Journal of Zhejiang University-SCIENCE C (Computers & Electronics)
ISSN 1869-1951 (Print); ISSN 1869-196X (Online)
www.zju.edu.cn/jzus; www.springerlink.com
E-mail: jzus@zju.edu.cn

A new algorithm based on the proximity principle for the virtual network embedding problem*

Jiang LIU†, Tao HUANG†‡, Jian-ya CHEN, Yun-jie LIU

(*Key Laboratory of Universal Wireless Communications of Ministry of Education*, *Beijing University of Post and Telecommunications*, *Beijing 100876*, *China*)

†E-mail: joe_lj@126.com; htao@bupt.edu.cn

Received Jan. 1, 2011; Revision accepted May 17, 2011; Crosschecked Aug. 26, 2011

Abstract: The virtual network embedding/mapping problem is a core issue of network virtualization. It is concerned mainly with how to map virtual network requests to the substrate network efficiently. There are two steps in this problem: node mapping and link mapping. Current studies mainly focus on developing heuristic algorithms, since both steps are computationally intractable. In this paper, we propose a new algorithm based on the proximity principle, which considers the distance factor besides the capacity factor in the node mapping step. Thus, the two steps of the embedding problem can be better integrated and the substrate network resource can be used more efficiently. Simulation results show that the new algorithm greatly enhances the performance of the revenue/cost (*R*/*C*) ratio, acceptance ratio, and runtime of the embedding problem.

Key words: Virtual network embedding, Proximity, Revenue/cost (*R*/*C*) ratio, Acceptance ratio, Runtime
doi:10.1631/jzus.C1100003 **Document code:** A **CLC number:** TP393

1 Introduction

Network virtualization has been considered as a promising way of allowing multiple kinds of networks (including experimental ones) to run on a shared substrate network (Turner and Taylor, 2005; Feamster *et al*., 2007). It could even become a foundation of the future Internet (Anderson *et al*., 2005; Bavier *et al*., 2006).

The virtual network embedding problem is an essential part of network virtualization. The main object of the embedding problem is to assign the virtual network requests reasonably to the substrate network with node and link constraints being satisfied. For example, if a substrate link is 1 Gb/s, the sum bandwidth of virtual links which are embedded on it should not be over 1 Gb/s. These node and link constraints make the embedding problem nondeterministic polynomial-time hard (NP-hard) (Andersen, 2002). Thus, in practice, it is common for researchers to find heuristic-based algorithms to accomplish embedding in polynomial time, with a sacrifice in performance. Meanwhile, an embedding algorithm should also consider admission control and online processing for a practical application.

There have been several studies reported on this issue. Some restrict the search space by considering only link constraints, while assuming that the node mapping condition is known beforehand (Ricci *et al*., 2003; Lu and Turner, 2006; Zhu and Ammar, 2006). Some discuss the offline problem (Fan and Ammar, 2006; Lu and Turner, 2006). Some do not consider admission control (Fan and Ammar, 2006; Lu and Turner, 2006; Zhu and Ammar, 2006). Yu *et al*. (2008) considered node constraints, link constraints, admission control, and online processing together. However, the node mapping and link mapping algorithms

‡ Corresponding author

* Project supported by the National Basic Research Program (973) of China (Nos. 2007CB310701 and 2011CB302900) and the National Science and Technology Major Projects (No. 2010ZX03004-002-02)

are isolated, which results in room for improvement.

In this paper, we propose a new algorithm based on the proximity principle for the virtual network embedding problem. The proximity principle considers the distance factor besides the capacity factor in the node mapping step. Thus, the two steps of embedding can be better integrated and the substrate network resource can be used more efficiently. The idea comes from a simple thought that in a two-step procedure, the first step, or the fundamental step, is more important and should be optimized with consideration of the following step. Based on this idea, we add a proximity principle into the embedding algorithm to integrate node mapping and link mapping. We also build a simulation model and evaluate our new algorithm.

2 Background

In this section, the virtual network embedding problem will be introduced, including basic definitions and formulations. Then we describe the simulation model in detail.

2.1 Virtual network embedding problem

The virtual network embedding problem is concerned mainly with how to assign the virtual network requests reasonably to the substrate network with node and link constraints being satisfied. The problem is important since it is an essential mechanism in every virtualization network. Meanwhile, the problem is NP-hard in mathematics, so the study on it will give reference to other NP-hard problems. The intuitional description of the virtual network embedding problem is shown in Fig. 1.

