

Correspondence should be addressed to Hongyan Cui; yan555cui@163.com

Received 22 July 2014; Accepted 4 August 2014; Published 31 August 2014

Academic Editor: Haipeng Peng

Virtualization provides a powerful way to run multiple virtual networks on a shared substrate network, which needs accurate and efficient mathematical models. Virtual network embedding is a challenge in network virtualization. In this paper, considering the degree of convergence when mapping a virtual network onto substrate network, we propose a new embedding algorithm based on topology mapping convergence-degree. Convergence-degree means the adjacent degree of virtual network's nodes when they are mapped onto a substrate network. The contributions of our method are as below. Firstly, we map virtual nodes onto the substrate nodes with the maximum convergence-degree. The simulation results show that our proposed algorithm largely enhances the network utilization efficiency and decreases the complexity of the embedding problem. Secondly, we define the load balance rate to reflect the load balance of substrate links. The simulation results show our proposed algorithm achieves better load balance. Finally, based on the feature of star topology, we further improve our embedding algorithm and make it suitable for application in the star topology. The test result shows it gets better performance than previous works.

1. Introduction

The concept of network virtualization is put forward for the first time by Anderson et al. [1]. Researches in network virtualization involve many aspects [2–5]. Because of topological diversity of virtual networks, node and link resource constraints, online request, and access control, the embedding problem becomes the core problem of the network virtualization. In this paper, we mainly study the virtual network embedding problem.

In the environment of network virtualization, infrastructure providers (InPs) and service providers (SPs) play important roles [6, 7], which correspond to the substrate network (SN) and virtual network requests (VNs), respectively. Infrastructure providers focus on physical network construction and maintenance, while service providers concentrate on receiving users' virtual network requests and providing services. The main goal of virtual network embedding is how to assign virtual network requests to a shared substrate network optimally with node and link resource constraints being satisfied.

Previous work does not consider the adjacent degree when the virtual nodes connecting directly to each other are mapped onto the substrate network. The result is that the virtual nodes connecting directly to each other may be mapped on the substrate network far away from (multiple hops) each other; thus, one virtual link occupies many substrate links, and the utilization efficiency of substrate resource is extremely low. In this paper, considering the degree of convergence when mapping virtual networks to the substrate network, we propose a new algorithm based on maximum topology mapping convergence-degree. Convergence-degree means the adjacent degree of virtual network's nodes when they are mapped onto the substrate network. The maximum convergence-degree not only ensures virtual nodes are mapped onto the substrate nodes with abundant resources,

but also ensures the virtual nodes connecting directly to each other are mapped onto the substrate nodes nearby, so the topology of a virtual network gathers together when the virtual network is mapped onto the substrate network. Therefore, the cost and complexity of link embedding reduced and the efficiency increases significantly; especially, the load balance of substrate links is improved.

The paper's main contributions are as follows.

(1) Defining topology mapping convergence-degree, we realize a new algorithm. The algorithm largely decreases the complexity of embedding problem and improves the network utilization efficiency.

(2) The proposed algorithm defines the balance rate of link load to reflect the load balance of substrate links and improves it largely. To the best of our knowledge, there are few articles researching it.

(3) Based on the feature of star topology, we further propose a star topology embedding algorithm.

The remainder of this paper is organized as follows. In Section 2, we present the related work. Section 3 introduces the model and objective descriptions. Section 4 presents the proposed algorithm. In Section 5, we describe the performance simulation and analysis. Section 6 discusses star topology embedding algorithm. Section 7 gives conclusion.

2. Related Work

Because the virtual network embedding is an NP-hard problem [8], researchers have proposed many virtual network embedding algorithms through limiting problem space, which are mostly heuristic algorithms, such as ant swarm algorithm [9]. According to different control modes of virtual network embedding, virtual network embedding algorithms can be divided into centralized embedding algorithms and distributed embedding algorithms. Centralized embedding algorithms allocate resources by the central decision-making body, such as [10–14]. Distributed embedding algorithms are usually performed by the substrate nodes coordinately, such as [15]. According to the different embedding sequences of virtual nodes and virtual links, algorithms can be divided into first-order embedding algorithms and second-order embedding algorithms. In first-order embedding algorithms, node embedding and link embedding are completed at the same stage, such as [12–14], while node embedding and link embedding in second-order embedding algorithms are completed at the different stage, such as [10, 11, 15–18].

Yu et al. [11] present a second-order embedding algorithm supporting path splitting and migration. The algorithm can be divided into node embedding and link embedding stage. Firstly, authors use greedy way to embed all virtual nodes. Then, authors embed virtual links using K-shortest paths algorithm. Further, authors introduce path splitting and migration features and employ multicommodity flow algorithm in link embedding stage to improve algorithm's performance. Chowdhury et al. [10] propose an embedding algorithm with coordinated node and link mapping. The authors introduce the concept of metanode and transform the node and link resource constraints problem into mixed integer programming problem. Each virtual node belongs to one metanode. Each metanode contains a subset of physical nodes. Then, virtual link embedding problem with bandwidth constraints can be regarded as commodity flow problem between metanodes. At last, the authors propose two kinds of algorithms: D-ViNE and R-ViNE, while a generalized window-based embedding algorithm (WiNE) is further discussed. Inspired by Markov random walk model, Cheng et al. [13] propose a measurement of node resource.

NodeRank. The node's NodeRank value not only reflects the resource of the node itself but also reflects other nodes' resources in substrate network topology. Based on the idea, the authors put forward two kinds of virtual network embedding algorithms: RW-MaxMatch and RW-BFS.

Although there have been a large number of previous researches, the problem of virtual network embedding still has a lot of research space; especially there are few references in the load balancing of substrate links and algorithm's efficiency. So in this paper, we take into account the load balancing of substrate links and algorithm's efficiency while considering the network utilization efficiency.

3. Embedding Model and Objective Descriptions

In this part, we will give the model and objective descriptions of virtual network embedding.

