ISSN : 0974 - 7435 Volume 8 Issue 6

BioTechnology
An Indian Journal

FULL PAPER

BTAIJ, 8(6), 2013 [804-814]

Distributed and parallel virtual network embedding

Zhang Dong*, Gao Long
College of Mathematics and Computer Science Fuzhou University, (CHINA)
E-mail: zhangdong@fzu.edu.cn

ABSTRACT

Network virtualization has recently emerged as a promising solution for diversifying the future Internet architecture into separate virtual networks (VNs). The problem of efficiently embedding multiple independent VNs over a common substrate infrastructure is a challenging problem on cloud computing platforms and large-scale future network testbeds. To solve this problem, we take advantage of a Map-Reduce framework, decomposing a VN request into meta-requests, using a link-priority algorithm and pheromone transmission based on multi intelligent route nodes, which have the ability to distribute meta-requests and collect meta-request embedding results in the substrate, and assign VNs to the substrate physical network in a distributed and parallel manner. A distributed and parallel VN embedding protocol is proposed to communicate and exchange messages among substrate nodes to achieve successful embedding. Results of implementation and a performance evaluation of the distributed and parallel VN embedding algorithm in terms of embedding time, acceptance ratio, produced messages and revenue and cost are presented at the end of this paper.

KEYWORDS

Network virtualization;
Virtual network (VN);
Virtual network embedding (VNE);
Parallel and distributed;
Pheromone.

INTRODUCTION

After several decades of development, Internet has demonstrated its strong vitality and broad prospects. With the development of network technologies and applications, it is difficult for the traditional architecture of Internet to adapt to variety of network requirements and Quality of Service. etwork virtualization has become an important direction for the development of the future Internet[10]. By sharing an underlying network, infrastructure providers (InPs) rationally allocate network resources to provide heterogeneous coexistence and mutual isolation for multiple service providers (SPs). These VNs usually span multiple InPs, which may support services for different SPs in the same period and offer resource sharing between InPs. Existing VN embedding approaches mainly focus on serial management, assuming requirements of VNE are achieved sequentially with unified management. The management centres maintain the networks by collecting global information and allocating node and link resources in a unified manner. All InPs cannot have unified planning and centralized management, so it is difficult to reach the goal of total network virtualization. There is an urgent need to implement a VNE parallel algorithm, by which multiple InPs can construct VNs in a cooperative man-

ner and share open substrate resources.

Several researches have realized a distributed or concurrent VN embedding algorithm at different levels and through different concepts[7,8]. Among them, Houidi et al.[8] firstly proposed a VN embedding framework based on the use of underlying nodes that communicate and exchange information with each other (node-priority). On the basis of the structure of Houidi et al.[8], it designed a distributed VN embedding mechanism (DAVNM) which means that a VN request is firstly decomposed into a multiple subnet sequence star radiation structure, while regularly maintaining and updating the node power for sorting and finding the shortest path tree in the underlying network, and selecting the max-capability node as the root which is responsible for the star subnet embedding. The method in Houidi et al.[8] can reduce the load of the embedding management of the central node and realize concurrent VN embedding, but each underlying node must store and periodically update its ability for sorting and finding the corresponding shortest path tree. It is an offline building process that requires prior knowledge of a VN request, has no admission control mechanism, and the inter-node communication cost is high and VN building efficiency is low, so VN requests have poor acceptance rates. To improve efficiency and achieve parallel embedding of multiple VN requests through VN request decomposition of a multi-level structure, a method involving a sub-graph to perfect graph matching algorithm to realize VN embedding has been proposed[7], but the number of layers of decomposition is difficult to determine and the process uses centralized management of a single sub-graph and multiple sub-graphs to achieve concurrent embedding. Chowdhury et al.[2] proposed VNE across multiple InPs by unified management in an inter-domain and allowing each InP to enforce its local policies at the same time. However, VN embedding is still sequential in Chowdhury et al.[2]; Houidi et al.[9]. Fajjari et al.[6] proposed a scalable embedding strategy based on the Ant Colony metaheuristic. Leivadeas et al.[11] considered the social features of the physical network and proposed an evaluation of the socio-aware VNE paradigm. Di et al.[4] used mixed integer programming method to complete VNE. These methods completed the perception of global information and resource allocation through centralized control. In response to these issues, this article does not reduce the complexity of the problem and basic admission control mechanism; we provide and design a distributed and parallel building framework based on node autonomy using MapReduce[3] to realize a pheromone-transmission distributed online VNE. The major contributions of this paper can be summarized as follows:

- We formulate a parallel distributed framework to construct VNs based on multi intelligent route nodes simultaneously. To the best of our knowledge, this work is the first to design a parallel framework to solve VNE problems.
- Collecting local information about neighbour resources based on topology potential instead of global information in the periods of node ranking and embedding, which avoids excessive message exchanges among nodes throughout the entire network.
- To verify the parallel algorithm proposed in this paper, we define the common service interface protocols and implement them on PlanetLab, which is an open platform for developing and deploying planetary-scale services.

VN EMBEDDING MODEL AND PROBLEM FORMULATION

Substrate network model

We model the substrate network as an undirected graph denoted by $G_s = (V_s, E_s)$, where V_s is the set of substrate nodes and E_s is the set of substrate links. Each substrate node $v_s^i \in V_s$ is associated with weight value $C_s(v_s^i)$, which denotes the available capacity of the substrate node v_s^i. The capacity of a network node includes CPU processing capacity, storage and so forth; and the typical capacity of a network link is its bandwidth. Likewise, each substrate link $e_s(i, j) \in E_s$ between two substrate nodes i and j is also associated with weight value $W_s(e_s(i, j))$, which denotes the available bandwidth capacity associated with the substrate link.