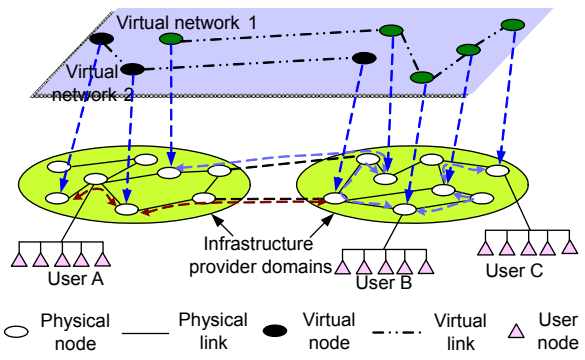


Fig. 1 Virtual network embedding

As shown in Fig. 1, two virtual networks are assigned to a shared substrate network, which is composed of two infrastructure provider domains, and provide their services to three groups of users. The system works as follows: The virtual network users, such as researchers or service providers, first apply for the virtual networks to the infrastructure providers. Then the embedding algorithm works to determine the embedding relationship between the virtual networks and substrate networks. Finally, the virtual network users can provide their experiments or services to the end users belonging to their virtual networks. Note that the two infrastructure provider domains indicate that there are multiple substrate network providers. This condition may cause an additional node constraint, such as the location constraint, for the virtual network embedding problem. It will be briefly discussed as a future work in the conclusion section.

2.2 Definitions and formulations

2.2.1 Substrate network

The substrate network will be modeled as an undirected graph $G^{\mathrm{S}}=(N^{\mathrm{S}},L^{\mathrm{S}},C_{\mathrm{N}}^{\mathrm{S}},C_{\mathrm{L}}^{\mathrm{S}})$, where N^{S} and L^{S} are sets of substrate nodes and links, respectively, and $C_{\mathrm{N}}^{\mathrm{S}}$ and $C_{\mathrm{L}}^{\mathrm{S}}$ are constraints of substrate nodes and links, respectively. $C_{\mathrm{N}}^{\mathrm{S}}$ and $C_{\mathrm{L}}^{\mathrm{S}}$ usually denote CPU capacity and link bandwidth, respectively.

2.2.2 Virtual request

Each virtual request will be modeled as an undirected graph $G^{\mathrm{V}}=(N^{\mathrm{V}},L^{\mathrm{V}},C_{\mathrm{N}}^{\mathrm{V}},C_{\mathrm{L}}^{\mathrm{V}})$, where N^{V} and L^{V} are sets of virtual nodes and links, respectively, and $C_{\mathrm{N}}^{\mathrm{V}}$ and $C_{\mathrm{L}}^{\mathrm{V}}$ are constraints of virtual nodes and links, respectively.

2.2.3 Embedding

The embedding can be defined as a mapping action M from a virtual request G^{V} to a subset of the substrate network G^{S}. The subset is the image of G^{V} in G^{S}, so it should satisfy the constraints $(C_{\mathrm{N}}^{\mathrm{S}},C_{\mathrm{L}}^{\mathrm{S}})$ from G^{S}. The mapping action can be expressed as follows:

$$M(G^{\mathrm{V}}):\ (G^{\mathrm{V}},C_{\mathrm{N}}^{\mathrm{V}},C_{\mathrm{L}}^{\mathrm{V}})\rightarrow(G^{\mathrm{S}},C_{\mathrm{N}}^{\mathrm{S}},C_{\mathrm{L}}^{\mathrm{S}}).$$

It can be further detailed as node mapping and link mapping:

$$M_{\mathrm{N}}(N^{\mathrm{V}}):\ (N^{\mathrm{V}},C_{\mathrm{N}}^{\mathrm{V}})\rightarrow(N^{\mathrm{S}},C_{\mathrm{N}}^{\mathrm{S}}),$$
$$M_{\mathrm{L}}(L^{\mathrm{V}}):\ (L^{\mathrm{V}},C_{\mathrm{L}}^{\mathrm{V}})\rightarrow(L^{\mathrm{S}},C_{\mathrm{L}}^{\mathrm{S}}).$$

From the expression above, it can be seen that the virtual network embedding problem is a two-step procedure, which is the foundation of our innovation.

2.2.4 Revenue and cost

Revenue is the profit of the embedding action. It is one of the main factors we use to judge the performance of an embedding algorithm. Revenue is defined according to the scale of the virtual request as follows:

$$\mathrm{Rev}(M(G^{\mathrm{V}}))=\alpha_{\mathrm{R}}\sum_{l^{\mathrm{V}}\in L^{\mathrm{V}}}\mathrm{BW}(l^{\mathrm{V}})+\beta_{\mathrm{R}}\sum_{n^{\mathrm{V}}\in N^{\mathrm{V}}}\mathrm{CPU}(n^{\mathrm{V}}),\quad(1)$$

where BW is the bandwidth of the request link, CPU is the capacity of the request node, and α_{R} and β_{R} are the weighting coefficients to balance the effect of BW and CPU. The weighting coefficients can also indicate the unit prices in revenue when the study comes to a practical application.