3.1. Substrate Network. The substrate network which InPs provide will be expressed as an undirected graph $G^s = (N^s, L^s, A_N^s, A_L^s)$, where N^s and L^s are sets of substrate node and link, respectively, and A_N^s and A_L^s are attribute sets of substrate node and link, respectively. The typical attributes of the node and link are the node's CPU capacity and the link's bandwidth.

3.2. Virtual Network Requests. The virtual network users, such as service providers and researchers, may need different virtual network requests. Similarly, each virtual network request will also be expressed as an undirected graph $G^v = (N^v, L^v, A_N^v, A_L^v)$, where N^v and L^v are sets of request node and link, respectively, and A_N^v and A_L^v are attribute sets of request node and link, respectively. Here, we also only consider the node's CPU requirement and the link's bandwidth requirement, which is estimated as long-range dependent traffic [19].

3.3. Embedding Model. The process of virtual network embedding M can be defined as finding a subgraph in undirected graph G^s for G^v satisfying the nodes' and links' attributes constraints (A_N^v, A_L^v). This process is expressed as follows:

$$M : G^v\left(N^v, L^v, A_N^v, A_L^v\right) \longrightarrow G^s\left(N^s, L^s, A_N^s, A_L^s\right). \quad (1)$$

The process can be further divided into node embedding process M_N and link embedding process M_L:

$$M_N : G^v\left(N^v, A_N^v\right) \longrightarrow G^s\left(N^s, A_N^s\right),$$
$$M_L : G^v\left(L^v, A_L^v\right) \longrightarrow G^s\left(L^s, A_L^s\right). \tag{2}$$

In the process of embedding, the substrate resources have to meet request's node and link resources requirements, namely,

$$\sum_{i \in N^v}^{N^v(i) \xrightarrow{M_N} N^s(j)} A_N^v(i) \le A_N^s(j), \quad j \in N^s,$$
$$\sum_{i \in L^v}^{L^v(i) \xrightarrow{M_L} L^s(j)} A_L^v(i) \le A_L^s(j), \quad j \in L^s, \tag{3}$$

where $N^v(i)$ and $N^s(j)$ represent the ith node of the virtual network request and the jth node of substrate network, respectively, $L^v(i)$ and $L^s(j)$ represent the ith link of the virtual network request and the jth link of substrate network, respectively, $A_N^v(i)$ and $A_N^s(j)$ represent the required resource of the ith virtual node and the available resource of the jth substrate node, respectively, and $A_L^v(i)$ and $A_L^s(j)$ represent the required resource of the ith virtual link and the available resource of the jth substrate link, respectively.

In order to make our work more practical, we adopt the time window model in Figure 1. We assume that virtual network requests arrive in a Poisson process and each request's lifetime obeys the exponential distribution. In one time window, the requests will be mapped according to their revenues. If one virtual network is embedded unsuccessfully, the request will be postponed into the waiting queue temporarily for subsequent embedding, while the virtual networks which lifetimes come to an end will release their resources in the time window.

3.4. Objective Descriptions

3.4.1. R/C Ratio. The revenue is the reward when virtual networks are being embedded successfully. For a virtual network which has been embedded successfully, we define the revenue $R(G^v)$ as follows:

$$R\left(G^v\right) = \sum_{l^v \in L^v} \mathrm{BW}\left(l^v\right) + \alpha \sum_{n^v \in N^v} \mathrm{CPU}\left(n^v\right), \tag{4}$$

where BW is the required bandwidth of virtual link, the CPU is the required CPU of virtual node n^v, and α is a weight coefficient balancing the influence of CPU and BW to revenue.

Cost is the consumption of substrate network resources for finishing virtual network embedding. We define the cost $C(G^v)$ as follows:

$$C\left(G^v\right) = \sum_{l^v \in L^v} \sum_{l^s \in L^s} \mathrm{BW}\left(F_{l^s}^{l^v}, l^v\right) + \beta \sum_{n^v \in N^v} \mathrm{CPU}\left(n^v\right), \tag{5}$$

where $F_{l^s}^{l^v} \in \{0, 1\}$ and $F_{l^s}^{l^v} = 1$ if substrate link l^s allocates bandwidth to virtual link l^v; otherwise $F_{l^s}^{l^v} = 0$. $\mathrm{BW}(F_{l^s}^{l^v}, l^v)$ is the bandwidth which the substrate link l^s allocates to the virtual link l^v. Similarly, β is a weight coefficient balancing the influence of CPU and BW to cost.

When we have defined $R(G^v)$ and $C(G^v)$, revenue/cost ratio (R/C) can be defined as follows:

$$\frac{R}{C} = \frac{R\left(G^v\right)}{C\left(G^v\right)} = \frac{\sum_{l^v \in L^v} \mathrm{BW}\left(l^v\right) + \alpha \sum_{n^v \in N^v} \mathrm{CPU}\left(n^v\right)}{\sum_{l^v \in L^v} \sum_{l^s \in L^s} \mathrm{BW}\left(F_{l^s}^{l^v}, l^v\right) + \beta \sum_{n^v \in N^v} \mathrm{CPU}\left(F_{n^s}^{n^v}\right)}. \tag{6}$$

R/C is directly related to net profit of InPs and reflects the resource utilization of substrate network. In this paper, we set $\alpha = \beta = 1$ but do not break generality.

3.4.2. The Balance Rate of Link Load. The usage of substrate links may overuse partly, while other parts are idle. Former researches rarely evaluate whether the usage of substrate links is balanced or not. In this paper, we define the standard deviation of link load as load balance rate to reflect the load balance of substrate links. The balance rate of link load has important practical significance for improving the reliability of the substrate network.

First, we define link load $L_l(t, l^s)$ as the stress of substrate link l^s in time t:

$$L_l\left(t, l^s\right) = \frac{\sum_{\forall l^v \uparrow l^s} \mathrm{BW}\left(l^v\right)}{\mathrm{BW}\left(l^s\right)}, \tag{7}$$

where $\sum_{\forall l^v \uparrow l^s} \mathrm{BW}(l^v)$ means the occupied bandwidth by the virtual links which are mapped onto the substrate link l^s.