Virtual network model

Similar to the substrate network, the VN is also

FULL PAPER

represented by an undirected graph $G_v = (V_v, E_v)$, where V_v is the set of virtual nodes, and E_v is the set of virtual links. The requirements on the virtual node and virtual link in terms of attributes of nodes and links of the substrate network are described as follows: the minimum required capacity of each virtual node $v_v^i \in V_v$ is denoted by $C_v(v_v^i)$, and the minimum required bandwidth capacity of virtual link $e_v(i, j) \in E_v$ between virtual nodes i and j is denoted by $W_v(e_v(i, j))$.

Virtual network emedding problem description

In fact, substrate network G_s is not able to host an infinite number of VN requests, as all substrate resources are limited. So finding a best embedding between G_v and G_s is necessary to maximize the VN request acceptance rate and the substrate provider's revenue while reducing the VN embedding cost. Based on the VN and substrate models, the VN embedding problem

(Fig.1) is defined by the embedding function EMBED:

$$G_v(V_v, E_v) \to G_s(V_{vs}, E_{vs}).$$

(1) The node embedding function between virtual nodes and substrate nodes: $V_v \to V_{vs} \subseteq V_s$ satisfies

$$\forall v_v \in V_v, \exists v_s^1, v_s^2, ..., v_s^k, (k \geq 1), \text{subject to } C_s(v_s^k) \geq C_v(v_v)$$

$$|V_v| = |V_{vs}|.$$

V_{vs} denotes the set of substrate nodes hosting the virtual nodes. The requirement $|V_v| = |V_{vs}|$ guarantees that different virtual nodes in the same VN G_v cannot be assigned to the same substrate node. $|V_v|$ is the number of VN nodes, and $|V_{vs}|$ is the number of substrate nodes hosting the VN nodes.

(2) Linking embedding functions between virtual links and substrate paths: $E_v \to E_{vs}$ Satisfying

$$\forall e_v \in E_v, \exists p, \textit{subject to } W_v(e_v) \leq min_{e_s \in p} W_s(e_s).$$

E_{vs} is the set of paths embedded by the virtual links, and p is one member of this set. Note that except for the end nodes, the substrate nodes associated with substrate path p just offer the service of transmission information, and thereby we can assume this service does not consume substrate node resources.

(3) Objections

Revenue denoted by $Rev(G_v)$ is defined as follows:

$$Rev(G_v, t) = \alpha_r \sum_{v_v \in V_v} C_v(v_v) + \beta_r \sum_{e_v \in E_v} W_v(e_v).$$

Above, α_r and β_r are the weighting coefficients to balance the effects of bandwidth of the link and capacity of the node, respectively.

Cost is $Cost(G_v, t)$ and is defined as follows:

$$Cost(G_v, t) = \alpha_c \sum_{e_v \in E_v} Hop(e_v) W_v(e_v) + \beta_c \sum_{v_v \in V_v} C_v(v_v).$$

$Hop(e_v)$ is the hop count of substrate paths assigned to the virtual link e_v; α_c and β_c are the weighting coefficients to balance the effects of bandwidth of the link and capacity of the node, respectively. Finally, the acceptance ratio of the system is defined by

$$\text{Acceptance ratio} = \frac{\text{Number of accepted requests}}{\text{Number of all requests}}.$$

DISTRIBUTED AND PARALLEL EMBEDDING FRAMEWORK

Inspired by MapReduce[3], this paper proposes a virtual network distributed and parallel embedding framework, taking full advantage of an underlying node which has the characteristics of automatic storage computing and routing forwarding capabilities, regards a substrate node as an autonomic computing individual and realises the virtual network task requests 'Map' and building 'Reduce' through a multi-node parallel processing mechanism. A distributed and parallel virtual network embedding framework based on MapReduce will improve the flexibility, scalability and reliability of the network.

As outlined in Figure 1, a virtual network request is submitted to one substrate node called the launching node and is then decomposed into many meta-requests. The meta-requests are distributed to different substrate nodes, and substrate nodes have the responsibility to complete the corresponding meta-requests in parallel.

Autonomic substrate route node

Each underlying node can become the launching

node in VN request embedding, and has the ability to decompose a VN request into many meta-requests. The substrate node consists of five components: decision, store, forwarding, produce and decomposition, and it can make decisions by analysing and evaluating its acquired information (meta-request, input message, VN request), and then execute corresponding actions. The executing actions primarily include: (1) produce message; (2) forward message; (3) store message; (4) decomposition of the VN request.

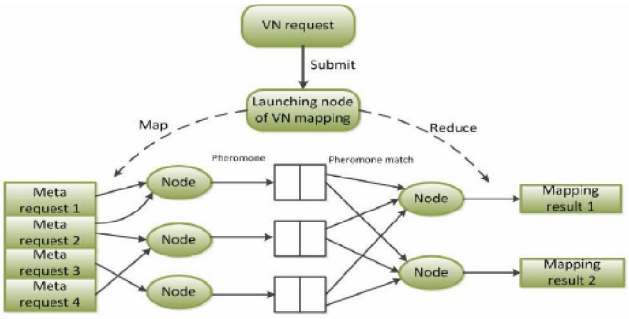

Figure 1 : The framework of distributed and parallel VN embedding

Communication protocol

A communication protocol is necessary to coordinate substrate nodes' statuses and exchange information among all substrate nodes. In this section, the protocol is defined based on six types of messages: START, MSG, CANCEL, INFO, RESULT and CONTROL. RESULT and CONTROL messages are used to communicate between the launching node of VN embedding and the launching node of meta-request embedding, and the other messages are used to communicate among the substrate nodes.