Cost is the price for finishing the embedding. It is defined according to the nodes' CPU and links' bandwidth used for the virtual request in the substrate network:

$$\mathrm{Cost}(M(G^{\mathrm{V}}))=\alpha_{\mathrm{C}}\sum_{l^{\mathrm{V}}\in L^{\mathrm{V}}}\mathrm{HOP}(l^{\mathrm{V}})\mathrm{BW}(l^{\mathrm{V}})+\beta_{\mathrm{C}}\sum_{n^{\mathrm{V}}\in N^{\mathrm{V}}}\mathrm{CPU}(n^{\mathrm{V}}),\quad(2)$$

where $\mathrm{HOP}(l^{\mathrm{V}})$ is the hop count of the virtual link l^{V} when it is assigned to a set of substrate links. α_{C} and β_{C} are the weighting coefficients to balance the effect of BW and CPU. They can also indicate the unit prices in the cost. Note that the intermediate CPU, which is used to forward data for a multiple-hop virtual link, is determined by the BW it carries and the hops it takes. Thus, it is contained in the BW part of the cost model and the effect could be shown up by adjusting the coefficient α_{C}. However, we will not discuss the intermediate CPU in detail since the current model is widely used and qualified to validate our new algorithm.

The *R*/*C* ratio is defined as follows:

$$R/C=\frac{\mathrm{Rev}(M(G^{\mathrm{V}}))}{\mathrm{Cost}(M(G^{\mathrm{V}}))}=\frac{\alpha_{\mathrm{R}}\sum_{l^{\mathrm{V}}\in L^{\mathrm{V}}}\mathrm{BW}(l^{\mathrm{V}})+\beta_{\mathrm{R}}\sum_{n^{\mathrm{V}}\in N^{\mathrm{V}}}\mathrm{CPU}(n^{\mathrm{V}})}{\alpha_{\mathrm{C}}\sum_{l^{\mathrm{V}}\in L^{\mathrm{V}}}\mathrm{HOP}(l^{\mathrm{V}})\mathrm{BW}(l^{\mathrm{V}})+\beta_{\mathrm{C}}\sum_{n^{\mathrm{V}}\in N^{\mathrm{V}}}\mathrm{CPU}(n^{\mathrm{V}})}.\quad(3)$$

The *R*/*C* ratio is an important factor to judge the performance of an embedding algorithm, since it indicates the efficiency of substrate network resource. In this paper, we assume $\alpha_{\mathrm{R}}=\beta_{\mathrm{R}}=\alpha_{\mathrm{C}}=\beta_{\mathrm{C}}=1$ to normalize the simulation results. Thus, the *R*/*C* ratio is restricted in [0, 1], and 1 indicates the optimum embedding.

2.2.5 Acceptance ratio

The acceptance ratio indicates how many requests are successfully accepted from all the requests. It is defined by the following formulation:

$$\text{Acceptance ratio}=\frac{\text{Number of accepted requests}}{\text{Number of all requests}}.\quad(4)$$

2.2.6 Runtime

Runtime indicates the average time we use for each embedding time window, which is introduced in detail in Section 2.3. Runtime is defined by the following formulation:

$$\text{Runtime}=\frac{\text{Total run time}}{\text{Number of time windows}}.\quad(5)$$

2.3 Model of embedding

To make our work more practical, we present an embedding model based on the architecture in previous work (Yu *et al*., 2008). The model not only considers the performance of embedding, but also takes online processing and admission control into account. Thus, our work could be easily migrated to a practical embedding system.

The core idea of the architecture is the time window. We assume that the virtual request arrives in a time window in two conditions: first, a new virtual request arrives randomly following a Poisson distribution; second, the request that failed in the last time

window will be postponed into the waiting queue and embedded in the current time window. Note that the postponed request could be rejected after failing for certain times set up beforehand. The virtual request leaves in two conditions: first, a request will survive for a random time and finish with releasing the resource it occupied; second, some dangerous or illegal requests could be closed and the resource could also be released. Thus, we execute the embedding procedure once in each time window by the following steps:

1. Release the resource of requests left in the last time window, including finished and closed requests.

2. Count arrived requests in the current time window, including new and postponed ones.

3. Sort the virtual requests by their revenues to embed the request with the largest revenue first. Then map the requests to the substrate network. If successful, update the state of residual resource in the substrate network; else, postpone the request into the waiting queue, or reject it after failing for certain times set up beforehand.

Fig. 2 shows the flow of the embedding model described above.

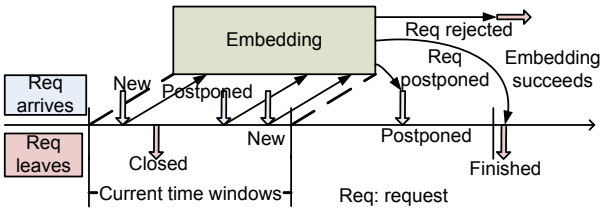


Fig. 2 Time window model of embedding

As shown in Fig. 2, two new requests and one postponed request arrive in the current time window. They are sent into the embedding algorithm, generating the following results: one is rejected, one is postponed, and one is embedded successfully and finished in two time windows. Note that a former mapped request may be found to be dangerous and closed in the current time window.

3 The proximity algorithm

In this section, we first explain the motivation of our work. Then we describe the proximity algorithm we proposed in detail. Finally, we briefly discuss the complexity of the new algorithm.