Then, we define the standard deviation of link load as balance rate of link load *LinkLoadRate*:

$$LinkLoadRate = \sqrt{\frac{1}{|L^s|} \sum_{l^s \in L^s} \left[L_l\left(t, l^s\right) - L_a\left(t\right)\right]^2}, \tag{8}$$

where $|L^s|$ is the number of links in the set of L^s. $L_a(t) = \sum_{l^s \in L^s} L_l(t, l^s)/|L^s|$ is the average stress of links L^s in time t.

3.4.3. RunTime. The average running time consumed by a time window can reflect the efficiency of an algorithm. We define *RunTime* as follows:

$$RunTime = \frac{\text{Total run time}}{\text{Number of time windows}}. \tag{9}$$

4. The Maximum Convergence-Degree Algorithm

In this section, we will give the motivation of our proposed algorithm. Then we present the algorithm in detail. Finally, the complexity of the algorithm is discussed briefly.

4.1. Motivation. In previous research work, such as [10, 11], each virtual node is mapped in isolation in node embedding

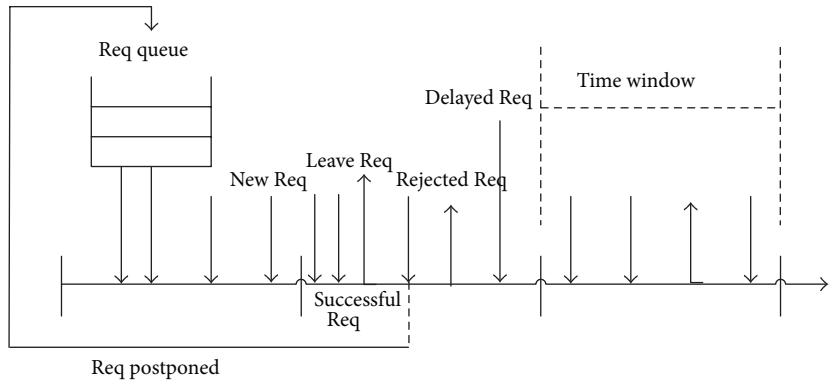


Figure 1: Model of time window.

stage. Although paper [13, 16] considers the connection relationship of substrate nodes, paper [16] only considers the substrate nodes which directly connect to the occupied substrate nodes, while the treatment for other substrate nodes is the same as paper [11]. Paper [13] defines the transfer probability matrix for all the virtual and substrate nodes, so the algorithm is fairly complex.

In order to overcome the above problems, we propose a new algorithm based on the maximum topology mapping convergence-degree. The proposed algorithm considers the topologies for both virtual network and substrate network. The maximum convergence-degree ensures the virtual nodes connecting directly to each other in a virtual network are embedded onto the substrate nodes nearby. The following example presents our algorithm's superiority.

As shown in Figure 2, (a) is the substrate network. Request1 and request2 represent two virtual network requests. The embedding result of the algorithm in paper [11] is (b). (c) is the embedding result by the algorithm proposed in paper [16]. The embedding result of our proposed algorithm is (d). For our proposed algorithm based on maximum topology mapping convergence-degree, when embedding request1, we choose the substrate node (5) to map the virtual node (c) after virtual nodes (a) and (b) being mapped onto the substrate nodes (1) and (6), because node (5) not only has the abundant resource (node's CPU capacity and link's bandwidth), but also the sum of the shortest paths between node (5) and nodes (1) ans (6) is minimum (the sum of shortest paths is 2). The reason we consider the sum of the shortest paths between node (5) and nodes (1) and (6) is that node (c) connects directly to nodes (a) and (b) in the virtual network (if node (c) only connects directly to node (a), we will only consider the shortest path between the candidate substrate node and node (1)). This mapping way ensures that the virtual nodes connecting directly to each other in a virtual network are mapped onto the substrate nodes nearby, so the topology of a virtual network gathers together when the virtual network is mapped onto the substrate network. From the results (a) and (b), we can see the substrate resource is used inefficiently since one virtual link occupies many substrate links, while some substrate nodes and links are overloaded. From the intuitional view, we can see the embedding result of our algorithm is more reasonable. We will prove the superiority of our proposed algorithm in Section 5.

4.2. Details of Maximum-Convergence-Degree Algorithm. In this part, we will introduce our algorithm in detail. Our proposed algorithm belongs to the second-order embedding algorithm, which can be divided into node embedding stage and link embedding stage. We employ the time window model described in Figure 1.

In node embedding stage, the virtual networks will be sorted according to their revenues in a time window. We select a virtual network with the maximum revenue to map. Then, we sort the nodes of the virtual network in descending order according to their required resources VR. VR is defined as follows:

$$\mathrm{VR}\left(n^v\right) = \mathrm{CPU}\left(n^v\right) \sum_{l^v \in \mathrm{Neib}(n^v)} \mathrm{BW}\left(l^v\right), \tag{10}$$

where $\mathrm{Neib}(n^v)$ is the link set connecting directly to the virtual node n^v. Then we select a virtual node with the largest VR in this virtual network and embed it onto the substrate node. Considering the degree of convergence when embedding virtual networks onto the substrate network, the substrate node n^s we choose not only meets virtual node's CPU requirement, but also has the maximum convergence-degree CSR. CSR is defined as follows:

$$\mathrm{CSR}\left(n^s\right) = \frac{\mathrm{CPU}\left(n^s\right) \sum_{l^s \in \mathrm{Neib}(n^s)} \mathrm{BW}\left(l^s\right)}{N_{\mathrm{path}} + \varepsilon}, \tag{11}$$

where $\mathrm{Neib}(n^s)$ is the link set connecting directly to the substrate node n^s. $\mathrm{CPU}(n^s)$ is the residual resource of node n^s. $\mathrm{BW}(l^s)$ is the residual bandwidth of link l^s. N_{path} is the sum of the shortest paths between the candidate substrate node n^s and substrate nodes n^i $(i = 1, 2, \ldots, k,\ k \neq s)$, where n^i $(i = 1, 2, \ldots, k,\ k \neq s)$ are substrate nodes which have been assigned to the virtual network's nodes connecting directly to the node n^v. ε is an infinitesimal number in order to avoid the condition that N_{path} is equal to 0. $\mathrm{CPU}(n^s) \sum_{l^s \in \mathrm{Neib}(n^s)} \mathrm{BW}(l^s)$ reflects the abundant degree of the node resource including