- START: The START message is sent from the launching node of VN embedding to all substrate nodes to trigger and start the distributed and parallel virtual network embedding algorithm and asks for the substrate network resource status. This message also initiates decomposing the VN request into meta-requests.
- MSG: This message is used to reply to the START message of the launching node of VN embedding.
- CANCEL: A CANCEL message is sent to some substrate nodes to stop a VN request once its meta-request cannot be embedded well (e.g. cannot be embedded into the substrate network). That is to say, any information related with this VN request will be eliminated, and any resources (e.g., node and link resources) having been occupied by this VN request will be released.
- INFO: The INFO message is used to exchange node capacities and surrounding link capacities among all substrate nodes.
- RESULT: Using the RESULT message, the launching node of meta-request embedding notifies the launching node of VN embedding of the meta-request embedding results. The results include: (1) embedding success or not (2) which virtual node is assigned to which substrate node and (3) which virtual link is assigned to which substrate path.
- CONTROL: Using the CONTROL message, the launching node of VN embedding assigns the meta-request to the launching node of meta-request embedding.

Implementation of the distributed and parallel VN embedding algorithm

As substrate network resources are limited, some meta-requests of each VN request have been embedded, namely each VN request occupies some substrate resources and the rest of the meta-requests cannot be embedded. The result is that some requests for VN embedding will fail in a parallel framework. To avoid this, we define the maximum parallel size as MAX. To adapt to the current substrate traffic and resources, the value of the parallel size is variable. At first, the parallel size is set as n ($1 < n < MAX$), and n VN requests are submitted to the launching node of VN embedding, and after some processing, m VN requests return success, that is to say, n - m VN embedding requests fail. The n - m VN requests will be submitted to the launching node of VN embedding one by one, the result is that p VN embedding requests will succeed, and the parallel size becomes the minimum of MAX ($p + m + 1$) that allows all VN embedding requests to succeed.

The launching node of VN embedding receives a VN request then sends the START message to all the substrate nodes and at the same time decomposes the VN request into meta-requests. The substrate node gets local topology information and informs the launching node of VN embedding using the MSG message. The launching node of VN embedding chooses the launch-

FULL PAPER

ing node for meta-request embedding of each VN request by understanding the global topology and then determines and notifies the launching node of meta-request embedding of the meta-request information by the CONTROL message. The launching node of meta-request embedding will execute the VN embedding algorithm by INFO message exchange. The results of the VN embedding algorithm execution will return to the launching node of VN embedding by the RESULT message.

The meta-request embedding of one VN request occurs in parallel and the embedding of n (parallel size as n) VN requests is also in parallel, namely the beginning of one VN request embedding does not depend on the ending of another VN embedding request.

DISTRIBUTED AND PARALLEL VN EMBEDDING ALGORITHM

VN request decomposition

To implement distributed virtual network embedding and reduce the complexity of the entire VN request embedding process, a virtual network request will be decomposed into elementary clusters (links, star or tree clusters)[8]. Likewise, for the convenience of distributed and parallel embedding and to enhance VN self-healing capabilities, before the virtual network is embedded into the substrate network, the virtual network request will be decomposed into meta-request set G_v^{meta} (Map). G_v^{meta} is a set of virtual links that includes three different subsets:

$$G_v^{meta} = E_v^{double} \cup E_v^{single} \cup E_v^{none}$$

$$E_v^{double} \cap E_v^{single} = \varnothing \quad E_v^{double} \cap E_v^{none} = \varnothing \quad E_v^{single} \cap E_v^{none} = \varnothing .$$

(1) double-node virtual link $E_v^{double} = \{[e_v^i, e_v^j]\}$ $[e_v^i, e_v^j]$ denotes a virtual link (e_v^i, e_v^j) and two virtual nodes e_v^i and e_v^j associated with it.

(2) single-node virtual link $E_v^{single} = \{[e_v^i, e_v^j)$ or $(e_v^i, e_v^j]\}$ ÿ$[e_v^i, e_v^j)$ or $(e_v^i, e_v^j]$denotes a virtual link (e_v^i, e_v^j) and one virtual node e_v^i or e_v^j associated with it.

(3) none-node virtual link $E_v^{none} = \{(e_v^i, e_v^j)\}$, (e_v^i, e_v^j) denotes a virtual link (e_v^i, e_v^j) without any virtual node.

The VN request will be decomposed into three types of virtual links. The link resource value of the double-node virtual link is larger than that of the single-node virtual link, which in turn is larger than that of the none-node virtual link.

Pheromone and pheromone matching judge algorithm

The pheromone defined here are different from those provided in Dorigo et al.[5]. The traits of the pheromones provided in this paper are as follows: Transmission by different speeds in the substrate network; One type of pheromone will block the transmission of another pheromone;Once pheromones meet or virtual link embedding fails, the corresponding pheromone at the node will disappear instantly;A pheromone will not be sent to nodes which it has already passed. The transmission speed of a pheromone is:

$$V = \frac{s_1}{s_2 \ln\left(\frac{\Omega}{\Delta} + e - 1\right)}$$

where, s_1 denotes the link resource value of a meta-request (double-node virtual link, single-node virtual and none-node virtual link) and s_2 denotes the current substrate link's resource. Ω denotes the amount of pheromone in the current substrate link; Δ denotes the total amount of current pheromone.

TABLE 1: Pheromone matching judge algorithm

1	When the node receives a pheromone, it will judge if this pheromone is new.
2	If (pheromone is not new)
3	judge whether the pheromone meets with matching pheromone
4	if (meets) the node sends a message to launching node for reserving the resource; then sends messages to the entire network to cancel the corresponding pheromone. else forward this pheromone and store it
5	else do nothing;

Node assessment and double-nodes virtual link embedding algorithm

(1) Node-priority: After one double-node virtual request link is produced, the launching node of VN embedding will choose the top two nodes v_{s1},v_{s2} $\{C(v_{s1}) > C(v_{s2}) > \ldots > C(v_{sn})\}$ in the substrate node resource rank list to embed the virtual nodes v_{v1} and v_{v2} $(C(v_{v1}) > C(v_{v2}))$ of this double-node virtual request link,

then applies the pheromone matching judge algorithm to complete the embedding of the virtual link (v_{v1},v_{v2}).