3.1 Motivation

We used the algorithm in a previous work (Yu *et al*., 2008) as a baseline algorithm. It finishes the virtual network embedding in two steps: mapping virtual nodes using the greedy algorithm and mapping virtual links using the *k*-shortest path algorithm. Both algorithms are mature and easy to implement. However, they are executed irrespectively with each other. Thus, an optimal node mapping decision may not be suitable for link mapping. For example, we find such unreasonable situations hard to avoid in the baseline algorithm: the nodes selected by the greedy algorithm are usually far away (multiple hops) from each other; thus, the nodes and links between them will be overloaded. Meanwhile, the substrate resource is used inefficiently since one virtual link occupies many substrate links.

To overcome these problems, we propose a new algorithm based on the proximity principle. The idea comes from the simple thought that in a two-step procedure, the first step, or the fundamental step, is more important and should be optimized with consideration of the following step. Thus, in the new algorithm, we increase the priority of the substrate nodes that are connected to the former selected ones. This procedure considers the distance factor as well as the capacity factor for a substrate node. Thus, the following link mapping will benefit from that, which results in a good integration of the two mapping steps. Fig. 3 explains the advantage of our new algorithm by an example.

As shown in Fig. 3, when mapping request 1, both algorithms will select A-B-C. When mapping request 2, the baseline algorithm will still select A-B-C since the substrate nodes A and C have the largest capacity. This embedding result will make the link A-B-C and node B heavy-loaded, and the substrate resource is used inefficiently since the single virtual link d-e occupies two substrate links. In contrast, the proximity algorithm will select substrate node D instead of C for the virtual node e. This is because, after request 1 and virtual node d have been assigned, the substrate node D has a better priority than C considering both capacity and distance factors. Thus, the proximity algorithm evens up the system load and uses the substrate resource more efficiently.

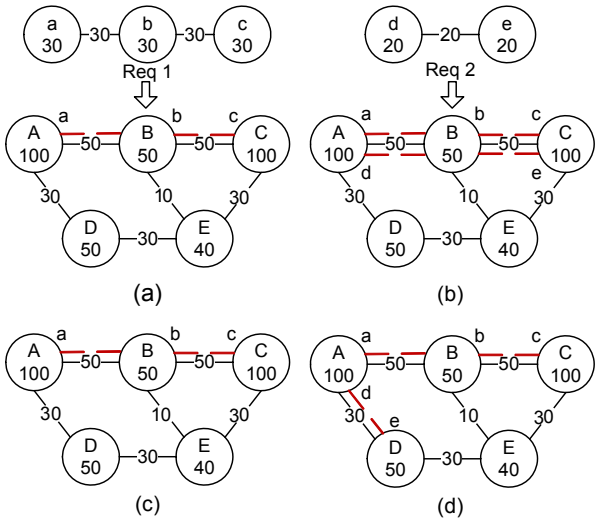


Fig. 3 Advantage of the proximity algorithm

(a) Mapping request 1 (Req 1) with the baseline algorithm; (b) Mapping request 2 (Req 2) with the baseline algorithm; (c) Mapping Req 1 with the proximity algorithm; (d) Mapping Req 2 with the proximity algorithm. a–e: request nodes; A–E: substrate nodes. Numbers within the nodes and links denote node CPU and link BW, respectively

3.2 Details of the proximity algorithm

In this subsection, we introduce the details of our new algorithm. The core objective of the new algorithm is to increase the utilization of the substrate network resource using the proximity principle. The simulation in Section 4 shows that the acceptance ratio and runtime become better at the same time as the byproducts. We use the time window model introduced in Section 2. The details of the proximity algorithm are as follows: After virtual requests are sorted by their revenues (Section 2.3), we first select one with the maximum revenue. Then we sort this request's nodes by their required resources, which are defined as

$$\mathrm{RR}(n^{\mathrm{V}}) = \mathrm{CPU}(n^{\mathrm{V}}) \sum_{l^{\mathrm{V}} \in L(n^{\mathrm{V}})} \mathrm{BW}(l^{\mathrm{V}}), \qquad (6)$$

where $L(n^{\mathrm{V}})$ denotes the set of adjacent virtual links of n^{V}. Then we select one with the maximum required resource, and assign it to the substrate node with both enough CPU and maximum weighted available resource, which is defined as follows:

$$\mathrm{WAR}(n^{\mathrm{S}}) = \mathrm{Corr}^{m}\mathrm{CPU}(n^{\mathrm{S}}) \sum_{l^{\mathrm{S}} \in L(n^{\mathrm{S}})} \mathrm{BW}(l^{\mathrm{S}}), \qquad (7)$$

where $L(n^{\mathrm{S}})$ denotes the set of adjacent substrate links of n^{S}. Corr is the correlation factor for the substrate node n^{S}, and the coefficient m is the number of the substrate nodes which is not only already assigned in the same request, but also directly connected to n^{S}. This procedure can make sure that if some substrate nodes are already selected, the nodes connected to them will have a better chance to be selected. Note that we use an exponential model for the weighted coefficient instead of a linear model since the exponential model is simpler and performs better.