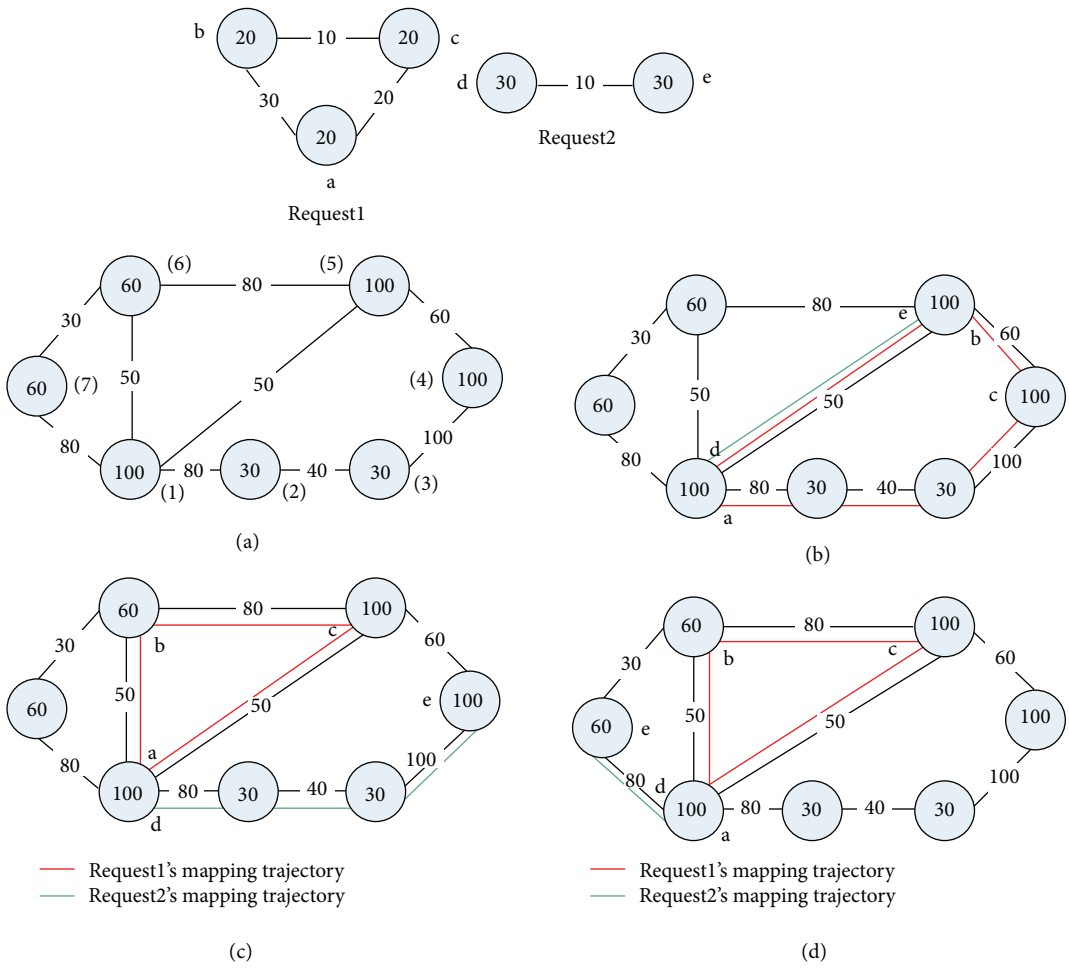


FIGURE 2: (a) Substrate network. (b) Embedding result of baseline algorithm proposed in paper [9, equation (11)]. (c) Embedding result of proximity algorithm proposed in paper [11, equation (16)]. (d) Embedding result of our proposed algorithm.

CPU and link resources. N_{path} reflects the adjacent degree of virtual nodes connecting directly to each other when they are mapped onto the substrate network; thus, CSR reflects the degree of convergence when mapping virtual nodes onto the substrate network. The bigger the value of CSR, the higher the degree of convergence.

The procedure of node embedding stage is executed in Algorithm 1. When all virtual nodes in this virtual network are embedded successfully according to Algorithm 1, we conduct link embedding. We use K-shortest paths algorithm when substrate links do not support path splitting, while we employ multicommodity flow algorithm when substrate links support path splitting. The link embedding stage is executed in Algorithm 2.

4.3. Discussion of Complexity. Compared with algorithms proposed in paper [11, 13, 16], our algorithm defines the topology mapping convergence-degree and just adds calculation factor $N_{\text{path}} + \varepsilon$ in node embedding stage. Therefore, the increased complexity in node embedding stage can be ignored.

Because our algorithm considers the degree of convergence when mapping virtual network onto the substrate network, this mapping way ensures that the topology of a virtual network gathers together in substrate network when it is embedded successfully. So the complexity of link embedding stage is greatly reduced, while the running time of our proposed algorithm also reduced significantly. In Section 5, the simulation result will prove our algorithm's superiority.

5. Performance Evaluation

In this section, we will present simulation settings and compare our algorithm (maximum-convergence-degree algorithm) with baseline algorithm proposed in paper [11], proximity algorithm proposed in paper [16], and RW-MaxMatch algorithm proposed in paper [13] in the performance of revenue/cost ratio (R/C), balance rate of link load *LinkLoadRate*, and algorithm's running time *RunTime*. Finally we present the simulation results and analysis.