(2) Link-priority: According to an autonomous node's characteristics of local topology-awareness, the underlying physical nodes can obtain the resource capacity of surrounding links and nodes. The launching node of VN embedding can obtain the entire substrate network information through message communications. The launching node of VN embedding will perform the following: □the substrate network topology will be decomposed into the node-pair set S_a, which is different from the that of the virtual network. The link is unique in this set, however the nodes are not. After the double-nodes virtual request link is produced, the substrate node-pair set S_a will be simplified as set S_b according to the virtual link request's resource. Simplifying rule: the resource of the physical node pair's link is more than that of the double-node virtual request link. □then the set S_b will be simplified as set S_c. Simplifying rule: the combined resource of the physical node pair is greater than that of the double-node virtual request link. If the set S_c is null, then transform set S_b into S_d based on whether the combined resource of the physical node pair is larger than that of the double-node virtual request link; then set S_c is obtained from S_d by a node pair merger, the merger rule is:

$$\begin{pmatrix} 0 & w_{12} & \dots & w_{1n} \\ w_{21} & 0 & \dots & w_{2n} \\ M & M & M & M \\ w_{n1} & w_{n2} & \dots & 0 \end{pmatrix}$$

$$\text{if}\,(w_{ij}=0)\quad w_{ij}=\max\{\min(w_{ik},w_{kj}),\ k\neq i,j\}\ .$$

Select a node pair that has the maximum link resource from set S_c, then the two nodes of the node pair will be used to host the double-node virtual request link.

Overall algorithm of distributed and parallel embedding

This paper focuses on the VN decomposition and addresses how to complete VN embedding in a distributed and parallel manner. To achieve distributed and parallel VN embedding, the virtual request is decomposed into three types of meta-requests, as previously described. In this distributed and parallel framework, each node of the substrate network is an autonomous

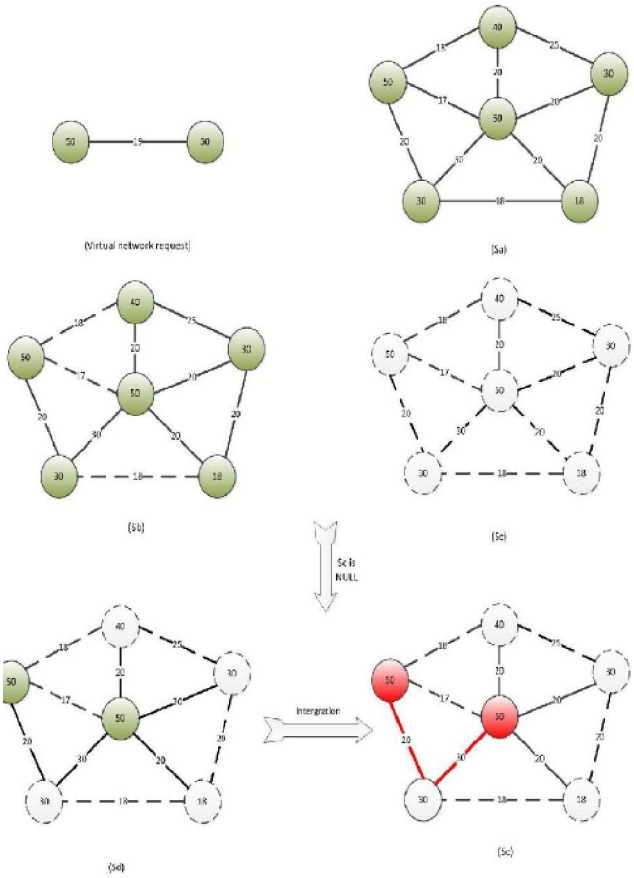

Figure 2 : The diagram of node assessment

individual which can be a launching node of a meta-request, so embedding of double-node virtual links can be completed in parallel. If all double-node virtual links map successfully, then single-node and none-node virtual links build a complete virtual network in parallel. Likewise, different virtual network embedding can also occur in parallel. The double-node virtual link can be embedded into the substrate network by (1) or (2) in section V.C. As for single-node virtual link embedding, we can also use the method in item (2) (V.C), as one end node being fixed is the only difference between the single-node virtual link and the double-node virtual link.

We also use the pheromone matching judge algorithm to complete none-node virtual link embedding. The overall algorithm of distributed and parallel embedding is outlined in TABLE 2.

PERFORMANCE EVALUATION AND COMPARISON

Simulation environment

Our proposed algorithm has been implemented and

FULL PAPER

tested on the PlanetLab platform. PlanetLab is a global research network that supports the development of new network services. The platform can emulate a real substrate network and each node slice deployed is programmed in PlanetLab as an agent that can meet the distributed algorithm requirements through interactions and messages with other node slices. In this section, we give the detailed description of the simulation environment and analyse the performance of the new algorithm by comparing it with other algorithms.