After the substrate nodes are selected, we use the k-shortest path algorithm to solve the link mapping problem. It is an approximation approach since link mapping is also NP-hard (Kleinberg, 1996; Kolliopoulos and Stein, 1997).

The detailed steps of the proximity algorithm are shown as follows.

Algorithm 1 The proximity algorithm

```
1  sort the requests according to their revenues in descending
     order
2  for each request in order do
3    sort the virtual nodes according to their required
       resources in descending order
4    for each node of the request in order do
5      assign it to the substrate node with both enough CPU
         and maximum weighted available resource
6      if failing in Line 5 then
7        reject the request and postpone it into the waiting queue
         break
8      end if
9    end for
10 end for
11 sort the already mapped requests according to their
     revenues in descending order
12 for each request in order do
13   for each virtual link of the request do
14     search the k-shortest paths for increasing k
15     if there exist paths that satisfy the virtual link constraint
         then
16       map the virtual link to the shortest path
17     else
18       reject the request and postpone it into the waiting queue
         break
19     end if
20   end for
21 end for
```

3.3 Brief discussion of the complexity

Compared to the baseline algorithm, the new algorithm we proposed adds a computational cost in

finding the coefficient m. However, since we need only to check the nodes already selected in the same request, the added computational cost will be negligible.

Moreover, since the new algorithm considers node mapping and link mapping at the same time, the link mapping in the new algorithm will be much easier than the baseline algorithm. Thus, the total runtime will decrease. We will demonstrate this by recording the runtime of both algorithms in simulation.

4 Simulation and analysis

In this section, we first introduce the simulation environment and parameter setting, and then analyze the performance of the new algorithm by comparing it with the baseline algorithm.

4.1 Simulation environment and parameter setting

We used the GT-ITM tool (Zegura *et al*., 1996) to generate the substrate network. It is a widely used method for network topology generation and simulation. The substrate network has 100 nodes and around 500 links, which is close to the scale of a medium-sized Internet service provider (ISP). The nodes' CPU and links' BW both follow a uniform distribution from 0 to 100 units.

The virtual network requests were generated following the previous work (Zhu and Ammar, 2006). The number of virtual nodes follows a uniform distribution from 2 to 10. Each pair of virtual nodes randomly connects to each other with a probability of 0.5. The virtual nodes' CPU and links' BW follow a uniform distribution with the size increasing from 10% to 90% of the substrate network's CPU and BW. This configuration not only evaluates the performance of the new algorithm in different conditions, but also shows the influence of the increasing requests' scale. The arrival of the virtual network requests follows the Poisson distribution with the mean of five requests per time window. The duration of the requests is exponentially distributed with the mean of 10 time windows. We ran the simulation in each condition for 500 time windows and used the average values to make our simulation results statistically stable. We can infer from the duration that the evaluation system will become stable after about 10 time windows, which makes 500 time windows enough for the stability of the results.

DELAY is a parameter to indicate the time windows a virtual request will wait. Corr (Eq. (7)) is the main reflection of the proximity principle. The effect of Corr will be simulated in the following section.

The factors we used to judge the performance of the embedding algorithm are as follows: R/C ratio, acceptance ratio, and runtime (Eqs. (3)–(5)).

4.2 Results and analysis

4.2.1 Impact of increasing request's BW: CPU=50%, BW=10%–90%, DELAY=3, Corr=2

As shown in Fig. 4, while the request's BW increases from 10% to 90% of the substrate network's BW, the performance of the new algorithm keeps better than that of the baseline algorithm: the R/C ratio is larger (28% on average); the acceptance ratio

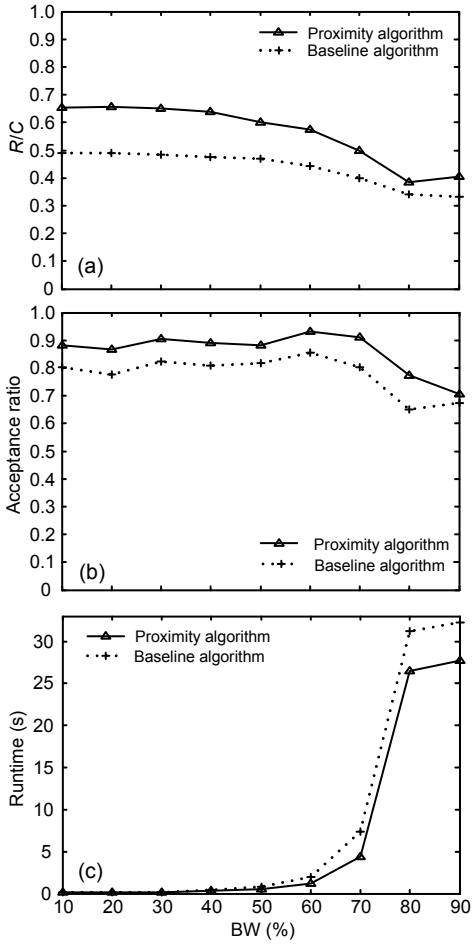


Fig. 4 The R/C ratio (a), acceptance ratio (b), and runtime (c) with increasing BW
CPU=50%, DELAY=3, Corr=2

is larger (10% on average); the runtime is less (22% on average). These results show that our proximity algorithm uses the substrate network resource more efficiently, which is reflected by the *R*/*C* ratio. Meanwhile, more requests are accepted since we squeeze out some room for them by enhancing the utilization of the substrate network. Moreover, the runtime is also saved. We believe it is mainly because the link mapping step of the proximity algorithm maps the virtual link in a shorter distance, with a higher success ratio. Thus, the time is saved by avoiding backtracking.