(1) Sort the virtual networks in descending order according to their revenues $R(G^v)$ in a time window.
(2) **for** each virtual network with maximum revenue **do**
sort the virtual nodes according to their required resources VR in descending order
(3) **for** each node with the largest VR **do**
assign the node onto the substrate node which not only meets its' CPU requirement, but also has the maximum convergence-degree CSR
(4) **end for**
(5) **if** all nodes are embedded successfully **then**
return NODE_MAPPING_SUCCESS
(6) **else**
postpone the virtual network into the waiting Queue
return NODE_MAPPING_FAILED
(7) **end if**
(8) **end for**

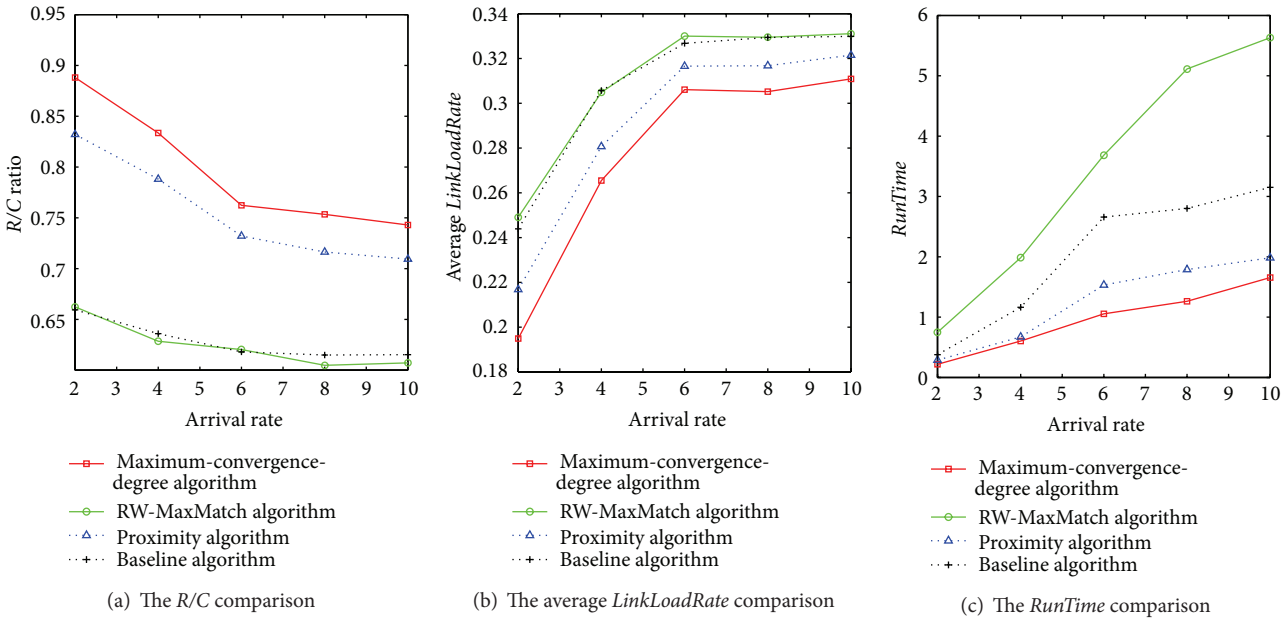
Algorithm 1: Maximum-convergence-degree algorithm node embedding stage.

(1) sort the virtual networks mapped successfully in the node stage according to their revenues $R(G^v)$ in descending order
(2) **for** each virtual network with maximum revenue **do**
(3) **for** each virtual link of the request **do**
(4) **if** substrate network does not support path splitting **then**
map virtual link using the k-shortest paths algorithm
(5) **else**
map virtual link using the multi-commodity flow algorithm
(6) **end if**
(7) **end for**
(8) **if** all links are embedded successfully **then**
return LINK_MAPPING_SUCCESS
(9) **else**
postpone the virtual network into the waiting queue
return LINK_MAPPING_FAILED
(10) **end if**
(11) **end for**

Algorithm 2: Maximum-convergence-degree algorithm link embedding stage.

5.1. Simulation Environment. We generate substrate network topology adopting the GT-IMT tool [20]. The substrate network is set to have 100 nodes and about 570 links. The computing resources of nodes' CPU and the bandwidth of links are configured to real numbers uniformly distributed between 50 and 100.

We set 100 time units as one time window. The virtual networks arrive in a Poisson process with an average arrival rate λ per time window. Each virtual network's lifetime obeys exponential distribution with an average of 10 time windows. For each virtual network, the number of virtual nodes follows a uniform distribution between 2 and 10. The connection probability among virtual nodes is 0.5. The required resources of virtual link's bandwidth BW and virtual node's CPU are uniform distribution between 0 and Max ($10 \leq \text{Max} \leq 100$); namely, $E(\text{BW}) = E(\text{CPU})$ is increased from 5 to 50, where $E(\,)$ is the mean. The DELAY which each virtual network waits for subsequent mapping is set to 2 time windows. For reaching a



(a) The *R/C* comparison (b) The average *LinkLoadRate* comparison (c) The *RunTime* comparison

FIGURE 3: Performance comparison under increasing arrival rate λ. $E(\text{BW}) = E(\text{CPU}) = 25$. DELAY = 2. Splitting ratio is 0.

steady state, we simulate 500 time windows, corresponding to about 2500 virtual networks.