TABLE 2 : Overall algorithm of distributed and parallel embedding

Parallel While(VN_i){
If parallel($e=[e_v^i, e_v^j]$){ // double-node virtual request link
Substrate nodes do self-assessment;
Virtual nodes e_v^i, e_v^j are embedded to the substrate nodes e_s^i, e_s^j;
// e_s^i, e_s^j are neighbours or are very close in the substrate network
If e_v^i, e_v^j can't find the fit substrate node to embed, this result indicates that the entire virtual network embedding fails.
}
Else if parallel($e=[e_v^i, e_v^j)$ or $(e_v^i, e_v^j]$){ // single-node virtual request link
Substrate nodes do self-assessment;
Virtual node e_v^i is embedded to substrate node e_s^i(virtual node e_v^j has already been embedded to the substrate node e_s^j) or virtual node e_v^i is embedded to substrate node e_s^j(virtual node e_v^i has already been embedded to substrate node e_s^i);
Substrate nodes e_s^i, e_s^j produce pheromone respectively and then broadcast it;
The first meeting of the same pheromones that have passed through the path that will support the virtual linkmeans that the virtual link maps successfully.
During a period of time, if a pheromone meeting has been not occured the results indicate that the virtual link embedding fails, that is to say, the entire virtual network embedding fails.
}
Else if parallel($e=(e_v^i, e_v^j)$){ // none-node virtual request link
Virtual node e_v^i has been embedded to the substrate node e_s^i, virtual node e_v^j has been embedded to substrate node e_s^j;
Substrate nodes e_s^i, e_s^j produce the pheromone respectively and then broadcast it;
The first meeting of the same pheromone that have passed through the path that will support the virtual linkindicates that the virtual link maps successfully.
During a period of time, if a pheromone meeting has been not occurred the results indicate that the virtual link embedding fails, that is to say, the entire virtual network embedding fails.
}
}

In our simulation, we use the GT-ITM tool to generate the substrate network topology and 50 VN topologies. The substrate network topology has 15 nodes, and its link connectivity probability is 0.5. The computing capacity at substrate nodes and bandwidth capacity on the links follow a uniform distribution from 50 to 100. A substrate network node can be easily deployed by defining the characteristics of a node slice in the PlanetLab platform. We use socket communication programming-based TCP to connect one node slice to another. To do so, we specify a substrate network. We assume that VN requests arrive at average rates of 1 VN per minute, 2 VNs per minute, 3 VNs per minute and 4 VNs per minute for simulation purposes. In each VN, the number of nodes follows a uniform distribution from 4 to 10. The average probability of connectivity between any two nodes in the VN request is set to 0.5. The computing requirements of a virtual node and the bandwidth requirement of the virtual link both follow a uniform distribution, from 1 to 5 and from 1 to 9, respectively. We set the initial parallel size of VN embedding as 5.

Simulation environment

Under different virtual network request arrival rates and complete embedding of 50 virtual network requests, we compare our method (the link-priority algorithm) with traditional algorithms on VN acceptance ratio, the number of messages and the revenue/cost (R/C) ratio.

(1) Figure 3 shows that under different virtual request arrival rates, the link-priority algorithm can produce a higher acceptance rate than the node-priority algorithm; at the same time, the link-priority algorithm is superior to RW-MM-SP[1] and DAVNM[8] regarding VN acceptance ratio.

The reason is that the node-priority, RW-MM-SP and DAVNM algorithms may map close virtual nodes into substrate nodes which are far away, however, the link-priority algorithm can avoid this case. As

the number of VN requests increases, the node-priority, RW-MM-SP and DAVNM algorithms can consume more substrate resources, which is not the case with the proposed link-priority algorithm, thus it achieves better results than the other three algorithms regarding VN acceptance ratio.

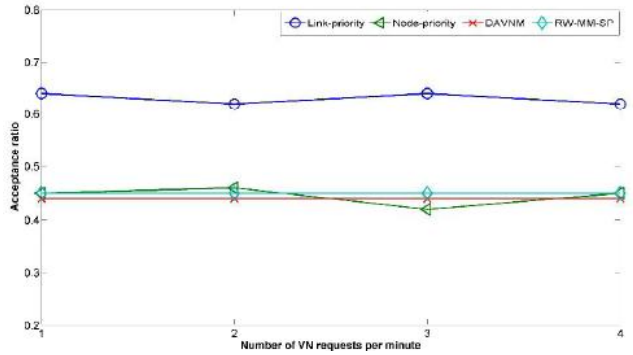

Figure 3 : Comparison of acceptance ratio

When the VN request arrival rate is 2, the node-priority algorithm has a somewhat higher acceptance ratio than RW-MM-SP and DAVNM; when the VN request arrival rate is 3, its acceptance ratio is a little lower than that of the two algorithms. This is due to the characteristics of the pheromone; as the VN request arrival rate accelerates, the number of messages in the substrate network will increase, and these slightly influence the VN embedding acceptance ratio, which can be seen from Figure 3.

(2) The number of failed VN embedding requests will affect the underlying network traffic, so the proposed algorithm uses the waiting method to determine whether a VN embedding request fails or not. When the embedding time is longer than 3.5 m, the embedding will be considered a failure. In contrast, the node-priority algorithm produces more messages than the proposed algorithm as shown in Fig.4. Meanwhile, when embedding the same number of VN requests, the DAVNM algorithm also produces more messages than the node-priority and the proposed link-priority algorithms. This illustrates that using the pheromone proposed in this paper to embed the VN request results in fewer messages thus improving overall performance.

(3) Figure 3 and Figure 5 show that the algorithms have different virtual network acceptance rates, so they have different embedding revenues. The proposed link-priority algorithm has a higher acceptance ratio, so it represents more revenue. The link-priority algorithm can avoid the embedding of close virtual nodes into substrate nodes which are far away, indicating that the link-priority algorithm has lower cost than the other three algorithms. The revenue/cost (R/C) ratio is shown in Figure 5 for the various algorithms. The link-priority algorithm has a higher R/C than the node-priority, RW-MM-SP and DAVNM algorithms. At times, the node-priority algorithm has a higher acceptance ratio, so it can have a higher R/C ratio than the RW-MM-SP and DAVNM algorithms; Note that RW-MM-SP and DAVNM algorithms have the same acceptance ratio, so they also have the same R/C (Figure 3, Figure 5).