For both algorithms, while the BW increases, the *R*/*C* ratio and acceptance ratio decrease, and the runtime increases sharply, meaning that the request's BW scale affects the system performance, especially for the runtime. Note that the acceptance ratio increases a little as the BW increases up to 60%. This is because we map the virtual request in descending revenue order to increase the utilization of substrate resource. Thus, as the BW increases, a virtual request with larger revenue could fail in link mapping and the corresponding resource will be saved for the following smaller requests. Thus, there exists a chance that two or more small requests, instead of one large request, are mapped successfully. That is why the acceptance ratio increases a little. However, it is a small effect compared with BW after BW increases up to around 60%. Thus, the acceptance ratio decreases sharply by that point.

4.2.2 Impact of increasing request's CPU: CPU=10%–90%, BW=50%, DELAY=3, Corr=2

As shown in Fig. 5, while the request's CPU increases from 10% to 90% of the substrate network's CPU, the performance of the new algorithm keeps better than that of the baseline algorithm: the *R*/*C* ratio is larger (33% on average); the acceptance ratio is larger (7% on average); the runtime is less (28% on average). These mean that in all CPU conditions, our proximity algorithm has a better performance than the baseline algorithm.

For both algorithms, when CPU increases, the *R*/*C* ratio decreases a little, while the acceptance ratio decreases clearly. Compared with the results in Section 4.2.1, we see that CPU affects the acceptance ratio more than BW. The runtime situation is different from that in Section 4.2.1 in two aspects: first of all, the CPU has less effect than BW on the runtime, which means mapping nodes needs less computational cost than mapping links; second, the runtime in Section 4.2.2 decreases along with the increasing CPU. We believe this situation occurs because the large CPU fails in node mapping. Thus, some runtime will be saved by avoiding mapping corresponding links.

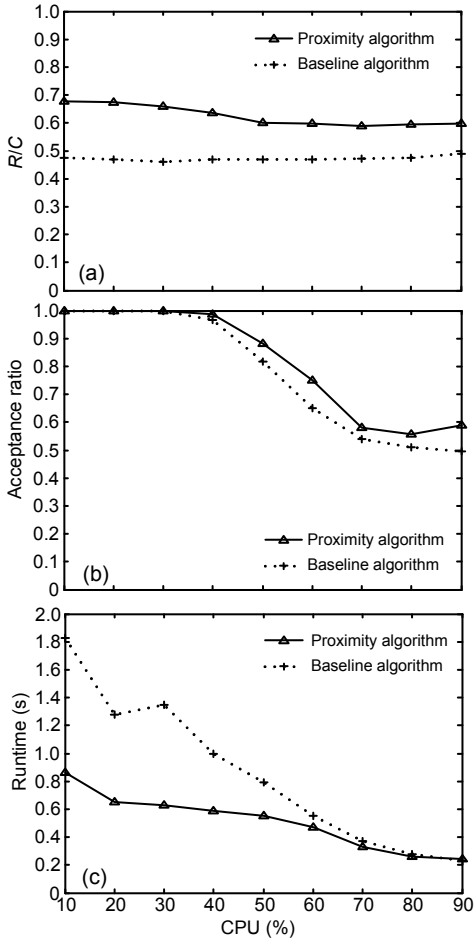


Fig. 5 The *R*/*C* ratio (a), acceptance ratio (b), and runtime (c) with increasing CPU
BW=50%, DELAY=3, Corr=2

4.2.3 Impact of increasing Corr: CPU=10%–90%, BW=50%, DELAY=3, Corr=1–4

As shown in Fig. 6, while Corr changes from 1/2 to 4, the *R*/*C* ratio continues to increase. After Corr>2, the *R*/*C* ratio grows slowly, which means the effect of Corr is limited by that point. Thus, we choose Corr=2 in our simulation to keep the balance of the distance factor and the capacity factor. Note that while Corr=1, the proximity principle is not working, which makes

the proximity algorithm equal to the baseline algorithm. While Corr<1, the proximity principle works opposite to the condition Corr>1, which means the substrate nodes closed to the already chosen ones will have a lower chance to be selected. Thus, while Corr=1/2, the performance of the proximity algorithm is worse than that of the baseline algorithm (Fig. 6).