5.2. Simulation Results and Analysis

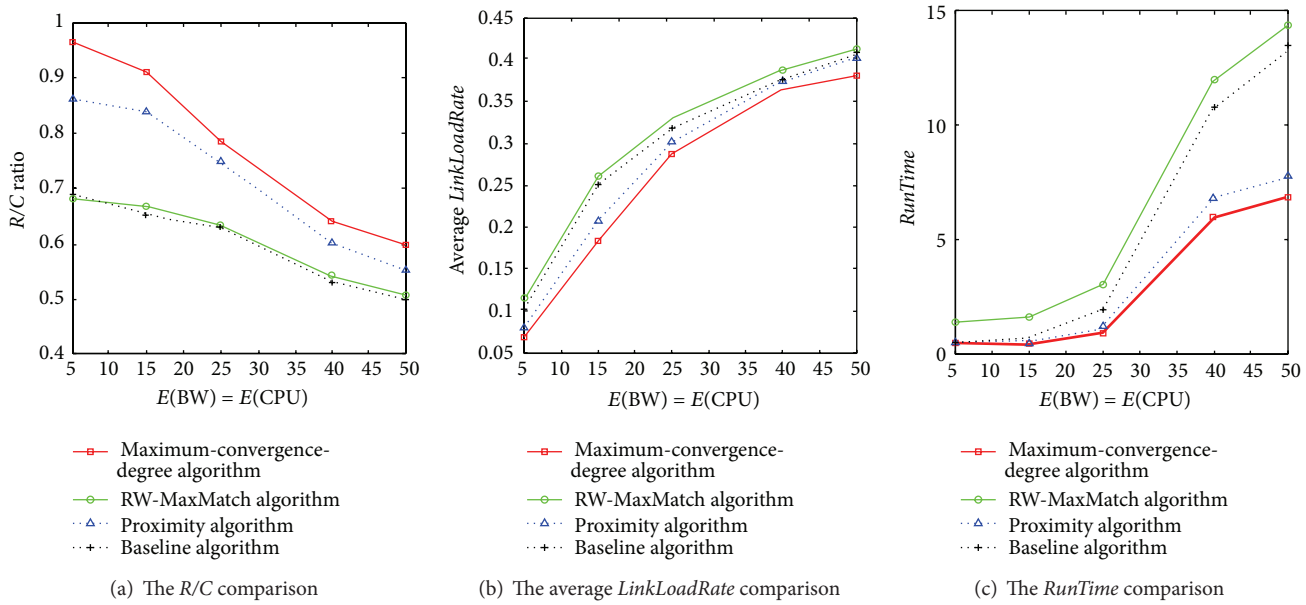
5.2.1. Maximum-Convergence-Degree Algorithm Produces Higher R/C Ratio. From Figures 3(a) and 4(a), we can see our proposed algorithm produces higher *R/C* under different VNs arrival rate and VNs required resources. Because we consider the topology mapping convergence-degree in the node embedding stage, it ensures the topology of a virtual network gathers together when the virtual network is mapped onto the substrate network. Each virtual link is mapped onto the substrate links with fewer hops; thus, the consumption of substrate bandwidth resources reduces greatly in link embedding stage, and *R/C* is improved greatly.

5.2.2. Maximum-Convergence-Degree Algorithm Makes Use of the Substrate Links More Balanced. From Figures 3(b) and 4(b), we can see that the load balance of substrate links in our proposed algorithm is always better than baseline algorithm, RW-MaxMatch algorithm, and proximity algorithm under different VNs arrival rate and VNs required resources. This indicates that our proposed algorithm uses the substrate more evenly and reasonably, and it will not show that parts of substrate links are idle, while other parts of substrate links are overused. The reason is when virtual nodes in a virtual network connecting directly to each other are mapped onto the substrate nodes nearby, the probabilities which one virtual link is mapped onto multiple substrate links and one substrate link is occupied by several virtual links will decrease, so it will not show the loads of some substrate links are lighter, while some parts are heavier. The load balance of substrate links has important practical significance for improving the reliability of the substrate network.

5.2.3. Maximum-Convergence-Degree Algorithm Reduces the Running Time Greatly. From Figures 3(c) and 4(c), we can see our proposed algorithm reduces the running time greatly and improves the algorithm efficiency under different VNs arrival rate and VNs required resources. As stated above, because we consider the topology mapping convergence-degree in node embedding stage, it ensures the virtual nodes in a virtual network connecting directly to each other are mapped onto the substrate nodes nearby; thus, each virtual link is mapped onto the substrate links with fewer hops. On one hand, the running time of searching shortest paths for virtual links decreases, and on the other hand, with a higher success ratio in the link embedding stage, the running time is saved by avoiding backtracking. So the running time of the algorithm is reduced greatly, and the efficiency is improved greatly.

5.2.4. The Effect of Path Splitting Feature. Path splitting is a subsidiary feature of substrate network brought up by previous work [11]. It splits one virtual link into small pieces to assign more virtual networks to the substrate network. In order to demonstrate the superiority of our algorithm in different path splitting ratios, we also compare the algorithms performance with the splitting ratio increasing from 0 to 100%, while other parameters are set as follows: $E(\text{BW}) = E(\text{CPU}) = 25$. Arrival rate = 5. DELAY = 2.

As shown in Figure 5, our proposed algorithm is always better than baseline algorithm, proximity algorithm, and RW-MaxMatch algorithm in the performances of revenue/cost (*R/C*), balance rate of link load *LinkLoadRate*, and running time *RunTime*. Therefore, we can conclude that our proposed



(a) The R/C comparison (b) The average *LinkLoadRate* comparison (c) The *RunTime* comparison

FIGURE 4: Performance comparison under increasing $E(\text{BW}) = E(\text{CPU})$. Arrival rate = 5. DELAY = 2. Splitting ratio is 0.

algorithm is also suitable for the substrate network supporting any link splitting ratio.

6. The Star Topology Embedding Algorithm

There are many different kinds of network topologies, and common topological types are star, ring, tree, and mesh, and so forth. Different topologies are suitable for different types of applications. Among them, the star topology is the most commonly used, because it is easy to extend and convenient for management, such as Ethernet and CDN (content distribution network). We have proposed the embedding algorithm based on topology mapping convergence-degree in the above without considering topology feature. Of course, the proposed algorithm also applies to specific topology types. However, we believe there are more efficient embedding algorithms for specific topology types. Therefore, considering the importance of star topology, we further discuss star topology embedding algorithm on the basis of our previous work.