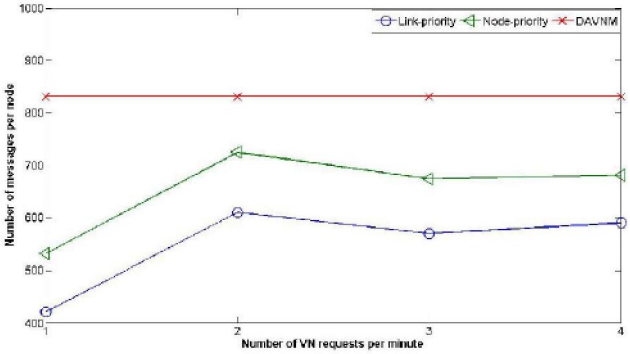

Figure 4 : Comparison of number of VN embedding messages

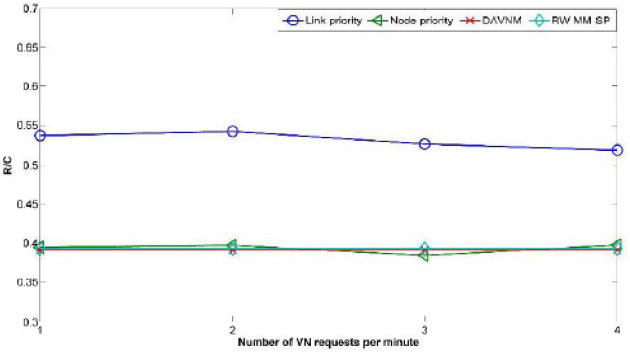

Figure 5 : Comparison of embedding R/C

By comparison of several algorithms (Figures 3-5), we see that link-priority has good performance regarding VN acceptance ratio, embedding time and revenue/cost (R/C).

Figure 6 shows that with the number of VN request arrivals per minute from 1 to 4, the time taken to build 10 VN requests gradually decreases and the time taken to build 20, 30, 40 and 50 VN requests fluctuates. When the substrate network has abundant re-

FULL PAPER

sources, the arrival rate of VN requests is low enough to allow efficient VN embedding. With the increase of VN requests, the underlying network resources are gradually reduced and some unsuccessful virtual network embedding (embedding time is more than 3.5 minutes) leads to adjustment of the parallel size which can account for the time fluctuations observed in the 20, 30, 40 and 50 VN requests with different VN request arrival rates.

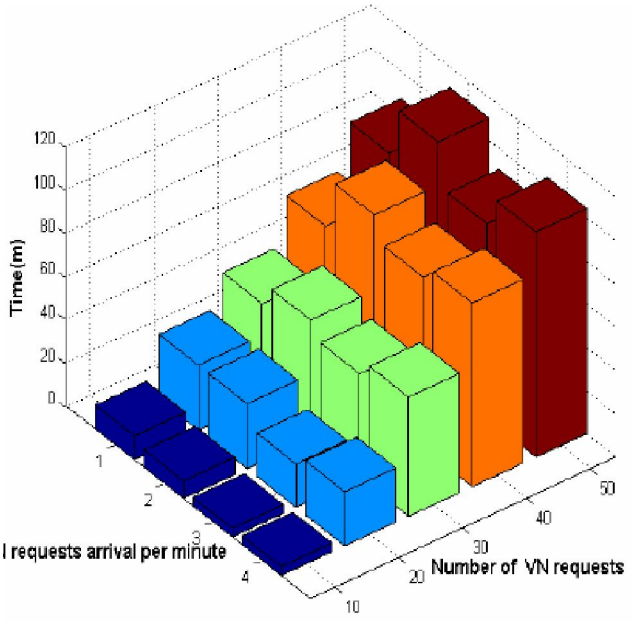

Figure 6 : Time taken by the link-priority algorithm for different number of VN requests at different VN arrival rates

As shown in Figure 7, with the increase of VN requests under each arrival rate, the underlying network resources are gradually consumed and lead to a decline in the VN acceptance ratio. Note the similarity in the acceptance ratio for the same number of VN requests with different arrival rates. This phenomenon can be attributed to the adjustable parallel size discussed in Section III.C

As Figure 8 shows, the number of messages is almost the same under different VN request arrival rates when the number of VN requests is 10 and 20, respectively. However, when the number of VN requests is 30, 40 or 50, respectively, the number of messages varies under different VN request arrival rates. The lower number of unsuccessful VN embedding attempts among 10 and 20 VN requests than among 30, 40 and 50 VN requests leads to this result.

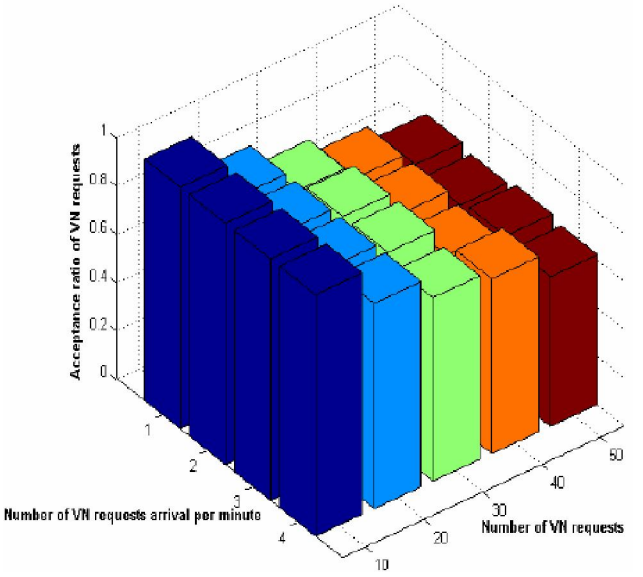

Figure 7 : VN acceptance ratio of the link-priority algorithm with different number of VN requests at different VN arrival rates