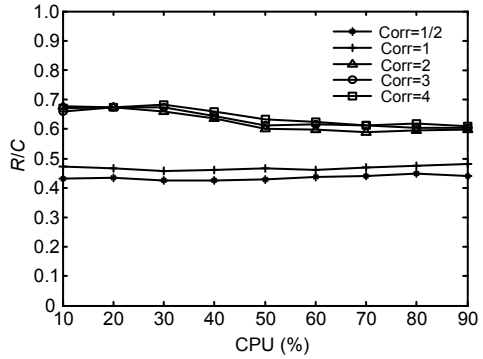


Fig. 6 Performance with increasing Corr for the proximity algorithm
BW=50%, DELAY=3

4.2.4 Path splitting

Path splitting is an assistant function of substrate network brought up in previous work (Yu *et al*., 2008). It splits the virtual link into small pieces in order to assign more requests to the substrate network. The baseline algorithm with path splitting increases the revenue clearly, but the cost is also increased; thus, the *R*/*C* ratio is poor. To demonstrate that our proximity algorithm is more suitable for the path splitting function, we simulate the proximity algorithm with path splitting and compare it to the baseline algorithm. In the simulation, the splitting ratio indicates the ratio of requests allowed to be split. Other parameters are as follows: CPU=50%, BW=50%, DELAY=3, Corr=2.

As shown in Fig. 7, while the splitting ratio increases from 0 to 100%, the performance of the proximity algorithm is better than that of the baseline algorithm: the *R*/*C* ratio is larger (31% on average); the acceptance ratio is larger (8% on average). Note that the runtime of the proximity algorithm is more (7% on average), but it is close to that of the baseline algorithm. Moreover, the total gain of the new algorithm is still positive with consideration of the *R*/*C* ratio and the acceptance ratio. Thus, we can conclude that the proximity algorithm is more suitable for the path splitting function.

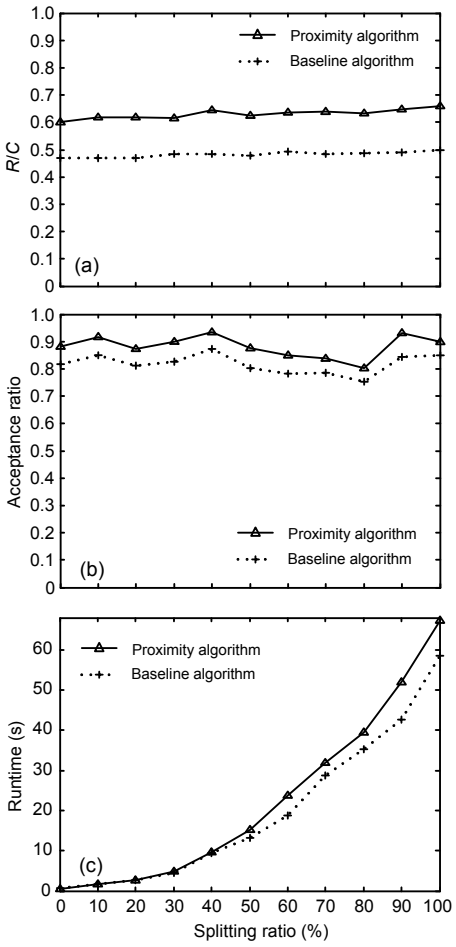


Fig. 7 The *R*/*C* ratio (a), acceptance ratio (b), and runtime (c) with the increasing splitting ratio
CPU=50%, BW=50%, DELAY=3, Corr=2

5 Conclusions

The main concern of the virtual network embedding problem is how to develop an algorithm both effective and easy to implement. By this study, we believe that we have found a better balance between the two factors. Simulation shows that the new algorithm based on the proximity principle greatly enhances the performance of the embedding in almost all conditions. Meanwhile, the runtime is saved in most simulation conditions.

There are still some issues that should be further discussed. First, the node mapping step is usually restricted in practice, such as in location or special functions. However, we believe that the proximity principle is still useful in such situations with some adjustments. Second, weighted available resource is

defined as an exponential function. We believe other models should be developed for different application conditions. Last but not the least, the scale of the virtual network is briefly mentioned in this paper. We believe it is an important issue, since it affects not only the performance of the embedding algorithm, but also the management policy of the substrate network, such as the charge problem.

References

Andersen, D.G., 2002. Theoretical Approaches to Node Assignment. Available from http://www.cs.cmu.edu/~dga/papers/andersen-assign.ps [Accessed on Sept. 20, 2010].

Anderson, T., Peterson, L., Shenker, S., Turner, J., 2005. Overcoming the Internet impasse through virtualization. *IEEE Comput. Mag.*, **38**(4):34-41.