We know the feature of star topology type is that all nodes in the topology connect directly to a central node. Data transmission and exchange are controlled by the central node. The central node in the star topology is the most important node, so we think the central node should be mapped firstly in the process of mapping a star topology. Due to the fact that other nodes in star topology connect directly to the central node, if the central node is mapped onto a substrate node n^s with a large degree $d(n^s)$, the probability of the star topology being mapped successfully will increase, and the cost of link embedding stage will be reduced. Thus, the substrate node n^s we choose to map the central node not only has enough available resources, but also has the maximum $\text{DSR}(n^s)$. $\text{DSR}(n^s)$ is defined as follows:

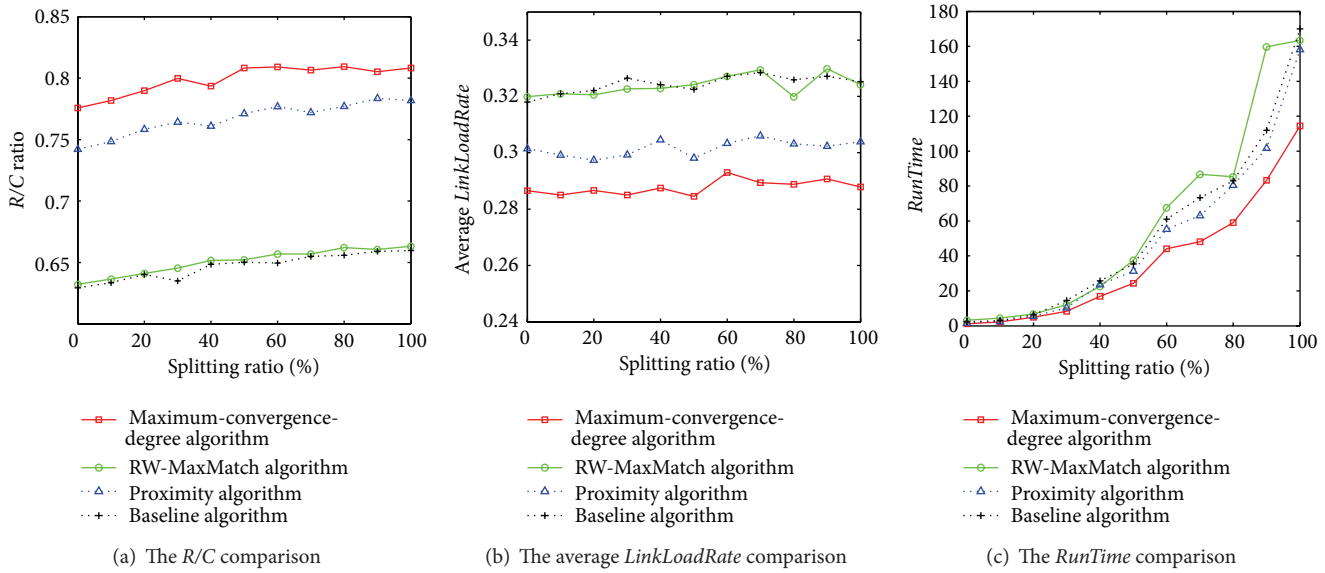
$$\text{DSR}\left(n^s\right) = \text{CPU}\left(n^s\right) d\left(n^s\right) \sum_{l^s \in \text{Neib}(n^s)} \text{BW}\left(l^s\right), \quad (12)$$

where $d(n^s)$ represents node's degree, namely, the number of links associated to the node n^s. After the central node is mapped successfully, the mapping methods of other virtual nodes and virtual links are the same as maximum-convergence-degree algorithm we have proposed.

In order to verify the effect of our improved star topology embedding algorithm, we compare the simulation results with previous works in the simulation environment of splitting ratio as 0, $E(\text{BW}) = E(\text{CPU}) = 25$, Arrival rate = 5, DELAY = 2, while other simulation parameters are the same as Section 5. The result is shown in Figure 6, where we can see the performances of star topology embedding algorithm are further improved. The result confirms that there are more efficient embedding algorithms for specific topology types, and we will research specific embedding algorithms according to the features of other different topology types in the future.

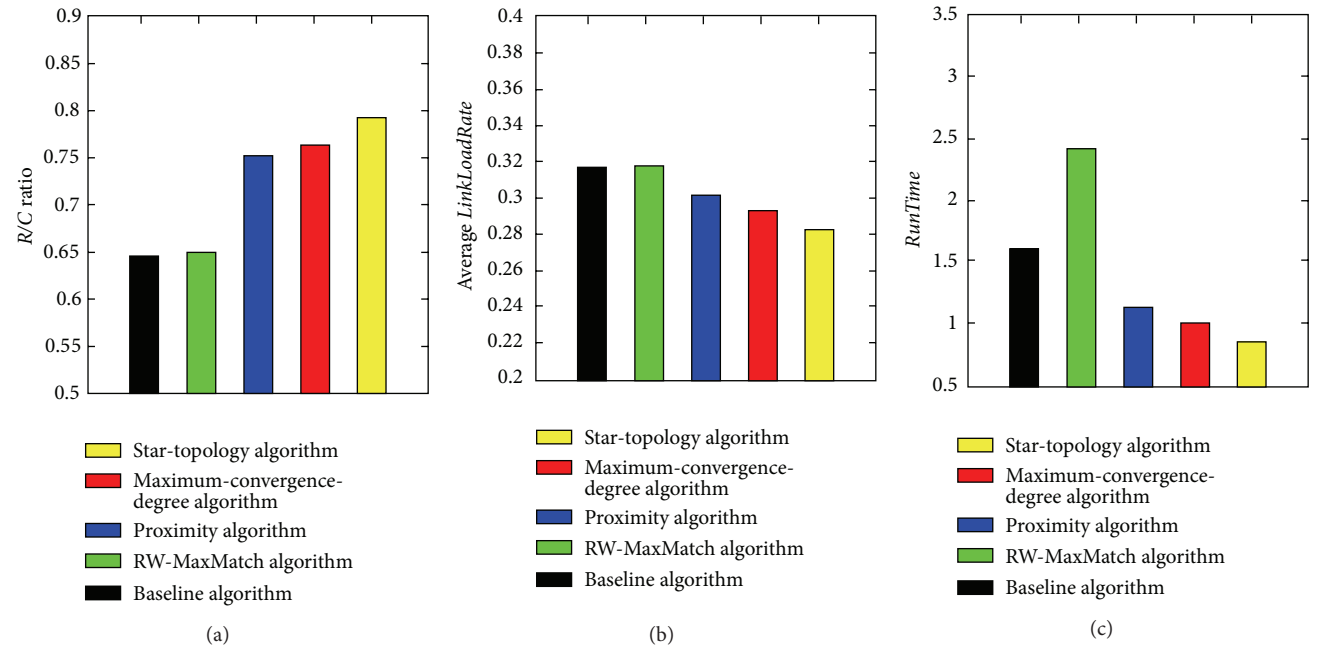
7. Conclusions

In this paper, we propose a new algorithm based on maximum topology mapping convergence-degree. The proposed algorithm ensures the virtual nodes connecting directly to each other in a virtual network are embedded onto the substrate nodes nearby; thus, the topology of a virtual network gathers together when the virtual network is mapped onto the substrate network, and the efficiency of link embedding



(a) The *R/C* comparison (b) The average *LinkLoadRate* comparison (c) The *RunTime* comparison

Figure 5: Performance comparison under different splitting ratios. $E(\text{BW}) = E(\text{CPU}) = 25$. Arrival rate = 5. DELAY = 2.