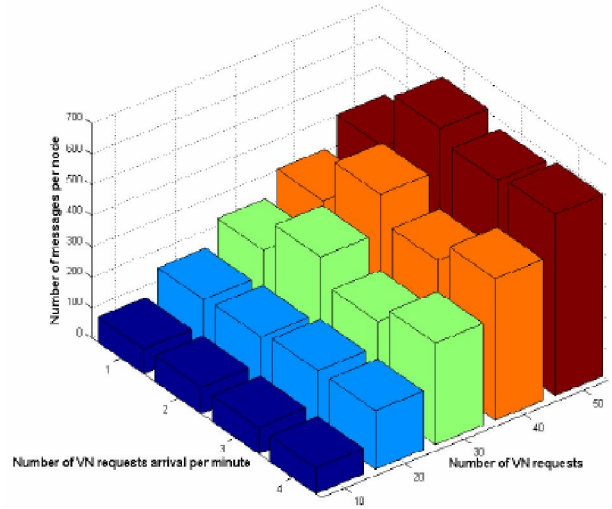

Figure 8 : Number of messages for different numbers of VN requests embedded into the substrate network of the link-priority algorithm

TABLE 3 shows that the revenue for 50 VN embedding requests is almost the same under different VN request arrival rates and the cost is almost the same under different VN request arrival rates. The acceptance ratios of 50 VN requests under different VN request arrival rates are also almost the same that can explain the result.

We compare the link-priority and the node-priority algorithms in terms of acceptance ratio, VN embedding time, number of embedding messages, embedding revenue and cost under different VN request arrival

rates when the number of VN requests is 50. Figure 9 shows that link-priority produces a higher acceptance than node-priority under different VN request arrival rates. The number of successful VN embedding requests will affect the VN request's average embedding time and the number of messages in the substrate network, and we use the waiting method to determine embedding failure in the link-priority algorithm. So the node-priority algorithm needs more time and messages as shown in Figures 10 and.11, respectively.

TABLE 3 : Revenue and cost of the link-priority algorithm for 50 VN requests at different VN arrival rates