Bavier, A., Feamster, N., Huang, M., Peterson, L., Rexford, J., 2006. In VINI Veritas: Realistic and Controlled Network Experimentation. Proc. Conf. on Applications, Technologies, Architectures, and Protocols for Computer Communications, p.3-14. [doi:10.1145/1151659.1159916]

Fan, J., Ammar, M.H., 2006. Dynamic Topology Configuration in Service Overlay Networks: a Study of Reconfiguration Policies. Proc. 25th IEEE Int. Conf. on Computer Communications, p.1-12. [doi:10.1109/INFOCOM.2006.139]

Feamster, N., Gao, L., Rexford, J., 2007. How to lease the Internet in your spare time. *ACM SIGCOMM Comput. Commun. Rev.*, **37**(1):61-64. [doi:10.1145/1198255.1198265]

Kleinberg, J., 1996. Approximation Algorithms for Disjoint Paths Problems. PhD Thesis, MIT, USA.

Kolliopoulos, S.G., Stein, C., 1997. Improved Approximation Algorithms for Unsplittable Flow Problems. Proc. 38th Annual Symp. on Foundations of Computer Science, p.426-436. [doi:10.1109/SFCS.1997.646131]

Lu, J., Turner, J., 2006. Efficient Mapping of Virtual Networks onto a Shared Substrate. Technical Report No. WUCSE-2006-35, Washington University, USA.

Ricci, R., Alfeld, C., Lepreau, J., 2003. A solver for the network testbed mapping problem. *ACM SIGCOMM Comput. Commun. Rev.*, **33**(2):65-81. [doi:10.1145/956981.956988]

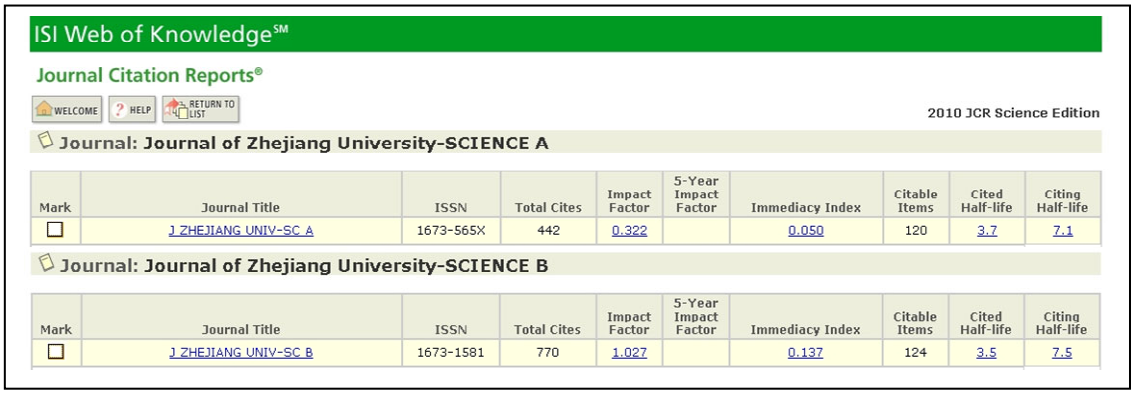
Turner, J.S., Taylor, D.E., 2005. Diversifying the Internet. Proc. IEEE Global Telecommunications Conf., p.755-760.

Yu, M., Yi, Y., Rexford, J., Chiang, M., 2008. Rethinking virtua1 network embedding: substrate support for path splitting and migration. *ACM SIGCOMM Comput. Commun. Rev.*, **38**(2):17-29. [doi:10.1145/1355734.1355737]

Zegura, E.W., Calvert, K.L., Bhattacharjee, S., 1996. How to Model an Internetwork. Proc. 15th IEEE Int. Conf. on Computer Communications, p.594-602.

Zhu, Y., Ammar, M., 2006. Algorithms for Assigning Substrate Network Resources to Virtual Network Components. Proc. 25th IEEE Int. Conf. on Computer Communications, p.1-12. [doi:10.1109/INFOCOM.2006.322]

2010 JCR of Thomson Reuters for *JZUS-A* and *JZUS-B*



ISI Web of Knowledge℠

Journal Citation Reports®

WELCOME | HELP | RETURN TO LIST

2010 JCR Science Edition

Journal: Journal of Zhejiang University-SCIENCE A

Mark	Journal Title	ISSN	Total Cites	Impact Factor	5-Year Impact Factor	Immediacy Index	Citable Items	Cited Half-life	Citing Half-life
☐	J ZHEJIANG UNIV-SC A	1673-565X	442	0.322		0.050	120	3.7	7.1

Journal: Journal of Zhejiang University-SCIENCE B

Mark	Journal Title	ISSN	Total Cites	Impact Factor	5-Year Impact Factor	Immediacy Index	Citable Items	Cited Half-life	Citing Half-life
☐	J ZHEJIANG UNIV-SC B	1673-1581	770	1.027		0.137	124	3.5	7.5

JZUS-A is an international “Applied Physics & Engineering” reviewed-Journal, covering research in Applied Physics, Mechanical and Civil Engineering, Environmental Science and Energy, Materials Science, and Chemical Engineering. *JZUS-B* is an international “Biomedicine & Biotechnology” reviewed-Journal, covering research in Biomedicine, Biochemistry, and Biotechnology. *JZUS-A* and *JZUS-B* were covered by SCI-E in 2007 and 2008, respectively.