(a) (b) (c)

Figure 6: Performance comparison under $E(\text{BW}) = E(\text{CPU}) = 25$. Arrival rate = 5. DELAY = 2. Splitting ratio is 0.

increases significantly. The simulation results show, under a wide range of virtual networks and substrate network conditions, our proposed algorithm largely enhances the network utilization efficiency and decreases the complexity of the embedding problem; especially, the proposed algorithm improves the load balance of substrate links. Furthermore, based on the feature of star topology and its importance, we propose star topology embedding algorithm, and the performances of star topology embedding algorithm are further improved.

In our future research, we will further discuss how to realize the algorithm based on software defined network and how to optimize the method according to the actual restrictions such as networks node's position, reliability, and so forth.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (61201153), the National 973 Program of China under Grant (2012CB315805), the Prospective Research Project on Future Networks in the Jiangsu Future Networks Innovation Institute (BY2013095-2-16), the National Basic Research Program 973 of China (2012CB315801), and the Fundamental Research Funds for the Central Universities (2013RC0113).

References

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

[2] W. Yeow, C. Westphal, and U. Kozat, "Designing and embedding reliable virtual infrastructures," *ACM SIGCOMM Computer Communication Review*, vol. 41, no. 2, 2011.

[3] Y. Zhou, Y. Li, G. Sun, D. Jin, L. Su, and L. Zeng, "Game theory based bandwidth allocation scheme for network virtualization," in *Proceedings of the 53rd IEEE Global Communications Conference (GLOBECOM '10)*, pp. 1–5, December 2010.

[4] I. Houidi, W. Louati, D. Zeghlache, and S. Baucke, "Virtual resource description and clustering for virtual network discovery," in *Proceedings of the IEEE International Conference on Communications Workshops (ICC '09)*, pp. 1–6, Dresden, Germany, June 2009.

[5] A. Belbekkouche, M. M. Hasan, and A. Karmouch, "Resource discovery and allocation in network virtualization," *IEEE Communications Surveys and Tutorials*, vol. 14, no. 4, pp. 1114–1128, 2012.

[6] N. M. Mosharaf Kabir Chowdhury and R. Boutaba, "Network virtualization: state of the art and research challenges," *IEEE Communications Magazine*, vol. 47, no. 7, pp. 20–26, 2009.

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

[8] D. Andersen, "Theoretical approaches to node assignment," Working Paper, 2002.

[9] L. Li, J. Kurths, H. Peng, Y. Yang, and Q. Luo, "Exponentially asymptotic synchronization of uncertain complex time-delay dynamical networks," *The European Physical Journal B*, vol. 86, no. 4, article 125, 9 pages, 2013.

[10] M. Chowdhury, M. R. Rahman, and R. Boutaba, "ViNEYard: virtual network embedding algorithms with coordinated node and link mapping," *IEEE/ACM Transactions on Networking*, vol. 20, no. 1, pp. 206–219, 2012.

[11] M. Yu, Y. Yi, J. Rexford et al., "Rethinking virtual network embedding: substrate support for path splitting and migration," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 2, pp. 17–29, 2008.

[12] J. Lischka and H. Karl, "A virtual network mapping algorithm based on subgraph isomorphism detection," in *Proceedings of the 1st ACM Workshop on Virtualized Infrastructure Systems and Architectures (VISA '09)*, pp. 81–88, 2009.

[13] X. Cheng, S. Su, Z. Zhang et al., "Virtual network embedding through topology-aware node ranking," *ACM SIGCOMM Computer Communication Review*, vol. 41, no. 2, 2011.

[14] J. Shamsi and M. Brockmeyer, "QoSMap: QoS aware mapping of virtual networks for resiliency and efficiency," in *Proceedings of the IEEE GLOBECOM Workshop*, pp. 1–6, Washington, DC, USA, November 2007.

[15] I. Houidi, W. Louati, and D. Zeghlache, "A distributed virtual network mapping algorithm," in *Proceeding of the IEEE International Conference on Communications (ICC '08)*, pp. 5634–5640, Beijing, China, May 2008.

[16] J. Liu, T. Huang, J. Chen, and Y. Liu, "A new algorithm based on the proximity principle for the virtual network embedding problem," *Journal of Zhejiang University: Science C*, vol. 12, no. 11, pp. 910–918, 2011.

[17] H. Cui, W. Gao, Y. Liu et al., "A virtual network embedding alogrithm based on virtual topology connection feature," in *Proceedings of the IEEE 16th International Symposium on Wireless Personal Multimedia Communications (WPMC '13)*, pp. 1–5, 2013.

[18] X. Chen, Y. Luo, and J. Wang, "Virtual network embedding with border matching," in *Proceedings of the 4th International Conference on Communication Systems and Networks (COMSNETS '12)*, pp. 1–8, January 2012.

[19] M. Li, W. Chen, and L. Han, "Correlation matching method for the weak stationarity test of LRD traffic," *Telecommunication Systems*, vol. 43, no. 3-4, pp. 181–195, 2010.

[20] E. W. Zegura, K. L. Calvert, and S. Bhattacharjee, "How to model an internetwork," in *Proceedings of the 15th Annual Joint Conference of the IEEE Computer Societies (INFOCOM '96)*, vol. 2, pp. 594–602, San Francisco, Calif, USA, March 1996.