NUMBER OF VN REQUEST ARRIVALS PER MINUTE	REVENUE	COST
1	2103	3914
2	2177	4012
3	2100	3987
4	2029	3910

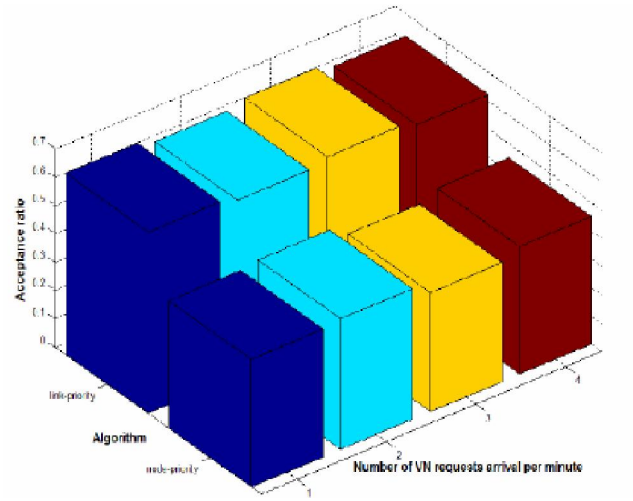

Figure 9 : Comparison of acceptance ratio

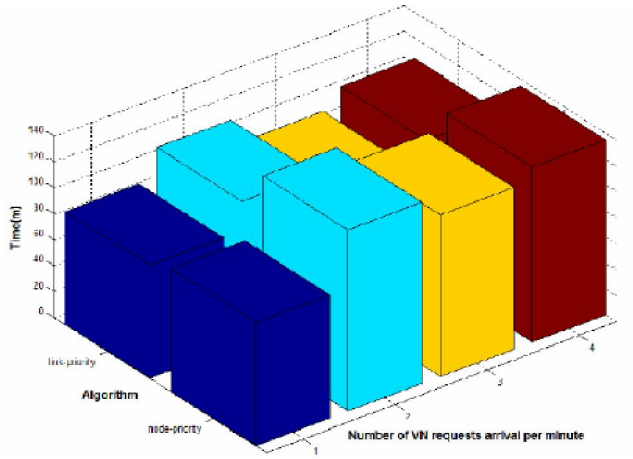

Figure 10 : Comparison of time

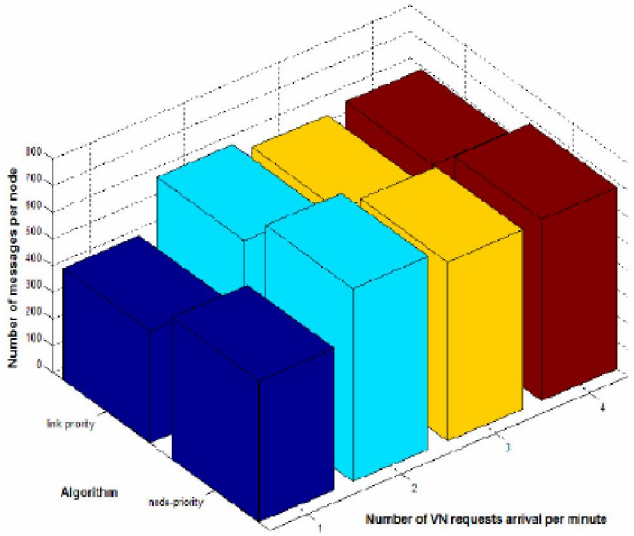

Figure 11 : Comparison of number of VN embedding messages

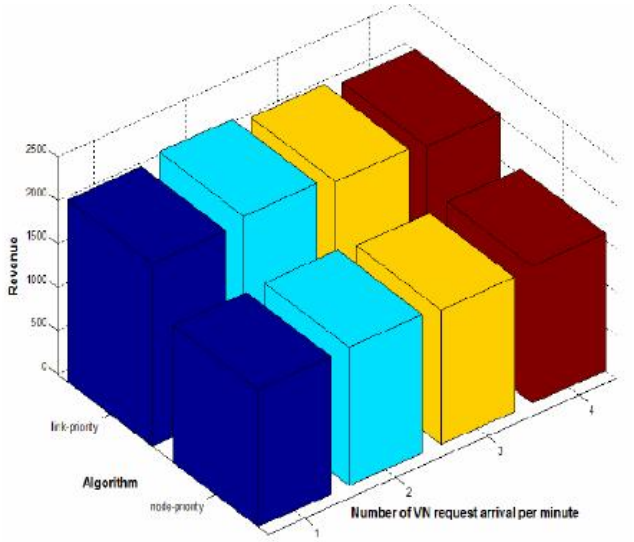

Figure 12 : Comparison of embedding revenue

Figure 12 and Figure 13 show the comparison of revenue and cost, respectively. Combined with Fig.9, the two algorithms can produce different acceptance ratios so they lead to different revenues. The link-priority algorithm has high acceptance, so it has high revenue. Although the two algorithms have different revenue, they have almost the same cost, as shown in Figure 13. The reason is that the node-priority algorithm allows distant virtual nodes to be embedded into the substrate network.

As the number of VN requests varies from 10 to 50, from Figures 6-8 we find that with VN requests increasing, the resources of the substrate network gradually decrease, the VN acceptance ratio also de-

FULL PAPER

creases and the embedding time will increase. The increase in VN request embedding failures increases the messages traffic in the substrate network, which in turn affects the VN request acceptance ratio and embedding time.

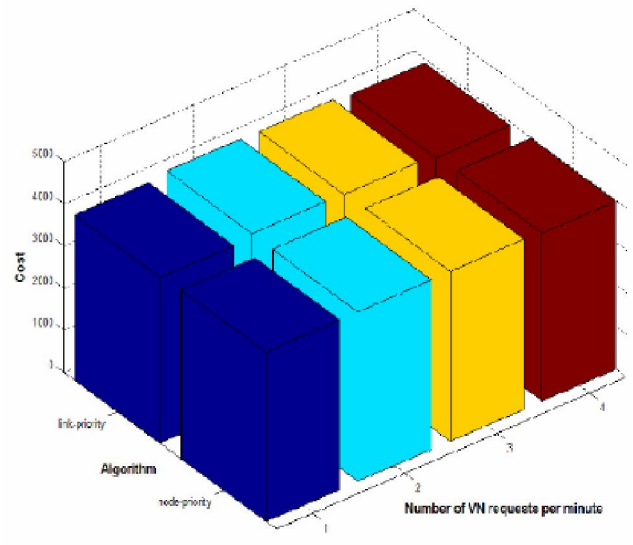

Figure 13 : Comparison of embedding cost

From the comparison of the link-priority and node-priority algorithms shown in Figures 9-13, the link-priority algorithm has better performance than the node-priority algorithm in terms of acceptance ratio, embedding time and revenue.

CONCLUSION AND FUTURE WORK

This paper presents the design of the distributed and parallel VN embedding framework. Inspiration of the thought of map-reduce, we decompose VN request into meta-requests. The distributed and parallel VN embedding algorithm can be a potential starting point for self-management of virtual network. In the future, we will provide the intelligence algorithm to substrate nodes in order to decrease the substrate traffic.

ACKNOWLEDGMENT

This paper is supported by the Natural Science Foundation of Fujian Province, China under GrantNo.2011J05150.

REFERENCES

[1] X.Cheng, S.Su, Z.B.Zhang, H.C.Wamg, F.C.Yang, Y.Luo, J.Wang; Virtual network embedding through topology-aware node ranking. ACM SIGCOMM Comput. Commun. Review., **41**, 39-47 **(2011)**.

[2] M.Chowdhury, F.Samuel, R.Boutaba; PolyViNE: policy-based virtual network embedding across multiple domain. Proceedings of the 2nd ACM SIGCOMM workshop on Virtualized infrastructure systems and architectures, August 30-September 3, New Delhi, 49-56 **(2010)**.

[3] J.Dean, S.Ghemawat; MapReduce: simplified data processing on large clusters. Commun. of the ACM, **51**, 107-113 **(2008)**.

[4] H.Di, H.F.Yu, A.Vishal, L.M.Li, G.Sun, B.H.Dong; Efficient online virtual network mapping using resource evaluation. Journal of network and systems management., **20**, 468-488 **(2012)**.

[5] M.Dorigo, M.Birattari, T.Stutzle; Ant colony optimization. IEEE Computational Intelligence Mag., **1**, 28-39 **(2006)**.

[6] I.Fajjari, N.Aitsaadi, G.Pujolle, H.Zimmermann; VNE-AC: virtual network embedding algorithm based on ant colony metaheuristic. Proceedings of IEEE International Conference on Communications, June 5-9 2011, Kyoto, 1-6 **(2011)**.

[7] T.Ghazar, N.Samaan; Hierarchical approach for efficient virtual network embedding based on exact subgraph matching. In Proceedings of IEEE Globecom, December 5-9, 2011, Houston, 1-6 **(2011)**.

[8] I.Houidi, W.Louati, D.Zeghlance; A distributed and autonomic virtual network mapping framework. Proceedings of the 4th International Conference on Autonomic and Autonomous System, March 16-21, 2008, Gosier, 241-247 **(2008)**.

[9] I.Houidi, W.Louati, W.Ameur, D.Zeghlache; Virtual network provisioning across multiple substrate networks. Comput. Networks., **55**, 1011-1023 **(2011)**.

[10] A.Khan, A.Zugenmaier, D.Jurca, W.Kellerer; Network virtualization: hypervisor for the Internet?. IEEE Commun. Mag., **50**, 136-143 **(2012)**.

[11] A.Leivadeas, C.Papagianni, S.Papavassiliou; Socio-aware virtual network embedding. IEEE Network. **26**, 35-43 **(2012)**.